Université de Strasbourg

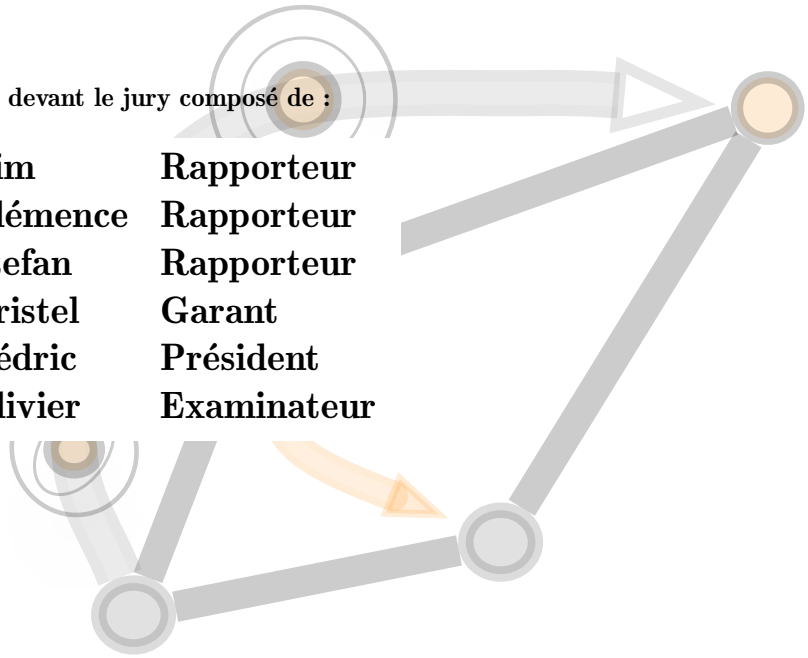
Habilitation à Diriger des Recherches

présentée par

Pascal MÉRINDOL

Build and Measure Routing Systems

About Computing and Monitoring the Path Forward



Soutenance prévue le 11/07/2022 devant le jury composé de :

Griffin	**Tim**	**Rapporteur**
Magnien	**Clémence**	**Rapporteur**
Schmid	**Stefan**	**Rapporteur**
Pelsser	**Cristel**	**Garant**
Bastoul	**Cédric**	**Président**
Bonaventure	**Olivier**	**Examinateur**

UFR de mathématique et d'informatique Université de Strasbourg

ICUBE Laboratoire des sciences de l'ingénieur de l'informatique et de l'imagerie

Laboratoire ICube UMR 7537, UFR Mathématique & Informatique

merindol@unistra.fr

Résumé

Les réseaux informatiques sont devenus en quelques décennies un outil incontournable que ce soit dans le cadre professionnel ou dans la sphère privée. Certaines des opportunités rendues possibles par cette émergence rapide sont fantastiques : l'accès à l'information n'a jamais été aussi simple et ludique, et de nombreuses tâches fastidieuses et répétitives n'ont plus lieu d'être grâce à l'automatisation de nombreux services. Internet est non seulement le support assurant la connectivité physique de l'ensemble de ces communications, mais il est aussi composé des briques logicielles nécessaires au transfert et au traitement des informations sous-jacentes, les protocoles. Mes activités scientifiques s'inscrivent dans ce contexte : comment améliorer et évaluer les performances des réseaux actuels et leurs protocoles afin de tendre vers un Internet plus robuste, réactif et adapté aux nouveaux besoins et services de demain ?

Le présent document a deux objectifs : il s'agit d'une part d'établir un panorama de mes travaux de recherche passés et présents, et d'autre part de définir un projet de recherche à moyen et long terme. Pour cela, après avoir décrit et illustré mes contributions majeures dans trois chapitres distincts articulés thématiquement, le dernier chapitre s'attellera à la description de mes projets futurs. Alors que les trois premiers chapitres s'efforceront de mettre en exergue les perspectives de mes travaux en cours, le dernier sera quant à lui dédié à mes projets à moyen et long termes pour approfondir mes activités dans le domaine et abordera aussi une évolution thématique dans d'autres domaines connexes comme l'algorithmique distribuée.

Les trois principales thématiques abordées dans mes travaux traitent essentiellement, et respectivement, (*i*) de routage IP dans les réseaux de coeur (fournisseurs d'accès filaires), (*ii*), de mesures IP à large échelle et de la supervision des domaines de l'Internet et, enfin (*iii*), de réseaux contraints aux données sensibles comme l'Internet des Objets. Pour chacune de ces thématiques, je présenterai mes principales réalisations qui y sont liées en prenant soin de détailler les extensions et travaux futurs envisagés.

Plus précisément, et en particulier, je m'intéresse aux techniques de re-routage rapide (prévention locale des pannes), d'équilibrage de charge (ou routage multi-chemins), et aux chemins à faible délai / haut débit dont la qualité de service requise est contrainte selon plusieurs dimensions. De manière générale, l'objectif du premier chapitre (*i*), est de proposer des méthodes algorithmiques efficaces assurant une convergence rapide et correcte des protocoles de routage intra- et inter-domaines. Nous nous intéresserons aussi au calcul de chemins multi-contraints et à plusieurs paradigmes de routage (e.g., au saut par saut ou par la source).

Dans le cadre de la seconde thématique (*ii*), je travaille sur l'élaboration de systèmes de mesure et de supervision. Ces activités reposent sur des outils de mesures actives et passives permettant de mieux comprendre et de *surveiller*, voire de révéler des parties méconnues d'Internet. Les défis sont nombreux mais peuvent se résumer ainsi : comment collecter efficacement des données fiables (sans biais) dans un système aussi large, hétérogène et complexe, dans lequel les exceptions sont nombreuses ? Nous étudierons notamment l'écosystème MPLS, les plateformes de supervision multi-sources et les détours d'acheminement survenant à l'intérieur des domaines qui composent Internet.

Enfin, en ce qui concerne les travaux présentés dans le troisième chapitre (*iii*), il s'agit de déployer des algorithmes de routage spécifiques aux propriétés des réseaux sans fil – en particulier dans les réseaux de capteurs et pour l'Internet des Objets en général. Deux problématiques seront abordées : l'économie d'énergie et la confidentialité des données, également développée dans un autre contexte centrée sur l'analyse de politiques d'accès au sein des architectures micro-services réparties. Comment prolonger la durée de vie des réseaux de capteurs contraints en ressource et comment assurer l'anonymisation et la sécurisation des données ? notamment celles émises par les objets connectés aux réseaux de bordure de l'Internet ou celles relevant d'une architecture multi-parties.

Mon projet de recherche se décline quant à lui en plusieurs directions de recherche. À moyen terme, il s'agit entre autres de développer mes activités concernant la convergence rapide pour déployer un plan de données reprogrammable efficace, de mesurer le déploiement de technologies comme *Segment Routing* ainsi que l'étude de nouveaux protocoles de routage économes en énergie pour les réseaux multi-radios. À plus long terme, mon intention est d'évoluer progressivement vers l'étude d'algorithmes de communications dont les propriétés peuvent être vérifiées formellement en me tournant vers les systèmes distribués en général, les algorithmes auto-stabilisants en particulier.
Ce rapport se concentre essentiellement sur les modèles, les méthodes utilisés et les principaux résultats obtenus dans le cadre de mes recherches plutôt que de détailler ses aspects plus techniques (que ce soit algorithmiques ou en terme d'implémentation) ou d'analyser ses performances internes. L'objectif est de fournir en priorité les concepts et résultats généraux des contributions qui ont davantage de chances d'intéresser et d'être ré-utilisés par la communauté sur le long terme.

Abstract

This report summarizes my main research achievements in the area of computer networks. In a few decades, computer networks have become an essential tool whether in the professional context or in the private sphere. Some of the opportunities made possible by this rapid emergence are fantastic: access to information has never been so easy and pleasant, and, thanks to the automation of many services, many tedious and repetitive tasks no longer need to be. Internet does not only ensure the physical support of such communications, but also provide the necessary pieces of software, i.e., the protocols and their rules used to forward the underlying data. My scientific activities fall within this context: how to improve and evaluate the performance of current network protocols in order to tend towards a more robust Internet, responsive and adapted to the new needs and services of tomorrow?

This document has two objectives: on the one hand, it is a question of establishing an overview of my past and present research work, and on the other hand, it is about defining a medium and long-term research project. For that, after having described and illustrated my major contributions in three distinct chapters articulated thematically, the last chapter will get down to the description of my future projects. While the first three chapters will highlight the perspectives of my current work, the last one is dedicated to the mid and long-term projects I aim at developing in my activities, and I will finally but shortly introduce a possible thematic evolution in other related fields such as distributed algorithms.

The three main themes addressed in my work mainly deal with, and respectively: (*i*) IP routing in core networks (wired access providers), (*ii*), large-scale measurement and monitoring of domains of the Internet and, finally (*iii*), routing within constrained networks such as the Internet of Things. For each of these themes, I will present my main related achievements, taking care to detail the extensions and future work envisaged.

More precisely, and in particular, I am interested in techniques of fast re-routing (local failure prevention), load balancing (or multi-path routing), and low delay / high bandiwth paths for which the required quality of service is constrained in several dimensions. In general, the objective of the first chapter, (*i*), is to propose efficient algorithmic methods ensuring fast and correct convergence of intra- and inter-domain routing protocols. We are also interested in the computation of multi-constrained paths considering several routing paradigms (e.g., hop by hop or source routing).

Within the framework of the second theme (*ii*), I am working on the development of measurement and supervision systems. These activities are based on active and passive measurement tools allowing to better understand and *monitor* IP networks, for example to reveal hidden parts of the Internet. The challenges are numerous but can be summed up as follows: how to efficiently collect reliable data (without bias) in such a large, heterogeneous and complex system, in which many exceptions arise as the rule? In particular, we will study the MPLS ecosystem, multi-source monitoring platforms and routing detours occurring within intra-domains, or autonomous systems, for the (external) transit traffic.

Regarding my work presented in the third chapter (*iii*), it is about deploying routing algorithms specific to the properties of wireless networks - especially in sensor networks and for the Internet of Things in general. Two issues will be addressed: energy saving and data privacy. How to extend the lifespan of resource-constrained sensor networks? and how to ensure the confidentiality of the data emitted by connected objects within edge networks of the Internet (e.g. with transformations like anonymization and aggregation)? The question of privacy and security will also be overviewed in the context of multi-party architectures involving several autonomous actors: in addition to our work on secured workflows using micro-services, we will look at policy verification and analysis considering metagraphs.

Finally, my research projects are divided into several research directions. In the medium term, this involves, among other things, developing my activities concerning fast convergence to deploy an efficient reprogrammable data-plane, measure the deployment of technologies such as *Segment Routing* as well as

proposing novel energy-efficient solutions for multi-radio IoT networks. In the longer term, my intention is to gradually move to distributed algorithms and distributed systems, in particular with the study of self-stabilizing algorithms. Relying on the formal foundations in use in this context, I aim to design networking algorithms and protocols having provable properties.
This report mainly focuses on models and methods used to conduct my research activities rather than detailing its technical aspects (such as algorithmic and implementation details) or deeply analyzing its practical performance. The goal is to first describe the general concepts and results of my contributions that are more likely to be reusable in the future than their internal details.

Acknowledgments/Remerciements

In French only: merci for your understanding.

Merci à ma famille et mes amis pour le soutien quotidien qu'ils m'ont offert durant la rédaction de cette habilitation à diriger des recherches. Merci Julie pour tous ces moments où tu as pris soin de moi, de nous et des enfants. Tu es ma bénédiction : compréhensive, attentionnée et presque toujours délicate, tu ne m'as que trop souvent pardonner d'avoir l'esprit ailleurs ou contrarié, ce travail t'est dédié à toi plus qu'à tout autre personne, merci ma chère colombe nacrée. Merci à ma fille Lou de m'avoir fait père et si fier, tu m'as aidé et inspiré bien plus que tu ne le crois. Je sais qu'un jour, pas suffisamment assez bien lointain, c'est toi qui m'apprendra et me fera découvrir des tonnes de belles choses en t'agaçant à peine. Merci à toi mon fils Clarence de nous communiquer tant de joie et d'énergique tendresse tous les jours, tu es doué de tous les talents. Tes sourires sont si malicieux et charmants qu'ils nous remplissent tous d'allégresse. Votre présence dans cet effort a été si précieuse que je ne suis pas sûr de pouvoir m'acquitter de cette dette autrement qu'en vous offrant une station de ski toute entière : merci pour votre réconfort quotidien, ce travail en est le fruit.

Merci à mes parents, ma sœur, Nicole, Chantal, Jean-Pierre, mes grands parents, oncles, tantes, neveux et nièces, cousin(e)s de près ou de loin, vous avez tous contribué à m'éveiller, m'apporter toutes les nuances de vos personnalités : j'ai beaucoup de chance d'avoir grandi parmi vous en Ardèche, en Bretagne, en ville, à la montagne, à la campagne et ailleurs. Merci pour tous les moments passés ensemble, les belles choses que vous m'avez données à voir, et merci d'avance pour tous les instants à venir.

Merci à mes amis, vous croyez en moi plus que moi-même et que de raison, et ça fait du bien ! Fier d'avoir grandi et évolué à vos côtés pour mes plus anciens compagnons d'infortunes adolescentes et existentielles, heureux d'avoir eu la chance de vous rencontrer, je vous dois tous beaucoup plus que des raclettes. Tous ceux avec qui j'ai passé des vacances ou des moments inoubliables se reconnaîtront : merci pour tous ces instants partagés en excellente compagnie ! Les meilleurs sont encore à venir si l'on s'en donne la peine...

Merci à tous mes collègues d'ici et d'ailleurs et plus particulièrement les doctorants, encadrants et co-auteurs avec qui j'ai eu le privilège de travailler. Vos noms sont déjà suffisamment cités dans ce manuscrit pour m'abstenir de vous faire l'offense de vous nommer ici. J'ai pris beaucoup de plaisir à travailler à vos côtés et systématiquement appris et développer beaucoup de nouvelles méthodes et idées grâce à vous. Ce manuscrit est le résultat d'un investissement collectif considérable en amont dans lequel j'ai savouré chaque instant, même les difficiles; a posteriori, c'est en surmontant ces difficultés à plusieurs que j'ai éprouvé mes plus grandes satisfactions et que le fruit de ce travail a pu mûrir.

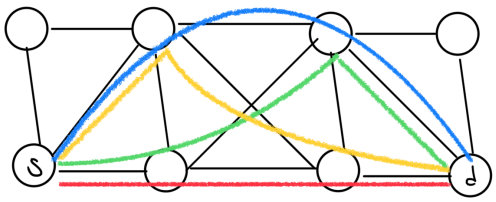
Enfin, merci à tous les membres de mon prestigieux jury : les rapporteurs en premier lieu pour l'effort qu'a dû demander la lecture de ce (trop) long manuscrit - je suis très fier de vous compter dans mes (re)lecteurs; les examinateurs pour avoir accepté l'invitation et m'honorer de leur présence et de leurs questions; ainsi que Cristel ma garante, pour son aide et son soutien tout au long du processus d'habilitation et, plus encore pour ces trois thèses encadrées ensemble.

Merci à vous tous, ceux présents à ma soutenance comme les absents, de m'avoir donné les moyens et le courage d'aller au bout de cette route pour y triompher modestement avec le présent manuscrit. Grâce à vous et aux ailes qui me poussent dans le dos, j'aurai aussi bientôt l'habilitation de rouler très vite sur routes goudronnées, souvent plus prosaïquement appelée *permis de conduire*, dès l'an prochain je vous l'assure !

Contents

Chapter I

Introduction: Build and Measure Internet Routing Systems



Building and measuring IP routing systems is the essence of my ongoing research in computer networks. While routing is basically about finding and deploying forwarding paths, such a computation is not enough: not only it should be efficient, correct and scalable, but verifying the ground performance with appropriate measurement tools and monitoring systems is also essential.

My research work is indeed all about Internet Protocol (IP) Routing & Forwarding and means to ensure and understand the underlying performance of such systems. In particular, **my first contributions consist in new architectures and routing schemes for a better reliability and quality of service support**. That is for example using multiple forwarding paths for load balancing and fast-rerouting, or providing incremental means to ensure graceful loop-free topological changes. In any cases, my goal is to enhance current routing approaches by proposing new practical features with simple and efficient algorithms.

On the other hand, by measuring and monitoring real IP networks relying on current deployed routing protocols, I gain knowledge about their real limitations and weaknesses. This second field of research not only motivates and inspires my first objective but is an independent activity in itself. However, **evaluating the current forwarding ecosystem requires to design complex fine-grained measurement platforms**. There exists many technical challenges and difficulties to overcome in this area of research as *the sum of exceptions are often the rule* (like layer-2 networks, missing hops or third-party addresses).

To some extents, I have also explored the same space of contributions in other playgrounds like Internet of Things (IoT). In particular, I am interested in reducing the energy consumption of wireless devices (to extend their battery life) and guaranteeing to the users that their privacy concerns are correctly enforced (to ensure the desire level of data confidentiality).

Finally, I will develop my mid- and long-terms research projects in the last chapter: I aim to solve new challenges and problems not only revisiting and unifying some of my previous works but considering new paradigms and methods. On the practical side, **I will consider the opportunity offered with programmable hardware to design both new algorithms for fast convergence and novel monitoring functions**. On the other hand, I expect to rely on distributed system theory to tackle more general fault models and provide provably correct self-stabilizing protocols.

Most of my research projects started with **the design of distributed routing algorithms and protocols**, in general for deploying traffic engineering, e.g., enabling multi-path routing and ensuring loop-free network reconfigurations. I am interested in fast-rerouting, load balancing and multi-constraint graph algorithms to compute appropriate routes. The first chapter of this document is the essence of all my work: proposing new IP routing mechanisms, simple but valuable enough to be easily and efficiently deployed in real IP networks. It is also about evaluating the benefices and the limits of my contributions to assess such techniques. During my post-doc and then with Ph.D. students, I investigated the opportunity to model and solve distinct networking problems such as the following ones:

Sec. II.1 Computing efficiently the two best first hop disjoint paths – Two Best First Hops algorithm (TBFH);

Sec. II.2 Finding the minimal loop-free sequence to shut down a link or an entire node – Adjusted Greedy Backward Algorithm (AGBA);

Sec. II.3 Ensuring the transparent convergence of BGP for any IGP changes – Optimal Protection Technique for Inter-intra-domain Convergence (OPTIC);

Sec. II.4 Computing efficiently the Delay Constrained Least Cost Paths with Segment Routing as a third constraint – Best Exact Segment Track for 2-Constrained Optimal Paths (BEST2COP).

These four problems are briefly introduced in Sec. I.1 with their context and motivations and are then more formally presented in Chapter II. Although their objectives differ, they all rely on similar graph theory constructs and models that ease the reasoning about basic IP routing features, and their possible extensions with more complex or advanced protocols and technologies like Multi Protocol Label Switching (MPLS), Segment Routing (SR) ([106]) or BGP (all observed from a measurement perspective in the following chapter). This first chapter provides an overview of research questions tackled in the three first chapters (related to the three fields of research mentioned before) with their context and motivations. It also introduces the projects I have in mind for the next years, in particular to supervise Ph.D students and explore new research directions. Generally speaking, this manuscript does not report an exhaustive list of my research activities on such topics but rather provides a selected panorama of my most significative technical contributions, ongoing projects and perspectives. My will is to exhibit their relevance both in term of variety and consistency. They not only share a common technical background (routing protocols) but also rely on the same theoritical tools (graphs in particular).

On the other hand, I am also interested in **measuring such routing systems**, checking their correctness and evaluating their performance. Not only the systems I design but the ones which are currently deployed, in order to study their limits. In particular, MPLS is one of the core technology used to deploy Traffic Engineering (TE). In this regard, it is interesting to understand the nature of this technology; many of its aspects deserve attention like its quantification, usages and best practices, and, last but not least, reveal its hidden part and to which extent it may obfuscate topology discovery. Moreover, I am interested in IP monitoring with both passive captures and active probing tools, in order to be able to ensure large scale Service Level Agreements (SLA) verification ([233]) in particular and study routing and forwarding patterns in general. For example, we deployed a multi-sources measurement platform in RENATER, whose aim is to correlate routing transitions with packet losses and loops. Finally, I also study anomalies like forwarding detours, analyzing both their causes and consequences. More precisely, these three measurement topics leads to the following challenges and related contributions:

Sec. III.1 How to measure MPLS tunnels and reveal hidden ones? – Trace the Naughty Tunnels (TNT);

Sec. III.2 How to deploy a fine-grained monitoring platform to quantify routing events and their impacts? – Dynamic Changes Analysis with Routing Traces (DCART);

Sec. III.3 Do IP forwarding detours occur and why? – Forwarding Detours (FD).

These three challenges are briefly introduced in Sec. I.2 with their context and motivations and then developed more technically in Chapter III.

Finally, while most of my contributions relate to wired IP networks, I am also interested in other fields of applications like IoT or Wireless Sensor Networks (WSN), and **wireless and edge networks** or clouds in general. Three objectives in particular caught my attention:

Sec. IV.1 How to Save the Battery Life of Devices?

Sec. IV.2 How to Protect the Data Privacy in IoT Networks?

Sec. IV.3 How to Ensure Security in Multi-Party Workflows?

These three challenges are briefly presented in Sec. I.3 and then discussed in Chapter IV with their context and motivations.

In the routing and measurement chapters, I also briefly introduce some other contributions that has led to significant publications (last section of these two chapters). While the first routing chapter (Chapter II) is the basis and the core of my scientific activities, the two others, and mostly the second on measurements, helps me to better understand and revisit the first, its context of deployment and inherent practical limitations. **Each section of each chapter focuses on a given problem statement and provides illustrations, definitions and notations rather than developing algorithms, discuss their complexity and elaborate on technical details or performance analysis.** Interested readers are invited to read related references to better understand the internal technical details of our contributions and their related performance. Finally, the last chapter (Chapter V) is dedicated to my research projects. I will introduce and detail some on my ongoing work and long term research plans. It provides both technical development, challenges and theoretical directions I aim to solve and explore in the future.

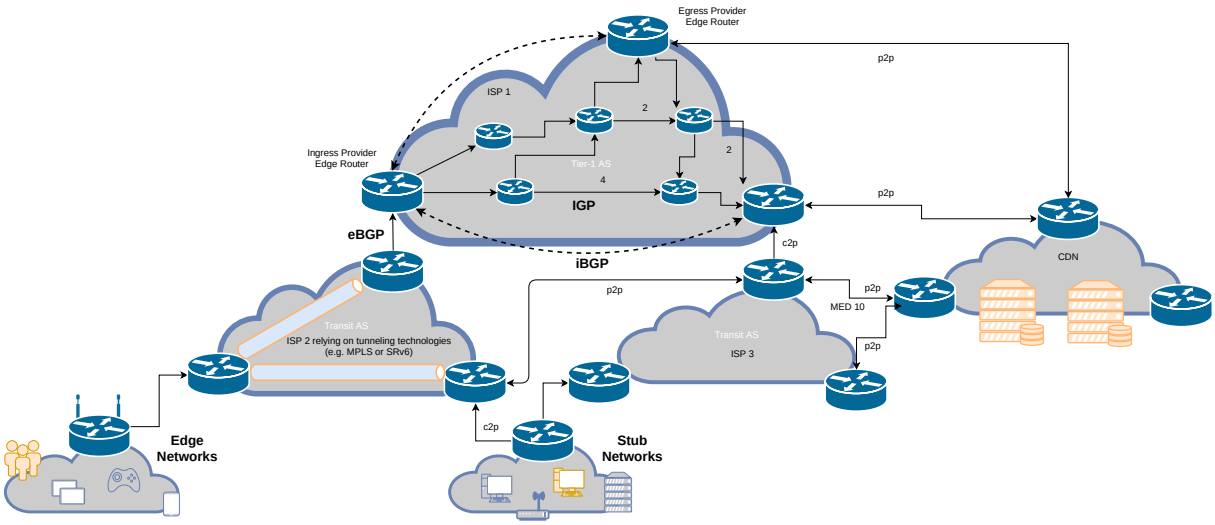


Figure I.1: An overall illustration of the general context of my activities. This figure sketches a global IP network inter-connecting Autonomous Domains (these ASes can be Transit or Tier-1 ISP, Edge and Enterprise Stub networks, and Content Delivery Networks (CDN)) having diverse peering relations (customer-to-provider, c2p, or peer-to-peer, p2p possibly tuned with the MED) and independent incentives. Between ASes, BGP is in charge of the global connectivity (in particular with eBGP sessions between border routers), while each AS has its own internal structure on which its IGP control the paths in use for the transit traffic (thanks to iBGP sessions to internally select the best external routes). Within an ISP, tunnels are often deployed on the top of the weighted internal structure to provide both scalability and TE features. Overall, thanks to such a physical and logical structure that lie on several technologies and protocols, users in edge and stub networks are able to communicate together, and typically retrieve the data they want to access within Content Delivery Networks (CDN).

Before continuing into more details, Figure I.1 provides a general illustration of the context of applications of my research in IP networks. I am interested both in the internal connectivity (within the IGP) and the global one (with BGP and its two sub components, eBGP and iBGP for external and internal routes exchange sessions respectively). On this figure that sketches the Internet hierarchy, several ASes (including a Tier-1, having no provider by definition, on the top) are interconnected with distinct

relations (customer-to-provider ones, c2p, or peer-to-peer ones, p2p, here possibly tuned with the MED). Such relations determine filters they apply on routes (and their local preferences), e.g. not advertising a route from a provider to another one to avoid offering them free lunch (i.e. a transit path including a valley). Within an AS, tunneling technologies are deployed to ensure scalability and possibly TE. Border routers, or provider edge routers, act as ingress-egress points for the transit traffic flowing end-to-end. Looking at the AS scale details hidden under their tunnels, networks rely on link weights to control the paths assigned to such flows and possibly distributed their load across the ressource according to several requirements (flow needs, congestions, etc). Here, in the example of ISP1, ECMP paths between edge routers form a DAG to offer path diversity as the basis of features like resiliency and TE. Overall, thanks to such an infrastructure, users in edge and stub networks (having no customers by definition) can communicate together, and typically retrieve the data they want to access within CDN. For this general picture to work, many network practices and protocols co-exist more or less harmoniously, and my research activities specifically focus on the study and the design of routing protocols involved in such a global infrastructure.

The following table of content summarizes both the structure of the introduction and the remainder of the document in general. While this introduction only provides a high level view of the context and motivations behind my work in distinct topics and areas, the four next chapters develop most of my contributions and propositions with more technical details. Not only there exists links between sections of a given chapter (e.g., BEST2COP, our solution developed for SR domains Sec. II.4 has inspired the revisit of *TBFH* in Sec. II.1) but also across chapters, e.g., the description of MPLS in Sec. III.1 and the hierarchical forwarding model introduced in Sec. III.3 is useful to understand chapter II in general and Sec. III.2 in particular. In each conclusion of each chapter, we start to draw the short term perspectives and ongoing projects in their respective field of research while the most promising ones are technically developed in the last Chapter V providing an overview of my plans for the future. It consists of several proposals ranging from designing smart data-planes to the study of self-stabilizing algorithms for network protocols.

Contents

I.1 IP Routing and Forwarding: Towards Reliability & Quality of Service

The context of application of my research, and of most of my contributions, is about routing in core wired IP networks; at two scales in particular, inter- and intra-domain systems, or Autonomous System (AS). Routing is an essential network layer feature implemented between the transport layer (data retransmissiion and congestion control) and the link layer (physical medium access) of the TCP/IP/Access model. Routers must implement several protocols to manage both local, metropolitan and wide area connectivity. Internet Service Providers (ISP), that is **each domain of the Internet**, indeed rely on at least two routing protocols to deploy forwarding paths among routers: the later is about setting internal best routes, while the former is intended to share global connectivity knowledge driven by economical relationships. The two co-exist to ensure full reachability and enable qualitative and reliable paths using adaptive mechanisms to react to topological and configuration changes. The Internet is indeed composed of independent domains known as AS. Within the Internal Gateway Protocol (IGP), with protocols such as OSPF or IS-IS, the goal is to provide intra-domain connectivity while Border Gateway Protocol (BGP) allows them to trade transit traffic. AS exchange routes through eBGP, while iBGP enables their dissemination among border routers. Since several distinct routes may exist for a given BGP prefix, border routers determine the best route towards each prefix by running the BGP decision process. This process consists, for each prefix, in comparing routes thanks to a lexicographical order based on a set of ranked attributes. As of 2021, the number of BGP prefixes has reached 900K [173], making this process computationally expensive.

Usually, internal routing is performed using a *link state protocol* while external routes are exchanged with a *path vector protocol*. The two differ in many aspects not detailed here, but the main difference is that intra-domain routing protocols are *safes* by design, i.e. they converge systematically towards a unique solution (deterministically), while the de facto inter-domain routing protocol currently in use, BGP, is not [313]: it may diverge because it is does neither verify isotonicity or monotony[1]. Both kind of **routing schemes are slow to converge**, but it is order of magnitude slower with BGP [292], even when sufficient conditions are met to prevent routing oscillations at the control-plane (like the absence of a dispute wheel [149, 132][2]), i.e. the ones required and in use for the safety of the convergence.

While the routing process (signalization and computation) is performed at the control-plane, the data-plane deals with packet forwarding performed *hop by hop* (with distributed intra-domain routing protocols, or simply IGP, like OSPF and IS-IS) or orchestrated at edges with *source routing* (with MPLS and/or SR – see sections III.1 and II.4 respectively). In practice, both kinds of routing paradigms are used depending on the needs. Best effort traffic is generally load balanced on best equal cost paths (Equal Cost MultiPath (ECMP)), while the re-routed traffic or TE flows can rely on provisioned routes with source routed forwarding paths using loose routing with encapsulation. Such paths are either setup in advance (with Resource ReSerVation Protocol for Traffic Engineering (RSVP-TE) control-plane) or encoded in each packet with SR. This later is a vibrant technology gathering traction from router vendors [15], network operators and academic communities [236, 339] while the former, the legacy RSVP-TE, or respectively Label Distribution Protocol (LDP) for best effort traffic (both along with the MPLS data-plane that is still relevant for SR although SRv6 has obvious practical advantages in this case) exhibits some significant overhead in large scale deployment [106]. With BGP, we will see later (in Sec. II.3) that such internal routes are taken into account in the internal decision process of iBGP, its internal counterpart to expose routes within the AS.

[1] That is, some BGP configurations (resulting from the routing policies and filters applied on a given topology), exhibiting neither *isotonic* nor *strictly increasing* behaviors, can – or will always – fail to converge. For example, filters introduced with the use of route maps can result in the loss of the *distributivity* of the route comparison operator on the route concatenation one. Moreover, the AS path length is only the second or third criteria in use and, as such, secondary regarding greedy selfish objectives, i.e. local financial preferences.

[2] In practice it translates in what is called the *Gao and Rexford rules*, or simply the *AS relationships of Griffin*:

- `import rules`: prefer customer routes over peer routes over provider's ones;
- `export rules`: export customer routes to all but filter provider or peer learned routes to send them only to customers.

Such rules are enforced (and make sense) because they follow economical incentives of each ISP in a self-fish manner (using the locally cheapest routes and not offering any *free lunch*); with them, sufficient conditions ensuring the absence of any dispute wheel are supposed to be met by design to ensure a global safety property. References [150, 73, 84], [Gamba, 2017] and [Rodau, 2017] provide more insights on this topic studying in particular the Algebraic causes behind control loops.

The ultimate goal of the main share of my work is about improving current routing protocols with efficient TE algorithms for enabling various kinds of forwarding paths. In particular, one of my primary motivation is to **reduce the convergence time** required to adapt to any topological change. Many critical services like health monitoring, energy infrastructures or financial flows require minimal disruption to perform well. More generally, with the growing demand for quasi-instantaneous communication services such as real-time video streaming, cloud gaming, and industry 4.0 applications, TE and performance guarantees become increasingly important [273]. A loss of connectivity greater than half a second, or even less in some critical cases, is not acceptable regarding strict SLA. Moreover, one of the challenge to tackle is to avoid any inconsistency and side-effect problems like forwarding loops, superfluous intermediary changes, flapping, etc. The main difficulty is to provide simple but efficient algorithms coming with strong guarantees, i.e. **provably safe and bounded in their worst-time case complexity**. Only such well controlled algorithms have a chance to be deployed one day at a large scale. Manageability and high availability are critical properties for IP networks. Unfortunately, with link-state routing protocols commonly used in such networks, topological changes lead to several kind of anomalies (e.g. transient forwarding loops or simply blackholes in the FIB) inducing service disruption, reducing the ability of operators to frequently and reactively perform management operations without affecting compliance to SLA.

In the following, we will see how to provide reliability and quality of service with the use of IP fast-rerouting and load balancing, safe routing updates and multi-constrained routes.

In the first chapter of this report, we will tackle four distinct but related problems, in particular the three first ones. They all relate to routing convergence more or less directly, and, as such, impose a strict constraint concerning the **time efficiency of the solution**. We develop several algorithms addressing the following challenges:

1. prevent any local failure by efficiently pre-computing all new best local routes (first hop distinct);
2. ensure graceful node or link shutdown with safe sequence of link state updates to deal with any planned routing event (e.g. router maintenance operation);
3. avoid superfluous BGP updates and extra-processing thanks to global IP fast-reroute;
4. enable delay constrained least cost (DCLC) routes in SR domains with an efficient multi-metric SPC algorithm. This last objective is orthogonal from the three previous ones related directly to the routing convergence, and their combination is let for future works.

Each of them deals with graph theory constructs developed in Chapter II. Computing only best (equal cost) paths is not enough and there is a need to extend existing methods to fill the gap. Depending on the objective (backup or TE routes) and the context (BGP or IGP only), our constructs differ in both their modelization and the data structures in use. Although the motivations behind each objective may look similar at the first glance, i.e. finding and deploying paths efficiently to offer a valuable routing experience for users, they slightly differ as they are complementary.

First, it is about finding alternative backup routes to avoid traffic disruptions in case of (unexpected) failures. Indeed, intra-domain changes are frequent [231], and may provoke numerous significant outages as described in [MDP+18]. Moreover, since BGP is tightly coupled with the IGP in use due to the *hot potato routing* rule of its decision process, the impacts of internal changes on the BGP convergence also constitute a challenging issue. While many fast re-routing schemes have been proposed both at the intra- [278, 43] and inter-domain (local or remote) scale [68], none guarantees a fast re-routing of *transit traffic* towards *optimal BGP routes* after *any single internal event*. However such events may lead to long-lasting connectivity loss [325, 110], performance degradation [351] and non-negligible churn [76]. Our aim is to propose routing alternatives that can be combined to enable failure protection for both the internal and the transit traffic, respectively in Sec. II.1 and Sec. II.3.

Second, my aim is to enable graceful updates not triggering any anomalies (including traffic disruptions) for planned operations. IP networks indeed need to frequently undergo topological modifications, e.g., to support hardware replacement, software upgrades, and configuration updates [270, 231]. Those modifications can induce forwarding anomalies, which, in turn, can reduce the ability of operators to frequently and reactively perform management operations [233] without affecting compliance to SLA.

Some ISPs have defined procedures to re-route the traffic out of a link [324] or a router [238] before shutting it down for maintenance. However, anomalies like forwarding loops can still arise in spite of these procedures. In Sec. II.2, I describe the challenges we succeed to resolve. Note that the same logic applies for sections II.2 and II.1: while the former explicitly prevent forwarding loops during transient periods of topological changes, the later is also designed to avoid such anomalies when unplanned events occur – making the network un-synchronized (indeed, while the node detecting first the failure has a backup route, the others may still consider obsolete routes). That is why I propose an all-in-one solution to deal with both kind of changes (planned or unplanned) at the beginning of chapter V.

Third and finally, since latency is critical in modern networks for various applications, we design an algorithm to compute constrained paths. The constraints on the delay are indeed increasingly stringent. For example, in financial networks, vast amounts of money depend on the ability to receive information in real-time. Likewise, technologies such as 5G slicing, in addition to requiring significant bandwidth availability, demand strong end-to-end delay guarantees depending on the service they aim to provide, e.g., less than 15ms for low latency applications such as motion control for industry 4.0, VR or video games [273]. For such interactive applications, the latency is at least as critical as the IGP cost. With bounded delays, the traffic can benefit from paths allowing for sufficient interactivity. It is thus relevant to both minimize the IGP cost and add an upper constraint on the latency to respectively enforce the ISP policies and the flow requirement. Computing such paths requires to solve a well known problem, Delay Constraint Least Cost (DCLC), which is NP-Hard [369]. DCLC has already been extensively studied [135, 152] but we revisit several aspects. In Sec. II.4, I describe the solution we propose to efficiently solve the problem in the context of SR domains.

At a high level, most of the algorithms we propose relies on usual Shortest Path Computation (SPC) models using variants of *Dijkstra or Bellmann-Ford* algorithms. However, not only our solutions extend them in several ways but also required new models and graph transformations. The time complexity of our algorithms is bounded (usually quadratic in V at worst) to ensure real time convergence, even in large complex networks. We evaluate each of them both theoretically (worst-case) and also experimentally (average and realistic cases). However, in this document I will only provide the overall formal framework with illustrations, not the most technical contributions such as experimental evaluations and algorithmical details. Interested readers are invited to read the referenced papers. Each section of chapter II first states and illustrates the tackled problem, then sketches the solution and finally provides insights about its performance and complexity. I also point limitations of proposed schemes and possible future works in their regard.

I.2 Internet Measurements: Understanding, Characterizing & Monitoring IP Networks

This field of research is very exciting technically speaking as Internet is not only large and complex but also wild and secret. Measuring and quantifying its properties (both structural and logical) is challenging because their complexities and heterogeneity can be costly to capture well from distinct perspectives (e.g., some links and relations can be hidden at first glance) and difficult to trace regarding its size (how efficiently coordinating the vantage points with efficient large scale campaigns?). Deploying and managing the necessary computing resource and hardware is complex but there exists several more or less open and flexible projects and platforms [1, 8, 67]. Overall, this topic is about designing models and measurement systems, including both active and passive tools, in order to analyse, characterize and, last but not least, understand IP networks. Only internal monitoring, as a subclass of measurements in general, has access to privileges and confidential information like internal protocols logs with routing traces. Others only rely on external data such as the ones obtained with `traceroute` and `ping` or more evolved tools like Scamper [220].

While each actor and stakeholder of the Internet knows its sub-system well, the way the whole behave together is something else. All interactions between protocols, and their possibly induced interferences (their distinct control loop may lead to antagonist decisions), become quiclky complex to analyze as many exceptions or even bugs arise. While a *holistic* study is the ultimate objective, *reductionism* is the reality. Collecting data at large scale requires many approximations, and **sampling the Internet is performed at**

several grains: Internet maps can be declined at the IP link-level, router level, PoP or AS levels. It is also possible to derive layer-2 networks like MPLS clouds or Ethernet Switches that are respectively hiding links and single point of failure.

Internet is one of the most complex distributed systems currently deployed. Internet topology discovery aims at analyzing such a system, generally using active and passive probing. However, most of current tools and methods come with several limitations. In particular, some MPLS clouds might obfuscate collected traces. The resulting Internet maps, their inferred properties, and the graph models on which topology generators are made are thus incomplete and inaccurate.

On the contrary to Internet topology discovery that aims to understand networks from the outside (and as such remain as general as possible with few requirements and assumptions), monitoring a network from the inside has several advantages as some privileges (e.g. no or less filters) and specific data access (e.g. the expected router level map observed with control plane messages) lead generally to a much more fine-grained view of the targeted network. Monitoring IP networks plays an important role in improving and managing their design. If ones aims to test whether SLA are verified, it is mandatory to keep track of connections. Last but not least, troubleshooting networks is unavoidable as a day to day task and evolved tools like PerfSonar [367] are popular and widespread. In brief, **monitor its (own) network is not only an option, but a precious service answering a critical need**. Many tools are available, from active topology discovery to traffic passive captures. While probing tools can be intrusive, monitoring high speed links is challenging. Modeling the routing is necessary to understand some kind of **anomalies due to BGP scalability limitations**. Why, when and how they arise? Indeed some limited routers may rely on a default gateway for the incoming transit traffic they forward. Such workarounds can lead to *persistent forwarding deflections*, i.e. not isotonic internal path, and rooting loops at worst. More surprisingly, detours from the optimal paths can result in their turn in *BGP lies*. That is forwarding paths being not consistent with control plane advertisements.

In Chapter III, I will develop three of the main aspects of my work in this research field:

1. What is the impact of MPLS tunnels on the Internet?
2. How routing changes are disruptive for the IP traffic?
3. Why internal routes of several ISP are not optimal?

After addressing each of these three questions in a dedicated section, we will briefly review in section III.4 some others of my contributions in the field like topology discovery with `mrinfo` and `traceroute` probing, including several analysis on the data collected (e.g. on the degree distribution).

For now more than twenty years and the seminal paper of Jean-Jacques Pansiot [262], the Internet topology discovery has attracted attention from the research community [95, 157]. First, numerous tools have been proposed to better capture the Internet at the IP interface level (mainly based on `traceroute`) and at the router level (by aggregating IP interfaces of a router through *alias resolution* [192]). Second, the data collected has been used to model the Internet [265], but also to have a better knowledge of the network ecosystem and how it is structured and organized by operators. The protocols configured in each AS (the IGP) and between them (with BGP relationships) have strong impacts on what can be observed from the outside: they deform the visible topology and its properties. In this regard, one technology is prevalent in the current routing control and forwarding data-plane systems, MPLS. It is one of the most popular and deployed tool by operators for enabling TE and internal scalability. While new path probing tools has been proposed recently, like the Paris-traceroute extension detailed in [341] or YARRP [50] to speed up probing campaigns, none of them focuses specifically on hidden MPLS tunnels. For example, `tracebox` [91] reveals the presence of middleboxes along a path while reverse `traceroute` [191] is able to provide the reverse path (i.e., from the target back to the vantage point) and `tracenet` [330] focuses on subnetworks. Passenger [308] and Discarte [307] extend `traceroute` with the IP record route option whereas Marchetta et al. [229] have proposed to use the ICMP Parameter Problem in addition to the Record Route option in `traceroute` [144]. To some extent, Discarte, Passenger and also Drago [230] with the ICMP Timestamp option, allow to reveal hidden routers along a path. However, such options are likely to be filtered, and our aim is to go beyond and not rely on any specific options nor messages. In section III.1, we present our measurement work dedicated to MPLS tunnels: classifying them and revealing hidden tunnels reshaping the Internet being our main contributions.

Measuring networks and their performance can be achieved at a finer grain. For example, with a valuable (internal) monitoring system, one can be interested in the relationship between packet losses and routing changes in an operational network. More precisely it is about analyzing the anomalies and outages disrupting critical services both at the intra- and inter-domain scales relying on multiple data sources to perform and understand correlations among them (studies like [40] only compare active and passive methods for packet-loss measurements but they do not cross them with other data sources). To do so we designed and deployed several measurement models and primitives, in particular we deployed two monitoring platforms over RENATER and GEANT, respectively the French and European research and education networks. Few research has targeted flexible multi-sources monitoring. Indeed, most previous works focused on specific measurements restricting their analysis to the monitoring of the routing protocol, e.g., [344] and [302]. Others, for example in a study of the Sprint's IP backbone [174, 231] use more data sources, but only analyze the impact of link failures occurring during maintenance windows on the IGP convergence period. Generally speaking most existing proposals only correlate a couple of indicators, e.g., IS-IS intra-domain routing changes with trouble tickets in [239], ICMP messages to detect loops and packet losses during routing changes in [351]. Often routing changes are not monitored through an accurate control plane listener but only by running `traceroute`, and the main focus is often on the BGP control plane (e.g., [103] or [250] to monitor a single end-to-end inter-domain paths but with a fine-grained NTP synchronization). To the best of ur knowledge, only NICE (Network-wide Information Correlation and Exploration), presented by Mahimkar et al. [225, 226, 227] use both passive and active monitoring for performing wide correlations. They use sophisticated statistical tools and multi-resolution methods to perform such correlations instead of relying on accurate enough timestamps as we will do in our similar work. Moreover, we investigate distinct aspects specific to our objectives, e.g., including ECMP and anomalies like forwarding loops and flapping, and our outcomes are different. Section III.2 summarizes our main achievements with DCART in particular.

Finally, the scale and the inherent complexity of a network large as the whole Internet make it prone to several kind of errors and failures challenging to observe and measure. Having privileges to access internal data is not enough to detect all kind of anomalies, more advanced ground techniques may be required. For example, to mitigate scalability issues of the Internet continuous growing (over the last 8 years, the full Internet feed has doubled in size, reaching ∼900K prefixes at the end of 2021 [172]), some operators filter prefixes, perform prefix aggregation and use default routes. Indeed, the sustained increase in the number of prefixes advertised on the BGP has led Autonomous Systems to exchange more update messages [101, 171, 162], and to suffer from scalability issues. Indeed, considering the current trend, maintaining a full Forwarding Information Base (FIB) may be challenging, specially for AS incapable of upgrading their network devices regularly [368, 329, 11]. In this context, networks operators have found alternatives to endure with legacy routers unable to maintain a complete FIB in memory. For example, in a BGP-free core, tunneling techniques reduce the size of the FIB on core routers [82]. In addition, partial iBGP dissemination relying on route-reflector hierarchies may also boost scalability [343]. This technique allows routers to maintain less BGP peers and, in some rare cases, may even prevent the full redistribution of BGP prefixes within the AS [345]. In addition, memory-constrained routers may aggregate routes to limit the number of FIB entries [315]. Other type of workarounds consist in storing a partial-FIB [38, 190], and redirecting traffic via default routes towards more capable routers (e.g. having a full-FIB). Some network operators even apply this technique on switches with IP capabilities [27]. Despite being effective, such workarounds may result in routing inconsistencies, i.e., in routers along a forwarding route mapping the same IP addresses to different IP prefixes. In turn, the exit AS border routers associated with these distinct prefixes may potentially differ. For some prefixes, forwarding detours may occur, i.e., traffic may deviate from best IGP paths. Section III.3 provides an overview of my activity in this field.

In each section of this chapter, I will provide the necessary background to introduce at a high level the tools and platforms we develop to answer these research questions. Moreover, rather than providing technical details, I will describe main results and lessons learned thanks to our measurement campaigns. Finally, I will pinpoint weaknesses of existing schemes to propose novel research options and directions in the field.

I.3 Energy, Security & Privacy Concerns in Internet of Things and Multi-Party Architectures

Routing is also a necessary feature for wireless edge networks. However, requirements and constraints greatly differ from wired core networks. While most changes lie at the physical and link layers because the medium is much more unstable and complex to share, usual routing protocols must at least be adapted to address such challenges (e.g. neither OLSR is just a variation of OSPF [12] nor LOADng is an extension of RIP [80]). Wireless networks are diverse in their architectures and deployment paradigms with many potential services and technologies at stake. Two contexts in particular attracted my attention: Sensors Networks (WSN, [360]) and Internet of Things (IoT, [363, 213]); with two respective objectives: energy ([206, 207]) and privacy ([311]). Moreover, in the context of multi-party workflows deployed in the cloud [269], I am also interested in security architectures using micro-services ([178]) and verification and policy control methods used in the cloud [182]. More precisely, my work in this field relates to these three problems:

1. reducing energy consumption in WSN having a converge-cast data collection model $n \rightarrow 1$ ([359]): by deploying a tree rooted at the sink maximizing the number of leafs (such nodes are then passive regarding the forwarding), energy can be preserved to extend the network life time;

2. ensuring privacy for IoT data-streams ([195, 358]) within a multi-domain context: by aggregating the data and filtering it directly within the network, the level of anonymity of confidential and sensitive streams can be increased by design;

3. guaranteeing data security both at rest and in transport in the context of multi-party workflows deployed in the cloud. Not only we propose a micro-services architecture but we are interested in policy verification and analysis ([279]).

Chapter IV will briefly develop these three goals and summarize our main achievements in their respective field of application. **Rather than describing technical details of the solutions and schemes we proposed as it is done for the other chapters, I will only provide a brief overview of my (limited) contributions in these research topics**. In particular, I will focus on the problems offered by this challenging context: wireless edge networks are indeed much more constrained (e.g., in terms of processing capacity) than wired transit ones and subject to more difficulties and requirements to address [310] (e.g., to energy limitation and interferences due to an unstable medium). Finally, in the context of IoT communications and the shift to the cloud, the privacy and security concerns become even more relevant [97, 371]. We have designed and proposed novel network architectures in order to mitigate the impact of leaks in both an IoT context considering a NDN paradigm ([26]) on the one hand, and the deployment of workflows in multi-party clouds on the other hand. While we have designed a data advertisement algorithm for in-network aggregation for the former project, we have proposed a micro-service architecture allowing isolation for the latter. In addition we have considered metagraphs [45] and hypergraphs [34] as a basis to verify and analyze the enforced policies.

Wireless networks are more complex and fragile architectures than core networks but they represent great challenges for researchers in networking as they pose many problems, e.g., battery, interferences and constrained hardware ([242]). One promising technology to deal with such difficulties is the use of two radios to possibly split signalling and data transmissions. This is one of the model I aim to consider in my future works in this field.

I.4 Research Projects: from Smart Forwarding to Multi-Radios in Distributed Systems

Network constraints evolves over time with the emergence of new services and technologies. These last years, new paradigms arise like software defined networking [185] or smart data-planes [306]. The hardware also evolves with new capabilities like the support of several radios in IoT [138] and higher bandwidth. In the next years, I aim to explore the new challenges offered by such technological changes,

in particular the opportunity to take the control of the data-plane for designing agile routing systems supported by accurate measurements. More precisely, my projects relate to these five problems:

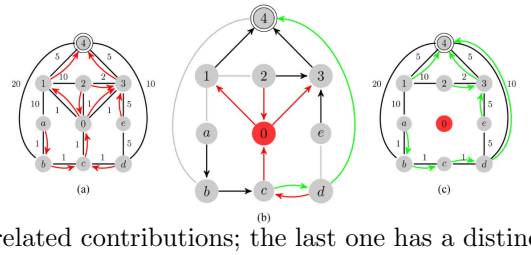
1. designing a comprehensive safe routing convergence framework leveraging simple assumptions (e.g. symmetric valuation);
2. relying on programmable hardware to enhance smart and fast data-plane decisions to recover from faults;
3. studying multi-radio multi-topology theoretical opportunities for wireless networks;
4. defining new measurement primitives for modern networks embedding recent technologies;
5. looking at less restrictive assumptions when considering distributed faults (i.e. byzantine ones rather than only analyzing the fail-stop model) and consider the network stabilization in general (with robust self-stabilizing properties).

Chapter V will introduce all these practical problems I will try to tackle in a more or less near future. While the first topics are in fact already ongoing projects almost mature, the last ones (3-5), and the latter in particular (5), represent the long term goals and methods I aim to investigate and rely on to gradually move to distributed systems theory to solve dynamic graph problems. Overall, these five problems are all related, each being a distinct aspect of a global routing framework based on forwarding inputs.More specifically, in short or medium terms I aim to develop the former topics (1-2) by directing my first Ph.D. thesis. Most of these projects involve the researchers of my group or are collaborations with other abroad colleagues working in their respective field. My long term aim is to formally study routing stabilization considering recent data-plane and technological opportunities. I envision both to build protocols taking advantage of new forwarding and routing features (e.g. new programmable hardware, SR or multi-radio support) to design stable but agile networks, and verify them with formal means of distributed algorithms.

This report will now start with the presentation of my previous and ongoing works to build and measure Internet routing systems in several contexts of applications. In each section of the following chapters, I will first highlight the main research questions I aim to tackle and, if needed, then divide them into sub-problems to be solved to reach the proposed research objective. Generally speaking, rather than providing a summary of my publications and present works, my intention is to underline what I consider to be my most valuable contributions in the field. After a brief description of the background and the related works, my goal is to quickly move to the core contribution of my past and current proposals. This attempt essentially consists in providing either the formal framework used to model and solve the underlying research problems (in particular in Chapter II), or the main technical challenges addressed and their outcomes (in particular in Chapter III). Finally, note that I revisit some of my past works under the prism of new technological opportunities like SR or programmable hardware. In this case, and especially in Chapter V, I elaborate on a number of new technical details (e.g., sketch of algorithms or proofs) that has not been published yet.

Chapter II

Designing and Improving Internet Routing Protocols is the First Key



Improving IP Routing in core networks is my very first objective. **Fixing current limitations concerning the convergence time** being my main concern, I study means to reduce this period to make its impact marginal even for critical services (e.g. delay constrained ones). In this chapter, I present my four main related contributions; the last one has a distinct goal but also offers path diversity. First, it is indeed about enabling path diversity to make IP networks flexible and support *fast-rerouting & load balancing* with efficient Shortest Path Computation (SPC) algorithms, i.e. with TBFH. Second, as routing transitions implies *transient micro forwarding loops*, we develop several algorithms, e.g. AGBA, able to prevent them. For planned routing updates, e.g. maintenance or reboot operations, *AGBA* only relies on incremental link state updates that do not require any change in current protocols. Third, and while the two first proposals are designed for intra-domain routing protocols (IGP), we tackle the problem of the *superfluous BGP convergence due to hot-potato routing* in case of IGP events with OPTIC.

Contents

Before starting, let us define the formal context of this work with the basic notations introduced in Table II.1. They are general graph notations common to all sections of this report. They relate to the most basic best path computation concepts while more specific definitions and refinements are then progressively introduced in the following (sub)sections.

Table II.1: General Basic Notations to Define LoopFree Rules

Notations	Definitions
$G(V, E, w)$	directed graph G with a set of vertices V (or nodes), a set of edges[1] E, and a positive valuation $w : E \longrightarrow \mathbb{N}^+$ of edges
n, m	respective cardinal of V and E, i.e, $n = \|V\|, m = \|E\|$
$e = (u, v)$	edge $e \in E$ linking node u to node v
$succ(u)$	set of successor nodes of u
$deg(u)$	incoming or outgoing degree[2] of node u, e.g., $deg^+(u) = \|succ(u)\|$
$p_1(s, d)$	a shortest path connecting s to d using an additive metric regarding w
$c_1(s, d)$	IGP cost of the path $p_1(s, d)$, or simply the best distance from s to d

Each edge is weighted with the w function that usually models a given link property[3] such as its delay or capacity. With an additive metric relying on the most usual algebra for intra-domain routing purpose, i.e. $(min, +)$, computing the best routes from all sources $s \in V$ and towards all destinations $d \in V$ implies to find a shortest path $p_1(s, d) = e_1, e_2, \cdots, e_i, \cdots, e_t$ for each source-destination couple minimizing the sum $\sum_{i=1}^{t} w(e_i) = c_1(s, d)$, that is the cost offering the best (i.e. shortest) distance regarding the metric in use. In practice, considering the *hop by hop forwarding paradigm*, the best known SPC algorithm[4] to compute shortest paths from one source to all destinations in a digraph, possibly including circuits and having positive valuations, is the one of *Dijkstra*. Its complexity can be summarized as $O(n \log n)$ for sparse graphs using a binary heap as the priority queue.

As with most usual SPC algorithms, the one of *Dijkstra* relies on a structure called a Priority Queue (PQ) to perform its distance updates at each iteration. This structure is used to store and evaluate the distance of visited paths and supports three methods: *extract_min*, *decrease_key* and *insert_key*. The first operation allows to find the current minimal cost path in the PQ, while the two others respectively modify and creates a new path cost entry in the PQ. The time complexity of a SPC algorithm depends on the kind of PQ used.

Let us denote x the cost of its *extract_min* operation (being both the search of the minimum key and its suppression), r the cost of *decrease_key*, and i the cost of *insert_key*. The worst case complexity of Dijkstra is then bounded by $O((x + i) \times n + r \times m)$ with any PQ. With an array list as PQ, we have $x = n$, $r = i = 1$ whereas a binary heap reduces the cost of x to $\log n$ while the cost of i and r are increased to $\log n$. The optimal structure to implement an efficient PQ for the Dijkstra algorithm is the *Fibonacci* heap designed by Tarjan et al. [83, 124, 59]: the amortized cost of the minimum extraction is then $x = \log n$ while $r = i = 1$ on average. With a Fibonacci heap as PQ, the amortized time complexity of Dijkstra is thus only $O(n \log n + m)$. However, while it is very attractive in theory for very large and dense graphs, in practice, for relatively small and sparse IP intra-domain networks (typically with $n \approx 100 - 10,000$ as measured in [MFB+11] and with $m \approx 2 - 5n$ on average [MDBP10]), a binary heap is generally enough to achieve very good performance[5]. Indeed, the complexity is then simply in $O(n \log n)$

[1] While their existence is assumed symmetric, i.e., $(u, v) \in E \Rightarrow (v, u) \in E$, their valuation can be asymmetric in the general case (otherwise we can consider a weighted but non directed graph model).

[2] In practice, we will consider them as equal: $pred(u) = succ(u)$ and $deg^-(u) = deg^+(u)$. The directed graph is a graph (each edge exists in both direction) where only the directed valuation can differ (not the edge existence).

[3] Note that relying on an unified valuation taking multiple characteristics into account may result in a loss of relevant informations (e.g. to verify a strict constraint)

[4] SPC stands for Shortest Path Computation and does not necessarily compute All Pairs Shortest Paths (APSP) as, according to the technology in use, it may not be required.

[5] In general, a binary heap still provides better results than with a Fibonacci heap as PQ for $n \leq 10,000$ (because of its overhead not well counter-balanced for small graphs) and is already better than simpler structures like lists or buckets (that can benefit from graph with small diameters or having few distinct distances) for $n \geq 100$. Note that with very simple graphs or, for example with constant valuation, the complexity is linear in $n + m$.

with both structures. Since our proposals often rely or enhance this algorithm, we face the same situation: **taking into account the real IP network properties, we can expect a complexity of $O(n \log n)$ for each SPC** using simple operational data structures for modeling the PQ, that leads notably to a complexity of $O(n^2 \log n)$ to retrieve All Pairs Shortest Path (APSP).

Each router runs this algorithm considering itself as the source and then only install in its FIB the best next-hop(s) for each destination, that is the first hop of the computed shortest paths. Eventually, the *isotonicity*[6] and the *monotony*[7] of this basic algebra ensures that local decisions are globally consistent even when enabling/deploying all paths having the best equal cost (ECMP) in a distributed manner (hop by hop). In other words, within an IGP, there is neither control loops nor forwarding loops as long as the hop by hop composition leading to ECMP routes remains stable: the traffic can be safely and locally load balanced on each router having multiple ECMP next-hops.

The other available algorithms and methods will be introduced when necessary. In particular in section II.4 where we will discuss the more general case of multi-metric networks.

[6] That is, sub shortest paths are also shortest paths: the optimality of sub-best paths.
[7] The underlying additive metric leads to a strictly increasing function with positive weights.

II.1 TBFH: the Two Best First Hops Algorithm

This first section presents a summary of the work developed during my post-doc at the Université catholique de Louvain with Pierre François and Olivier Bonaventure. Jean-Jacques Pansiot, my former Ph.D. director, was also part of this adventure as almost always until his retirement in 2015. The two main contributions we produced at the time are detailed in [MCP09] and mostly in [MFB+11]. **I am currently revisiting some of the opportunities my previous contributions enable with SR and more recent programmable hardware. In that sense, many of the contributions presented here are in fact novel proposals, in particular when they are technically detailed with new proofs.** Some of the extensions in progress are already detailed here while I continue with my most relevant progresses at the beginning of chapter V.

Protecting the traffic in case of failures is a common objective in IP networks [304]. There exists several technologies to (more or less) easily achieved full protection, e.g. MPLS (see section III.1 in the next chapter), Segment Routing (with TI-LFA [43]) or Loop Free Alternate (LFA) variants in general [290] (along with other IP-FRR methods) relying on IP-in-IP encapsulation [209] or packet marking. In particular, section II.4 presents SR in details for TE. In the usual and local IP context, full protection means that each router locally protects each of its outgoing link (possibly its neighbor) to handle all single link (or node) failure within the network by pre-installing backup entries to immediately react to any failure. Such a complete protection is generally achieved in a distributed fashion and is neither necessarily optimal nor correct: sub-optimal transient paths can be the only ones available for a while (in particular when calibrating timers to speed up the convergence period [145]), or even forwarding loops, can occur before reaching the full re-convergence. The induced overhead of the re-routing protection scheme should also be taken into consideration (e.g. with the states, messages and the overall signalization/computational cost of the technology in use). The originality of our proposal is to propose a lightweight computation process that allows, in only two SPC runs, to select and validate all post-convergence next-hops, in less than a segment with symmetric weights.

Avoiding transient loops in case of failures is generally enforced in the conditions at play for re-routing the traffic. This is notably the basis of all LFA variants as such rules work correctly in network differing from one event: as long as neighbors differ from one routing/forwarding state at most the traffic is safely detoured. Typically, the node detecting the failure is asynchronous with respect to it previous uptream nodes, that may now turn to be its downstream ones in the new configuration. LFA, R-LFA and more advanced TI-LFA variants enforce the release point to be safe and avoid the failed component because its pre- = its post-convergence paths. However, implementations generally rely on timers (Shortest Path First (SPF) interval and throttling) and does not deal with all kinds of transient loops (e.g. they handle only local ones, transitively adjacent to the failure, as remote routers do neither pre-compute anything nor rely on SR). In our work, we aim to achieve a fully integrated (fast re-)routing solution with no timers, preventing all kind of transient loops and mitigating the use of transient sub-optimal paths. Moreover, such a solution should be efficient both in terms of signalization and computation, that is mitigating any kind of overhead. To enable all these advantages, we will see that the only required assumption is to opt for symmetric weights and, on the technological side, the use of an encapsulation technique such as SR and possibly a flexible data-plane allowing embedded forwarding updates.

The question I am to address in this section is the following:

Research Question

How to Efficiently Protect an IP Network From Local Failures to Ease and Fasten its Safe Routing Convergence?

Such a general objective leads us to the following contents (that is split between this section and section V.1 in the last chapter):

1. In this initial background section (Sec. II.1.a), we will discuss how efficiently computing the best available local re-routing paths, i.e. the post-convergence ones for each destination and each local failed link, and what are the other current options;

2. Second, in Sec. II.1.b, I will detail how to compute, validate and deploy such optimal backup routes with *TBFH*. In particular, we will already observe that **symmetric weights ease the computation of post-convergence paths and their deployment**;

3. Sec. II.1.c then provides and discusses necessary extensions to this work to fit with SR (and TI-LFA). On the contrary to the previous section, this is a **fully novel part that is the launching pad for what is then extended in my initial short term projects** (Sec. V.1). In particular, this section introduces generalized properties for remote failures and discusses opportunities to perform data-plane driven forwarding updates;

4. In Sec. V.1.a, I will continue to elaborate on symmetric weights showing that such an assumption also eases the detection and prevention of all transient forwarding loops (not only the ones transitively local to the failure) in a similar fashion as in Sec. II.2 but with less overhead and to deal with unexpected events. On this basis, I will then briefly introduce the baseline for a fully integrated optimal and safe SR re-routing solution not requiring any timers. I will also expose the difficulties and technical challenges to achieve such an ideal convergence;

5. Finally, in Sec. V.1.b, based on previous observations and their outcomes, I expose my overall objective: reaching both a full IP protection and prevent all kinds of transient anomalies. Opportunistically, that is using implicit SR updates, as well as data-plane operations to detect forwarding loops, distant and local failures can be solved with data-plane driven updates. Neither timers, nor explicit synchronisation are required when the graph valuation is symmetric. All elaborated features enable a quick optimal convergence, that is globally correct while as fast and efficient as possible, i.e. with a low computing and incremental operational overhead not requiring all devices to support data-plane updates at line-rate.

The detailed table of content of the first part of this ongoing project is given here, it includes half of the current work (whose the remainder is available in section V.1). This decomposition in two steps is eased thanks to section II.2 which is the natural link between the two parts.

II.1.a Compute, Validate and Deploy Alternate Paths

II.1.a.1 About Basic IP Protection Techniques

I can now start to introduce the graph problem we solve in this section starting with the most basic rules to satisfy to deploy safe backup routes. With notations provided in Table II.1, one can define simple loopfree forwarding rules for extending load balancing and/or safely enabling fast re-routing techniques, e.g. respectively with **DC** (Downstream Criteria) or **LFA**[8]. Such rules are used in many contexts [357, 349, 288, 270] and can be extended or generalized with more complex frameworks [258, 116, 298]. Many

[8]They are two of the most popular and typical sufficient conditions in use because they are simple to deploy as they only assume the basic hop by hop forwarding paradigm.

works exist in this area and I will not survey them in detail as the literature is too diverse on this topic. Interested readers are invited to read the following surveys and related works: [74, 186]. From the seminal paper [128], there is now many existing works [201, 115], some tackling many link or node failures [100], multipath-routing [298, 200] or other using control [253] or data-driven methods [211, 75]. Load balancing and path deployment methods have been proposed in conjunction [187, 352, 248]. The originality of our approach is to provide the most efficient algorithm(s) to compute all kinds of *post-convergence* LFA in SR domains (offering source loose routing encapsulation), not depending on the router degree. *TBFH* requires only two SPC runs (for all destinations) and exhibits numerous other advantages when assuming symmetric weights: a low deployment overhead and the ability to handle remote and node failures easily. We generalize this idea within a lightweight SR framework proposal that is able to quickly, fully and correctly update the network and its forwarding paths, without generating transient loops as in [284] and the work developed in the next section II.2.

Let me now formally define the basic principles and rules on which our proposal are made (as many others in the literature that has been cited previously):

Loop Free Rules A neighbor $v \in succ(s)$ is safe regarding the triplet (s, d, e_1) (source, destination and failed link – the first edge $e_1 \in p_1^1(s, d)$), that is loopfree according to a given objective (load balancing or only re-routing), if:

$$\textbf{DC} : c_1(v, d) < c_1(s, d) \quad \text{(Eq. II.1)}$$

$$\textbf{LFA} : c_1(v, d) < c_1(s, d) + c_1(v, s) \quad \text{(Eq. II.2)}$$

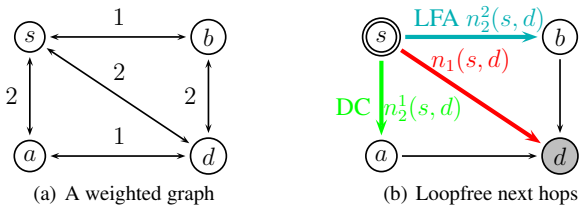
Such loopfreeness conditions allows to correctly re-route the traffic (and avoid transient loops) even when the network states are desynchronized of one event, that is the failure of e_1, the first edge of the best path between s and d[9]) but they depend of the context of application (LB or fast re-routing only). **While the first rule (a generalization of ECMP) enables load balancing and can also be used for re-routing (as LFA is a looser condition than DC), the second only allows for fast-rerouting.** Indeed, the second rule assumes that only the node detecting the failure (here s) applies it, not both s and v at the same time (since loops may occur otherwise). There exists more subtle rules [357, 209, 290, 116] (e.g. variants or generalization for more complex protection) whose some allow for *simple* paths instead of *elementary* ones, but they have at least the same complexity requirements as the two first: typically computing the best distances, $c_1(v, d), \forall v \in succ(s)$, that is at least the Shortest Path Tree (SPT) rooted at each neighbor of the source node. **This naive method thus results in the computation of** $deg(s) \leq n$ **SPT**, that is at worst $O(n^2 logn)$ for some source routers (for sparse networks where $m \approx nlogn$ at worst or relying on the most efficient heaps such as the *Fibonnacci* one). For maximum degree router, it thus becomes equivalent to running a complete APSP algorithm.

In the following, we will introduce the algorithms and technologies necessary to both and respectively perform efficient computations (i.e. not depending on the router degree) and reach a full protection coverage. Indeed, for the latter aspect, such rules are not enough to ensure the protection of all links (or nodes) towards all destinations. That is there exists cases where the graph structure and/or the weight assignment is not suited to ensure full protection. Note that we assume that the graph is bi-connected by design of the network (otherwise full protection cannot be achieved in any cases).

ECMP and Alternate Next-Hops As current IP networks enable ECMP and so the use of all multiple equal cost paths co-existing for a given destination, let us now introduce new notations accounting for it. From the point of view of a given source router s, ECMP extends its SPT towards all destinations into a Directed Acyclic Graph (DAG)[10]. In particular, the set of best equal cost paths from a source s to a

[9] For the sake of simplicity, let us first ignore ECMP at this stage: we consider only the best lexicographical path (when equal cost paths exist they are arbitrarily sorted according to the identifier assigned to each node and the protection is granted by design). Some routers learn the change (typically the one detecting the failure) before others and this desynchronization may trigger inconsistent forwarding states if not handled with care. With the notations provided later, we can denote e_1 as (s, n_1^1) or simply (s, n_1) as our advanced protection scheme is only required if there is no available ECMP alternate.

[10] Indeed, the equal cost condition holds and so ensure the absence of any circuit – strictly decreasing distances along paths which thus are elementary.



(a) A weighted graph (b) Loopfree next hops

Figure II.1: Alternate next-hops vs. loopfree validations rules. On the right digraph, we illustrate the best path made of a single next-hop (red), and alternate ones made of two hops starting with the DC (green) and local LFA (teal blue) hops, provided the symmetric weighted digraph on the left. While $p_1^1(s,d) = e_1 = (s, n_1) = (s, d)$ with $c_1(s,d) = 2$ (since $|\mathcal{D}_1| = 1$ the index j is useless here), we have $p_2^1(s,d) = (s,a),(a,d)$ and $p_2^2(s,d) = (s,b),(b,d)$ with $c_2 = 3$ (the lexicographical order indeed allows to tie break the two paths in $\mathcal{D}_2$). The next-hop $n_2^1 = a$ (green) is actually validated for both DC and LFA (respectively with conditions Eq. II.1 and Eq. II.2) because $c_1(a,d) = 1 < c_1(s,d) = 2$ while the next-hop $n_2^2 = b$ (teal blue) is only validated for LFA (the DC condition is not verified; indeed b can also use s otherwise and so create a pesristent forwarding loops).

destination d forms a sub-DAG, denoted $\mathcal{D}_1(s,d)$ in the following. Considering hop by hop forwarding, such a DAG locally and practically results in $\mathcal{F}_1(s,d)$, the set of best first hops used at s towards d with ECMP. All these paths, along with their first hops, share the same best (unique and equal) cost simply denoted $c_1(s,d)$. Let us now extend these notations to $\mathcal{D}_i(s,d)$, or simply $\mathcal{D}_i$, $\mathcal{F}_i$ and c_i which respectively denotes the set of best paths, first hops and cost considered when all links in $\bigcup_{j=1}^{i-1} \mathcal{F}_j$ fail. We order next-hops and paths in each subset arbitrarily, e.g. with a lexicographical tie-break (since their respective costs are the same, we look at them as equivalent in each subset). Note that we have by construction a strict inequality among c_i, i.e., $c_1 < c_2 < \cdots < c_{i-1} < c_i < \cdots$ because we consider the failure of all next-hops in the previous set. Such alternative paths, for $i > 1$, and in particular with $i = 2$, are used only if better ones fail while all paths in a given $\mathcal{D}_i$ (and respectively their next-hops in $\mathcal{F}_i$, and in particular for $i = 1$ respectively) can be used simultaneously to perform load balancing (not only re-rerouting). I envision to investigate cases where $i > 2$ in my perspectives but let me now focus on $i \leq 2$.

Focusing on single link or node protection (i.e. one failure at a time), we are mainly and more specifically interested in $(\mathcal{F}_2, c_2)$ and paths in $\mathcal{D}_2$ when $|\mathcal{F}_1| = 1$, that is no local ECMP and multiple next-hops. Indeed, if $|\mathcal{F}_1| > 1$, the single link protection is natively ensured with ECMP for couple (s,d): when a local link fails in the set of next-hops $\mathcal{F}_1$, let say n_1^1, its ECMP counterparts $n_1^j \mid j \neq 1$ are enough to provide optimal backup paths (i.e. post-convergence ones as we will detail later). More interestingly, next-hops in $\mathcal{F}_i$ with $i > 2$ are *backup of the backups*, that is they are not necessary when only considering single link failure[11] or when looking at them one after the other. In other words, when the unique next-hop in $\mathcal{F}_1$ fails, we can simply update the sets and their indexes: $\mathcal{F}_1 \leftarrow \mathcal{F}_2$, $\mathcal{F}_2 \leftarrow \mathcal{F}_3$ if $|\mathcal{F}_1| = |\mathcal{F}_2| = 1$ and so on if necessary. One failure after the other, we do not anticipate more than one at a time. That is why we will mainly focus on $\mathcal{D}_2$ and related notations in the following without more objective than getting all next-hops in $\mathcal{F}_2$ (not *only one* of them as we aim to retrieve *all* post-convergence paths after any single link/node failure).

On the contrary, if $|\mathcal{F}_1| > 1$, ECMP applies and there is no need for more protection (either local or remote); more specifically if $|\mathcal{F}_i| > 1$, we will say that ECMP applies locally for the $\mathcal{D}_i$ and we will denote p_i^j a given path in such a DAG, arbitrary sorted at the j^{th} rank, and the same for its next-hop $n_i^j \in \mathcal{F}_i$. Computing all equal cost paths requires to slightly extend the basic *Dijkstra* algorithm: it is enough to store all parents whose costs are equal to the best one (and not only the first visited parent using a strict condition). The goal is not to extend the multiple paths during the path exploration, but to retrieve a posteriori the DAG following such multiple parents in a backward fashion. Using this simple method, one can determine the subset of destinations $V_D^s \subset V$ for which there is a need of additional protection:

[11] Note that while our technique can be easily extended to deal with single node failure, we do not consider the case of multiple independent concurrent failures.

$d \in V_D^s$ if and only if $|\mathcal{F}_1(s,d)| = 1$.

Fig. II.1 illustrates the two basic loopfree rules (DC and LFA) and previous notations. As there is no ECMP for the couple (s,d), additional post-convergence next-hops are required: while the blue one complies with both DC and LFA rules, the red one only satisfies the LFA condition. In the following, we will only focus on the re-routing objective for the sake of simplicity.

II.1.a.2 Efficiently Compute, Validate and Deploy Alternative Paths

With these extended notations, the problem solved in this first section is now easy to state, at least at first glance. Simply put, it consists in computing the two best distinct first next-hops towards all destinations from a single source with the most efficient possible algorithm (not linear in $deg(s)$).

Problem 1. *The Best First Hop Disjoint path problem (BFHD)*
Given a graph $G = (V, E, w)$ and a source $s \in V$, compute for each destination $d \in V$ the two best first hop disjoint paths (either link or node distinct). That is for any best next-hop $f(s,d)$ (being the first hop of the shortest path towards d), compute its best alternates $f_2(s,d)$ if necessary (taking ECMP into account, they are the best next-hop in $E \setminus (s, f(s,d))$ for the link protection).

More precisely, given a source node $s \in V$, the first challenge we aim to solve is:
How to efficiently compute $\forall d \in V$, not only the best equal cost paths $\mathcal{D}_1(s,d)$ towards each destination d, but also, $\forall d \in V_D^s$ all the best equal ones not using $n_1^1(s,d)$, the unique first hop in $\mathcal{F}_1$? That is computing $\mathcal{D}_2(s,d)$, c_2 and so $\mathcal{F}_2$ for all $d \in V_D^s$.
With our notations accounting for multiple equal cost paths and this new problem formulation, the practical challenge we tackle becomes easier to state, at least at first glance. Not only we look for a single (local, remote or directed) LFA alternative for each destination, but we are interested in computing *all post-convergence paths* while ensuring loopfreeness after and during any (local and remote) network changes.

In particular, when symmetrical weights apply, we will show[12] that deploying such multiple post-convergence LFA does not lead to a significant overhead as only one encapsulation at most is enough to cover all cases. The only drawback is that we may solicit a remote LFA instead of a local one (when it exists) to deploy post-convergence path as detailed later: it induces a bit more overhead but, on the other hand, does not introduce transient forwarding changes. Simply put, our first objective consists in computing the sets of pre- and post-convergence first next-hops towards all destinations from a single source s with the most efficient possible algorithm (not depending on $deg(s)$). In any cases, with or without ECMP, at least two next-hops are necessary, either they both belong to $\mathcal{F}_1$ (and no more computation is required for the destination) or we have $|\mathcal{F}_1| = 1$ and $|\mathcal{F}_2| \geq 1$; since ECMP does not apply, more alternate paths are necessary, i.e. $V_D^s \neq \emptyset$, and **we aim to compute loopfree candidate next-hops enabling to deploy all post-convergence paths and so ensure a full ECMP routing transition handling local or even remote single failure**. Formally, we define an *alternate path* as a path not using the failed link. Their first hops lie in the set $\mathcal{F}_2$ if $\mathcal{F}_1$ consists in only one next-hop. We will first demonstrate that only one routing detour is enough to protect any destination d with (all ECMP) next-hop(s) in $|\mathcal{F}_2|$.

In the following, we will show that using only few SPC, that is usually 2 – or in any cases less than a constant factor of 4 (when the ISP cannot rely on SR), instead of at worst $|V|$, we can achieve the exact same result. By focusing its computation efforts on the two best first hop distinct paths, for all destinations $d \in V$, *TBFH* ensures both computing efficiently and better transient re-rerouting options. Computations are performed *locally* on s: instead of computing as many SPT as its amount of neighbors (one rooted at each), s just has to extend its initial SPC by computing not only one but several best paths, at least two, whose the second best. Since computations are local to s – both stages of *TBFH* being launch from this unique source root, distances of the form $c(v,d)$ are not directly available but can be retrieved from best alternate paths[13] and loopfree rule formulations slightly adapted.

[12] This result is known and used in practice [209], but we investigate a new construction proof based on transverse edges and specific to post-convergence paths and that also has the merit to be both simple to understand and efficient to compute.

[13] Distances like $c_1(v,s)$, necessary for LFA when weights are not symmetric, can be obtained easily by computing a reverse SPT rooted towards s (just reverting the weights of each edge and applying an usual SPC). Assuming symmetric weights can be convenient and seems reasonable for most ISP as it is a realistic assumption. However, for the sake of generality, we will discuss this assumption and simply pinpoint cases where it would ease the computation.

As a recall, focusing on the first hop is because link state routing schemes in IGP works this way[14], intra-domain protocols, like IS-IS and OSPF, indeed rely on the *hop by hop* forwarding paradigm. Looking at this specific first hop problem allows to locally and immediately provide an alternate path without waiting that the whole IGP converges globally, for example after any link failure (local to s, the neighbor-wide failure being discussed later). The goal is to offer a local fast-rerouting transient exit before all routers actually converge to their new (global) optimal paths, using an incremental SPC algorithm, or at worst a full SPC. However, the alternative route should be **loop-free**[15] and using immediately and blindly f_2 as the best local second option can trigger transient loops[16]. Although p_2, as the optimal alternate path, looks to be the best candidate to speed up the convergence, it may not locally satisfy the loopfree requirement while there exists other hop(s) doing so for a given condition.

Loopfree rules designed exclusively for fast-rerouting, in order to react to specific failure, rely on less restrictive conditions and so provides a better coverage. With technology like SR and Topology Independent Loop Free Alternate (TI-LFA), it becomes easier to achieve full protection thanks to the use of loose source routing. In its simplest form, LFA, by forcing just one local hop, cannot guarantee more than a pretty good coverage with a favorable design. In both cases, the alternate next-hops are enabled only when the failure is detected avoiding so the presence of loops as long as only one (link or node) failure occurs at a time. The algorithms introduced in the next section, *TBFH* and its variants, can provide any kind of valid loopfree next-hops (at least one if there exists such), and the ones we call post-convergence next-hops in particular. However its **complexity is not dependent from the source node degree**: this is a remarkable property considering existing proposals that usually require $deg(s)$ SPC computations. Within SR domains, *TBFH* not only provides transient backup routes for a limited overhead, but the best of them, that is leading to the second best next-hop, the local optimal one after the failure. This post-convergence property mitigates intermediary changes[17], at least locally, and offer the best achievable transient re-rerouting detours before the IGP re-converges widely.

II.1.b An Efficient Multi-Modal Solution for Load Balancing & Fast-Rerouting

II.1.b.1 Transverse Edges to Decompose the SPT

Let us now briefly expose why and how such *alternate paths* can be computed efficiently, that is with a single additional SPC, just one run accounting for all destinations in V_D^s. Note that by alternate paths (or next-hops respectively) we formally mean that they do not use (or are equal to resp.) one of the link (s, n_1^j) (one of next-hop n_1^j). They are only local backups. To simplify the following, and without loss of generality, we can consider specifically n_1^1 as the ranking among next-hops in a given subset is by definition arbitrary (as they share the same cost, the tie-break is a lexicographical order in practice). If $|\mathcal{F}_1| > 1$, next-hops n_1^j are alternate ones regarding the others (ECMP protects them all from a single failure). There is no need to run a second SPC stage for these destinations.

Thus, we are interested in the destinations for which ECMP is not sufficient as such equal cost alternate paths do not exist (the subset V_D^s). We will show that non equal cost alternate paths share similar structural properties with equal ones although it requires one more SPC to compute next-hops for those destinations (extending *Dijkstra* with an equality condition being not enough). Indeed, given a root node s, the set of edges[19] of a graph can be partitioned into four subsets according to the SPT[20] rooted at s:

- Edges in $N_1(s)$ connecting s to first hops of primary best paths (the failed links to be considered in the branches);

[14] as well as with a distance vector model.

[15] Routes should map to elementary forwarding paths, being free of any circuit at any time.

[16] Note that we tackle this *always consistent forwarding* problem in the second section of this chapter.

[17] that can hamper TCP flows as with per packet multipath round robin. ECMP generally uses per flow load balancing to ensure a *congruent* flow/path assignment. This aspect is developed in the next chapter when comparing detours to multipath routing (and load balancing with ECMP in particular).

[18] Here we do not consider explicitly the multigraph case (with parallel links), but it can be handled with few extensions.

[19] We consider both directions of each edge.

[20] Note that this is true for any arbitrary SPT in the DAG $\bigcup_{\forall d \in V} \mathcal{D}_1(s,d)$.

Table II.2: Alternate Path Terminology, The Partition of Edges and the case of Remote Failures

Terms/Notations	Definitions
$\mathcal{F}_i, c_i, \mathcal{D}_i$	i^{th} sorted subsets of next-hops $\mathcal{F}_i(s,d)$, their related respective cost $c_i(s,d)$ and paths $p_i^j(s,d) \in \mathcal{D}_i(s,d)$ from s to d. In practice, we consider $i \leq 2$, and the order between $i=1$ and $i=2$ discriminates best paths from alternate ones used iff all best ones are unavailable (one or several local failures forbid their use)
$n_i^j(s)$	A next-hop $\in \mathcal{F}_i$ arbitrarily sorted at the j^{th} rank (e.g. with a lexicographical tie-break). Note that we have $\lvert\mathcal{D}_i\rvert \geq \lvert\mathcal{F}_i\rvert$: several paths sharing the same next-hop can exist (the index j is specific to each set)
$N_1(s)$	A set of primary next-hops $\{h \in succ(s) \mid h = n_1^1(s,h)\}$ regarding a given (arbitrary) SPT rooted at s
V_D^s	Set of destinations d for which $\lvert\mathcal{F}_1\rvert = 1$ (that is ECMP does not protect them)
$branch_h(s)$	Subtree of a SPT of s rooted at a neighbor $h \in N_1(s)$; it includes all nodes $d \in V$ such that $n_1^1(s,d) = h$ and all edges $e \in E$ such that $e \in p_1^1(s,d)$ with $d \in branch_h(s)$
transverse edge	An edge $e=(u,v)$ is transverse if it connects two nodes belonging to distinct branches, $u \in branch_h(s)$ and $v \in branch_{h'}(s)$ $(h \neq h')$, or if it connects the source root $s=u$ and a node $v \neq h$[18] in a $branch_h(s)$
$trans(G,s)$	Set of all transverse edges considering a root node s and its resulting lexicographical arbitrary SPT on a graph $G(V,E,w)$
internal edge	An edge $e=(u,v) \notin branch_h(s), \forall h \in N_1(s)$ is internal regarding h if it connects two nodes u and $v \in branch_h(s)$
simple alternate paths $Pt(s,d)$	A path $p=(s,...,u,d)$ such that $(s,...,u) = p_1^1(s,u)$ and (u,d) is a transverse edge. Their union forms a set denoted $Pt(s,d)$.
1-alternate path	A path $(s,...,u,v,...,d)$ is 1-alternate if it contains exactly one transverse edge (u,v) and if $(s,...,u,v) \in Pt(s,v)$ while d and u do not belong to the same branch
$p_2^j(n,d,l), n_2^j(n,d,l)$ and $c_2(n,d,l)$	Post-convergence paths $\in \mathcal{D}_2(n,d,l)$, their related next-hops $\in \mathcal{F}_2(n,d,l)$ and their common cost considered from a node n towards d when an arbitrary (and so possibly remote) link $l \in E$ fails
$\mathcal{D}_2^*(l), \mathcal{F}_2^*(l)$	Post-convergence paths $\subseteq \mathcal{D}_2(n,d,l)$, their next-hops $\subseteq \mathcal{F}_2(n,d,l)$ that are respectively not in $\mathcal{D}_1(n,d)$ and $\mathcal{F}_1(n,d)$

- Edges belonging to subtrees of the SPT forming *branches*;
- *Transverse edges* connecting two distinct branches or connecting the root s and a branch without being the first hop of a primary path;
- *Internal edges* linking nodes of the same branch without belonging to this branch.

The union of these four subsets entirely captures E: as we consider all nodes as destinations, they all belong to a given branch. A direct consequence of this SPT decomposition is the fact that an alternate path contains at least one transverse edge (we recall that an alternate path is a valid backup and should avoid the failed next-hop in $N_1(s)$); since such a path needs to switch from the primary branch to another branch (to avoid the failed link), by construction, it has to use a transverse edge to re-enter the initial branch. Whatever the ranking used among next-hops used in $\mathcal{F}_1$, it remains true for any lexicographical tie-break (ordering among n_1^j) and turn the original DAG into a strict SPT.

Fig. II.2 illustrates this edge partition in a simple graph. Note that we consider an easy case where the weights of each link is fixed to 1 such that the metric that we use is the path length in hop number (uniform valuation). The SPT rooted at s includes three branches illustrated by three different colors (black, grey and white nodes). In this topology, there are two *transverse* edges (dashed arcs) and two *internal* edges (dotted arcs). The plain edges form the SPT that is arbitrarily picked in $\bigcup_{\forall d \in V} \mathcal{D}_1(s,d)$ because of ECMP. Indeed, with an uniform valuation, we have on this example $V_D^S = \{i,j,f,g,n,m,o,h,e,a\} \subset V \setminus s$ such that remaining nodes $\{k,l,b,c,d\} = V \setminus V_D^S$ are destinations protected thanks to ECMP. Thus they can be attached to other branch arbitrarily, e.g., $k \in branch_n(s)$ instead of $branch_i(s)$. That is we consider $n_1^1(s,k) = i$ on this STP decomposition (and $n_1^2(s,k) = n$). We do not consider these destinations to be protected and it is obvious that such equal cost paths are 1-alternate ones because of the partition.

The edges i_1 and i_2 are called internal because they connect nodes belonging to the same branch, whereas edges t_1, t_2 and t_3 are transverse edges because they connect nodes belonging to different branches. A path must contain nodes of at least two different branches to be an alternate path ex-

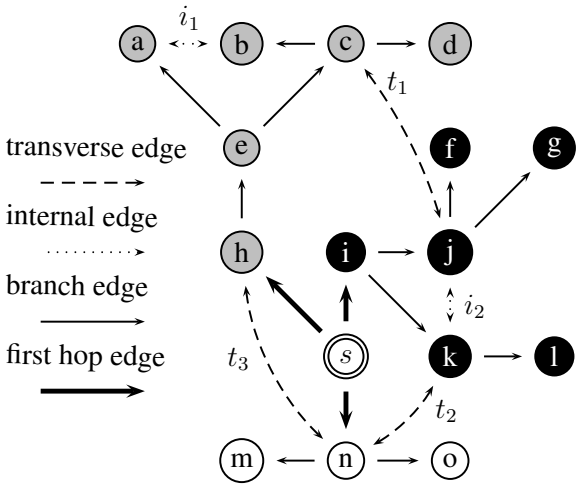


Figure II.2: *Transverse* vs. *internal* edges. A partition of edges considering an uniform valuation and s as the root of the SPT.

cept if the first hop n of the alternate path corresponds to a transverse edge (we can consider s as belonging to a branch reduced to only it). In any case, this implies that an alternate path contains at least one transverse edge. More generally, in the hop by hop forwarding context, an alternate path is necessarily a k-alternate path for some $k \geq 1$ to be first-hop distinct from the primary one.

Fig. II.2 illustrates the 1-alternate paths terminology. Edges (s, h), (s, i) and (s, n) are the three first hops (thick arcs) linking s to the three branches. Path (s, h, n) is a *simple alternate* path and (s, i, j, c, e) is a 1-alternate path.

In practice, it means that router s uses three primary next-hops n_1^1 for its forwarding plane: $N_1(s) = \{h, i, n\}$. If the destination belongs to:

- $\{a, b, c, d, e, h\}$, s uses h as primary next-hop (and we have i acting as the secondary ECMP backup, $n_1^2(s, \cdot)$, for destinations b, c, d);
- $\{i, j, k, l, f, g\}$, s uses i as primary next-hop (and we have $n = n_1^2(s, k) = n_1^2(s, l)$);
- $\{m, n, o\}$, s uses n as primary next-hop (and we do not have any ECMP backup for destinations in $branch_n(s)$).

As a last example, having a post convergence next-hop $h = n_2^1(s, n)$ (here the first hop of a simple alternate path with $|\mathcal{F}_2(s, n)| = 1$ in particular) verifying rule[21] (Eq. II.2) means that s can use h as a LFA next-hop towards destination n. By generalizing to all nodes in $branch_n(s)$, s can use h as a LFA next-hop for any destination in $\{m, n, o\}$. Indeed, there exists a 1-alternate path verifying rule (Eq. II.2) towards each of these destinations.

II.1.b.2 Optimal Alternate Paths

Let us now describe the properties of alternate paths that contain only one transverse edge: 1-alternate paths. Let $\circ$ be the operator representing path concatenation and ϵ be the notation for an empty path. Several alternate paths using the same first hop towards a given destination may exist. Considering a destination d, we focus on the shortest path among the set of 1-alternate paths using the same first hop towards d. To refer to those paths we use the term *optimal 1-alternate path*. An optimal 1-alternate path, denoted $p^*(s, d)$, can be decomposed as follows[22]:

$$p^*(s, d) = p' \circ (u, v) \circ p''$$

[21] This rule is rewritten later as rule ()Eq. II.4).

[22] We rely on DAG to both denote the set of its links and all possible sequences of them that lead to a path from the source to the destination. We can then denote $p \in \mathcal{D}_1$ to refer to a given path with edges in $\mathcal{D}_1$.

with $p' \in \{\mathcal{D}_1(s,u) \cup \epsilon\}$, edge $(u,v) \in trans(G,s)$, and $p'' \in \mathcal{D}_1(v,d)$ not containing any transverse edge regarding the STP decomposition chosen on s. Note that a simple alternate path $p' \circ (u,v) \in Pt(s,v)$ is by construction entirely disjoint from the (arbitrary) best primary path $p_1^1(s,v)$.

We have the following property:

Property 1. *If there exists an alternate path p, of cost c, from s to d and not using n_1^1 (for each (s,d)), then there exists at least one path p^*, of cost $c^* \leq c$, containing only one transverse edge. Such a path is an optimal 1-alternate path and can be denoted either:*

- $p^* = p_1^j(s,d)$ *with* $j > 1$ *if* $|\mathcal{F}_1| > 1$ *($d \notin V_D^s$);*
- *or* $p^* = p_2^j(s,d)$ *with* $j \geq 1$ *and a cost equal to* c_2 *($d \in V_D^s$).*

Interested readers can find the proof in [MFB+11]. The intuition is that since a transverse edge is required by design (to avoid the failed link in the primary branch), there always exists at least one shortest alternate path made of a single transverse edge. In the following, since we focus on cases where ECMP is not enough to protect all destinations, I simplify notations for $p_1(.,.)$ and $n_1(.,.)$ respectively: the rank of the path (or next-hop) is omitted as superfluous for the reasoning. Fig. II.2 illustrates Property 1: for example, the path (s,n,h,e,c,j) goes through $branch_n(s)$ and $branch_h(s)$ to finally reach the transverse edge t_1 and $branch_i(s)$. This path contains two transverse edges (t_3 and t_1) whereas there exists a simple alternate path $\in Pt(s,j)$ that can be decomposed as $p_1(s,c) \circ (c,j)$ whose cost is here strictly shorter than the previous one (both paths are common from node h). However, the optimal 1-alternate path, a post-convergence one, is given by $p^* = p_1(s,n) \circ (n,k) \circ p_1(k,j)$. Note that while $p_1(s,n)$ consists of the unique link $(s,n) \in branch_n(s)$, $(n,k) = t_2$ and $p_1(k,j) = i_2$ respectively consist of a transverse and internal edges.

II.1.b.3 Our Algorithm to Compute the Two Best First Hops

A direct consequence of Property 1 is that if there exists an alternate path p from s to d (regarding the primary path p_1^1 and its first hop n_1^1 in particular), then at least one of the optimal alternate paths towards d is a 1-alternate path. For a given destination, it is not possible for a k-alternate path (having k transverse edges with $k > 1$) to have a strictly lower cost than an optimal 1-alternate path. Indeed, by definition, the best simple alternate path reaching a given branch at the same node has necessarily a cost lower or equal to any other alternate path reaching this branch (Property 1).

This opportunity motivates the design of our algorithm, *TBFH*, not running more than two SPC [MFB+11]: the goal is to take advantage of Property 1 by focusing on the search for optimal 1-alternate paths and their equal cost variants as detailed later. This algorithm, *TBFH* (that stands for Two Best First Hop), is able to retrieve both $\mathcal{F}_1$ and $\mathcal{F}_2$ in only two Dijkstra like SPC runs at worst, even if their respective cardinal is greater than 1.

From now on, and for the sake of simplicity, we will only consider destinations in V_D^s. Indeed, although other destinations are also protected with optimal 1-alternate paths, they simply come for free as ECMP applies and there is no need to compute additional alternate paths (a single Dijkstra extended run is enough). That is we can now focus on retrieving the entire set $\mathcal{F}_2$ and all its next-hops n_2^j, not only one of them. This ensures both optimality and the opportunity to load balance the traffic on the whole post-convergence paths range. Note that $|\mathcal{D}_2| \geq |\mathcal{F}_2|$ as ECMP can also apply remotely from s: while *TBFH* is able to retrieve the entire DAGs (both $\mathcal{D}_1$ and $\mathcal{D}_2$), it does not need to return more than the reduced set of local next-hops ($\mathcal{F}_i$ with $i = 1, 2$). Hop-by-hop forwarding implies that each router opts for its own route locally, providing a simple and resilient distributed routing scheme but requiring consistent actions (e.g. complying with ECMP, DC or LFA rules)[23]. Let then extend property 1 to enlarge the scope of *TBFH* and show that since it computes the whole $\mathcal{D}_2$ DAG, it can extract all its related (and local) next-hops.

To deploy loop-free backup routes, one needs to compute the primary next-hops (ECMP) and another set of candidate next-hops towards each destination if necessary. A candidate next-hop is the first hop

[23] It is also worth to notice that the load balancing is performed locally: it does not account from the remote diversity of paths. That is the load is locally uniformly distributed with respect to $|\mathcal{F}_i|$ and not to $|\mathcal{D}_i|$ (with specific path-congruence mechanisms [Otto, 2016, Del Fiore, 2021] for preserving TCP flows in case of load balancing). Even if one hop later, the number of paths drastically differs, the load balancing is blind to this regard.

of a computed alternate path. Each candidate next-hop is associated to the cost of its alternate path for further validation ensuring loop-free forwarding. We will now show that *TBFH* requires only one additional SPC while it results in all local post-convergence next-hops as candidates and can validate all of them.

TBFH: Mode of Operations For this purpose, let us briefly describe how *TBFH* behaves. After a first Dijkstra run extended to account for ECMP, *TBFH* requires the set V_D^s as input for its second stage. This stage also consists of a Dijkstra run extended to account for ECMP but on a modified graph and only targeting destinations in $V_D^s \neq \emptyset$. This modified graph is a transformation of the original one: the primary next-hops $n_1^1, \forall d$, as well as transverse edges in $trans(G, s)$ are removed. However, the latter are replaced with new virtual edges modeling simple transverse paths. That is, for all transverse edges $(u, v) \in trans(G, s)$, *TBFH* adds one virtual directed edge directly connecting s to v such that its weight is the one of the simple transverse path in $Pt(s, v)$ it virtually models (to ensure the next corollary it also stores all next-hops computed towards v in the first stage). The result is a graph G' where s is an isthme whose deletion partition G' in as many connected components as branches found in G. On this basis, *TBFH* runs its second stage with G' as input and the same ECMP extension as in its first stage; internal edges are visited inside each component but there is no more paths existing among node in distinct components. Note that it results in practice in $|N_1(s)|$ independent sub-Dijkstra runs for the same reason. Our algorithm uses two distinct phases of computation whose details and complexities are given in [MFB+11]. The first stage performs the computation of the set of primary next-hops and partitions the graph into several non connected components. These connected components are given as an input for the second stage which performs a SPC on each of them and then returns a set of candidate couples (alternate next-hop, alternate cost). The separation between those two phases is useful to uncouple the primary next-hops computation from the computation of candidate alternate next-hops. Hence, it is possible to forward packets on primary next-hops while the computation and validation of alternate next-hops is performed as an independent task. Finally, as soon as all equal cost paths towards destinations in V_D^s are computed *TBFH* terminates.

Theorem 1. *TBFH solves the BFHD problem*
TBFH allows, in only two SPC, any node $s \in V$ to compute an optimal 1-alternate path in $\mathcal{D}_2(s, d)$, towards all destinations $d \in V_D^s$.

Interested readers can find the proof, the detailed algorithm and complexity in [MFB+11]. *TBFH* having a cost of 2 SPC, its complexity can be summarized as $O(nlogn(n))$ stressing again its independence regarding the node degree. The following corollary elaborates on the capacity of *TBFH*: while it computes $\mathcal{D}_1$ in its first stage, all paths in $\mathcal{D}_2$ can be retrieved in its second stage, even if this set may include k-alternate paths with $k > 1$.

Corollary 1. *For all destinations $d \in V_D^s$, TBFH is able to compute any post-convergence next-hop n_2^j (having so a cost of c_2) even if its related path(s) include(s) more than one transverse edge.*

That is, if there exists a post-convergence path $p_2^j(s, d)$ involving strictly more than one transverse edge, i.e. p_2^j is a k-alternate path with $k > 1$, we only have to prove that *TBFH* also computes it in the same fashion as all (1-alternate) paths in $\mathcal{D}_2$. This set of paths includes the paths having more than one transverse edges only if they share the same cost as the 1-alternate paths having only a single transverse edge.

Proof. Thanks to property 1, we know that there exists at least one 1-alternate path in $\mathcal{D}_2$, denoted p^*, with a cost of c_2. *TBFH* computes such paths by design as they are extension of simple alternate paths (see [MFB+11] for the more formal proof). Let us first decompose p^* as $p' \circ (u, v) \circ p''$ and look at $p = p^1 \circ (u', v') \circ p^2 \circ (u, v) \circ p''$. The decomposition of p^* is general enough for 2−alternate paths as we simply consider (u, v) as the second transverse edge in use. Reaching this last transverse edge (through the branch containing its invertex) already induces an extra cost as we consider destinations in V_D^s, we will see that the first transverse edge (u', v') in p cannot also induce an additional cost. Overall the path p is a 2-alternate path using a common transverse edge with p^*, with a first transverse edge (u', v') (with p^1 and p^2 being best sub-paths), either the cost c of the subpath $p^1 \circ (u', v') \circ p^2$ is strictly greater than

the one of p' or equal (by definition of transverse edges). If it is strictly greater, it contradicts the initial condition about p being a post-convergence paths: $p \circ (u, v) \circ p''$ cannot exhibit the cost c_2 in case of local failure (the same cost as p^*); thus costs of these subpaths are necessarily equals and we can deduce that we have $c = c_1(s, u)$, i.e. reaching u either indirectly via u' (with or without additional internal and/or branch edges in p^2), or directly via the arbitrary branch it belongs to, exhibit the same cost (and can be retrieved backward after the SPC of the first stage of TBFH).
Paths having more than $k = 3$ transverse edges lead to the same result in general: only the latest transverse edge has to considered as previous $k - 1$ alternate subpaths share equal costs. Indeed, since the latest transverse already implies an additional cost, other previous transverse edges cannot add any extra cost. For a 3−alternance path, it means that its sub-path to its first transverse edge (and resp. second) share the same cost as the one of its common 2−alternate path (and resp. 1−alternate path) and, as such, are given as inputs of the second stage of TBFH. We can then simply reason by induction to conclude for any k. □

Thus, extending both stages of *TBFH* to retrieve all equal cost paths (here towards v in the first stage in particular) is enough to capture the post-convergence ones in $\mathcal{D}_2$ including more than one transverse edge. In particular, if they exist, all equal cost paths towards destination v, the outvertex of a transverse edge leading to simple alternate path $Pt(s, v)$, computed in the first stage should be taken into account by storing their distinct parents (as well as indirectly v', u' and finally their related best first hops) not only the ones of their branch. That is retrieving all post-convergence paths can be achieved in the same ECMP fashion as extending Dijkstra with multiple parent inheritance in case of equal costs.

Finally, observe again that according to the arbitrary SPT decomposition, an edge can be either denoted as transverse or belongs to a branch when ECMP exists: the 1-alternate subpath $p^1 \circ (u', v') \circ p^2$ leading to a 2-alternate path p we have described is also a primary best path towards u with another tie-break and reciprocally with p'. The same reasoning can be extended with k-alternate paths having more than two transverse edge (more than the two described in the path p of the proof).

On Fig.II.2, let us consider the failure of link (s, n): if we add a link between s and k of weight 2 as a local transverse edge in this SPT decomposition, and set $w(t_3) = 2$, we can observe that the 2-alternate path $(s, k), (k, n)$ has the same cost as $(s, h), (h, n)$ and $(s, i), (i, k), (k, n)$. *TBFH* can compute all these post-convergence paths, but the question is now: how to deploy them without introducing forwarding loops? this does not come for free in any cases (like with ECMP, DC or even LFA for fast-rerouting purpose only).

In the next two sections, we will describe how to actually use and modify *TBFH* to validate and deploy post-convergence paths or at least increase the coverage of the loopfree rules specified so far relying only on basic IP means.

II.1.b.4 Refined Conditions at Play

In practice, loopfreeness can be checked and modeled by several conditions more or less refined and difficult to evaluate (and deploy). With our new specific notations and for source-destination couples (s, d), and for any $n_2^j \in \mathcal{F}_2$ (such rules are always valid with ECMP for next-hops in $\mathcal{F}_1$), the two conditions already stated before in rules Eq. II.1 and Eq. II.2 can now be reformulated as:

$$c_2 - w(s, n_2^j) < c_1 \quad \text{(Eq. II.3)}$$

$$c_2 - w(s, n_2^j) < c_1 + c_1(n_2^j, s) \quad \text{(Eq. II.4)}$$

Note that the (upper bound) approximation we have on $c_1(n_2^j, d)$ (that is $c_1(n_2^j, d) \leq c_2(s, d) - w(s, n_2^j)$) is enough because, if the equality does not hold, it implies that $s \in p_1(n_2^j, d)$ and, as such, the rules cannot be satisfied anyway. However, there is no guarantee that the valid DC or LFA next-hop lies necessarily in $\mathcal{F}_2$: that is, all n_2^j may not verify the considered rule even if there exist another (not as well ranked) neighbor in $\mathcal{F}_i$ with $i > 2$ that does. In the next section, we will show how to solve this challenge and tune our algorithm to offer such guarantees in any cases either with the use of SR (and also reach a complete protection in any situation) or relying only on the traditional IP hop by hop paradigm (but with a limited protection if weights are not symmetric).

While conditions at play should be *sufficient* to avoid loops, they are generally not designed to be *necessary* as the complexity of a given condition, as well as its actual deployment requirements, may not scale. The simplest and most popular example is certainly **ECMP**; it allows both for load balancing and fast-rerouting as ECMP next-hops are not only loopfree when used in case of a (single) failure, but permanently loopfree as they can be used simultaneously without considering any specific topological change to be enabled. The Downstream Criteria (DC) condition or more sophisticated conditions, like the ones mentioned in [357], also provide the same LB opportunity as ECMP but still with a limited coverage (although greater or equal than ECMP). As such conditions are quite restrictive for arbitrary networks, not specifically designed and tuned to favor their usages, they are generally not enough to ensure full protection for all bi-connected graphs (having non symmetric weights in particular). LFA and its variants up to TI-LFA progressively relax the assumption (because next-hops are only used in case of single failure and can be picked remotely and successively) using an arbitrary number of encapsulation.

(a) A graph with asymmetric weights

(b) Pre-/post-convergence edges and paths ($p_1^j(\cdot, d)$ vs. $p_2^1(\cdot, d)$)

Figure II.3: One asymmetric weight is enough to prevent full protection with basic fast-reroute techniques.

II.1.b.5 Validating & Deploying Backup Paths Relying on Usual IP Forwarding

With usual IP Forwarding, simply based on the destination, there is no direct way to address all (kind of) routing changes with simple technical means (without relying on SR as in the next section or using other more complex proposals requiring more in depth protocol changes [253, 201, 200]).

Limits of Standard Solutions With asymmetric weights, there exists graphs which require more than one level of encapsulation to protect the impacted traffic (without introducing forwarding loops). Fig. II.3 illustrates how and why problematic cases occur.

Neither LFA, U-TURN nor R-LFA ([278]) are enough to face such a situation and provide full protection coverage. Indeed, considering node a as the source and d as the destination, there is no alternate next-hop to protect link (a, d). Neither node b satisfies the LFA condition to protect d, nor nodes b and c can offer a U-TURN (node c is not a LFA for b for the same destination). Finally, there is no R-LFA, as the P and Q spaces[24] do neither intersect nor contain adjacent nodes (here Q is empty while P only includes b): due to the asymmetric weights, c uses b to forward traffic to d considering its pre-convergence path (in red) while a cannot reach c with its P space[25].

This can occur with or without ECMP, here with as c also use link (c, d) to reach d. One can argue that forcing c to make the good choice, i.e. deactivating next-hop b but not d, can arrange the situation (the same situation can occur on P instead of Q and this looks even easier to fix locally). However just setting $w((c, d))$ to 5 instead of 4 worsten the situation: c just now strictly prefers b and one cannot adapt

[24] For a destination d and a source s, the P-space is the set of destination nodes reachable from s using its set of best paths not including link (s, n_1) in the initial pre-convergence DAG (i.e. the merged DAG resulting from the union of destination oriented DAG $\mathcal{D}_1(s, d), \forall d \in V$ is pruned from destinations whose at least one path includes (s, n_1)), while the Q-space is the set of source nodes that can reach d without using (s, n_1) (with the same conditions, all best reversed paths from the sources should avoid the failed link towards d). Basically, it then consists of two pruned DAGs, one rooted at source s and the other, a reverse one, rooted at destination d. If they do not have a node in common or at least two adjacent nodes, R-LFA is not deployable.

[25] With symmetric weights, let now say 4 in both directions, the situation is not the same as c is then in the P space of a, and the last link can be forced to avoid ECMP in one adjacency segment.

its choice. On the contrary[26], setting $w((c,d))$ to 1 enables one option for R-LFA (the generalization of the underlying valid U-TURN - but still no LFA): while the two spaces still do not intersect, they are adjacent as Q now includes c that is adjacent to b in the P space. Forcing the traffic (e.g. with one encapsulation or just pushing it here) to reach b and pushing it (again) to c guarantee a remote loopfree as c now only uses link (c,d) to reach d. That is asymmetric valuation makes the problem harder to solve in practice.

On the contrary, with the previous SPT decomposition and in the restricted case of symmetric weight, one can prove that with one encapsulation (plus at worst one more local push), full protection can be achieved – as we will indeed prove in the next section specific to the the use of SR. In general, just one local push is simply enough; indeed, with LFA one can expect already a good coverage (typically around 70% in general, with usual ISP topological patterns [290], and more if carefully designed to favor it [296]). However, in some cases, only when weights are asymmetric, multiple segments can be required.

Forcing TBFH to Visit Valid Alternate Paths Generally speaking, in order to compute a complete valid set of purely local alternate next-hops (e.g. LFA or DC) towards any destination with any link valuation (providing the maximal but not necessary full protection), it is possible to use a variant of *TBFH*: Two Valid First Hops (TVFH). According to the objective, TVFH just needs to take a modified link valuation into account for the second stage of *TBFH*. Let us denote w' the new valuation function. If the goal is to find loop-free next-hops for re-routing purposes using the LFA rule, the function w' must return the same weight for all outgoing links minus the best distance from the candidate to s:

$$w'((x,y)) = \begin{cases} \lambda - c_1(y,x) & \text{if } x = s \ (\lambda \in \mathbb{N}) \\ w((x,y)) & \text{otherwise.} \end{cases}$$

Therefore, TVFH computes an optimal LFA coverage although the time complexity does not depend on the degree of s. In this case, we need to perform two additional SPC computations compared to *TBFH* because we also need to compute the *reverse costs* $c_1(y,s),\ \forall y \in succ(s)$.

Note that with TVFH, the optimality of the computed valid alternate paths is not guaranteed. There may exist a shorter valid alternate path because weights of each outgoing link of the calculating node have been modified. For instance, with the DC rule, the w' valuation function considers the best alternate path from the neighboring node, not from the root node. To perform a good tradeoff between alternate path optimality and coverage, we may consider the union of the output of *TBFH* and TVFH. For a given pair (s,d), if there exists a valid local post convergence next-hop, s computes it using *TBFH*, otherwise, if there exists a valid next-hop towards d, s can compute it using TVFH.

II.1.c More than IP Backups Routes: One Routing Segment can be Enough

With Segment Routing, the full coverage protection problem is easy to solve even in the general case of asymmetric weights. Indeed, using several segments, full protection is provably achievable, segment per segment at worst (using link by link source routing). With our graph decomposition, it is easy to find and compute the necessary list of segments. However, some cases may require more than one or two segments and require, as well, more computing ressource. Let us start with the easy symmetric case that can be solved very efficiently with SR.

II.1.c.1 The Symmetric Case: Easy to Handle Properly

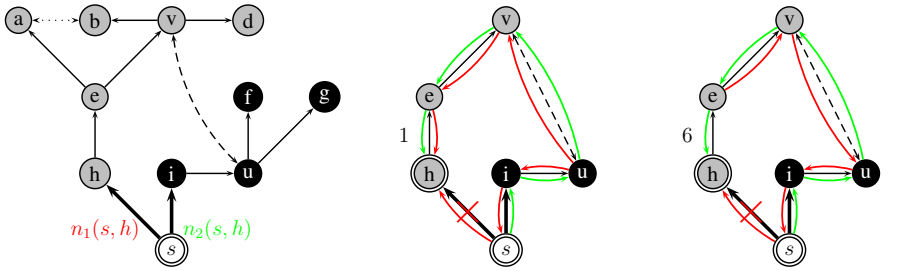
First, it is worth to notice that looking only at a destination d such that $d \in N_1(s) \cap V_D^s$ is relevant to then generalize the protection difficulty for a whole branch (and all its attached destination). Indeed, such a destination, denoted now h for convenience and considering only destinations in V_D^s (as $|\mathcal{F}_1| = 1$ we can simplify and state that $n_1 = n_1^1$), is, as we will see, a good representative of all destinations in the same branch because we have $h = n_1(s,d), \forall d \in branch_h(s)$: we will indeed show that protecting h is the most

[26] With 2 or 3, it is also the case as the asymmetry is not enough. Note that a cumulated difference of 3, between the two directions of the path, looks to be the minimal condition to create problematic cases requiring more than one encapsulation. However, depending on the implementation of ECMP, that is enabling or not partial next-hops deactivation slightly modifies such minimal conditions.

difficult task, and so the only case to be actually considered when the valuation function is symmetric. In the following, the terms *protecting* and *protection* will refer to link protection only, that is considering only the specific case of the failure of a link (s, n_1). Finally, we assume that the underlying graph G is **bi-connected**: for any failure, there exists at least one alternate path for all sources and destinations. Note that the case of the node protection (where the failure occurs router-wide on n_1 and concerns all its incoming and outgoing links) is not explicitly handled for the sake of brevity and because just requiring some simple extensions left for further works.

The two first sub-figures in Fig. II.4 illustrate notations and conditions used in the following. It is a refinement of Fig. II.2 for destination $h = n_1$ to illustrate the IGP convergence process and possible loops occurring. The same SPT decomposition applies with a uniform symmetric valuation (expect in Fig. II.4.c), but let us consider that $w(t_3)$ is high enough (such that we can remove it) to make all optimal 1-alternate paths using only $t_1 = (u, v)$ in this new representation. Red edge denote old, pre-convergence, paths in use while green ones indicate new, post-convergence paths.

This figure shows that adjacency (u, v) is by construction the good candidate for P and Q spaces when the two sets do not intersect but are adjacent. In Fig. II.4.b, the P space includes node u while the Q space includes node v thanks to symmetric weights. This encapsulation can also easily be enforced with SR by design. Note that $h = n_1$ is the most difficult destination to protect in its branch $branch_h(s)$, as for other destinations in it, one node segment towards u is enough for nodes e, a, and the candidate LFA i is enough for destinations b, v, d.



(a) Arbitrary SPT decomposition (lexicographical) considering an uniform symmetric valuation ($\forall e \in E, w(e) = 1$)

(b) Pre-/post-convergence paths towards h with symmetric weights ($p_1^1(\cdot, h)$ vs. $p_2^1(\cdot, h)$)

(c) Pre-/post-convergence paths towards h with asymmetric weights

Figure II.4: With a symmetric valuation (b), the protection of the link (s, h) is easy to ensure (one segment at most is enough) while it may require several encapsulation levels when weights are asymmetric (c). The post-convergence path (in green) from s towards h is $p_2^1(s, h) = (s, i, u, v, e, h)$ if the link (s, h) fails (here we consider the worst-case destination as $h = n_1^1(s, h)$), while the pre-convergence path (given in red) is $p_1^1(s, h) = (s, h)$. Link (u, v) is a transverse edge not necessary to force in (b), node segment u is enough, while it is in (c) with more edges because of the asymmetry.

One Segment is Enough Let us now start with the formal proof showing that **symmetric valuations enable SR to protect all links in a bi-connected network with a bounded overhead**. This complete protection can be achieved with one segment at most[27] in any symmetric cases and does not lead necessary to the same paths as with R-LFA because here we force the optimal backup post-convergence path to be used (not only it provides the local optimal path but it also avoids the use of transient intermediary paths from s[28]). The backup path is then the best available and potentially specific to each destination. Our first lemma and its corollary focus on protecting h as a destination (considering the failed link (s, h) with $h = n_1(s, h)$ as $h \in n_1(s)$ – other nodes in $succ(s)$ are not necessary to protect as destinations because they are not used to forward any transit traffic).

[27] With SRv6, adjacency segments can be globally advertised while it is generally not the case with SRv4 relying on MPLS. Thus with SRv4, two segments may be necessary: one node segment to reach the router being the invertex of the local link to force, the transverse edge in particular.

[28] As we will see latter on in this section, this property comes with even more advantages (whose some are developed in the last section) like the opportunity to quickly update the upstream routers and so avoid transient loops.

Lemma 1. *For all $h \in N_1(s) \cap V_D^s$ (considered as destinations to protect), and for each (u, v) denoting the (unique) transverse edge of any 1-alternate path in $\mathcal{F}_2$, we have $c_1(v, h) < w((s, h)) + c_1(v, s)$ if w is symmetric according to the direction of the edges ($w((u, v)) = w((v, u)), \forall (u, v) \in E$).*

Proof. Let us denote h the destination that we consider in $N_1(s) \cap V_D^s$ and (u, v) the (unique) transverse edge of any alternate path reaching it. With symmetric valuations, we know that $c_1(v, h) = c_1(h, v)$ for all couples, and with strictly positive weights, we also have $c_1(v, h) = c_1(h, v) < c_1(s, v) = c_1(v, s) < w((s, h)) + c_1(v, s)$ yielding the initial statement ($c_1(h, v) < c_1(s, v)$ because h is on the best path from s to v). The first inequality is an obvious inherent property of the SPT decomposition in sub-branches when considering strictly positive weights (with c being isotonic, sub path of best paths are also best paths but with strictly lower costs): here h and v belong to the same branch rooted at h, and path from h to v is a sub best path of $p_1(s, v)$. □

With this lemma, we can now introduce the notion of complete protection with one segment for all destinations (and all possible link failures) in the network. However, let us start with an immediate corollary focusing on destination h to better ease the progression. This corollary and its proof introduce technical notions related to segments and their relations with transverse edge.

Corollary 2. *Considering a graph $G(V, E, w)$ whose the valuation w of edges is symmetric, protecting the primary next-hops in $N_1(s)$ can be achieved with post-convergence backup paths requiring each one segment at most for any $s \in V$. More precisely, $\forall h \in N_1(s) \cap V_D^s$, any optimal 1-alternate path $p^* \in \mathcal{D}_2(s, h)$ requires exactly one segment to be safely deployed*[29].

Proof. Let us denote h the destination that we consider in $N_1(s) \cap V_D^s$ and (u, v) the (unique) transverse edge of the optimal 1- alternate path reaching it, denoted $p^* = p_2^j(s, h) \in \mathcal{D}_2$ or simply p_2 for convenience (it is actually a post-convergence backup path computed with *TBFH* along with its next-hop n_2). Relying on the previous inequality $c_1(v, h) < w((s, h)) + c_1(v, s) = c_1(s, h) + c_1(v, s)$ is sufficient as (v being the latest endpoint of the tunnel not using the failed link), we also know that: (i), v can be reached using only one adjacency segment in any cases (with adjacency (u, v) in particular) using an underlying path not including the failed link (s, h) by construction (because $u \notin branch_h(s)$) and, (ii), this rewritten inequality with $w((s, h)) = c_1(s, h)$ (by definition of $h \in N_1(s)$) implies that the sub-pre-convergence path from v to h is not sensitive to the failure of (s, h) as this link does not belong to it. As they are equal to pre-convergence paths, the post-convergence paths do neither include (s, h); before and after the failure, the two are the same with respect to this change (their edges exactly coincide). For (i), either $u = s$ and thus v is a local LFA (or an ECMP alternate) not requiring any segment, or $u \neq s$ such that $u \in branch_{h'}(s)$ with $h' \neq h$ and whose each node is reachable via a single node segment at most (a branch being a sub shortest path tree). If adjacency segments are globally advertised, it is worth to notice that the adjacency segment (u, v) both forces link (u, v) and the node segment to its upstream node u to be used in a single action. Thus, thanks to lemma 1, we can conclude that any post-convergence backup option being 1-alternate path, and also denoted p_2 for convenience since at least one exists in $\mathcal{D}_2$ (property 1), can be deployed relying on one encapsulation action at most: forcing the transverse link (u, v) with an adjacency segment in the most difficult case (if $u \neq s$ and each of its upstream node in p_2 does not satisfy the LFA criteria), none if n_2 satisfies locally the LFA condition (one local push from s to it is enough) or one node segment otherwise (u or one its upstream node in p_2 does satisfy LFA). Note that the easiest case to protect occurs thanks to ECMP: in such a case neither any segment nor specific actions are required. □

Theorem 2. *Considering a graph $G(V, E, w)$ with w being a symmetric valuation of edges, for any $s \in V$, we have: $\forall d \in V_D^s$, backup post-convergence re-routing paths in $\mathcal{D}_2$ require exactly one segment to be deployed (a local push at best, an adjacency segment at worst), using their latest transverse edge as an adjacency segment at worst.*

If $\mathcal{F}_2 \neq \emptyset$ there exists at least one 1-alternate representative $p^* \in \mathcal{D}_2$ simply denoted as p_2 to stress its post-convergence nature.

[29] In the following, we will consider that adjacency segments are globally advertised for the sake of clarity (i.e. not requiring two segments with the MPLS data-plane). Moreover, we assimilate a local push (a simple LFA) to a local segment.

Proof. As in the previous proof, it is enough to demonstrate that $c_1(v,d) < c_1(s,d)+c_1(v,s)$ for the same two reasons as before (i and ii). While (u,v) still denotes the transverse edge of a path $p_2^j(s,d) \in \mathcal{D}_2$ (here d simply replaces h in all notations), the only remaining difficulty is to generalize the previous proof when $d \in V_D^s \setminus N_1(s)$. For any $d \in V_D^s$, we have by construction $c_1(v,d) \leq c_1(v,h) + c_1(h,d)$ with $h = n_1(s,d)$ and since $c_1(v,h) < c_1(v,s) + w((s,h))$ (lemma 1 does not require node v to be the same for $p_2^j(s,h)$ and $p_2^{j'}(s,d)$ – there exists anyway an 1-alternate path towards h having the same v as in $p_2^j(s,d)$), we can directly conclude $c_1(v,d) \leq c_1(v,h)+c_1(h,d) < c_1(v,s)+w((s,h))+c_1(h,d) = c_1(v,s)+c_1(s,d)$ yielding again the initial statement. □

To summarize, if weights are symmetric, since $c_1(v,n_1) < c_1(v,s)+c_1(s,n_1)$ for any 1-alternate path p^* from s to $h = n_1$ using v as the outvertex of the transverse edge (u,v) in p^*, we have $\forall d \in V_D^s$ the following statement: the 1-alternate post-convergence path ($p^* := p_2$) is encodable with one adjacency segment at worst, its transverse edge (u,v).

This theorem can be extended into a more general result thanks to corollary 1: any post-convergence path p_2^j computed with *TBFH* can be encoded within one segment (at most) even if it includes more than one transverse edge. While, at first glance, one may question if it is the case for k-alternate paths in $\mathcal{D}_2$, we recall that such paths are made of subpaths having the same cost up to the penultimate transverse edge and so are also reachable in one segment because SR natively supports ECMP for the previous subpaths and their transverse edges.

Corollary 3. *For all destinations $d \in V_D^s$, and given arbitrary symmetric weights, TBFH is able to encode all paths $p_2^j \in \mathcal{D}_2$, i.e all the post-convergence ones (having so a cost of $c_2 > c_1$ but possibly more than one transverse edge), in one adjacency segment at worst (i.e. the latest transverse edge of the alternate path).*

In practice, deploying all these paths in parallel can be costly as their number can be exponential in theory (ECMP may lead to a large varierty of paths even if its resulting DAG looks limited at first glance). A good compromise can be to deploy only $|\mathcal{F}_2|$ first-hop distinct paths, that is exactly a path for each of alternate protecting next-hops $n_2^j \in \mathcal{F}_2$ computed with *TBFH*. Other more or less diverse variants are possible, but note again that a generalization may result in a significant overhead, i.e. at worst a specific segment for each SR distinct path. However, one can notice that even if the number of paths is actually exponential, they can all be used with $m+n$ distinct segments at worst (since one segment is enough for all).

For destinations $d \in V \setminus V_D^s$, we recall that segments are not strictly necessary (because SR natively handles ECMP, the encapsulation may be used for other purposes like not installing external routing entries within internal routers). With symmetric weights and thanks to corollary 1, the above corollary becomes obvious enough to omit a formal proof. Indeed, k-alternate paths with $k > 1$ does not require more detours than their counterpart including only a single transverse edge, the optimal 1-alternate denoted p^* (or sometimes p_2 to highlight its post-convergence ability as before). If among post-convergence paths, there exists such a k-alternate path denoted p, note first that using the same and only segment necessary to deploy its 1-alternate counterpart p^* towards link (u,v) is enough to activate both paths p and p^* in parallel with the same segment thanks to the ECMP ability of SR: they indeed do share the same cost towards u and v and are thus both used at the same time for d (if this ultimate segment is actually added because required at worst). This also implies that encoding path p cannot need more than this segment, more likely a less expensive action would be required like a local push or just a node segment if a node before u in path p verifies the LFA condition for d. This discussion thus becomes again an implementation choice/tradeoff (to mitigate the SR overhead in terms of number of parallel but specific SR paths to deploy): either using the same segment for all possible related ECMP paths ending at the same v or using distinct ones (as for distinct $p^* \in \mathcal{D}_2$ not considering the same transverse edge) to shorten the subpart of paths where a segment is needed. Another option is to enable only one or a subset, but the opportunity exists to use them all if one aims to target fine grained load balancing with a smart load balancing module for source routing (we develop this aspect in the general multi-metric SR case in section II.4). We will anyway rely on this corollary later on, in an opportunistic manner for deploying any kind of post-convergence paths (not only local ones computed for fast-rerouting).

Since p^* requires at most one segment at worst (thanks to Theorem 2 when weights are symmetric), it is hence the same even if multiple transverse edges are necessary (only the last one is enough it to

encode the path at worst), as well as for any post-convergence paths, e.g. in $\mathcal{D}_2$ and computed with *TBFH* towards destinations in V_D^s. Paths in $\mathcal{D}_2$ never need more than a single segment: an adjacency one at worst, a simple LFA push or a node segment in general and does not even need to be computed at best (if ECMP occurs for d: $|\mathcal{F}_1| > 1$).

About the End of a Loop & Remote Failures In addition, note that Fig. II.4 also illustrates two interesting properties involving pre- and post-convergence paths (as well as the forwarding loops they may lead to and more formally introduced in section II.2), their nature (e.g. typically 1-alternate path), and the (remote) LFA condition and P-/Q-spaces: these three concepts are indeed closely related. First, it is about transient forwarding loops and LFA conditions: while the potential forwarding loops made of pre- and post-convergence edges are prevented at u in the first sub-example (b), this is only the case at e in the second sub-example (c). However, on both cases, once solved, no other loops occur in sequence. Second, while at worst the P and Q space are adjacent with symmetric weights as respectively with u and v in (b), this not the case in (c) where the Q space is empty (or resumed to as h, the destination itself) for s towards h. Note it remains true even if the failure does not occur locally to s: for example, if we assume a failure on link (h, e) and look at destination d in Fig. II.4 (a) and consider symmetric weights as in sub-figure (b), we can observe that the same property holds. As soon as the P-space ends at u, the Q-space starts at v. These two properties will be turned as (sufficient) features for designing our algorithms in section V.1.b. Let us denote p_2 any path in $\mathcal{D}_2$ (implicitly with $d \in V_D^s$) and $p_1(y, d)$ a path in $\mathcal{D}_1(y, d)$ with $y \in p_2$. Note that while p_2 consists of post-convergence edges, $p_1(y, d)$ only have pre-ones by definition.

The first one can be stated as follow. Its main interest lies in the fact that it can be extended to show that symmetric weights implies that a packet does not need to be derouted twice, locally or not to the failure.

Property 2. *From a node $y \in p_2 \subset \mathcal{D}_2(s, d)$ verifying the LFA condition for a destination $d \in V_D^s$, and so being upstream or equal to the transverse outvertex of path p_2, not only its post-convergence edges coincide up to d with pre-convergence ones (without assumption on the valuation), i.e. there exists a path $p_1(y, d)$ whose edges are equal to the ones of p_2 (and since respectively their pre- and post-edges coincide no loop can occur up to the destination), but assuming symmetric weights we also have the reciprocal: if pre- and post-convergence outgoing edges coincide at a node $y \in p_2 \subset \mathcal{D}_2(s, d)$, then y satisfies the LFA condition and post-edges of p_2 continue to coincide with the ones of $p_1(y, d)$ up to d.*

A node having its pre- and post-convergence next-hops equals does not need to update anything (either the loops have been prevented before or will be after later on the path). Only node updating their next-hops by adding a new post-edge are required to operate a form of synchronisation (either explicit or implicit with several technologies at play as we will see). It occurs only once on a path (if there is no ECMP alternative), that is at most one series of loops are traversed.

Proof. The first implication is trivial as node y, where the LFA condition applies, neither uses s nor (s, n_1) to reach d (to avoid loop involving s), such that it is not impacted when (s, n_1) fails. Hence, no change occur on y as well as for its downstream nodes in p_2 (isotonicity of best paths) and no divergence between pre- and post-convergence edges can occur anymore. The second direction of the implication is provable because it contradicts Theorem 2 otherwise. Indeed, two cases may occur if y does not satisfy the LFA condition for $d \in V_D^s$ although its outgoing pre- and post-convergence edges are the same: either downstream edges regarding s continue to coincide until reaching a node $y' \neq y$ where no additional segment is required to reach the d or they do not (continue to coincide up to y'). Note that such a node $y' \in p_2$ necessarily exists (the transverse outvertex v of path p_2 being the last candidate in p_2 with symmetric weights), satisfies the LFA condition and is reachable with one segment as stated by Theorem 2. In the former case, we can conclude to an immediate contradiction since it implies that y does not use s to reach d, as edges in p_2 and $p_1(y, d)$ (as well as the ones of $p_1(y', d)$) would then continue to coincide up to d thanks to the first implication, and as such y should also verify the LFA condition as y'. Thus, only the latter case should be considered: at least one post-convergence edge $e = (y_1, y_2) \in p_2$ between y and y' does not coincide with the pre-convergence one exiting node $y_1 \in p_1(y, d)$. However reaching y, as well as y_1, already requires an encapsulation as $c_2 > c_1$: either and at best one local push if $y \in \mathcal{F}_2$, or an

adjacency segment at worst if $y = v$ is the last resort candidate (the outvertex of the unique transverse edge (u, v) in p_2), a node segment otherwise. Thus, forcing such a diverging edge e to actually deploy path p_2 would require one more segment in addition to this preliminary action contradicting so Theorem 2. Indeed, each divergence between pre- and post- edges costs one segment for a safe deployment. Since the two cases imply a contradiction, we can conclude that symmetric weights also imply that y verifies the LFA condition and as such ensures that pre- and post-convergence paths will coincide up to d. □

Note that the proof relies on the fact that we consider a destination d in V_D^s such that ECMP is not able to protect d: the path p_2, having a higher cost requires a segment or at least a local push with LFA. With asymmetric weights the equivalence is not true because forward and return paths between intermediary nodes affected by the failure can be distinct. An edge being both pre- and post- is not always safe with asymmetric paths. On the contrary, the symmetric property will be useful to design our transient loop-free distributed solution in section V.1.b as well as to develop the study of distant loops (not including s) and remote failures.

For the second property, let us introduce some new notations accounting for remote failures. The main idea behind this extension is to generalize the use of SR and TI-LFA to deal with any arbitrary single failure during the convergence (not only for local re-routing purpose). We will denote $p_2^j(n, d, l) \in \mathcal{D}_2(n, d, l)$ all the paths (along with their next-hops $n_2^j(n, d, l) \in \mathcal{F}_2(n, d, l)$) sharing the same cost $c_2(n, d, l) > c_1(n, d)$ (again the index j is not bounded the same for both sets as $|\mathcal{D}_2(n, d, l)| \geq |\mathcal{F}_2(n, d, l)|$ but we do not need to associate respective indexes explicitly). Note that, as we have done so far, we can ignore the ECMP case as we claim for a strict order $c_2(n, d, l) > c_1(n, d)$; however, this time, the failure and ECMP can apply locally or not. First, if $l \notin \mathcal{D}_1(n, d)$, there is no need to compute $\mathcal{D}_2(n, d, l)$ as n is not impacted by the change. Second, it is worth to notice that a node n cannot trigger a loop[30] for d after the failure of $l = (r, t)$ if its best cost does not change for d (again $\mathcal{D}_2(n, d, l)$ has no interest because $c_2(n, d, l) = c_1(n, d)$ otherwise): there exist enough path diversity to prevent the failure almost silently (just removing some next-hops along paths, locally or downstream). With equal cost, two cases can indeed occur and the set $\mathcal{F}_1(n, d)$ is enough in both but need to be updated in the former:

- either n has no descendant having a local ECMP backup for avoiding l; it is then as before because n actually handles the failure as a source s with $l = (s, n_1)$, and we can ignore such a case: the next-hop just need to be deactivated locally, i.e. $\mathcal{F}_1(n, d)$ is cut off of the next-hops leading to a single best path $\in \mathcal{D}_1(n, d)$ having (n, t) as first hop. In practice, next-hops can be each associated with a sub-DAG rooted at them (the sub-one induced by the removal of it in $\mathcal{D}_1(n, d)$) and if (r, t) belongs to all paths from n towards d (can be unique) in such a sub-DAG, the next-hop is removed;

- or n can benefit with a remote ECMP. That is, for each path $p_1^j(n, d)$ there exists a downstream node $x \neq n$ in $p_1^j(n, d) \setminus p_1^j(r, d)$ that locally and optimally handle the failure (r, t) (with respect to n as a source). Although the set $\mathcal{D}_1(n, d)$ is cut off of a subset of the local ECMP next-hops of x (that deactivates its next-hop leading to l remotely regarding n) but there is no need to reduce $\mathcal{F}_1(n, d)$, because there exists multiple paths for each sub-DAG rooted at each of its element (at least one path per sub-DAG does not use l and remains available).

Finally, although this kind of situations can also occur even if the cost increases (some next-hops are removed using the same condition – all paths in its sub-DAG are using l), $\mathcal{D}_2(n, d, l)$ should be computed in more details to check whether new paths $\notin \mathcal{D}_1$ and mostly next-hops $\notin \mathcal{D}_1$ are necessary. Indeed, it is possible that $\mathcal{F}_2(n, d, l) \nsubseteq \mathcal{F}_1(n, d)$ when new next-hops are locally necessary (only if the cost increases but not necessarily reciprocally). Let us denote $\mathcal{F}_2^*(l) \subseteq \mathcal{F}_2(n, d, l)$ the subset of next-hops that are not in $\mathcal{F}_1(n, d)$ and $\mathcal{D}_2^*(l)$ the paths they lead to (their first hop is new: $n_2^j(n, d, l) \notin \mathcal{F}_1(n, d)$). Only those post-convergence paths can trigger loops[30] and so requires the use of a segment at n, as others are just subset of pre-convergence ones as previously (ECMP) or remote new ones not local to n (there is no segment requirement up to the descendant using the new next-hop and possibly triggering a loop).

On the contrary as for the local case where failures concern direct neighbor, and although sets $\mathcal{D}_1$ and $\mathcal{D}_2$ are exclusive as well as $\mathcal{D}_1(n, d, l)$ and $\mathcal{D}_2(n, d, l)$ and $\mathcal{F}_2$ regarding $\mathcal{F}_1$, it is not the case for $\mathcal{F}_2(n, d, l)$

[30] In the following, in particular with corollary 5, we will demonstrate that not only such cases cannot generate a loop, but they cannot be involved in any circuit.

regarding $\mathcal{F}_1(n,d,l)$. Indeed, with remote failure both (new) exclusive post-convergence paths and paths whose pre- and post-edges coincide can co-exist. But only the former can lead to loops (because the later only deactivates some next-hop as with ECMP).

Property 3. *With symmetric weights, even if the failure is not local, that is considering post-convergence paths $p_2(n,d,l) \in \mathcal{D}_2^*(l)$, towards a destination d, for any node n (as source) and any failure l in general, only a single segment is necessary and sufficient to encode these new paths safely (without introducing forwarding loops).*

As for the local failure case, we are interested in particular in nodes introducing new next-hops on the post-convergence path (as others does not implies local forwarding changes and are eventually similar to cases where downstream nodes optimally handling the failure). A node activating a new next-hop equal to the old pre-one does not need to handle a loop possibly occurring later on the path (we will later generalize this result to nodes not implied in circuit), the first downstream node introducing a new next-hop can do the same to prevent itself the loop. I argue it is superfluous to do so as packets looping once in a circuit can still occur[31] and we prefer to restrict the use of SR encapsulation to necessary cases involving loops actually occurring considering an efficient opportunistic approach developed in chapter V.

Intuitively, thanks to the symmetric valuation assumption, this generalized property (3) of Theorem 2 remains true in the case of remote (single) failures because when forcing a transverse edge in order to deploy any kind of alternate path, the Q-space also and always *follows* the P-space even in such generalized conditions (at worst, otherwise they intersect and requires a node segment or just a local push instead of an adjacency one). It is not specific to the pre-computed path for a local failure (the TI-LFA backup) but, more generally, related to its post-convergence characteristic in case of symmetric weights. Indeed, the P-space (the pre-convergence paths in this case) can be compared to the post-convergence path and the node where they differ is in the Q-space, or at worst one of its successor is.

Proof. Let us prove that a post-convergence path $p_2(n,d,l) \in \mathcal{D}_2^*(l)$ (verifying $c_2(n,d,l) > c_1(n,d)$ by definition), with a given next-hop $n_2(n,d,l) \in \mathcal{F}_2^*(l)$, is equivalent to specific k-alternate paths used to prove corollary 1 (but note that paths in $\mathcal{F}_2^*(l)$ may be distinct from local re-routing paths in $\mathcal{F}_2(n,d)$ to remain general). Since $n_2(n,d,l) \notin \mathcal{F}_l(n,d)$ and $l \notin p_2(n,d,l)$ by definition, the nodes and links in between n and r in any path $p_1^j(n,d)$ are not used neither in $p_2(n,d,l)$ (otherwise it contradicts the isotonicity of best paths). Hence, the partition of edges described in section II.1.b.1 is still valid, not only covering and protecting a first hop but a whole first sub-branch (as well as a SPC algorithm like *TBFH* computing $p_2(n,d,l)$ can be extended to return the best path in the sub-graph of G induced by the removal of all nodes and edges belonging to the path $p_1^j(n,d)$ up to l). That is, with a generalization of property 1 and corollaries 1 and 3, we can easily observe that one segment is enough to deploy such a path $p_2(n,d,l)$ (and, more generally, any path $\in \mathcal{D}_2^*(l)$). Indeed, if there exists one alternate path avoiding link l to reach d from n, it does include at least one transverse edge to reach back the branch to which d belongs to – the one including the sub-branch $p_1^j(n,d)$ (because of the partition of edges, the sub-branch is not protectable otherwise), and at least one of these alternate paths is a 1-alternate one for the same reasons as in property 1. The resulting corollaries (1 and 3) also remain valid for such paths thanks to the ECMP support of SR. □

By combining these two last properties, one can easily envision to deploy an ideal convergence model with SR. Indeed, locally or remotely, respectively pre- or on the fly computed backup paths are loopfree by design thanks to only one detour at most. Such paths only require one operation to be deployed at the node introducing a new forwarding state by adding one more next-hop. Moreover, these paths also are optimal and it is tempting to use this characteristic to silently update all nodes. **TI-LFA backup paths computed with *TBFH* are exactly the new best ones, what is the best technique to gracefully enforce the forwarding states to switch from old to backup = new ones?** Indeed, deployed as it is, timers or an explicit synchronization is required to know when nodes are ready to stop using backup paths. We

[31] While using SR systematically at each next-hop activation can mitigate this effect, it also induces overhead and new technical challenges as calibrating timers to eventually switch to new states (and stop encapsulating). Moreover, nodes where the pre-convergence next-hop(s) diverge from the new one(s) can still lead to such *one tour loops* as long as the new next-hop is not activated.

will show later in details in chapter V how one can deploy an ideal convergence model only requiring data-plane operations to update all nodes correctly even if some nodes only support the most basic SR features (and no data-plane driven simple updates and tests).

Finally, considering a possibly asymmetric valuation of edges, the guarantees provided until here do not hold anymore. However, two options follow to deploy post-convergence backup paths for each destination to protect. Let me show how retrieving such (longer) list of segments.

II.1.c.2 Compute and Encode Longer Segment Lists in Case of Asymmetric valuation

On the one hand, we have seen that, whatever the valuation, when $c_1(v,d) < c_1(v,s) + c_1(s,d)$ with v being the outvertex of the unique transverse edge (u,v) of path p_2 (the best 1-alternate path for the couple (s,d)), the local router can easily support the change as being illustrated in Fig II.4.b. Either there exists only one segment or less to use, ranging from a local push (LFA) (e.g. i for destinations $\{b,v,d\}$) to U-TURN and R-LFA, requiring only a (remote) adjacency segment to work properly at worst (considering symmetric valuation, only this case occurs for any $h \in N_1(s)$ and so $d \in V$). This comes with the advantage of requiring very few computation to detect the closest unique exit point (i.e. the end of the encapsulation) as illustrated in Fig. II.4.b: at worst, this point is u for destinations $\{a,e\}$ with one node segment (to u that is here also a possible U-TURN). With R-LFA, or using an adjacency segment, and considering the destination h itself, u also require to force the packet on interface (u,v) while it can be at best $n_2^1 = i$ (i.e. a LFA). Solving the problem is then just about looking for the node(s), and/or the link(s), to force as segments in order to avoid forwarding loops. **Selecting the first node in a path p_2^j satisfying the LFA condition using a binary approach is efficient enough to be performed for each destination independently** (such a node is located between $n_2^1 = i$ and v according to the destination considered in the example). In other words, this approach looks for the *earliest exit* in the SR tunnel (made of only 1 segment)[32].

On the other hand, if $c_1(v,d) \geq c_1(v,s) + c_1(s,d)$, more computation is required to deploy the re-routing paths in $\mathcal{D}_2(s,d)$ as well as the induced overhead and number of segments. Indeed, more than one segment is then necessary (this occur only with asymmetric weights because the condition cannot be satisfied otherwise – since we have $c_1(v,n_1^1) < c_1(s,v)$ because v and n_1^1), in that case the computation is more challenging and expensive. Indeed, one needs to compute several Shortest Path Directed Acyclic Graph (SPDAG) (not only rooted at s or its neighbors to exclude the failing element, but looking at branches of the SPDAG rooted at remote nodes possibly not using it for other destinations) to provide the correct and complete list of segments. **More precisely, one needs to compute at least as many SPDAG as their necessary numbers (i.e. minimal in terms of #segments) for a given path to encode** (or, more generally, considering a SR graph obtained with an APSP, i.e. a All Pair Shortest Path algorithm running Dijkstra for each node as source resulting in a complexity of $O(n^2 logn)$ for sparse networks – see also section II.4).

In practice, the sketch of the procedure to retrieve the latest exit SR list for a given path $p_2 \in \mathcal{D}_2^*(l)$ is the following[33]: looking at the SPDAG rooted at s, it is about greedily extracting the latest (intermediary destination) node $x_n \in p_2 := x_1...x_n x_{n'}...d \mid x_1...x_n \in \mathcal{D}_1(s,x_n)$, possibly force one (more) hop $x_{n'}$ with an adjacency segment $(x_n, x_{n'})$ if necessary when the link is not in any $\mathcal{D}_1(x_n,.)$ considering SR with MPLS or systematically with SRv6 [34] and continue like this iteratively from x_n, or its forced neighbor $x_{n'}$, towards x_{n2} or $x_{n2'}$ and so on with intermediary steps $ni, ni+1$ until reaching d (with $d = x_{nk} \vee x_{nk'}$ in k steps/segments). Overall p_2 is decomposed as follow: $p_2 := x_1...x_n x_{n'}...x_{n2} x_{n2'}...x_{ni} x_{ni'}...d$. In summary, one can determine the next segment to push by finding the latest common node in the P space

[32] One can save computing time, at the detriment of the SR overhead, by just looking directly at the *latest exit* without any refinement, with an adjacency segment for v or a node one for u (if sufficient).

[33] For fully enabling ECMP, or just find one minimal path in term of number of segments, it is more challenging as each option has to be considered. The encoding may be embedded within *TBFH* or in SPC algorithm to ease the exploration and computation of the best options. For example, on can look for all the best paths in number of segments, or rather consider all paths requiring less than a given number of segments.

[34] It is worth to notice that MPLS adjacency segments are used only if they are strictly necessary (no node segment can replace them) because their weight is not equal to the best distance linking the two extremities of the link. On the contrary, since an adjacency comes at the same cost as a node segment with global SRv6 advertisements (instead of a node + an adjacency for MPLS), it is exactly the opposite: adjacency segments are always useful as one more next-hop can be forced for free. Nodes $x_{ni'}$, transverse edges, are used systematically as soon as the P space ends even when it intersects the one.

– from the last explored node starting initially from s and continuing with x_{ni}, to look at the next latest x_{ni+1} such that $x_{ni}...x_{ni+1} \in \mathcal{D}_1(x_{ni}, x_{ni+1})$ until reaching d.This kind of greedy solution can be proven minimal [LAM+22] but requires as input the SR graph or at least a partial APSP as in Sec. II.4.

Fig. II.4.c illustrates the second option when multiple segments are required: here node v, the outvertex of the transverse edge, still prefers to use s in its best path towards h, using so the failing link (s, n_1) to reach h. This is because the weight of the link (e, h) is high and there does not exist better alternate path towards h not using it or the failed link. This kind of situation leads to more overhead and computations: from s and its SPDAG, one can determine that the largest common portion it has with the second best DAG computed with *TBFH* is towards $x_n = u$. Then, continuing from u and its SPDAG, we have $x_{n2} = e$ and $y_{n2'} = h$, the adjacency (e, h) requires to be forced to eventually and safely reach h. Overall, two segments need to be stacked, the node one towards u and the adjacency one (e, h) to reach h from e and so avoid link (s, h) without introducing any transient loops. However, although polynomial, such a greedy approach is more complex than it looks at first. There exists subtle technical difficulties not presented here that make the computation not so obvious, e.g., ECMP that has not been deeply discussed here, the choice between adjacency or node segments and multi-metric problems such as the one studied in section II.4 that generalizes such encoding approaches. Moreover, note that we rely here on a latest exit approach that is not the lightest model one can opt for (packets carry segments longer on the route). If one prefers to opt for an earliest exit model, on the contrary to the simple symmetric case where only one segment is required, it has to look backward from the destination towards the source, looking to intermediary latest exits in the reverse direction. Indeed, not only a SR graph or similar structure is required, the analysis should be reversed starting from the destination and looking back to the most distant source having the same subpath in common. Instead of looking for the most distant destination node as with the latest exit model, it searches for the most distant source that becomes the novel destination at the next step. Apart from this backward logic, the greedy approach remains the same, and once the actual source is reached the procedure is finished.

In section V.2.a, we develop a more advanced technique for node protection being able to construct latest or earliest segment lists not requiring the SR graph or any APSP. Only a variant of *TBFH* is needed (also requiring only two SPC runs at worst), but let us now conclude with another ongoing project in this context.

II.1.c.3 Discussions on Further Variants and Improvements

Finally, there exists two last options we can benefit with considering a SR context. First, for each destination d requiring more than one (adjacency) segment in the asymmetric case, one can rely on a given $p_2^j(s, h)$ path like the $p_2^1(s, h)$ one instead of considering all specific $p_2^j(s, d)$ paths. It allows for saving computing resources, but it also mitigate the SR deployment overhead and avoid computing more SPT/DAG. That is a single backup path towards h can be re-used for many other destinations as it is without introducing extra-computations (e.g. when the SR graph is not available) and operations (deploying each ECMP specifically with a distinct segment list). However, such a shortcut prevents for considering all ECMP optimal local backup paths (both in terms of distance and #segments), because loosing the post-convergence property and optimal solutions at the destination granularity.

Second, **the specific use of post-convergence TI-LFA paths can ease the micro-loops avoidance and allow to immediately update upstream routers involved in such loops**. To illustrate this last opportunity, let us ignore ECMP and denote r a router such that $s \in p_1^1(r, d)$ and $r \in p_2^1(s, d)$: it is respectively an upstream router of s regarding the pre-convergence path from r to d, but a downstream one regarding the backup path p_2^1 from s to d when the link (s, n_1) fails (i.e. there exists a transient forwarding loop between them). If r receives a packet destined to d but encapsulated towards a given intermediary TI-LFA segment, let say the adjacency (u, v) as a worst case example for the symmetric case with $r \in p_1^1(s, u)$, it can immediately update its forwarding state towards d with the same next-hop as the one it uses for u without waiting its control-plane to converge, because we have $p_1^1(r, u) \subseteq p_2^1(s, d)$ as well as[35] $n_2^1(r, d, (s, n_1)) = n_1^1(r, u)$.

In practice, upon such a reception, r just has to update its forwarding state towards d, setting its new

[35] I recall that $n_2^1(r, d, (s, n_1))$ denotes the first ranked post-convergence next-hop of r in the graph $G(V, E \setminus (s, n_1))$. This notation has been previously introduced for remote failure.

best next-hop according to the one it already uses for u as destination, i.e. $n_1^1(r,d) \leftarrow n_1^1(r,u)$. Thanks to the property of isotonicity of shortest intra-domain paths, we indeed have: $p_1^1(r,u) \subseteq p_2^1(r,d,(s,n_1)) = p_1^1(r,u) \circ (u,v) \circ p_1^1(u,d) \subseteq p_2^1(s,d) = p_1^1(s,r) \circ p_2^1(r,d,(s,n_1))$. If $r = u$, then we simply have $n_1^1(u,d) \leftarrow v$ and this occurs only for adjacency segment. For node segment only the first part is required, and for local push (basic LFA), no action at all is required (s can update its forwarding state safely without forcing its neighbor). We will generalize such a feature considering both ECMP and remote failures in the last chapter V.

The advantages of this second opportunity are twofold: (i) once updated, there is no more need to overload packet with a segment as the FIB entry is now ready for the new, post-convergence, forwarding state – because in the same state as it will be after the convergence; (ii) it eliminates transient micro-loops that can arise locally or remotely from the failure because routers do not have anymore to rely on (possibly badly configured) timers before stopping the encapsulation.

However, this ideal approach not only requires that all nodes support this feature and immediately update their FIB (at line-rate in the embedded data-plane pipeline), but also that the TI-LFA packets are not lost or delayed for any reason. That is why we will also develop efficient conditions to finely check if packets are looping with an efficient opportunistic approach able to implicitly adapt to the underlying situation (e.g. heterogeneous or incremental deployments). To benefit from both (i) and (ii) in practice, I will then develop in section V.1.b a new model based on extra advantages bring thanks to symmetric valuation: we will indeed see that loopy packets are easy to detect and enable to efficiently check for such conditions at the data-plane. This property indeed enables a simple and efficient synchronization system because the graph resulting from the merging of pre- and post-convergence paths is a very specific kind of directed *chordal* graph. While (i) mitigates the use of non elementary transient paths, only the outdated traffic is still encapsulated as long as necessary[36], (ii) is granted either because upstream nodes are updated before downstream ones send their traffic without encapsulation or because the (loopy) traffic is encapsulated otherwise. Sec. V.1 provides a detailed description of this ongoing work.

[36] In practice, up to now, one needs a safe timer implicit mechanism or an explicit synchronization one while our model stops the encapsulation at the packet granularity as soon as possible.

Title of the publication	Name of the venue	Year	Reference
Low Complexity Link-State Multipath Routing	Global Internet Symposium (**GIS**)	2009	[MCP09]
An Efficient Algorithm to Enable Path Diversity in Link-State Routing Networks	Computer Networks (**COMNET**)	2011	[MFB+11]

Table II.3: Summary of my publications related to multi-path computation algorithms (after my Ph.D. thesis)

Table II.3 lists my publications related to this topic. I aim to submit new papers in this field thanks to the extensions proposed so far as well as the ones introduced in V.1.

TBFH: Conclusions and Perspectives

Multi-path routing enhances the network reliability: it allows for load balancing and fast re-routing to circumvent congestions and failures. However, the overhead imposed by signaling messages [223], the time and space complexity of multiple routes computation can hamper its deployment [298]. In this section, I present an efficient algorithm, *TBFH*, which provides a greater path diversity than only considering equal cost routes (ECMP) with a very low overhead.

In particular, *TBFH* efficiently computes the two best first hop disjoint paths (link or node disjoint as discussed later with *TBFN*). The time complexity of *TBFH* does not depend on the degree of the calculating router, it requires only two SPC. Furthermore, we propose a general multipath forwarding scheme that provides load balancing and fast re-routing next-hops. One possible application of this scheme is to expose the forwarding diversity to the end hosts and allow them to control load shifting decisions thanks to a tagging mechanism.

We have considered several validation rules ensuring loop-free forwarding, e.g. the downstream criterion (DC) for load balancing, and the loop-free alternate rule (LFA) and its variants for local fast recovery. In a hop by hop forwarding context, the alternate next-hops computed with *TBFH* are called local post convergence next-hops. This set of next-hops minimizes the number of flow deflections in case of failure.

Our proposition can be incrementally integrated in OSPF or IS-IS by replacing the path computation algorithm without any additional messages in the control-plane. Our overall solution is scalable for large IP networks and all routers, even those having high degrees, and can operate in conjunction with routers using ECMP as well with non multipath-capable ones [MCP09, MFB+11].

The second part of the section has not been published in [MFB+11]: **I have revisited *TBFH* considering a SR context and demonstrate how it can be extended to improve the convergence of IP networks in general.** With SR, our re-routing solution can become much more general as well as the ability of our algorithm to efficiently retrieve and encode post-convergence TI-LFA backup paths. In particular, we have seen how any link can be protected with such paths made of only one segment in any cases considering symmetric valuation. We have provided several new results showing how *TBFH* is the most appropriate algorithm for such a computation (requiring only two SPC to compute all SR backup paths) as in addition it focuses on post-convergence paths. Such paths ensure direct convergence and as well as shortest TI-LFA backup transient paths in general, they do not require more than one segment in the symmetric case, exactly one if ECMP is not locally available. Not only we considered the symmetrical case with advanced details, but we also discuss efficient algorithms and several strategies (latest or earliest exit modes and both MPLS or SRv6 data-planes) for the general case (regarding both remote failures and loops). Moreover, we discuss several encoding strategies and start to introduce how and why SR can help to update the whole network with TI-LFA encapsulation very efficiently (neither requiring timers nor an explicit synchronisation). We will develop all these aspects with a detailed analysis of conditions at play in section V.1.

In the same chapter, we will also extend *TBFH* to deal with node protection and show the resulting complexity both in term of SR overhead (e.g. #segments) and computation. With a data-plane implementation of this new forwarding model, one can expect high performance to gracefully update the network (even with partial deployment). Finally, it is worth to notice that *TBFH* can serve as a basis for *OPTIC* (detailed in II.3) in a three-scale hierarchical forwarding architecture that can ensure almost instantaneous updates, even for the BGP transit traffic.

II.2 AGBA: the Adjusted Greedy Backward Algorithm

In this section, I will partially overview some of the work performed during the Ph.D. thesis of François Clad ([Clad, 2014], 2011-2014) [37] under the supervision of Jean-Jacques Pansiot and my own guidance as for the other thesis reported at the end of this manuscript. This is also a joint work with Pierre François (INSA Lyon) and Stefano Vissicchio (University College of London) with whom we have elaborated several algorithms and their proofs, in order to avoid transient forwarding loops in case of network updates or changes in general.

More precisely, as an evolution and generalization of the initial work done during the thesis of Pierre François [120], we develop incremental techniques to gracefully prevent any of such anomalies when performing maintenance operations – ranging from a single link weight change or a shutdown, up to the scale of an entire router (e.g. to perform a router-wide software update). Since the problem is increasingly challenging according to the scope of the change, in the following I will mainly introduce the core contribution given in our most general achievement, i.e. in [CVM+15]; but we published several other papers on the same topic (according to the scale of the modification, e.g. link, node or network wide in other works, several options and variants exist to tackle them): [CMV+13, CMP+14]. In a parallel IP measurements work [MDP+18], whose main results are given in Sec.III.2, we motivate the present problem by showing ground evidences of the presence of forwarding loops in a real Internet Service Provider (the French NREN, the RENATER network).
I will first motivate, position and illustrate our safe IP convergence problem with existing solutions. Then, I will formally introduce the framework we rely on to design our solution AGBA, in particular its ability to solve side effects like intermediate forwarding changes. The table of content of this section is the following:

II.2.a Performing Graceful Router-Wide Updates for Link-State Protocols

With OSPF or IS-IS as the link-state routing IGP in use, the state of all outgoing links of each router is (reliably) flooded over the network. Each router can then build a directed weighted graph representing the network and so compute its forwarding paths with a shortest path algorithm.Each topological modification triggers a convergence process (i.e., to flood new Link-State Advertisements (LSA), recompute updated paths, and install corresponding FIB updates) during which transient forwarding loops may occur [176]. Despite the additive nature of the underlying intra-domain metric (with positive weights) and its isotonicity, it does not prevent forwarding loops as the transient loss of synchronisation during the convergence provoke inconsistent topological states among routers. Such loops may cause in turn packet losses and delays increase [276, 366] and [MDP+18].

To avoid transient inconsistencies to provoke such anomalies, protocol based solutions have been investigated in the past, notably oFIB [122]. oFIB is a proposal to order the FIB update among routers in a way that ensures forwarding consistency during the convergence process. oFIB can prevent loops in the case of link and node shutdown, as well as during opposite up events. However, such a solution, as most others, requires both complex modifications to the specification of the IGP and a complete support in all routers of the network. An alternate solution, PLSN, is described in [303]. Loop avoidance is only granted one hop away from the rerouting node by controlling the time at which it actually updates

[37] He is currently working as an Engineer at Cisco Systems working on Segment Routing. We still continue to collaborate on several research projects including the one presented in Sec.II.4. His Ph.D.'s report gathers all our achievements in the field.

its FIB. Considering a fixed amount of time, a PLSN rerouting node delays its FIB update for a given destination if its new next-hop is not loop-free for the destination. PLSN then behaves like a reduced version of oFIB: it does not require router-to-router synchronization, but does not avoid neither remote transient loops.

Hitless network migration techniques such as [337] or more recent global updates techniques [346, 299, 284] could also be considered as globally safe reconfiguration methods for our purpose. They are however focused on network-wide changes, and require to temporarily maintain two IGP configurations or complex states in the whole network, and then finally switch from the initial to the final one on a per router basis. As such, it implies higher management overload, and longer reconfiguration processes than the lightweight technique we design to minimize the operational impact of a single router reconfiguration. Finally, solutions have also been investigated for the case of routing software upgrades on recent router architectures, which are able to fully dissociate the routing and forwarding engines. The *I'll Be Back* capability [301] allow the router to continue forwarding packets even if the routing process is down, while preventing possible forwarding loops. Our approach enables to solve the same problem (graceful router software upgrades) without requiring modifications of current routing protocols and with minimal control-plane overhead (but at the cost of local traffic shifts as described later).

Simply put, the objective of this work is to avoid transient anomalies triggered by distributed inconsistencies occurring during routing transitions, the question to answer is the following:

Research Question

How to Efficiently Ensure a Loop-free Convergence during Intra-domain Routing Changes Involving an Entire Forwarding Node?

The next section not only illustrates the question we aim to solve but also explain at a high level the distinct settings we propose. Since the routing convergence is a distributed process, it may lead to various transient inconsistent states as the routers can loose their synchronization (i.e., they temporarily do not share the same global network view) during the convergence.

II.2.a.1 Distributed Convergence and Transient Forwarding Loops

Shutting a Node: a First Basic Illustration Before formally introducing our model and framework in more details, let me first illustrate the core problem in Fig. II.5: it exemplifies a transient forwarding loop in the case of a router removal (the one denoted 0 in red). Namely, while the left subfigure and the right one represent the initial and final IGP topology, respectively, the central one illustrates how inconsistent information held by different routers may cause transient forwarding loops during protocol convergence. The red and green arrows respectively represent the next-hops for destination 4 before and after the removal of node 0. Black arrows represent next-hops that remain the same. If router c updates its next-hop to 4 before d, then c starts forwarding traffic for destination 4 to d, while d keeps forwarding traffic to c. This creates a transient loop between c and d, which will only be solved when d also has updated its next-hop.

In this work, we have proposed practical solutions to tackle such an issue without requiring any change in the specification of current routing protocols. Moreover, the solutions presented in this section apply to both symmetric and asymmetric link weights whereas the new option I sketch in chapter V only applies for the symmetric case.

Our framework relies on a progressive modification of IGP link weights. Those incremental changes allows for a loop-free ordering in the updates of all the forwarding paths. Practically, the IGP convergence is split into subsequent steps. During each step, weight modifications are flooded to all routers in the network through standard IGP LSA. To reduce their operational impact, our algorithms minimize the number of convergence steps, hence limiting the additional control-plane overhead. Instead of applying graceful operations on a per link basis and possibly generate long weight modification sequences [CMV+13], our strategy is not as naive and rely on the possibility of including weight changes for multiple links attached to the updated router in a single LSA. In the example of Fig. II.5, if node 0,

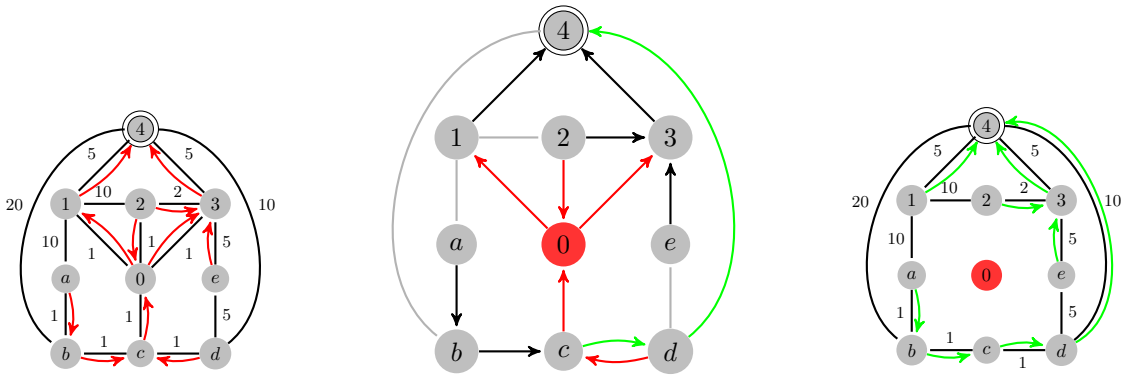


Figure II.5: A transient forwarding loop can occur between nodes c and d for destination 4 during the routing convergence.

	Intermediate	
	Transient Loops	Forwarding Changes
with μloop delay	*GBA*	*AGBA*
w/o μloop delay	*DGBH*	

Table II.4: Overview of our algorithmic contributions.

Avoiding Intermediate Forwarding Changes and

before its removal, sends an LSA to update the weights of links $(0, 1)$, $(0, 2)$ and $(0, 3)$ to values 4, 2 and 4 respectively, then d will switch to its final forwarding path while c will not. This guarantee no transient loop between c and d when c updates its FIB.

Several Settings for Several Challenges Computing minimal weight modification sequences for router-wide updates is challenging for three main reasons. First, all the destinations in the network must be taken into account, as our goal is to minimize the number of steps across them. Contrary to the link modification problem studied in our previous work [CMP+14], the solution space for the node shutdown problem is k-dimensional, with k being the degree of the router to be updated. Second, applying several weight increments in a single LSA may lead to the use of next-hops that are neither initial nor the final ones. To capture such intermediate next-hop changes, it is then not sufficient to rely on the initial and final routing states. Moreover, these intermediate changes provoke a new kind of disruptions such as flow deviations. In the example in Fig. II.5, the updated node 0 may transiently use node 2 as next-hop towards 4 during the convergence if the weights of 0 are set to the previously suggested values (4, 2 and 4, which are the minimal ones to avoid the loop between c and d). Third, those *intermediate forwarding changes* possibly lead to additional transient loops. In the initial state given in Fig. II.5, one of the shortest paths from 2 to 4 includes 0, with 2 being in an Equal-Cost Multi-Path (ECMP) state. Hence, a new kind of transient loop can occur between 0 and 2, an intermediate loop that depends on the LSA injected to avoid the potential loop between c and d. Intermediate forwarding changes are necessary but not sufficient conditions to trigger such loops. Finally, in the example, note that a uniform increase of 3 on all links avoid both kind of loops. However, targeting minimal sequences generally comes at the cost of applying non-uniform weight-increment LSA.

To deal with transient loops (intermediate and non), we develop multiple algorithmic contributions, which are summarized in Table II.4. Our algorithms target two different settings.

In the first setting, the next-hops of the reconfigured router are kept constant during the entire IGP convergence by temporarily disabling synchronization of the router data-plane and control-plane, e.g., through the *μloop delay* feature [208]. In this setting, no intermediate transient loop can occur, and the graceful sequence minimization problem is optimally solved by our **Greedy Backward Algorithm (*GBA*)**.

For every destination, and in a reverse fashion[38], *GBA* extracts the constraints that prevent transient loops resulting from the union of the initial and final forwarding paths. It then greedily computes minimal sequences of weight changes that verify all the extracted constraints.

In the second setting, data-plane and control-plane remain synchronized, and intermediate transient loops should be prevented algorithmically. For this setting, we propose two extensions of *GBA*. The first extension, called **Adjusted Greedy Backward Algorithm (*AGBA*)**, provably finds a minimal sequence that prevents both transient loops and intermediate forwarding changes. In particular, to avoid intermediate forwarding changes, it verifies that weight changes comply with a system of linear inequalities. Note that *AGBA* can also be used in the first setting if intermediate forwarding changes have to be avoided.

The second extension, called **Dynamic Greedy Backward Heuristic (*DGBH*)**, computes sequences that prevent any kind of transient loops (including intermediate ones) but not intermediate forwarding changes in general. Intermediate transient loops are simply prevented by augmenting the set of constraints. *DGBH* is a heuristic in the sense that the sequences it computes are safe but not provably minimal with respect to the loop prevention problem.

Generally speaking, to simplify the context of application, note that we consider the case of a non-urgent router update as for maintenance. Our approach can theoretically be combined with fast-reroute techniques like the ones presented in the previous section or the next one to address failure use cases. However, technically investigating their practical interactions is not addressed here, we rather aim to design an integrated solution as developed in the Chapter V. Besides, we do not consider neither the general theoritical case of fully arbitrary weights change on a given router, that is applying simultaneously increments and decrements on distinct links (and so potentially find shorter sequences than with always increasing ones we will consider). Such a relaxed problem is more challenging than the ones we look at here and discussed in more details in the conclusion of this section. Last but not least, we also assume that no network failure occurs while the sequence is applied on the network. Indeed, although it can happen in practice, independent concurrent modifications are rare enough [MDP+18](especially for planned ones as they can be scheduled) to restrict our analysis to the case of single isolated event (as we have done so far and in the remainder of the document). This simplification is common in this context as the number of failure combinations may quickly lead to intractable resolution in practice.

II.2.a.2 Modeling Forwarding Loops as Constraints

Basic Notations In link-state IGPs, forwarding paths are computed as the shortest paths on a weighted graph $G = (V, E, w)$[39]. The graph G is not static over time as it evolves due to failures or reconfigurations. In the following, to ease the reading without loss of generality in the case of a single event, we focus on the case of the router removal. We denote the initial IGP graph as G, the router to be removed as 0, and the final IGP graph as $G' = G \setminus \{0\}$, the subgraph induced by the removal of the node labeled 0, $G' = (V \setminus \{0\}, E \setminus \{(0, x) \in E \mid x \in succ(0)\})$.

Since multiple paths can have the same distance for some source-destination pair, the set of shortest paths from each source to a single destination forms a Directed Acyclic Graph and not only a tree, called *Reverse Shortest Path DAG* (RSPDAG). Hence, we denote as $RSPDAG(d, X)$ the set of forwarding paths computed by IGP routers towards a given destination d in a graph X. Transient loops can occur during the transition from G to G' if and only if $MP(d, G) := RSPDAG(d, G) \cup RSPDAG(d, G')$ contains circuits (see, for example, Fig. II.5). For the sake of simplicity, we use $MP(d)$, $RSPDAG(d)$ and $RSPDAG'(d)$ as shortcuts for $MP(d, G)$, $RSPDAG(d, G)$ and $RSPDAG(d, G')$ respectively.

Our convergence technique relies on a progressive modification of the weights configured on the outgoing links of 0. Formally, it consists in computing a sequence of intermediate weighted graphs $G_0, G_1, \ldots, G_n$ minimizing n, where $G_0 = G$, $G_n = G'$ and $\forall \in \{1, \ldots, n\}$, $G_i = (V, E, w_i)$, such that $\forall\, i \in \{0, \ldots, n-1\}$, $MP(d, G_i) := RSPDAG(d, G_i) \cup RSPDAG(d, G_{i+1}))$ contains no circuits. We generally refer to differences between weights in an intermediate graph $G_i \neq G$ and initial weights in G as *weight increments*, and we call a sequence $\{w_1, \ldots, w_n\}$ satisfying the previous property as a *weight increment sequence*. The term weight increment reflects our initial design choice: the weight of any link outgoing

[38] While a forward approach is enough for dealing with the link case, it does not work for the node one as demonstrated in [CMV+13] and [CMP+14].

[39] Where V is the set of routers, E provides adjacencies between routers, and $w : E \to \mathbb{N}$ the valuation mapping each directed link to its IGP weight.

from 0 remains always greater or equal to its initial weight. That is, since we aim at offloading traffic from the node to be removed, we do not consider sequences of weight modifications in which weights are decreased with respect to the initial state, as this can only make 0 more attractive. Nevertheless, we admit negative components in weight changes, e.g., if following positive increments as long as their sum remains positive. More formally, we define w_i such as $\forall i, \forall (u,v) \in E \neq (0,n)\ \forall n \in succ(0) : w_i(u,v) = w(u,v)$ and $\forall v \in succ(0), w_i(0,v) \geq w(0,v)$. We say that such sequences are *globally positive* and made of absolute increments as we authorize relative decrements among w_i only in the positive space regarding $w_0 = w$ $(\forall (u,v) \in E, w_i(u,v) > w(u,v))$.

On this basis, we model a weight increment as a vector v, having $|v|$ components. For any weight increment v, a component $v[i]$ corresponds to the weight increment applied to the i-th outgoing link. Vectors of the same size can be compared, and partial order relationships can be defined between them. In particular, we say that two vectors v_1 and v_2 of size $k \geq 0$ are equal, i.e., $v_1 = v_2$, if $\forall i \in \{1, \ldots, k\}$ $v_1[i] = v_2[i]$. Similarly, $>, \geq, <, \leq$ relationships, hold on vectors if they hold on all the corresponding components. In addition, given two vectors v_1 and v_2 (such that $|v_1| = |v_2| = k$), we say that v_1 is *positively greater* than v_2, denoted $v_1 >^+ v_2$ if $\forall\ i \in \{1, \ldots, k\}$

$$\begin{cases} v_1[i] > v_2[i] & (\text{if } \ v_2[i] \in \mathbb{N}) \\ v_1[i] \geq 0 & (\text{if } v_2[i] \in \mathbb{Z}_{<0}) \end{cases} \qquad \text{(Eq. II.5)}$$

From Loops to Necessary and Sufficient Conditions We now define the concept of loop-constraint to formalize property of the weight increment sequence that must hold to avoid transient loops. More precisely, we define *loop-constraint*, or simply *constraint*, as the weight increment interval associated to a single loop. For any given transient loop L, a loop-constraint l is a vector pair $l := (\underline{l}, \bar{l})$. Vectors $\underline{l}$ and $\bar{l}$ have one component per outgoing link of router 0 (i.e., $|\underline{l}| = |\bar{l}| = k$ with k is the degree of router 0), and respectively represent the set of minimum and maximal weight increments that *prevents* L. To compute numerical values of loop-constraints, we rely on *delta vectors* Δ. Given a router $x \neq 0$ and a destination d, $\Delta_d(x)$ is the vector of weight increments such that the shortest paths from x to d include both the initial and final paths (as computed in G and G', resp.). Let $C'(x,d)$ be the cost of the shortest paths from x to d in G', l_i be the i-th link outgoing from 0, and $C(x, l_i, d)$ be the cost of the shortest path (without circuits) from x to d via l_i in G. By definition,

$$\Delta_d(x)[i] = C'(x,d) - C(x, l_i, d)$$

Let $\vec{0}$ be the all-zero vector. Then, the loop-constraint l associated to a loop L to a destination d is defined as

$$l := (\underline{l} := \min_{\forall x \in L}(\Delta_d(x)), \bar{l} := \max_{\forall x \in L}(\Delta_d(x)))$$

Note that, for each destination d, the set of vectors $\Delta_d(x) \forall x \in N$ is totally ordered. Indeed, for any router x, we have $C(x, l_i, d) = C(0, l_i, d) - C(0, d) + C(x, d)$. This implies that each x has the same offset among its $\Delta_d(x)$ components for a given destination d. Moreover, note that $\Delta_d(x) = \vec{0}$ may imply an ECMP case on x potentially leading to an actual constraint.

By definition of Δ, the vector v_x verifying $\forall\ i \in \{1, \ldots, k\},\ v_x[i] = \max(\Delta_d(x)[i] + 1, 0)$ is the smallest set of increments to be configured on the outgoing interfaces of router 0, such that router x switches to its final state and no longer uses 0 to reach d. Hence, in order to satisfy a loop-constraint l such that $\underline{l} = \Delta_d(z)$ and $\bar{l} = \Delta_d(y)$, an intermediate vector v must be positively greater than $\Delta_d(z)$, but not greater than or equal to $\Delta_d(y)$. Besides, if $v_x[i] = 0$ the weight of l_i can be arbitrarily increased. The distance of shortest paths from the node y to d verifying $\bar{l} = \Delta_d(y)$ is strictly shorter than the ones using l_i.

As an example of Δ and constraint vectors, consider again Fig. II.5. In this figure, $\Delta_4(c) = (4\ 2\ 4\ -2)$ and $\Delta_4(d) = (2\ 0\ 2\ -4)$, where components respectively map to links $(0,1)$, $(0,2)$, $(0,3)$, and $(0,c)$. As an illustration, $\Delta_4(c)[1] = 4$ since $C'(c,4) = 11$ and $C(c,(0,1),4) = 7$. Adding 4 to the weight of link $(0,1)$ would make the path from c to 4 through $(0,1)$ as long as its final ones. Similar computations are applied to the other components of $\Delta_4(c)$ and $\Delta_4(d)$. According to those computations, forwarding paths from c (resp., d) are then ensured not to include 0 when weight increments greater than $\Delta_4(c)$ (resp., $\Delta_4(d)$) are applied to the outgoing links from 0. Moreover, the constraint l associated to the loop L between c and d is formalized as $l = (\underline{l} = \Delta_4(d), \bar{l} = \Delta_4(c))$.

By definition of l, applying weight increments positively greater than $\underline{l}$ (resp. $\bar{l}$) will cause the shortest paths from at least one router (resp. all the routers) in L not to traverse 0 anymore. In the previous example, applying a weight increment positively greater than $\underline{l} = \Delta_4(d)$ will cause d, but not necessarily c, to switch to its final shortest paths. Both c and d are guaranteed to switch to their respective final paths when the weight increments is positively greater than $\bar{l} = \Delta_4(c)$. To provably avoid a transient loop, we must then force weight increments changing only to forwarding paths of d, e.g. a relative increase of (3 1 3 0), before applying the final weights.

To formally state the problem of finding such intermediate weight increments, we introduce the following terminology. We say that a weight increment v ***meets*** a constraint $(\underline{l}, \bar{l})$ if $v >^+ \underline{l}$ and $\exists\ i \in \{1, \ldots, k\} \mid v[i] < \bar{l}[i]$. We also say that a weight increment v ***precedes*** a constraint l if $\exists\ i \in \{1, \ldots, k\} \mid v[i] \leq \underline{l}[i] \neq 0$, and that v ***follows*** l if $\forall\ i \in \{1, \ldots, k\} \mid v[i] \geq \bar{l}[i]$. Given a constraint l and a sequence of weight increments $\{v_0, \ldots, v_n\}$ with $v_0 = \vec{0}$ (initial state of node 0) and v_n containing all ∞ (as after the removal of node 0), a pair of consecutive vectors v_i and v_{i+1} constitutes an *unsafe transition* if either i) v_i precedes $\underline{l}$ and v_{i+1} follows $\bar{l}$; or ii) v_i follows $\bar{l}$ and v_{i+1} precedes $\underline{l}$. Trivially, a pair of consecutive vectors is said to form a ***safe transition*** with respect to a given constraint if it is not unsafe. In the previous example, the sequence of relative increments $\{\vec{0}, \vec{\infty}\}$ contains an unsafe transition for the constraint $l = \{(2\ 0\ 2\ 0), (4\ 2\ 4\ 0)\}$. On the contrary, both transitions in $\{\vec{0}, (3\ 1\ 3\ 0), \vec{\infty}\}$ are safe with respect to l since the second vector (3 1 3 0) meets l. Note that *MAX_METRIC* can be used in practice to enforce the final state $\vec{\infty}$.

A *safe sequence* contains safe transitions for all loop-constraints and the following theorem holds.

Theorem 3. *A weight sequence s avoids a loop L if and only if s contains only safe transitions with respect to the constraint corresponding to L.*

Theorem 3 implies that, for each constraint $(\underline{l}, \bar{l})$, at least one vector must meet the constraint for each transition from weight increments smaller than $\underline{l}$ to those greater than $\bar{l}$, and vice versa. Intuitively, *always increasing* sequences seem to be the natural candidate for targeting minimality. A sequence $s = \{v_0, \ldots, v_m\}$ is said *always increasing* if $\forall\ i \in \{1, \ldots, m\},\ v_{i-1} \leq v_i$. Each sequence step meets a given subset of constraints cumulatively. A simplified version of Theorem 3 holds for always increasing sequences.

Theorem 4. *An always increasing weight sequence s avoids a loop L if and only if s contains at least one vector meeting the constraint corresponding to L.*

In this section, we study the problem of finding minimal safe sequences with respect to all constraints. In particular, we present algorithms to compute always increasing sequences, that are provably safe and minimal (in the restricted context of globally positive updates). This implies that looking only at always increasing sequences does not limit our ability to optimally solve the safe router update problem. In other words, for any network and for any router removal, at least one minimal safe sequence is always increasing. The minimality holds because we restrict the problem to the case of globally positive sequences that explore only the positive space regarding the initial outgoing weights of 0. Indeed, as we will discuss later, it may exist shorter non strictly positive (or negative) sequences at the cost of possible intermediate changes and loops. In the conclusion of the section, we discuss more general and challenging problems and possible extensions.

II.2.b Towards an Efficient Minimal Solution

Now that I have introduced the necessary background, let me show how we solve these problems in practice. After having briefly introduced the baseline of all our solutions, we will focus on the most elegant of them, AGBA – this is the one optimally solving the more general problem with minimal assumptions (i.e. we do not consider μloop delay).

II.2.b.1 The Baseline: the Greedy Backward Algorithm

Let me first assume that the μloop delay feature is applied to the updated router 0 to showcase the baseline of all our solutions, *GBA*.

In this case, all the transient loops to be prevented can be extracted from the union of the initial and final RSPDAGs, since 0 postpones its next-hops changes. They are called ***static constraints***. In practice, at each transition, 0 freezes its own convergence during a delay sufficiently large to ensure the convergence of its neighbors. It allows to avoid intermediate transient loops, that are equivalently local to 0, but not forwarding changes in general (as illustrated in the next section). Hence, the problem of computing minimal safe sequences can be formulated as follows.

Problem 2. *Minimal Loop-free Problem (MLF)*
Given a set $cs = \{(\underline{l}_1, \bar{l}_1), \ldots, (\underline{l}_n, \bar{l}_n)\}$ *of static loop-constraints (resulting from a graph* G*, a source* s *and a failed node* 0 *towards all destinations in* V*), compute a minimal weight increment sequence which contains no unsafe transition for any constraint in* cs*.*

From a high-level perspective, *GBA* iteratively performs the following macro-steps:

I- Extract the *largest* constraint corresponding to a potential transient loop for each destination;

II- Backwardly compute a greedy weight increment that meets the maximum of extracted lower bound of constraints and update the set of constraints still to be met.

GBA stops when all the constraints are met.

Our algorithm is destination oriented, in the sense that it extracts constraints for each destination independently, from the merged DAG (*mdag*) obtained as the union of the initial (*RSPDAG*) and final (*RSPDAG'*) RSPDAGs. Before each intermediate vector computation, *GBA* only extracts the *last constraint* for each destination, i.e. the largest lower bound among the constraints associated to a destination. Second, *GBA* computes weight increments in a backward fashion, i.e. in the opposite order with respect to how they are to be applied. Using such a reverse order makes it possible to greedily build an update sequence of minimal length, as proved in [CMV+13]. Note that a greedy forward-based exploration of weight increments does not ensure minimality of the resulting sequence. This significant difference with previous works on graceful link operations [CMP+14] is due to an asymmetry in the way constraints may be satisfied: a vector v meets a constraint $(\underline{l}, \bar{l})$ if and only if $v >^+ \underline{l}$ and $v \not\geq \bar{l}$. While the first condition is a direct transposition of the scalar $>$, requiring each value in v to be greater than the value on same index in $\underline{l}$, the second condition allows all values but one to be greater than or equal to $\bar{l}$. The upper bound is far less restrictive than the lower one.

Before all, *GBA* starts by computing the set of *affected destinations* as the nodes that are reached through 0 by at least one source (other than 0 itself). Indeed, if node 0 is not used by any source to reach a given destination, no transient loop could appear for that destination. Then, for each affected destination d, our algorithm computes $RSPDAG(d)$, the initial forwarding graph towards d, while marking as SRC the subset of source nodes reaching d through 0. This subset makes it possible to avoid useless calculations: *GBA* only focuses on the subgraph that may evolve due to the removal of node 0. Thus, the merged graph $mdag(d)$, on which *GBA* detects cycles and their associated constraints, is computed as follows: $mdag(d) = G(SRC(d), E(RSPDAG(d) \bigcup RSPDAG'(d)) \bigcap (SRC(d) \times SRC(d))$. Δ values are then computed and associated to each node in $mdag(d)$. At this stage, a function checks whether transient loops could appear and, if so, computes the *last constraint* – without enumerating on circuits. If such a constraint exists, an offset value is then computed for each outgoing link of node 0. Otherwise, it means that no transient loop could possibly appear for this destination. This offset value reflects the *unattractiveness* of a link, and is equal to the difference of distance towards d through the associated link. Formally, we define $offset[d][x] = C(0, x, d) - C(0, d)$, where $C(0, x, d)$ represents the cost of the shortest elementary path in G from 0 to d through each successor x of 0. In the algorithm, we generalize for each node n in N (to provide sequences for all nodes $n = 0$). The purpose of such an offset is to avoid manipulating vectors when not necessary. Indeed, performing destination oriented operations does not require it since a total order exists among Δ for nodes in SRC. Eventually, the $mdag(d)$ is added to the global *MDags* set.

Once the *MDags* set is computed, our algorithm enters the second phase. At each round of the global loop, a new greedy vector v is computed (and added to the sequence S) as the smallest one that is safe with respect to the *last constraint* for all subgraphs in the *MDags* set. Then, for each destination d, the actual distance update m associated to this vector is computed in order to make $mdag(d)$ evolve

Destination 1	$c_1 = \{a, b, a\}$	$S_1 =$	$\begin{pmatrix} 7 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 \end{pmatrix}$,
	$c_5 = \{b, c, c\}$		
Destination 2	$c_2 = \{c, d, c\}$	$S_2 =$	$\begin{pmatrix} 0 & 10 & 8 & 0 \end{pmatrix}$
Destination 3	$c_3 = \{c, d, c\}$	$S_3 =$	$\begin{pmatrix} 0 & 6 & 8 & 0 \end{pmatrix}$
Destination 4	$c_4 = \{c, d, c\}$	$S_4 =$	$\begin{pmatrix} 3 & 1 & 3 & 0 \end{pmatrix}$

GBA	
$c_1, c_4 \rightarrow c_2, c_3, c_5$	$S_{GBA} = \begin{pmatrix} 7 \\ 1 \\ 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 9 \\ 10 \\ 8 \\ 0 \end{pmatrix}$

Table II.5: Destination and global sequences for the removal of node 0

accordingly. Note that a preliminary check is performed to know whether v could have an impact on $mdag(d)$. If m is not lower than the maximum Δ value among the nodes in $mdag(d)$, no constraint could have possibly been satisfied for d, so that it is not necessary to compute anything more for this destination. On the other hand, if m is lower than at least one Δ value in $mdag(d)$, another function is called (at least one node is impacted). This function modifies the graph, now considering v as the final weight assignment, and prunes all nodes that cannot be involved in any cycle. It then extracts the new *last constraint*, if any, and returns 0 otherwise. If there are no more constraints to be satisfied for this destination, it is removed from the *MDags* set. The main loop iterates this way until *MDags* is empty, meaning that all constraints are satisfied by the sequence S.

Table II.5 gives the sequences obtained by running *GBA* on the graph described on Fig. II.5, for each *affected destination* separately, and the global one. In this case, our algorithm provides a sequence that satisfies all loop constraints with only two intermediate updates.

With such a design, the following theorem holds:

Theorem 5. *GBA solves the MLF problem*
Given a graph $G = (V, E)$, GBA allows any node $s \in V$ to compute safe and minimal sequence of increments for any destination $s \in V$ for any link $\in E$ or node $\in V$ updates.

Formal properties and proofs demonstrating the safety and minimality of *GBA* are provided (and generalized for *AGBA*) in [Clad, 2014] and [CVM+15]. We also provide all algorithmic details for the internal procedures. There exist several ways to efficiently implement *GBA*, which can be tuned for a particular deployment: inside a router or in a management tool. While minimizing the worst case complexity appears to be the main goal in the former case, the average complexity becomes prevalent when considering the latter. We have implemented many "pruning processes" that reduce the number and the size of the graphs to be considered. These techniques (more detailed in [CVM+15]) limit graph manipulations to the strict necessary.

At the microscopic view, the core component of *GBA* we use is a circuit detection algorithm. It allows for initializing the constraint system and to extract the *last constraints* at each iteration of the main loop. This algorithm helps at two levels: first, it gives *GBA* the ability to definitively remove non relevant nodes and edges as soon as a given weight assignment removes them from the constraint system; second, it can be repeatedly applied on a clone of the remaining graph in order to extract new *last constraints*. This constraint extraction mechanism has a complexity of $|E|$ and is never called more than once for each node in an *mdag*.
Each procedure of *GBA* comes with a specific complexity:

- Last constraints extraction has a cost of $O(|V| \times |E|)$. Note that the *RSPDAG* computation has a complexity of $O(|V| \times (|V|ln(|V|) + |E|))$;
- The number of iterations of the main loop can be limited to a given length parameter $p << |V|^2$ (p being the targeted maximal sequence size). Inside the loop we have:

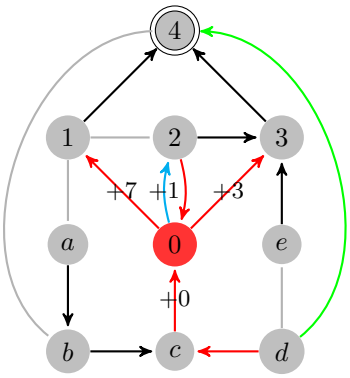


Figure II.6: Illustration of intermediate disruptions for destination 4.

- Vector manipulations for all destinations with a complexity of $pk|V|$ (k being the degree of node 0);
- The constraints update comes at a cost of $O(min(p \times |V| \times |E|), |V^2| \times |E|))$ for all destinations.

Eventually, *GBA* has a worst case complexity in $O(|V|^4)$ if node 0 has a degree of $k = |V|$ (or if $|E| \approx |V|^2$ in general). However, in practice it is worth noticing that p can be picked as an arbitrary low value such as $p \leq 5$ to limit the complexity of *GBA* to $O(|V|^3)$.

II.2.b.2 A Multi-Dimensionnal Problem with Extras Challenges

Applying non-uniform weight modifications on the outgoing links of node 0 allows for minimizing the length of an increment sequence. Indeed, using different weight increments over several outgoing links of 0 can help to satisfy a subset of constraints for different destinations in the same update step. In some cases, there is no equivalent uniform weight increase step. Unfortunately, such modifications can introduce new intermediate disruptions, namely intermediate forwarding changes and intermediate transient loops around 0.

In the following, we illustrate and describe those disruptions and how to deal with them within the *GBA* algorithm. Our extension to *GBA*, Adjusted Greedy Backward Algorithm (*AGBA*), computes provably minimal sequences preventing all kinds of intermediate disruptions. Interested readers can find our algorithmic alternative to the μloop delay feature, called *DGBH*, in the original paper. In [Holterbach, 2014], we also study how preventing all intermediate transient loops at the cost of slightly longer sequences.

Fig. II.6 depicts the shortest paths on the network in Fig. II.5 when applying the first vector, $(7, 1, 3, 0)$, computed by *GBA* for the removal of node 0. Aside from forcing node d to shift to its final path, this weight increment also makes 0 update its shortest paths to 4. More precisely, 0 starts using nodes 2 and 3 instead of 1 and 3 as next-hops, and forwarding traffic on path (0 2 3 4), that it does not use either in G or in G'. Note that, contrary to final paths that are expected to be used after the modification, such an intermediate path may not be sufficiently provisioned, hence leading to congestion. In this example, node 3 may act as a bottleneck on the paths used by 0 to 4, which are no longer disjoint. Even worse, a transient loop can occur between 0 and 2, since 2 was initially using 0, as highlighted by the red arrow from 2 to 0.

We will refer as *intermediate forwarding change* to any set of forwarding paths used by 0 after the application of weight increments and not coinciding with both its initial and final set of paths. Beyond increasing the risk of congestion, intermediate forwarding changes can translate to experiencing multiple temporary paths between some source-destination pairs before stabilizing on the final ones. Depending on the latency of each intermediate path with respect to the initial and final ones, this may increase the probability of out-of-order packet delivery, delay and TTL variations during the IGP convergence. All those variations may have a negative impact on control mechanisms implemented at the transport layer.

Intermediate forwarding changes can cause *intermediate transient loops*, as the loop between 0 and 2 in the example in Fig. II.6. Those loops depend on the shortest paths on intermediate forwarding

graphs obtained by applying non-uniform weight increments. As such, they do not correspond to cycles in the graph $MP(d)$. Note that these loops always include node 0 and induce two complications. First, intermediate transient loops are not captured by *GBA*, as shown by the example in Fig. II.6. Second, they map to *dynamic constraints* depending on the increment sequence itself (as opposed to the *GBA* constraints that can be computed through a static analysis on the initial and final RSPDAGs).

II.2.b.3 Avoiding Intermediate Forwarding Changes and Loops with *AGBA*

Preventing Changes: Maintain Initial Successors with a Linear System Since the root cause of intermediate next-hops leading to loops and new forwarding paths is induced by forwarding changes on node 0, a sufficient and necessary condition to avoid any intermediate edge consists in enforcing that 0 *maintains its next-hops* throughout the IGP convergence. We denote the component of a vector v associated to a link $(0, x)$ as $v[x]$.

Definition 1. *A node s is called **initial successor** of 0 to d if $(0, s)$ is the first edge of a path in $RSPDAG(d, G)$. We denote the set of initial successors of 0 to d as $S^*(d)$.*

Initial successors are next-hops used by 0 to reach d in G, In the example in Fig. II.6, nodes 1 and 3 are initial successors of 0 for destination 4, while 2 and c are not.

Definition 2. *Change Prevention Conditions (CPCs)*
Let d be a destination, s^* be an initial successor of 0 to d, and v be a weight increment. We define the intermediate forwarding Change Prevention Conditions as the set of inequalities

$$v[s] = v[s^*]$$

$$v[x] > v[s^*] - offset[d][x]$$

for each initial successor $s \in S^*(d)$ of 0, and for each other neighbor x of 0 such that $x \notin S^*(d)$.

As an illustration, consider again Fig. II.6 and let $s^* = 1$. The CPCs for destination 4 consists of inequalities $v[1] < v[2] + 2$ and $v[1] = v[3]$. Observe that CPCs are formulated with respect to a single initial successor (i.e., 1 in the example above). However, the correctness of the CPCs does not depend on the considered initial successor.

Moreover, for each neighbor $x \notin S^*(d)$, it must be $C(0, d) < C(0, x, d)$ by definition of initial successors. Hence, $offset[d][x] > 0$, and the following property holds.

Property 4. *Any CPC inequality can be written as $v[s^*] \leq v[x] + m$, with $m \geq 0$.*

Such CPCs impose that, for a given destination, paths via initial successors of 0 should be shorter than any other paths via a non initial successor (i.e., we aim to adjust their increments such that they do not results in intermediate shortest paths). Hence, verifying CPCs for a destination d guarantees that the shortest paths from 0 to d remain the same. This implies the following theorem (whose proof is reported in [CVM+15]).

Theorem 6. *If a weight increment v satisfies the CPCs for all destinations, no forwarding change occurs when v is applied.*

Since intermediate transient loops cannot occur in the absence of forwarding changes, the following corollary holds.

Corollary 4. *If a weight increment v satisfies the CPCs for all destinations, no intermediate transient loop occurs.*

Our *GBA* generalization, called *AGBA*, guarantees prevention of intermediate edges by enforcing accommodation of CPCs for all network destinations, it solves the following problem.

Problem 3. *Minimal intermediate Change-free and Loop-free Problem (MCLP)*
Given a set $\mathcal{C}$ of loop-constraints and a set $\mathcal{A}$ of CPCs, compute a minimal weight increment sequence that contains no unsafe transition for any constraint in $\mathcal{C}$, and no weight increment that violate any condition in $\mathcal{A}$.

Provided that all the loop-constraints and the CPCs are correctly enumerated, solving an MCLP instance implies preventing all possible convergence loops and forwarding changes in the corresponding network as per Theorems 3 and 6.

Avoiding Intermediate Changes with AGBA To solve the MCLP problem, at each iteration, *AGBA* post-process each weight increment gv as computed by *GBA*. To this end, *AGBA* adds two main algorithmic steps to each iteration of *GBA*. One in its initialization, the other within the main loop iteration to adjust the greedy vector.

First, *AGBA* computes every offset values and optimizes them across all destinations. In particular, for each destination, it computes all the offsets and identifies the initial successors. Moreover, for each pair initial successor and neighbor of 0, it only keeps the smallest offset, as it corresponds to the most constraining CPCs.

Second, *AGBA* modifies the greedy vector gv as computed by *GBA*, applying the following operations: 1) *vector sorting*, in which the components of gv are considered from the biggest to the smallest one (this corresponds to consider all the CPCs in decreasing order). The goal is to retrieve the up to date pivot component p;
and 2) *vector adjusting*, in which the current component of gv is modified to satisfy all the sorted CPCs. *AGBA* enforces the CPCs by imposing:

$$v[s] = m_d$$

$$v[x] = m_d - offset[d][x] + 1$$

where $s \in S^*(d)$, $x \notin S^*(d)$, and $m_d = max_{\forall s \in S^*(d)}(v[s])$. That is, given a weight increment, *AGBA* calculates the maximum component corresponding to an initial successor, which we call *pivot component*, and imposes that all the other components of the vector must enforce the CPCs with respect to such a pivot component. Consider again the example in Fig. II.6. The pivot component of the shown weight increment v is $v[1]$ and $m_4 = 7$. *AGBA* imposes that $v[1] = v[3] = 7$, $v[2] = 6$ and $v[c] = 2$. Eventually, the complete sequence computed by *AGBA* on the network in the figure is

$$S_{AGBA} = \left\{ \begin{pmatrix} 3 \\ 2 \\ 3 \\ 3 \end{pmatrix}, \begin{pmatrix} 7 \\ 6 \\ 7 \\ 7 \end{pmatrix}, \begin{pmatrix} 8 \\ 7 \\ 8 \\ 8 \end{pmatrix}, \begin{pmatrix} 9 \\ 10 \\ 9 \\ 9 \end{pmatrix} \right\}$$

which is two increments longer than the *GBA* one but one lower than the uniform sequence $\{3, 7, 8, 9, 10\}$. While it may appear as a large sequence increase in practice, our experiments have shown that the sequence increase is not significant in many realistic cases.

To conclude, the following theorems hold: *AGBA* finds minimal increment sequences solving the MCLP problem. Proofs are included in the appendix of the original paper [CVM+15].

Theorem 7. *For any MCLP instance $I =< \mathcal{C}, \mathcal{A} >$, AGBA always terminates in $O(|\mathcal{C}|)$ iterations.*

Theorem 8. *Correctness of AGBA*
The weight sequences computed by AGBA prevent both transient loops and forwarding changes.

Theorem 9. *Minimality of AGBA*
AGBA computes a minimal sequence solving any given MCLP instance.

Intuitively, *AGBA* is correct and optimal because CPC constraints are statics in the same vein as transient loops ones. The greedy behavior of *GBA* is then still ensured with respect to an additional kind of static constraints, i.e., the "minimal" resolution of a linear inequation system.

Avoiding Intermediate Loops with DGBH While *AGBA* enforces strong consistency guarantees during IGP convergence, this may increase the sequence length. But one can explore a different trade-off between consistency guarantees and sequence length. In particular, we investigate an algorithm that prevents transient loops but not necessarily intermediate forwarding changes, i.e., solving the following problem.

Problem 4. *Minimal Intermediate Loop-free Problem (MILP)*
Given a set $\mathcal{C}$ of loop-constraints, compute a minimal weight increment safe sequence that does not result in any intermediate transient loop on 0.

Since the MILP problem allows 0 to change its forwarding paths during the application of the increment sequence, we now face dynamic loop constraints, i.e., depending on the sequence being computed. In order to deal with those constraints, our greedy heuristic, called *DGBH*, potentially adds loop constraints at each iteration.

Theorem 10. *Correctness of DGBH*
The weight sequences computed by DGBH are sufficient to solve the MILP problem as they prevent intermediate transient loops (but not forwarding changes).

While sequences computed by *DGBH* are correct, they are not guaranteed to be minimal. Consider again the network in Fig. II.5. *DGBH* computes the following sequence.

$$S_{DGBH} = \left\{ \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \\ 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 7 \\ 3 \\ 5 \\ 0 \end{pmatrix}, \begin{pmatrix} 9 \\ 10 \\ 8 \\ 9 \end{pmatrix} \right\}$$

Vector (1 0 1 0) avoids the intermediate loop between 0 and 2. The given sequence is not minimal with respect to MILP. Indeed, consider the case in which (3 3 3 0) is used as second vector, preventing 0 to change its path (using an approach similar to the *AGBA* one). This vector would have prevented both the loop between c and d, and the intermediate one between 0 and 2, hence leading to a loop-free sequence with 3 metric increments. Nevertheless, adjusting vectors to prevent 0 to change its path forces some solutions of the MILP problem to be discarded, which can in turn lead to non-minimal sequences in other cases. Even worse, such adjustments can induce new intermediate loops while preventing some others.

Generally speaking, from a *GBA*-based perspective, two strategies can be adopted to prevent intermediate loops, namely, 1) modify the current greedy vector to avoid the intermediate change at 0; OR 2) add a constraint to the computation of the next greedy vector, to force another node participating in the loop to not use 0 before it switches. Unfortunately, none of the two strategies always leads to minimal sequences when applied independently. While the presence of alternative strategies at each step seems to force a combinatorial space exploration, the theoretical problem of efficiently solving MILP is left open. Fortunately, our evaluation have shown that a heuristic based only on the second strategy, i.e., *DGBH*, computes sequences as short as *GBA* in the vast majority of our experiments.

Title of the publication	Name of the venue	Year	Reference
Graceful Router Updates in Link-state Protocols	International Conference on Network Protocols (**ICNP**)	2013	[CMV+13]
Graceful Convergence in Link-State IP Networks A Lightweight Algorithm Ensuring Minimal Operational Impact	Transactions on Networking (**ToN**)	2014	[CMP+14]
Computing Minimal Update Sequences for Graceful Router-Wide Reconfigurations	Transactions on Networking (**ToN**)	2015	[CVM+15]

Table II.6: Summary of my publications related to the graceful convergence of IP networks

Tale II.6 summarizes my publication activities in this field. I now envision to propose new techniques dealing both with fast-rerouting and graceful convergence.

AGBA: Conclusions and Perspectives

In summary, in this second section I have described techniques to support graceful router updates in link-state routing protocols. They do not require changes to IGP specifications and are based on efficient algorithms finding minimal sequences of link weight increments that avoids transient forwarding loops. While we focused on router removal, our techniques can also address other use cases, like router addition (by applying the computed sequence in the reverse order), and arbitrary sets of weight increase xor[40] decrease on links maintained by a given router (by applying part of the sequences computed by our algorithms).

Our baseline algorithm, *GBA*, is correct and optimal when used in conjunction with local delay [208]. We then introduced *AGBA*, a generalized version of *GBA* that computes minimal sequences avoiding the use of intermediate paths and, as such, does not require local delay. In the original paper, we showed the practicality effectiveness of our algorithms. Even on large Tier-1 networks, they need few seconds to compute safe weight increment sequences, which are likely shorter than 5 steps. Such time efficiency indicates the possibility of including our algorithms in current routers' software. We can indeed conclude that our algorithms generally produce sequences of limited length, regardless of the *GBA* variation, in a much reasonable time. These results encourage us to consider the use of this solution on production routers. In the case of planned operations, transient inconsistencies could be avoided at the cost of a slightly extended convergence time.

However, there still exists several possible improvements and perspectives. First, we discussed a heuristic, *DGBH*, aimed at providing shorter sequences than *AGBA* while still not requiring local delay [208]to freeze the data-plane of the node to be shut. It comes at the cost of intermediate changes (on the contrary to *AGBA*), but targets safe shortest sequences than with it (*AGBA*) and greater or equal than with *GBA* and local delay. Although this method is correct, it is not provably optimal in this category. Even the complexity of this specific problem is unknown and left open, it may lead to a NP-hard challenge. Second, we only discuss the case of strictly increasing and positive increments while it is possible to look at negative increments that may help to satisfy more constraints within a single step, in particular at the beginning of the sequence. This could indeed allow for additional constraint combinations, thus leading to shorter sequences. This may look even more promising when applying arbitrary changes on link weights for TE optimization strategies requiring to both increase and decrease weights simultaneously (as our existing approaches are not proven optimal in terms of sequence length for this general case). However, it might not be realistic nor interesting from a practical networking perspective, as it would imply rerouting traffic through links that may not be provisioned enough, and possibly even attract more traffic into a router that one aims to shut down. Besides, we already have shown that modifying the set of next-hops used by the modified router may lead to another kind of inconsistencies and intermediary changes in general.

On a theoretical perspective, and especially if intermediate forwarding changes have a negligible impact on network performances, future works could take interest in formally studying the complexity of the Minimal Intermediate Loop-free Problem (MILP) and, if MILP is in P, finding a P-time algorithm to

[40] Indeed, our method is not provably optimal for usecases applying both weight increase and decrease at the same time (for outgoing links of a given router).

optimally solve it. Long-term objectives to provide ever shorter sequences would also include investigating the opportunity of performing weight updates of opposite sign to the intended modification. For example, considering link weight decrements before applying an always increasing sequence may, in certain cases, enable to reduce the overall sequence length. In the same spirit, it could be possible to perform weight reconfigurations on links farther away from the modified router. Finally, the approach could be further extended to the more general use cases of Shared Risk Link Groups (SRLG) and arbitrary k-links modifications anywhere in the network. Again, the formal analysis of the computational complexity of such minimization problems is left open for future works.

II.3 OPTIC: Optimal Protection Technique for Inter-intra-domain Convergence

In this section, I will present an overview of an ongoing work with Jean-Romain Luttringer (he started his Ph. D in 2019 and should defend in 2022 [Luttringer, 2022]), Quentin Bramas and Cristel Pelsser (who is the director of the thesis). This work started with his internship under my direction during his master thesis [Luttringer, 2019] in 2018/19. Then, we have continued this project, developing its analysis in a publication at INFOCOM [LBPM21]; I will develop here our main contributions and possible extensions. The originality of our proposal is that we tackle a specific kind of routing events for BGP: intra-domain changes leading to BGP re-convergence due to its hot-potato routing design. The following table of content shows the way we will rely on to expose and solve this problem along this section.

II.3.a Avoiding Superfluous BGP Updates in case of IGP events

Objectives and requirements This section introduces the objectives of our solution, *OPTIC* (**Optimal Protection Technique for Inter-Intra domain Convergence**). It is a multi-scale routing scheme minimizing the impact of IGP changes on the BGP convergence, while enforcing both hot- and cold-potato routing. It fulfills two design requirements, effectively making the transit traffic impervious to any IGP event. First, transit traffic is quickly re-routed towards the new optimal BGP route in a time equivalent to the IGP convergence (without running a full BGP decision process). We say that *OPTIC optimally protects* BGP prefixes, meaning that whatever the IGP event, the BGP prefixes are almost immediately reachable again through their best BGP route. Second, the background processing performed to anticipate any next future IGP event is manageable, i.e., at worst similar to BGP for the current event, but negligible when the event does not hamper the bi-connectivity offered with the border routers.

In this work, we aim to address several technical challenges having the following objective:

Research Question

How to Efficiently Ensure a Fast and Optimal BGP Convergence during Intra-domain Routing Changes Affecting the Traffic in Transit?

To achieve such requirements and answer this question, *OPTIC* efficiently computes groups of prefixes that have the same set of current and future optimal reachable gateways[41]. After an IGP event, instead of running the BGP decision process per prefix, OPTIC's convergence relies on a simple walk-through of each group of prefixes (in practice, a simple min-search on lists of distances). To limit the number of routing entries as well as the scale of the updates, the same set of routes is shared in memory by a group of prefixes: updating a shared set thus updates all grouped prefixes. Once the traffic is optimally re-routed, and only if the bi-connectivity among the set of border routers is lost, groups and their associated sets

[41] Since we also consider the failure of an external gateway, the term *gateway* refers to the external gateway of the neighboring AS by default (but can be limited to internal ones when the `next-hop-self` feature is used).

should be updated in background to *anticipate* any *next* future event. Constructing and updating such sets can be done efficiently as it does not require to consider independently each possible IGP event. With a single and simple computation, *OPTIC* encompasses all possible events while keeping sets at a manageable number and size.

We will start by describing the underlying context and current solutions and formally showcase the relationships between BGP and the IGP. Sec. II.3.b introduces the main building blocks of our proposal, from the data structures used to the procedures at play. Sec. II.3.c analyzes the operational advantages of our enhanced data plane scheme.

Operational Context We assume that *OPTIC* has access to a sufficient variety of routes to be able to construct its sets of protected routes. Having a good BGP route diversity is crucial. While route diversity is a well-known challenge [345, 332], several tools and designs exist to mitigate this issue [267, 232, 333].

When designing a network from the ground up with the aim of using *OPTIC*, the most obvious choice is to centralize every route learned by every gateway to a given route server designed especially to store them. This central point can either perform the computations of the routes itself, or relay them to every BGP speaker which would in turn perform the computation. Several BGP architectures exist that centralize external routes such as BMP [300], BGP SDN [131], BGP RPC [65], and containerized or standard route servers [179, 41], onto which *OPTIC* could easily be fitted.

While centralized architectures are achievable today, *OPTIC* can also be deployed incrementally in current distributed architectures. Routes learned by gateways can be gathered at route reflectors. These latter perform the computation of protecting sets and relay it to the gateways. The exchange of messages can be done through BGP Add-path. More precisely, routes can be gathered within the cluster through the use of BGP Add-path's mode *all*. Protecting sets computed through our algorithms can in turn be sent back to the gateway with the *Double AS Wide mode*, which is the best approximation of such sets currently possible through existing tools.

With the two approaches *OPTIC* has all routes to compute optimal backup routes and reach the IGP/BGP separation. The two approaches present different trade-offs in terms of memory used to store the routes and time complexity in computing its protecting sets.

II.3.a.1 Limits of Existing Solutions

Among previous studies enhancing BGP's convergence time, some suggest tuning the BGP Control-Plane through timers [272, 223], ghost-flushing [58], modifying update messages [266], using consistency assertions [68], or limiting path exploration [72]. None of these works prevent superfluous BGP convergence due to IGP events, neither do they allow the optimal protection of external destinations as we aim to do.

Other works reduce the impact of BGP events through data-plane anomaly detection [166, 167, 199] to shorten reaction time. They are however focused on external events and mainly aim to isolate the data-plane from the control-plane.

Closer to our work, BGP PIC [111] aims at mitigating the effect of network failures. PIC is explained more thorougly later on, but at a glance, PIC uses a specifically designed FIB that supports backup routes (usually, a single backup route). The structure storing the optimal and the backup routes can be shared by several prefixes, reducing the update time. PIC relies on a hierarchical FIB (Sec. III.3 provides a model illustrating such an architecture), allowing the transit traffic to benefit from the IGP convergence if an internal event makes the current path to the BGP NH unusable. However, as we will exhibit later on, PIC does not ensure that the transit traffic benefits from the new best BGP NH, nor the protection of the prefixes in all network configurations.

Fast-rerouting requires improved route visibility to learn backup routes, achievable through improved iBGP topologies [267], centralized architectures [131, 65] or BGP extensions. In particular, BGP Add-path [333] allows the exchange of multiple BGP routes. It enables fast re-routing, load-balancing, and reduces the iBGP churn. However, these solutions are mainly control-plane focused. They do not by themselves allow to fully benefit from the potential of the exchanged routes and thus do not guarantee the optimal protection of a destination.

Nevertheless, *OPTIC* could be fitted on top of these control-planes (in particular Add-path with its double IGP wide option) to benefit from better route visibility.

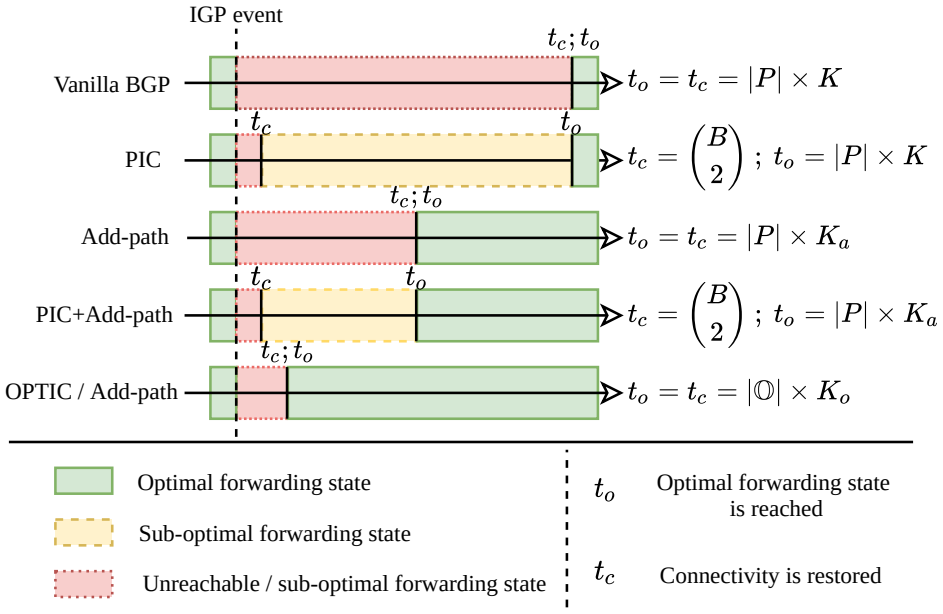


Figure II.7: Connectivity and optimal forwarding state restoration timelines according to different technologies after internal events, depending on the number of prefixes $|P|$ and the number of BGP entries (K_o when using *OPTIC* and K_a when using Add-path).

Fig. II.7 is a pedagogical illustration that does not provide a comprehensive comparison but shows typical cases to position OPTIC's objectives compared to current solutions. Since BGP routers only exchange their best route towards a given prefix, finding the new optimal forwarding states with vanilla BGP often require message exchange if a route becomes unusable. In any case, the router is required to perform a lexicographical comparison on all known routes (K) for each prefix (P).

PIC is designed to restore connectivity quickly by going through each of its sets of two gateways (whose number, for B border routers, can go up to $\binom{B}{2}$) and falling back to the backup route of the set, or by benefiting from the IGP convergence through its hierarchical FIB. However, afterward, finding the new optimal gateway may still require message exchanges and a lexicographical comparison for all prefixes. In worst cases, the set of two gateways is not sufficient to protect the prefix (both gateways are unreachable after the event). In such cases, the connectivity cannot be restored immediately: t_c can be as long as t_o.

Add-path allows to exchange subsets of routes through iBGP. With the double IGP-wide option in particular, the subsets of routes are likely to contain the new optimal gateway after any IGP event. Through adequate configuration, a BGP router can locally find the new optimal forwarding state by running the BGP decision process on the subset of routes sent through Add-path (K_a) for all prefixes $|P|$. This is however not fully guaranteed depending on the network connectivity. To ensure the protection of the prefixes upon any failure, all routes should be exchanged which scales poorly.

By combining PIC and Add-path, one can benefit from the enhanced connectivity restoration time of PIC and the advantages of Add-path. However, PIC and Add-path are not designed as a single entity and their union does not allow to reach the full potential of the available gateways. While PIC can restore connectivity quickly by walking through its $\binom{B}{2}$ sets, the time taken to restore the optimal forwarding state is ultimately the same as the one of Add-path alone. Our *OPTIC* solution can ensure immediate updates with the right data-plane (V.2.b) or at least fast ones, with a control-plane implementation, as all necessary gateways are pre-loaded within few backup shared backup group for most ISP (with respect to the number of BGP prefixes they handle).

OPTIC is designed to fully harness the potential of increased iBGP route visibility. By efficiently ensuring that the pre-computed sets of gateways *always* possess the new optimal path whatever the network configuration, *OPTIC* guarantees a fast switch to the latter (thus, $t_c = t_o$) after a single walk-through of our pre-computed sets of gateways. The number ($|\mathbb{O}|$) and size (K_o) of these sets are both

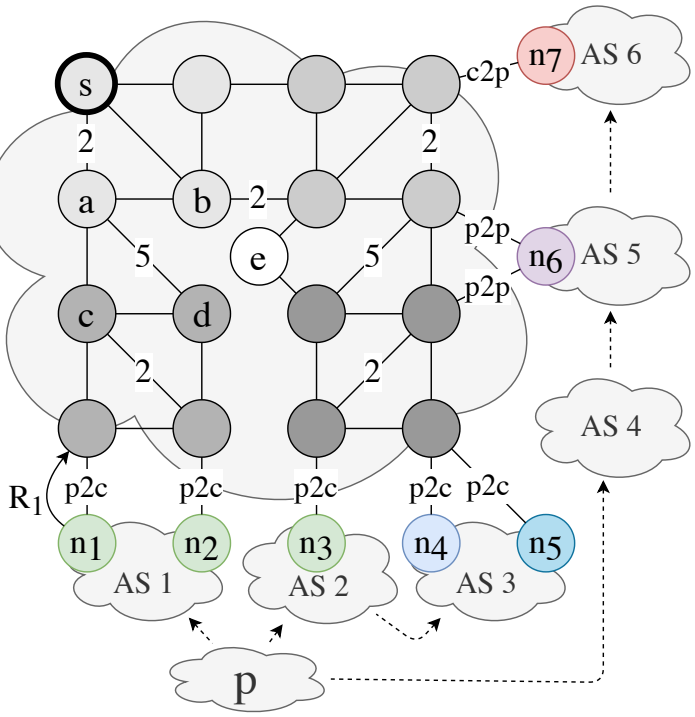


Figure II.8: This example consists of an AS that learns routes towards p via several border routers, focusing on the point of view of s. Each link from an internal border router to the BGP NH is labeled with the type of relation between the two ASes (p2c means provider-to-customer, p2p and c2p, peer-to-peer and customer-to-provider respectively, modeled in practice by a decreasing local preference). A route R_x is advertised by the BGP NH n_x. The routes announced by n_4 and n_5 are discriminated through the MED attribute. Unlabeled edges weight one.

limited, as shown in our evaluation. Restoring connectivity optimally does not require to work at the prefix granularity but at the set granularity instead ($|\mathbb{O}| << |P|$ in practice). In some degraded cases, the sets of gateways may need to be re-computed to handle any *future* IGP event (while the transit traffic already benefits from the new current optimal route). With *OPTIC*, this process does not rely on the slow lexicographical full BGP comparison anymore but rather on efficient updates of gateway structures and prefix groups, which remain stable when the network remains bi-connected after the change.

II.3.a.2 Lexicographically Ordering the Best BGP NH and their Internal Routes

We discuss here why IGP events require the re-convergence of BGP and why current solutions fail to address this challenge. Finally, we propose to reassemble both protocols gracefully.

BGP/IGP: an Intimate Relationship Let us start by showcasing the IGP-BGP interplay, resulting in the need to go through all BGP prefixes after IGP events. BGP routes are characterized by a collection of attributes of decreasing importance that can be locally modified by each router. Each attribute comes into play whenever paths could not be differentiated through the previous one. This route selection is called the *BGP decision process*. Note that the MED differs from the other attributes. It can be used in different ways, but should only be used to compare routes that originate from the same AS, breaking the total order of the decision process. While our solution can be adapted to any MED flavors, we consider this most general one.

When the inter-domain related attributes (local-pref, as-path and MED) of two routes are equal, routers choose the route with the closest exit point in terms of IGP distance (hot-potato routing). This criterion is at the core of the dependency between BGP and the IGP. To exhibit this interplay, we separate the BGP attributes into two sub-lists: β and α. β is composed of the attributes purely related

to inter-domain routing. They usually remain unchanged within an AS but some operators may configure them to change in iBGP [343]. Our work remains valid in both cases. Thus, for the sake of simplicity, we assume they are constant inside an AS. The attributes α, on the other hand, are, by construction, router-dependent and focus on intra-domain routing. Thus, a route R towards a prefix p and advertised by a gateway or BGP next-hop (BGP NH) n is characterized by the vector of attributes $\beta \circ \alpha$, with β = [LP, as-path length, origin, med] and α = [ibgp/ebgp, igp cost, router id]. Since attributes after the IGP cost are simple tie-breaks, and the one before can be seen as an extension to the IGP cost (an eBGP route has an IGP cost of 0), we can refer to α simply as the IGP distance towards the BGP NH, that is the cost $c_1(s, d)$ considering d as the BGP NH (the gateway associated to the route advertised).

It is then clear that IGP events may affect the ranking of BGP routes, for example in Fig. II.8. We state that $R_x \prec R_y$ if R_x is better than R_y according to the BGP decision process. We consider the routes R_1, R_2, R_3 and R_4 towards the prefix p announced by n_1, n_2, n_3 and n_4 respectively. The MED being irrelevant to the point, we consider that these routes have no MED for now. R_4 originates from a client and has an as-path length of 2, leading to the attributes $\beta(R_4)$ = [p2c, 2, -, -]. R_1, R_2 and R_3 are all characterized by the same $\beta(R_1,R_2, R_3)$ = [p2c, 1, -, -] and so are discriminated through their α distances (4 vs 5 vs 6). All have a better β than R_4. Thus, overall, $R_1 \prec R_2 \prec R_3 \prec R_4$ from the point of view of s.

While the inequality $R_1 \prec R_2 \prec R_3$ holds initially, this order is reversed after the failure of the link a→c as the IGP distances, taken into account by BGP, go from 4, 5 and 6 to 9, 8 and 6 respectively. After the failure, $R_3 \prec R_2 \prec R_1$, requiring to wait for the BGP decision process to find the new best route. However, note that inter-domain related attributes (β) are left unaffected by an IGP event, meaning that R_4 will remain less preferred than the other three routers after **any** IGP event in any cases.

Fast BGP re-routing upon IGP Changes Current solutions are not sufficient to guarantee fast BGP convergence when the IGP graph changes. The state-of-the-art solution would be a combination of BGP PIC [111] (implemented on many routers) and BGP Add-Path (for path diversity). PIC relies on the use of a Hierarchical FIB (HFIB). In a standard, *flat* FIB, routers only maintain a direct correspondance between the destination prefix and the local outgoing interface. Because the recursion performed in next-hops computation is flattened (regarding a forwarding model like the one developed in Sec. III.3.a.2), paths must be re-computed from scratch to correct the outgoing interface if needed (i.e., find the new best BGP NH, the corresponding IGP NH and the corresponding interface). With a Hierarchical FIB, routers maintain, for each prefix, a pointer to BGP NHs which in turn points to the IGP NH (instead of simply memorizing the associated outgoing interface). If an IGP event happens, the IGP NH can be changed immediately, impacting all the BGP NH that point to it, leading to a quick update that does not require to go through the recursive next-hop computation. To protect against the failure of the gateway, PIC stores at least the two best current BGP NH, which can be learned through Add-path. These sets of two best BGP NH are shared in memory by prefixes sharing the same best two BGP NH, reducing the scale of the updates. However, PIC only ensures partial sub-optimal protection and requires to run the usual BGP convergence afterward as illustrated in Fig II.8.

First, storing the two best BGP NH is not enough to protect the transit traffic against all failures when the network is poorly connected. Even if both n_1 and n_2 were stored, both become unreachable if a fails (due to the network not being node-biconnected), leading to a loss of connectivity until BGP re-converges and finds the new best available gateway, n_3.

Admittedly, these types of event are fairly rare in well-designed networks. However, less critical events may also lead to issues, in particular regarding forwarding optimality. After the failure of the link a→c, PIC's HFIB restores the connectivity to n_1 by updating the IGP NH used to reach n_1, allowing the transit traffic to go through n_1 again. However, this IGP event leads to a change in the α ranking of some BGP routes. After this failure, R_3 becomes the new best route. Restoring the connectivity to n_1 without considering the changes on α leads to the use of a sub-optimal route until BGP re-converges, violating so the hot-potato routing policy. Besides, the traffic may first be re-directed after the IGP convergence, and then re-directed once again after the BGP convergence, potentially leading to flow disruptions [324].

In both scenarios, retrieving the correct new optimal path requires the BGP decision process which does not scale well[42].

[42]Studies proposed ways to store reduced set of routes to enhance update times [314, 326] but not specifically to deal

Finally, even events that do not impact the connectivity state lead to useless computations. For example, if $s \rightarrow a$ fails, BGP still re-converges, and PIC re-considers its sets of gateways even though the latter are still viable, given the nature of the event.

With *OPTIC*, we aim at dealing with these cases more elegantly and efficiently, by guaranteeing to be able to switch immediately to the new best BGP NH whatever the internal event, for a low maintenance cost and while limiting unnecessary computations.

How to Reach a Symbiotic Coupling? We present here the necessary operational condition to untie the BGP convergence from IGP events. The question to address is: how to efficiently pre-compute the subset composed of every BGP route that may become the new *best* route upon *any* IGP change? We state that prefixes need to be *optimally protected*, as per Definition 3.

Definition 3. *Optimal Protection*
Let p be an external destination. We state that p is ***optimally protected*** *by a set O, if both pre- and post-convergence BGP NHs are stored within O. More precisely, O should verify the two following properties for any IGP change c:*

- *(i) It contains the best BGP NH n towards p before c occurs (pre-convergence NH for p);*
- *(ii) It contains the BGP NH of the new best path towards p after c occurs (post-convergence NH for p). It should be true for any kind of c: link or node event, n included, such as an insertion, deletion or weight update.*

Computing such sets naively may look costly, as predicting the optimal gateway for each specific possible failure and weight change is time-consuming. However, *OPTIC* computes and maintains these sets efficiently by *rounding* them. The size and number of such rounded sets are limited in most cases as we will see. Finding the new optimal gateway among these sets is performed through a simple min-search within each set (with no additional computation), updating so each group of prefixes depending on this set, and thus every prefix. Depending on how such sets of gateways (per group of prefixes) are designed, *OPTIC* can protect the traffic transiting in a BGP network against any link, node, or even SRLG (i.e., links sharing a common fate) single failure.

II.3.b An Efficient Optimal Solution

OPTIC mitigates the impact of IGP events on the BGP convergence without hampering neither hot- or cold- potato routing. It pre-computes sets of gateways bound to contain the current and future *optimal* gateways after any single IGP failure or change, *optimally protecting* every prefix. For the sake of simplicity, we only consider single node and link failures (including the gateway) as well as weight changes, but *OPTIC* can be extended for more general failure scenarios at the cost of complexifying the group management overhead.

The problem we aim to solve can be defined as follows:

Problem 5. *The problem of efficiently handling Optimally Protected Transit Routes (OPTR)*
Given an IGP graph G, a BGP route policy relying on $\prec$ and a source s, efficiently manage and update all optimally protected sets of BGP NH from s towards all prefixes.

The BGP route policy is given with a configuration that denotes preferences (route comparisons with $\prec$) among BGP routes learnt at s (including IGP distances if necessary). It is worth to notice that we look for the most efficient approach both spatially and temporally speaking. In particular, the set of BGP NH for a given prefix, or simply gateways, should ensure the optimal protection but are not required to be the minimal set (the inclusion is a sufficient condition as the equality is not necessary without the minimality requirement).

with IGP changes for transit traffic.

II.3.b.1 Sorting and Rounding BGP Routes: Efficient Data Structures at Play

Optimal protecting sets can be efficiently computed and maintained by sorting and *rounding* BGP routes in a specific way. Let us explain this concept at a high-level before formally detailing our solution to OPTR.

General idea Using the β (inter-domain attributes) and α (IGP distance) attribute separation, we can compute optimal protecting sets easily. Indeed, β is of higher importance than α within the BGP decision process, and IGP events can only affect α, leaving β unchanged. Thus, given the current optimal route, denoted R^{st}, with $\beta(R^{st}) = \beta^{st}$, the new optimal route after an IGP event is among the ones with the same best β^{st} – we simply need to find the one with the new best α. We can then easily avoid predicting which gateway will be the optimal one for a specific event; whatever the IGP event is (except the gateway failure possibly requiring to look for more gateways), the new optimal route is among $\{R \mid \beta(R) = \beta^{st}\}$. We thus create a *rounded* set that includes all routes sharing the same β. After the IGP event, since β attributes are unaffected, we simply need to consider that α may have changed and find within this rounded set the gateway with the lowest α (i.e., with a simple min-search). In Fig. II.8, such a set would be composed of n_1, n_2, n_3 as they share the same (best) β attributes. Indeed, any of these three gateways may become the new optimal gateway.

This is however not sufficient to deal with all failures. In particular, if the first rounded set only contains one gateway, a single failure may render all routes within the set unusable. If this scenario occurs (because there are no two node-disjoint paths towards the external prefix), more gateways are needed to optimally protect the prefix. Since β attributes are unchanged by an internal event, the new best route is, a priori, among the ones with second-to-best β attributes β^{nd}.

To form an optimal protecting set, we add rounded sets of β-tied gateways up until there is enough path diversity to ensure that no single failure may render all of them unreachable (there are two node-disjoint paths between the border routers and prefix p). By never adding less at a time than all gateways sharing the same β, we ensure that the final set contains all potential optimal gateways (as only α can be affected by internal events). This final set (composed of the union of rounded sets) is thus optimally protecting, and the new optimal gateway can be found through a simple walk-through of this set after any IGP event. If two prefixes share an equal optimal protecting set, they belong in the same group and share the same set in memory, reducing both the memory consumption and the number of entries to go through and update upon an event (as covering all shared sets covers all prefixes). In Fig. II.8, n_1, n_2, and n_3 provided enough path-diversity to ensure the prefix was protected, and shared the same best β. Thus, the optimal gateway after any internal event is bound to be one of these three, which then form an optimal protecting set (for all single possible failures).

Our solution requires to re-design both the control- and the data-plane to solve the OPTR problem. The control-plane refers to all learned BGP routes. It is restructured to ease the handling of the routes, their comparison in particular, for efficient computation of optimal protecting sets. The data-plane only contains the information necessary for the optimal forwarding and protection of all prefixes (i.e., the optimal protecting sets). The resulting structures are illustrated in Fig. II.9, which shows how the network depicted in Fig. II.8 would be translated within OPTIC's control-plane (left) and data-plane (right). To better illustrate our data structures, we assume here that n_4 has a better MED than n_5, while other routes do not possess any MED. The next paragraphs describe the control-plane structure, how we construct optimal protecting sets from it, and how they are used in the data-plane.

OPTIC's control-plane At the control-plane level, *OPTIC* stores every BGP routes learned within a sorted prefix-tree referred to as T, whose leaves form an ordered linked-list L, which contains rounded sets of routes sorted by decreasingly preferred β attributes. Both T and L are per-prefix structures. The set of all trees, for all prefixes, is referred to as $\mathbb{T}$. It is important to observe that, since α is not considered, the tree and the list stay stable upon IGP changes and that routes sharing the same β attributes are stored within the same leaf.

This observation implies that when an IGP event occurs, the BGP NH of the new optimal route belongs to the first leaf of the list L that contains at least one reachable gateway. In addition, a route from another leaf cannot be preferred to the routes of the first leaf. **While any route within the first leaf**

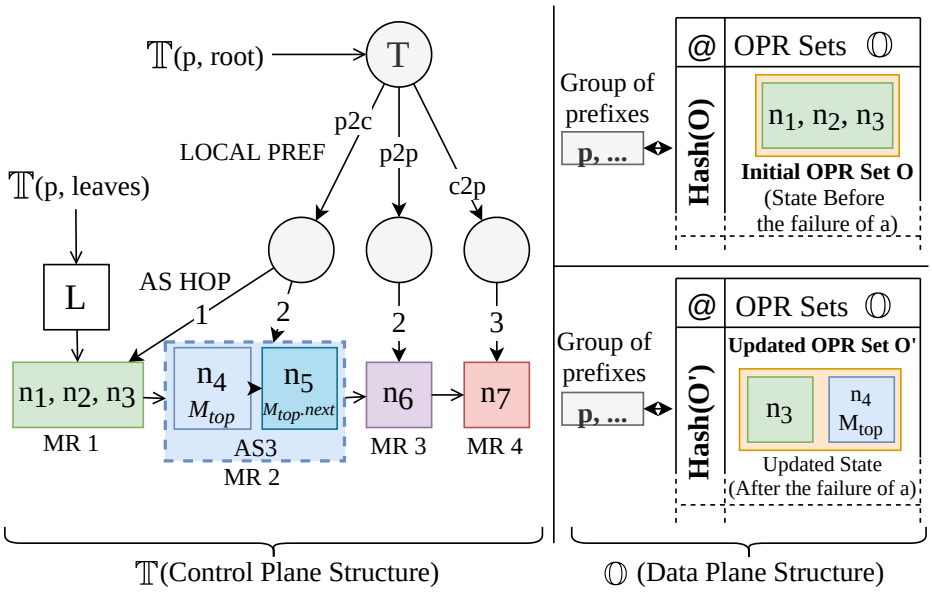


Figure II.9: OPTIC's data-plane and control-plane data structures. In the control-plane, routes are sorted within a prefix tree T whose leaves form a linked list L of structured BGP NH. MED-tied routes from the same AS are chained within a linked list inside their leaf. Only a sufficient optimally protecting subset O of routes is pushed to the data-plane.

can become optimal after an internal change, the order of the leaves themselves can not be modified by an IGP change.

The MED attribute can only be used to compare routes originated from the same AS, hence we cannot use it as a global, generic attribute. One can only consider a route with a greater MED if the route with a better one from the same AS becomes unreachable. Thus, routes discriminated by their MED (*MED-tied* routes) for each AS are stored within a sub-linked-list inside their leaf. This is illustrated in Fig. II.9 with n_4 and n_5. Both BGP NH share the same three first BGP attributes and are thus stored within the same blue leaf of T (MR2). As they originated from the same AS, we store them in a sorted linked list depending on their MED attribute. By doing so, we consider only the first route in the *MED-tied* list that is reachable (referred to as M_{top}), respecting the MED's semantics. Peculiar situations, such as routes not having a MED while others do, can be resolved by applying the standard ISP practices (e.g., ignoring such routes or using a default MED value). The leaves of the tree T thus form a sequence of rounded sets of gateways stable upon IGP changes. We call each set a *MED-aware Rounded set*.

Definition 4. *MED-Aware Rounded sets (MR)*
For a given prefix, a leaf of its prefix tree T is called a MED-Aware Rounded set. In particular, it contains all the routes having the same β attributes (MED-Excluded).

Getting Optimal-protecting Sets from the Control-plane For each prefix p, the first MR set contains, by construction, the optimal pre-convergence route. As stated previously, any BGP NH within the same MR set may offer the new optimal post-convergence route after an internal event. However, this first MR set is not always sufficient to protect p. In this case, the new optimal BGP NH is bound to be within the first MR set in L which contains a gateway that is still reachable. Consequently, *OPTIC* constructs an optimally protecting set by considering the union of the best MR sets, in order, until the destination prefix p is protected from any failure, i.e., there exist two node-disjoint paths. The union of such MR sets is referred to as an Optimal-Protecting Rounded (OPR) set for p and *OPTIC* handles such OPR to solve the OPTR problem. The formal definition is given in Theorem. 11. Its proof is not presented in the paper [LBPM21] but available in [Luttringer, 2019].

Theorem 11. *Optimal-Protecting Rounded sets (**OPR**)*
Let p be a prefix, and $M_1, M_2, \ldots$ be the sequence of MR sets in the list L. Let $O = \bigcup_{i=1}^{x} M_i$ with x minimal such that there exist two node-disjoint paths towards p (passing through O).

***Then, O, called the Optimal-Protecting Rounded set of** p, **is optimally protecting** p .*

Adding MR sets until the prefix p is protected means that there now exists enough path diversity to protect p from any single event. The number of routes necessary to protect a prefix depends on the resilience of the network. In bi-connected networks, two gateways are enough.

OPR sets computation does not require any (prior) knowledge of the IGP graph to cover all possible IGP events. Verifying the existence of two node-disjoint paths between the border router and p via O is enough and the lightest possible processing to test the protection property. Unless the protection property is affected by the event, OPR sets stay stable.

Using OPR sets in the data-plane Once OPR sets are extracted from the control-plane, we push them to the data-plane. The bottom part of Fig. II.9 shows OPTIC's data-plane. For a given prefix, only the OPR set O (and not the whole list L) that optimally protects p is pushed to the data-plane. The data-plane contains the meta-set $\mathbb{O}$ of all OPR sets for all groups of prefixes, indexed by their hash, as shown in Fig. II.9. Prefixes, when sharing the same OPR set, point towards the same set O. The hash index is content-based (see next sections for more details) and eases the management of $\mathbb{O}$. Allowing prefixes to share the same O reduces the amount of data that has to be stored within the data-plane, as well as the scale of the updates. Since O is constructed from a subset of L, prefixes can share the same OPR set O while having different control-plane structures L. Note, however, that in order to share an OPR set, prefixes must not only share the same best gateways but also the same MR set decomposition within the OPR set. Indeed, when retrieving the new best gateway within an OPR set after an internal event, one does not have to consider all the gateways of the OPR set. More precisely, one should only consider the first MR set (i.e., find the gateway with the minimum IGP cost within the first MR set of the OPR). Prefixes share the same OPR set only if they share the same MR-decomposition: simply sharing the same gateways is not sufficient.

To manage OPR sets within the data-plane, we first require a function able to extract, from L, the current OPR set. In short, the required MR sets are computed by first *(i)* creating a graph G' from G where we add a virtual node representing a remote prefix, then *(ii)* connecting in G' the gateways from MR sets, MR per MR to this virtual node, until there exist two node-disjoint paths towards the virtual node. Once this condition is verified, we know that the OPR set should be composed of the union of these MR. Such a function thus returns an OPR set as defined by Theorem 11.

Another way of finding the number of required MR set, more intuitive, would rely on checking the existence of a path from the current node to the destination for any possible failure on the best path towards p. However, this method is far less efficient than the one proposed here and does not allow to deal easily with ECMP. With this function, we can know design algorithms allowing to manage OPTIC's data-plane.

Let me briefly show how the OPR sets are updated in the data-plane when necessary. The optimal protection property may require to add gateways from the data-plane structure O (while removals are performed for efficiency). We start by extracting the OPR set O from the control-plane structure L. We then add the IGP distances towards each gateway (contained within D) to the structure. This is done for each gateway, including MED-tied ones. At the end of this stage, *OPTIC* retrieves the current optimal gateway within the first MR of O, O_{top}, i.e., the one with the lowest IGP distance. Once the OPR set O is updated, we compute its hash to check its existence within $\mathbb{O}$ and insert it if required. Finally, if no prefixes still use the previous O, it is removed from the data-plane. This procedure maintains the data-plane in an optimal-protecting state. Its limited complexity is often bypassed (after the bootstrap), as we expect OPR sets to stay stable in bi-connected networks. The complexity of such a procedure scales linearly in the IGP dimensions. Unused OPR sets could be kept transiently to mitigate the effects of intermittent failures.

II.3.b.2 Algorithms to Deal with BGP and IGP events

We describe here informally how OPR sets are updated upon a BGP or an IGP event to achieve optimal protection of all destinations.

Handling BGP Updates The question is how to maintain OPR sets upon a BGP update? being either an *Add* (i.e., a new route is learned) or a *Down* event (i.e., a withdraw that cancels a route)[43]. As a BGP update concerns a given prefix, only one OPR set O (the one that optimally protects p) is modified when necessary. Intuitively, checking whether the route R belongs (or should belong) to the leaves of T extracted to create the current O (i.e., if R belongs to the current O) is enough to know if the update is necessary.

First, our algorithm retrieves the route-tree T of the updated prefix p. Depending on the nature of the update, we update the control-plane structure T (and implicitly L) by either adding or removing the updated route. When performing these operations, we store the rank of the MR set containing the route R, *rMR*.

Using the rMR rank, one can check whether R belongs (or should belong) to O, e.g., by memorizing the number of MR sets used to form O. If R is not good enough to belong to the current OPR set, there is no need to consider it and the algorithm ends. Otherwise, if R is a newly added (resp. withdrawn) route, it must be added (resp. removed) from the data-plane structure O which can be found in $\mathbb{O}$ through its hash. In both cases, O has to be updated. Dealing with a BGP update is thus pretty straightforward and BGP events are likely to have no bearing on the data-plane.

IGP changes The challenging question is about the behavior of *OPTIC* upon an IGP change. Being either a modification on the existence (insertion or deletion – modeled by an infinite weight) or on the weight w of a link l (a node wide change can be modeled through its multiple outgoing links), we need to support efficiently all kind of changes.

Upon a routing event, the new IGP distances D are recovered. *OPTIC* then considers each O, covering so every BGP prefixes. For each relevant gateway (with the best MED for each AS, M_{top}) within O, we first check whether it is still reachable. Unreachable gateways are replaced by the next MED-tied route when possible or removed otherwise. Reachable gateways are first updated with their new best IGP distances. The whole group of prefixes using the set benefits from its new optimal path as soon as possible. Afterward, if necessary, we update OPTIC's structures in the background to anticipate any future internal event.

If the updated link l is a valuation change, there is no loss of reachability. Thus, O still contains two disjoint paths towards p and remains stable. For other kinds of events, O may need to be updated, as connectivity may have evolved due to the insertion or deletion of a link. If a link was added, the network connectivity may have increased and useless MR sets can be removed if O is not already minimal (e.g., containing two gateways). If a link was removed, O may have lost its protecting property and may have to be updated. . This update is used to prepare for a future event. We perform it in background afterwards and continue to walk through $\mathbb{O}$ to restore the optimal forwarding state of all groups of prefixes quickly.

The update aforementioned is performed at the prefix granularity (i.e., for each prefixes that used O that will be updated). Indeed, while these prefixes share the same O before the change, they do not necessarily share the same L. Since O may be updated by fetching information from L, they may point to distinct OPR sets after the update. Recall that this is a background processing phase where *OPTIC* may fallback to the prefix granularity to anticipate the next change only if node-bi-connectivity is not granted anymore. Note that the fast switch to the new optimal post-convergence gateway was already performed before. This switch is not done at the prefix granularity, it is performed only for each O instead.

In short, IGP updates eventually also result in simple operations. A BGP update just triggers a prefix tree manipulation: a single OPR set is re-computed only if the updated route is, or should be, part of the set. An IGP weight-change results in the walk-through of all OPR sets ($\mathbb{O}$) and a min-search to converge to the new optimal forwarding state followed by a background processing if necessary. We argue that **the cardinal of $\mathbb{O}$ will be orders of magnitudes lower than the number of BGP prefixes in most networks**. The failure or addition of a link or node results in the same walk-through, but could also

[43] A modified route can be handled through a *Down* followed by an *Add*.

require the background update of some OPR sets to prepare for a future event. More precisely, only when the network gains or loses its bi-connected property could some OPR sets be affected. New OPR sets then need to be re-computed for the prefixes of the groups that depended on the affected OPR sets. Instead of the number of prefixes, *OPTIC* convergence scales with the number and the size of the OPR sets. Consequently, to assess the viability of *OPTIC*, we aim at limiting their size (and so number). While the next section analyzes this performance aspect, let us first explore two possible extensions/improvements in the design of *OPTIC*. First, about the conditions on the graph properties allowing to use smaller optimally protecting sets, second about the way we may rely on smart data plane to implement *OPTIC*.

II.3.b.3 Extensions: Optimisations & Data-plane Management

Optimizations In this paragraph, we introduce some optimizations that allow to reduce the size of the OPR sets used by *OPTIC*.

Let us start with a fairly reasonable assumption: well-designed networks should offer bi-connectivity between border routers. Based on this realistic hypothesis (that we rely on for our analytical evaluation), one may consider in addition two kinds of reductions: (*i*) removing MED-tied entries from an OPR set and (*ii*) discarding all gateways in the second MR set except the best one (when the first MR set includes only one gateway). As the first optimization will neither be used or required further on, we will not dwell on it. Intuitively, since the MED attribute is of higher importance than the IGP cost, it may allow us to remove routes with lower MED from the set, as an IGP cost change will not make these routes optimal, only the failure of the overall best MED gateway can affect it.

The second optimization will be evaluated in our theoretical analysis and allows to keep at most one gateway from the second MR set when the first one contains a single gateway. If the current optimal gateway is not part of the path towards the first gateway of the second MR set, adding this second gateway is enough to form an OPR set. When the first MR set is made of only one gateway and the network bi-connected, *OPTIC* only needs to consider other gateways for the specific case of the optimal gateway failure (as other changes cannot make it less preferred than gateways from the following MR sets). If the second-best gateway does not use the first to reach p, its IGP distance will not be impacted by the current gateways' failure, and it will become, after the failure, the best gateway overall. This allows *OPTIC* to create many OPR sets containing only two routes.

On the Use of a Smart Data Plane for OPTIC The full potential of *OPTIC* comes when its fast re-routing updates are performed within the data-plane. Indeed, the groups do not need to be evaluated sequentially but can be updated at packet line-rate, at least the switch to the new best BGP next-hop(s). In section V.2.b, I present a novel forwarding primitive that we have started to implement in P4[44] to target a Tofino architecture[45] able to fully exploit the *OPTIC* advantages. The control and data-planes can gracefully interact thanks to shared registers in order to update each forwarding entry once and only once. While there exists some challenges to address due to the absence of basic control loops in P4 [193], I will show that one can efficiently implement *OPTIC* using both FRR[46] and recent flexible hardware (like Tofino ASIC). I aim to develop a proof a concept and evaluate its performance on an experimental testbed in a near future.

II.3.c The Gain of Grouping External Prefixes

II.3.c.1 A not so Theoretical Model

To react to an IGP event, *OPTIC* only operates a min-search in all OPR sets. OPTIC's performances thus mainly depend on the number of OPR sets ($|\mathbb{O}|$) and their sizes.

We present here a theoretical model capturing a wide variety of scenarios. This analytical approach is more suitable than experiments, as it is more general and provides a pessimistic order of magnitude of OPTIC's potential. This approach yields the same results as a simulation, but allows to easily explore

[44] https://opennetworking.org/p4/

[45] https://www.intel.fr/content/www/fr/fr/products/network-io/programmable-ethernet-switch/tofino-2-series.html

[46] https://frrouting.org/

numerous scenarios. It highlights what an ISP can expect by running *OPTIC* given only a few structural parameters on their networks. We investigate several ASes profiles (constructed from [221]'s data), varying the number of gateways, peers, clients, and providers, as well as the number of prefixes learned through each of the latter. We show that $|\mathbb{O}|$ remains manageable and/or close to the lower bound, being 99% smaller than the number of prefixes for stub networks.

Preliminary Model: counting #OPR sets We consider an AS (or a portion of it), with B bi-connected gateways advertising P prefixes in total. Each prefix p is advertised by a subset of $b \leq B$ of those gateways, chosen uniformly at random. The β of each prefix is represented by a value between 1 and ps (policy spreading) also chosen uniformly at random. For a given p, this implies that any subset of gateways of a given size $n \leq b$ all have the same probability to be the OPR set for p. Our model analyzes the number $|\mathbb{O}| = |\mathbb{O}_{B,P,ps}|$ of unique OPR sets depending on the number B of gateways, the number P of prefixes, and on the policy spreading ps. In practice, we decide to set b to a constant value (e.g., $b = 5$) greater than the median in [221].

In bi-connected networks, OPR sets can only take two possible forms: they are either composed of one MR set (all best β-tied gateways) or two MR sets, if the first one only contains a single gateway. Indeed, more than 2 MR sets are not necessary, as two gateways will by definition protect the prefix, and 2 MR sets will thus do so optimally. Similarly, a single MR set of two gateways or more will always be sufficient. Recall that OPR sets are shared if they share both the same gateways *and* the same MR-set decomposition. Both cases thus need to be treated independently.

We first compute the probability p_n of a prefix to have an OPR composed of a single MR set of n gateways. In this case, all n gateways must have have the same best β, leading to a probability of $\frac{1}{ps^n}$. In addition, all other $b-n$ gateways must not have a better β. This event has a probability of $\left(1 - \frac{i}{ps}\right)^{b-n}$, with i the β of the n best gateways. This may happen for each possible ps values, so the probability must be summed for $i = 1tops$ (to consider cases where the n first gateways have the best β, second-best β, ...). Finally, there are $\binom{B}{n}$ such possible sets of size n. Thus, the probability that a prefix has an OPR set composed of a single MR of n gateways is

$$p_n = \sum_{i=1}^{ps} \binom{b}{n} \frac{1}{ps^n} \left(1 - \frac{i}{ps}\right)^{b-n} \qquad \text{(Eq. II.6)}$$

An OPR set of size n may also be composed of 2 MR sets, if the first one only contains a single gateway (the second MR set will thus contains $n-1$ gateways). We thus compute the probability p'_n of a given prefix to have an OPR set of size n composed of two MR sets. For this to happen, $n-1$ gateways must have the same β attributes, while a single gateway must have better β attributes. In addition, all other $b-n$ gateways must have worst β attributes, leading to a probability of $\frac{1}{ps^{n-1}} \frac{i-1}{ps} \left(1 - \frac{i}{ps}\right)^{b-n}$, with i the attributes of the $n-1$ gateways of the second MR set. Any one of the b gateways may possess the best β attribute, and there are $\binom{b-1}{n-1}$ possible second MR set of $n-1$ gateways. Thus, this event occurs $b\binom{b-1}{n-1}$ times. Summing all the cases (for each possible ps), we have

$$p'_n = \sum_{i=1}^{ps} b\binom{b-1}{n-1} \frac{1}{ps^{n-1}} \frac{i-1}{ps} \left(1 - \frac{i}{ps}\right)^{b-n} \qquad \text{(Eq. II.7)}$$

From the equations Eq. II.6 and Eq. II.7, we can compute the probability that a given OPR set of size n is associated to at least one prefix.

First let us compute the probability that a given OPR set of size n composed of a single MR is associated to at least one prefix. There are $\binom{B}{n}$ such sets. On average, the number of prefixes associated with OPR sets of size n (with a single MR set) is $p_n P$. Thus, the probability that a given set of size n is *not* associated with any prefix is $\left(1 - \binom{B}{n}^{-1}\right)^{p_n P}$ (all prefixes chose another OPR set). So, the probability that a given OPR set is associated with at least one prefix is

$$\mathbb{P}_{B,ps,P,n} = 1 - \left(1 - \binom{B}{n}^{-1}\right)^{p_n P} \quad \text{(Eq. II.8)}$$

A similar reasoning can be done to find the probability that a given OPR set of size n composed of two MR is associated with at least one prefix, the main difference being that there are $b\binom{B-1}{n-1}$ such sets (b possible first MR set, and $\binom{B-1}{n-1}$ possible second MR set), leading to

$$\mathbb{P}'_{B,ps,P,n} = 1 - \left(1 - \left(B\binom{B-1}{n-1}\right)^{-1}\right)^{p'_n P} \quad \text{(Eq. II.9)}$$

From $\mathbb{P}$ and $\mathbb{P}'$, we can compute $|\mathbb{O}_{B,P,ps}|$, the number of distinct OPR sets. The quantity $|\mathbb{O}_{B,P,ps}|$ can be seen as the sum of distinct OPR sets of different sizes. From our assumptions, OPR sets of size n ($2 \leq n \leq b$) are in $\mathbb{O}_{B,P,ps}$ with the same probability. A particular subset of gateways of size n can be the OPR of a given prefix through its one MR set variant (with a probability $\mathbb{P}_{B,P,ps,n}$) or through its two MR set variant (with a probability $\mathbb{P}'_{B,P,ps,n}$), leading to a probability $\mathbb{P}_{B,P,ps,n} + \mathbb{P}'_{B,P,ps,n}$, to account for both possible configurations. There are respectively $\binom{B}{n}$ and $B\binom{B-1}{n-1}$ such sets, for $n = 2$ to b (as b gateways announce the prefix). Thus, we have

$$|\mathbb{O}_{B,P,ps}| = \sum_{n=2}^{b} \binom{B}{n}\mathbb{P}_{B,P,ps,n} + B\binom{B-1}{n-1}\mathbb{P}'_{B,P,ps,n} \quad \text{(Eq. II.10)}$$

We can also easily compute the number of distinct OPR sets when applying the optimizations mentioned in the previous sections. In particular when considering that when a single gateway has the minimum weight, then the OPR set can be shrink at size 2, not matter how many gateways have the second minimum weight (i.e., are in the second MR set). This optimization is indeed very likely to be possible in practice, as the properties required are pretty lenient (the best path to the best gateway does not contain the second best gateway). When considering the optimization, the probabilities do not change when considering OPR set of one MR set (all gateways must have the same β attributes, as expressed by the left side of equation II.3.c.1). However, when considering OPR set consisting of two MR sets, (right sight of equation), things change. As long as only one gateway has the best β attributes, the OPR set will be of size 2. Thus, with this optimization, the probability that an OPR set contains 2 MR set is independent of its size: it simply is the probability of a single gateway to have the best β attribute, i.e., p_1. The probability that such set is associated with at least one prefix is $1 - \left(1 - (B(B-1))^{-1}\right)^{p_1 P}$. From this, we can derive the number of distinct OPR set when considering the optimization:

$$|\mathbb{O}^{opt}{}_{B,P,ps}| = B(B-1) \times \left(1 - \left(1 - (B(B-1))^{-1}\right)^{p_1 P}\right) + \sum_{n=2}^{b} \binom{B}{n}\mathbb{P}_{B,P,ps,n} \quad \text{(Eq. II.11)}$$

II.3.c.2 Towards More Realistic Results

Break Down Into Classes In practice, neighboring ASes are partitioned in several classes (*eg.*, clients, peers, and providers), usually represented by the local-pref attribute. At the end of the decision process, we know that a prefix p is associated with a single class. Indeed, the local-pref depends only on the set of advertising neighbors for p: it belongs to the class of the neighbor having the highest local-pref.

This allows us to split the analysis by class. With this assumption, OPR sets are included inside a unique class of gateways, but as a counterpart, the policy spreading in each class is reduced (because gateways have the same local-pref inside a class). We use our former model to compute the number of distinct OPR sets in each class with $ps = b = 5$. This calibration is pessimistic enough as it only takes into account a limited AS length dispersion and always 5 learning gateways in the best class.

B_1, B_2 and B_3, denote respectively the number of gateways with local-pref 1, 2 and 3. Similarly, P_1, P_2 and P_3, denote respectively the number of prefixes originating from a gateway with local-pref 1, 2 and

3. We have $P_1 + P_2 + P_3 = P = 800000$ and $B_1 + B_2 + B_3 = B$. We can now compute $|\mathbb{O}|$ by assuming each class follows our basic model:

$$|\mathbb{O}| = |\mathbb{O}_{B_1,P_1,5}| + |\mathbb{O}_{B_2,P_2,5}| + |\mathbb{O}_{B_3,P_3,5}|$$

This sum gives the theoretical performance of *OPTIC* as it is the number of OPR sets each router has to manage.

Definition of the Lower Bound We define here the best theoretical performance an optimally protecting scheme could reach, to compare it with *OPTIC*. Such a scheme would have to store sets of at minima two gateways (less can not ensure protection). This lower bound also provides an estimation of the performances of techniques just aiming at providing (non-optimal) protection like [110]. In other words, with P_i prefixes and B_i gateways in a given class, the average minimum number of optimally protecting sets is the average number of distinct sets obtained when choosing P_i random subsets of two gateways (such sets are chosen uniformly at random).

Table II.7: Number of distinct OPR sets ($|\mathbb{O}|$) for several scenarios.

Type of AS	# gateways per class	# prefix per class	# distinct OPR sets	OPR sets median size	Lower bound
Stub	$(10; 20; 0)$	$(700K; 100K; 0K)$	3945	4	235
Tier 4	$(10; 25; 25)$	$(500K; 200K; 100K)$	11879	3	645
Tier 3	$(10; 50; 100)$	$(500K; 200K; 100K)$	46010	3	6219
Large Tier 3	$(10; 100; 500)$	$(500K; 200K; 100K)$	127433	2	73781
Tier 2	$(5; 500; 2000)$	$(500K; 200K; 100K)$	263219	2	197194
Tier 2 (other)	$(5; 500; 2000)$	$(450K; 250K; 100K)$	294886	2	205484
Tier 1	$(0; 50; 5000)$	$(0K; 600K; 200K)$	232180	2	199633
Tier 1 (other)	$(0; 50; 5000)$	$(0K; 400K; 400K)$	425786	2	394891

Evaluation on Fixed Break Down We now compute $|\mathbb{O}|$ for several AS categories; a *Stub* has few peers and even fewer providers from where most the prefixes originate; a *Transit* (Tier 2, 3 and 4) has a limited number of providers but from where the majority of the prefixes originate, more peers and possibly numerous customers; a *Tier 1* has few peers and a large number of customers. For *Transit* and *Tier 1*, we present different class and prefix break down. Note that our model is pessimistic, as, for *Tier 1* in particular, ASes may have more classes with $ps > 5$ (e.g., gateways can be geographically grouped). The number of gateways, and their partition into classes, are rounded upper bounds of realistic values obtained from [221]. Moreover, we did not assume any specific popularity of certain gateways. Using our complementary material [57], $|\mathbb{O}|$ can be computed for any parameters.

Table II.7 shows that the number of OPR sets is more than reasonable for Stubs and small Transit. For large transit, the distribution of the prefixes into classes has a great impact on $|\mathbb{O}|$. As expected, for Tier 1, the number of OPR sets is high, but *OPTIC* is close to the lower bound (there is not much room for possible improvements). The number of routes contained within each OPR set is limited, meaning that the min-search applied upon an IGP event has a limited computational cost. Finally, it is worth recalling that our analysis is pessimistic because uniform. Regional preferences or gateway popularity can strongly reduce the size of $\mathbb{O}$ in practice.

OPTIC: Conclusions and Perspectives

Because the IGP and BGP are entangled to enforce hot-potato routing at the AS scale, an IGP change triggers the full and slow BGP convergence. With *OPTIC*, we have re-designed this IGP/BGP coupling in a more graceful manner. We proposed efficient MED-aware algorithms and data-structures to anticipate and quickly react to any single IGP event (weight change, link or node failure, including the outage of BGP border routers). The forwarding data-structure we propose is simple and so efficient. At the data-plane level, *OPTIC* ensures a fast and optimal re-convergence of the transit traffic. Thanks to a two-scale routing model that provisions a fast optimal forwarding protection scheme for the hot-potato, it is able to provide a robust BGP transit within each AS.

In the control-plane, *OPTIC* updates its prefix-trees constructs in background to anticipate a future event when necessary (only after changes modifying the 2-node-connectivity network property). Otherwise, there is no need to prepare the next post-change state for each forwarding entry, the data-plane has still one step ahead from the next (single) failure. With *OPTIC*, the transit network becomes robust enough to support any single failure, including the one of the best gateway. We have developed an FRR[47] implementation for the control-plane [Alfroy, 2020] and now work at its interactions with a data-plane implementation developed in section V.2.b to maximize the potential of *OPTIC* (performing the optimal backup switches at line-rate).

Since nearly all calculations are performed per group of prefixes, *OPTIC* scales orders of magnitudes lower than the number of BGP prefixes. Our analytical evaluation shows that the number of entries to manage in the FIB is at worst 50% of the full Internet table for large Tier-1. It scales down to 25% for large Tier-2s and less than 1% for Stub AS, which represents 84% of all ASes in the current Internet. It is very convenient for such situations as it ensures an optimal protection convergence at a very small cost for this majority of AS.

We now envision to extend this work by implementing a prototype in P4 as developed in section V.2.b. Pushing more intelligence to the data-plane offers many advantages for such use-cases as already discussed previously and as we will see in details later. The most interesting of them is the update of the best gateway within the data-plane as it becomes immediate for each packet (instead of doing it globally within the control-plane). Then, the number of groups just becomes a problem of space, the temporal complexity does not matter anymore. The space taken by our solution in the FIB can be mitigated with methods like [75].

On the other hand, to evaluate our technique, we will have to develop more realistic scenarios than the ones used so far as our analytical model is too pessimistic according to what can be measured in practice. This is a topic I aim to investigate by refining our model and analysis with ground measurements. Using routing collectors is possible, but we need more than only the best routes; for an ideal analysis, one needs to access to all the routes learned by the routers at the finest granularity of details, and this becomes now partially possible with projects like PEERING[48], BGP Streams[49] or BGP mon[50]. Some iBGP architectures may hide the real network diversity (in terms of number of gateways) [280]. Many works look at the correctness and diversity of iBGP [151, 345, 343, **?**, 332, 267, 237] whose the architecture design may lead to even more challenges than the purely inter-domain related BGP ones.

From another standpoint, other problems and technical challenges exist. For example the *optimal management* of the MED, to handle and update groups in a efficient manner. Also, more generally speaking, groups can be refined for many usages: providing more diversity (to deal with more events), or being minimal (to scale better in large Tier-1) and possibly specific to set of failures. *OPTIC* can be extended also to meet other needs than only best effort transit traffic with hot-potato routing: one can adapt the algorithm in use when looking for more complex sets of internal routes. Here we consider a two-disjoint paths algorithm (link and/or node) but one can look for other TE objective and use our proposal for other objectives and services.

[47] https://frrouting.org/
[48] https://peering.ee.columbia.edu/
[49] https://bgpstream.caida.org/
[50] https://www.bgpmon.io/

II.4 BEST2COP: Best Exact Segment Track for 2-Constraint Optimal Paths

This section develops another ongoing work of the Ph. D. thesis of Jean-Romain Luttringer, again with Quentin Bramas and Cristel Pelsser but also with Thomas Alfroy (an internship student that I supervised to design a control plane on FRR for *OPTIC* [Alfroy, 2020] and which has now started a Ph. D. in our group). We published a NCA paper [LAM+20] and extend this first publication in a journal version [LAM+22]. The originality of this work is to bring the SR contraint into an already well known but difficult problem (DCLC to find multi-constrained routes) without increasing its overall complexity. Moreover, we argue it is critical to deploy these constrained routes without burdening the core network with many forwarding states. We thus propose an efficient source routing design with SR that offers novel improvements and strong guarantees. In particular, our algorithm, Best2Cop, encodes constrained paths based on a multi-metric SR graph to natively and efficiently account for ECMP delay diversity. We have shown that this metric is the best candidate to be approximated with low error margins. Best2Cop relies on an advanced Bellmann-Ford algorithm variant that can perform efficient lazy Pareto Front updates, and be easily parallelized. To be the best of our knowledge, Best2Cop is the most efficient existing algorithm to retrieve DCLC paths within SR networks.

The IGP cost is usually defined as an additive metric that reflects both the link's bandwidth and the operator's load distribution choices on the topology, or the delay but not both. Paths within an IGP are computed by minimizing this one dimensional cost. Thus, although delay constraints are increasingly important, they should not be enforced to the detriment of the IGP cost. With minimal IGP distances, the traffic benefits from high-bandwidth links and follows the operator's intent in managing the network and its load. With bounded delays, the traffic can benefit from paths allowing for sufficient interactivity. It is thus relevant to minimize the IGP cost while enforcing an upper constraint on the latency. Computing such paths requires to solve DCLC, an NP-Hard problem standing for Delay Constrained Least Cost. While the theoretical problem has been already quite well investigated, to the best of our knowledge, there does not exist any deploymed solution as it remains a complex operational task. In the following, we study the opportunity offered by SR to deploy such paths and so answer this question:

Research Question

How to Efficiently Compute, Manage and Deploy Multi-Constrained Paths within SR Domains?

The following table of content summarizes our progression in this last section of the chapter: from the problem statement to the proposed solution and its analysis.

II.4.a Computing Delay Constrained Least Cost Paths for Segment Routing

Segment Routing & MSD Segment Routing (SR) is a vibrant technology gathering traction from router vendors, network operators and academic communities [236, 339]. Relying on a combination of strict and

loose source routing, SR enables to deviate the traffic from the shortest IGP paths through a selected set of nodes and/or links by prepending routing instructions to the packet itself. Such deviations may for example allow to route traffic through a path with lower latency. These deviations are encoded in the form of *segments* within the packet itself. To prevent any packet forwarding degradation, the number of deviations (i.e. instructions) one can encode is limited to MSD (Maximum Segment Depth), whose exact value depends on the hardware. While this technology is adequate to support a variety of services, operators mainly deploy SR in the hopes of performing fine-grained and ECMP[51]-friendly tactical Traffic-Engineering (TE) [15], due to its reduced overhead compared to RSVP-TE [106]. Such a solution should thus not only encompass Segment Routing, but also fare well on large-sized networks of several thousand of nodes, as already observable in current SR deployments [236]. Indeed, while we showed in [LAM+20] that computing DCLC paths for Segment Routing (DCLC-SR) is possible in far less than a second on networks of up to 1000 nodes, scaling to ten or a hundred times more routers remains an open issue.

Moreover efficiently encompassing the Maximum Segment Depth constraint (MSD) is challenging. An SR router can only prepend up to *MSD* routing instructions to a packet at line-rate, i.e., ≈ 10 with the best current hardware. Although this limit does not prevent from deploying most DCLC paths in practice (if not all in easy cases), this constraint must still be taken into account. If ignored, the computed paths have no guarantees to be deployable, as they may exceed MSD. While this adds an additive metric to consider (the number of segments), Best2Cop manages, through adequate data-structures and graph exploration, to natively manipulate the list of segments and ensure that paths requiring more than MSD segments are removed from the exploration space.

Towards massive-scale networks & Strong Delay Guarantees for Solving DCLC Our proposals, Best2Cop and BEST2COPe (its extended variant for massive scale multi-area IGP networks), aim at enabling fine-grained TE on massive-scale networks efficiently using a divide-and-conquer approach. Indeed, massive networks usually rely on a standard physical and logical partitioning, as IGP protocols do not scale well as is. By leveraging this decomposition and designing Best2Cop to benefit from multi-threaded architectures, it becomes possible to solve DCLC-SR in a time suited for real-time routing. We evaluate our contribution with a new topology generator, YARGG[MFB+11], constructing realistic massive-scale, multi-valuated, and multi-area topologies based on geographical data. In this evaluation, our extension is able to solve DCLC-SR in ≈ 1 second for ≈ 100000 nodes.

DCLC is a well-known NP-Hard problem [369] that raises practical concerns like bounded error margins. While there exist several ways to solve DCLC [152, 135], they usually do not consider the underlying deployment technologies and real-life deployment constraints. We keep the latter at the core of our design. The nature of the concerned TE paths allows us to consider a stable latency metric (the propagation delay). This is essential as unstable metrics should not be considered nor advertised when performing routing [139, 141]. Furthermore, because of the arbitrary nature of the latency constraint and the inherent imprecision of the delay measurement, we argue that an acceptable error margin regarding the delay constraint is acceptable (more so than on the IGP cost). Our algorithm is designed to take advantage, if needed, of this acceptable margin to return the DCLC path efficiently in all situations to all destinations, with strong guarantees.

II.4.a.1 The DCLC Problem in SR domains

Although one may expect the IGP and delay metrics to be strongly correlated in practice, there are various cases where they may be drastically different. For example, the IGP cost may have been tuned arbitrarily by the operator. Heterogeneous infrastructures between countries or geographical constraints may also create this effect. This can be illustrated on real networks, as displayed by Fig. II.10. This map is a sample of the GEANT transit network [256].

As fibers often follow major roads, we rely on real road distances to infer the propagation delay of each link while the bandwidth, and so the estimated IGP cost, matches the indications provided by GEANT. A green link has an IGP cost of 1 while the IGP cost is 2 and 10 respectively for the yellow and pink ones.

[51] Equal Cost Multi-Path

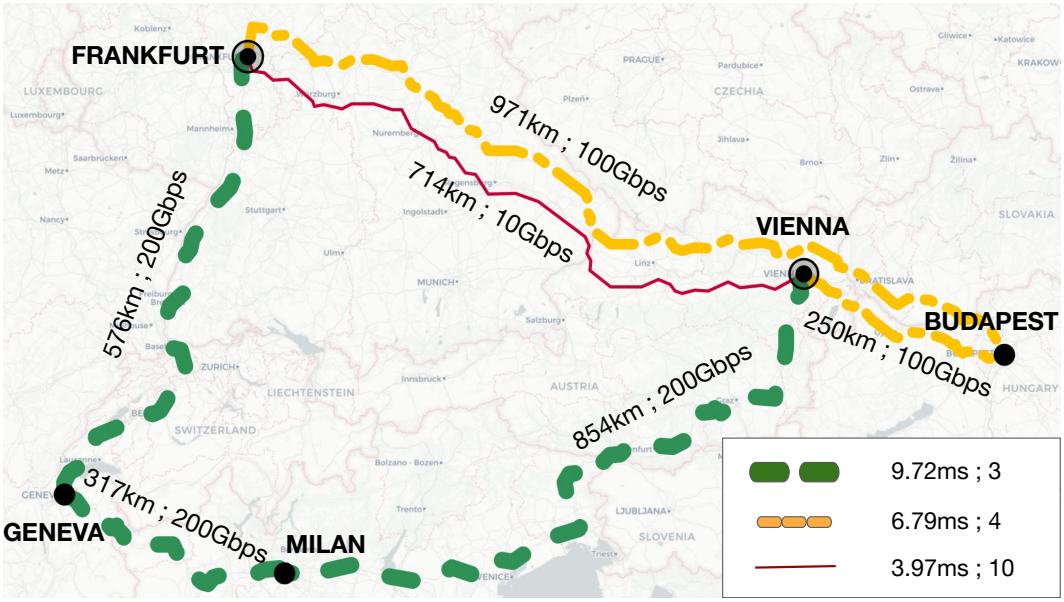


Figure II.10: An illustration to highlight the practical relevance of DCLC in the GEANT network. IGP costs are deduced from the bandwidth of each link. Depending on their needs (in terms of delay and bandwidth), applications can opt for three non-comparable paths between Frankfurt and Vienna.

Note that the two metrics are not correlated, hence all three paths shown between Frankfurt and Vienna offer diverse interesting options. They are non-comparable (or *non-dominated*) paths and form the *Pareto-front* of the paths between the two cities. Either solely the delay matters and the direct link (in pink) should be preferred, or the ISP prefers to favor high capacity links, and the green path, minimizing the IGP cost, should be used. The yellow path, however, offers an interesting compromise. Out of all paths offering a latency well-below 10ms, it is the one minimizing the IGP cost. Thus, it allows to provide strict Service-Level Agreement (< 9ms), while considering the IGP cost. These kind of paths, retrieved by solving DCLC, provide more options by enabling tradeoffs between the two most important networking metrics. Applications such as videoconferences, for example, can then benefit both from real-time interactive voice exchange (delay) and high video quality (bandwidth). In addition, IGP costs are also tuned to represent the operational costs. Any deviation from the shortest IGP paths thus results in additional costs for the operator. For all these reasons, there exist a clear interest for algorithms able to solve and deploy DCLC paths [105]. However and so far, while this problem has received a lot of attention in the last decades from the network research community [152, 135], no technologies were available for an efficient deployment of such paths.

Segment Routing Background and Practical Usages Segment Routing implements source routing by prepending packets with a stack of up to MSD *segments*. In a nutshell, segments are checkpoints the packet has to go through. There are two main types of segments:

- **Node segments.** A node segment v indicates that the packet should (first) be forwarded to v with ECMP (instead of its final IP destination). Flows are then load-balanced among the best IGP next hops for destination v.
- **Adjacency segments.** Adjacency segments indicate that the packet should be forwarded through a specific interface and its link.

Once computed, the stack of segments encoding the desired path is added to the packet. Routers forward packets according to the topmost segment, which is removed from the stack when the packet reaches the associated intermediate destination. Adjacency segments may be globally advertised, and thus be used the same way as node segments, or they may only have a local scope and, as such, can only be interpreted by the router possessing said interface. In this case, the packet should first be guided

to the corresponding router, by prepending the associated node segment. In the following, as a worst operational case, we consider the latter scenario, as it always requires the highest number of segments.

Segment Routing attracted a lot of interest from the research community. A table referencing most SR-related work can be found in Ventre et al. [339]. While some SR-TE works are related to tactical TE problems (like minimizing the maximum link utilization) taking indirectly into account some delay concerns [168, 159], most of the works related to SR do not focus on DCLC, but rather bandwidth optimization [52, 136, 61], network resiliency [114, 158], monitoring [217, 32], limiting energy consumption [69] or path encoding (the translation of path to segment lists) [153, 142]. Aubry [31] proposes a way to compute paths requiring less than MSD segments while optimizing an additive metric in polynomial time. The number of segments required is then evaluated. This work, however, considers only a single metric in addition to the operational constraints. The problem we tackle (i.e. DCLC paths for Segment Routing) deals with two metrics (in addition to the operational constraints). This additional dimension drastically changes the problem, which then becomes NP-Hard. Some works use a construct similar to ours in order to prevent the need to perform conversions from network paths to segment lists, [202] in particular. However, the same authors do not pretend to solve DCLC and, as such, do not tune the structure the same way (i.e. they do not remove dominated segments, as explained later on), and simply use their construct to sort paths lexicographically.

As aforementioned, while operators seem to mainly deploy SR to perform fine-grained TE, to the best of our knowledge, no DCLC variant exists for specifically tackling SR characteristics and constraints (except for our contribution). Using segments to steer particular flows allows however to deviate some TE traffic from the best IGP paths in order to achieve, for example, a lower latency (and by extension solve DCLC). A realistic example is shown on Fig. II.10 where the node segment *Vienna*, as well as considering Vienna as the destination itself, would result in the packets following the best IGP path from Frankfurt to Vienna, i.e., the green dashed path. To use the direct link instead (in plain pink) and so minimize the delay between the two nodes of this example, the associated adjacency segment would have to be used as it enforces a single link path having a smaller delay than the best IGP one (including here two intermediary routers). Finally, the yellow path, offering a non dominated compromise between both metrics (and being the best option if considering a delay constraint of 8ms), requires the use of the node segment *Budapest* to force the traffic to deviate from its best IGP path in green. Before converting the paths to segment lists (and actually deploy them with SR), such non-dominated paths need first to be explored. Computing these paths while ensuring that the number of segments necessary to encode them remains under MSD is at least as difficult as solving the standard DCLC problem since an additional constraint now applies.

DCLC (Delay-Constrained, Least-Cost), a Well-known Difficult Problem having many Solutions? DCLC belongs to the set of NP-Hard problems (as well as most related multi-constrained path problems). Intuitively, solely extending the least-cost path is not sufficient, as the latter may exceed the delay constraint. Thus, paths with greater cost but lower delays must be memorized and extended as well. These *non-dominated paths* form the *Pareto front* of the solution, whose size may grow exponentially with respect to the size of the graph. However, DCLC in particular, and related variants and extensions in general, does possess several interesting applications such as mapping specific flows to their appropriate paths (in terms of interactive quality). Thus, these problems have been extensively studied in the past decades. Many solutions have been proposed so far, as summarized in these surveys [196, 135, 152]: they range from to heuristics and approximations to exact algorithms, or even genetic approaches.

Heuristics. Because DCLC is NP-Hard, several polynomial-time heuristics have been designed to limit the worst-case computing time, but at the detriment of any guarantees. For example, [203] only returns the least-cost or least-delay path if one is feasible (i.e. respect the constraints). More advanced proposals try to explore the delay and cost space simultaneously, by either combining in a distributed manner the least-cost and least-delay subpaths [283, 370, 212] or by aggregating both metrics into one in a more or less intricate manner.

Aggregating metrics in a linear fashion [177, 30] preserves the *subpaths optimality principle* (isotonicity of best single-metric paths) and therefore allows to use standard shortest paths algorithms. However, it leads to a loss of relevant information regarding the quality and feasibility of the computed paths [369], in particular if the hull of the Pareto Front is not convex. Some methods try to mitigate this effect by using

a k-shortest path approach to possibly find more feasible paths [181, 183], but such an extension may result in a large increase of execution time and may not provide more guarantees. Other heuristics rely on non-linear metric aggregation. While it seems to prevent loss of relevant information, at first glance, such algorithms expose themselves to maintain all non-dominated paths (towards all nodes) as the isotonicity does not hold anymore (while it holds with linear metrics). Since the Pareto Front may be exponential with respect to the size of the graph, those algorithms either simply impose a hard limit on the number of paths that can be maintained (e.g. TAMCRA [87] and LPH [356]), or specifically chose the ones to maintain through previously acquired knowledge (HMCOP [194]). Finally, other works like [104, 155] rely on heuristics designed to solve a variant of DCLC, the MCP problem (Multi-Constrained Paths, the underlying NP-Complete decision version of DCLC – with no optimization objective). It mainly consists of sequential MCP runs using a conservative cost constraint iteratively refined.

Relying on heuristics is tempting, but their lack of guarantees can prevent to enforce strict SLAs even when a suitable path actually exists. One can argue it is particularly unfortunate, as DCLC is only *weakly* NP-hard: it can be solved exactly in pseudo-polynomial time, i.e. polynomial in the numerical value of the input [134]. Said otherwise, DCLC is polynomial in the smallest largest weight of the two metrics once translated to integers. Consequently, it is possible to design FPTAS[52] solving DCLC while offering strong guarantees [263].

Approximations. Numerous FPTAS have been proposed to solve DCLC and related constrained shortest path problems. All the following algorithms fall into this category. The common principle behind these schemes is to reduce the precision (and/or magnitude) of the considered metrics. This can be performed either directly, by *scaling and rounding* the weights of each link, or indirectly, by dividing the solution space into intervals and only maintaining paths belonging to different intervals (*Interval partitioning*) [291]. Scaling methods usually consider either a high-level dynamic programming scheme or a low-level practical Dijkstra/Bellman-Ford core with pseudo-polynomial complexity, and round the link costs to turn their algorithms into an FPTAS (see for example Hassin [161], Ergun et al. [102] or Lorenz and Raz [216] methods). Goel et al. [143], in particular, chose to round the delay instead of the cost and can consider multiple destinations (as our own algorithm).

Most interval partitioning solutions explore the graph through a Bellman-Ford approach. The costs of the paths are mapped to intervals, and only the path with the lowest delay within a given interval is kept. The size of the intervals thus introduces a bounded error factor [161, 331]. In particular, HIPH [318] offers a dynamic approach between an approximation and exact scheme. It proposes to maintain up to x non-dominated paths for each node and stores eventual additional paths using an interval partitioning strategy. This allows the algorithm to be exact on simple instances (resulting in a limited Pareto front, i.e. polynomial in the number of nodes, in particular when it is bounded by x) and offer strong guarantees on more complex ones.

Exact methods. Numerous exact methods have indeed also been studied extensively to solve DCLC. Some methods simply use a k-shortest path approach to list all paths within the Pareto front [252, 261]. On the other hand, Constrained Bellman-Ford [353] (ironically, also called Constrained Dijkstra as it uses a priority queue – denoted PQ in the following) explores paths by increasing delays and lists all non-dominated paths towards each node. Several algorithms use the same principle but order the paths differently within the queue, relying either on a lexicographical ordering, ordered aggregated sums, or a simple FIFO/LIFO ordering [234, 235, 60]. Most notably, A* Prune [210] is a multi-metric adaptation[53] of A* relying on a PQ where paths known to be unfeasible are pruned. Two-phase methods [277] first find paths lying on the convex hull of the Pareto front through multiple Dijkstra runs, before finding the remaining non-dominated path through implicit enumerations.

Finally, SAMCRA [336] is a popular and well-known multi-constrained path algorithm. Similarly to other Dijkstra-based algorithms, SAMCRA relies on a PQ to explore the graph but instead of the traditional lexicographical ordering, it relies on non-linear cost aggregation. Among feasible paths (others are natively ignored) it first considers the one that minimizes its maximum distance to the multiple constraints. Such a path ranking to deal with the PQ is supposed to increase its performance with respect to other PQ organizations.

[52] Fully Polynomial Time Approximation Scheme.

[53] This adaptation is exact, i.e. not a heuristic, as the estimated cost underestimates the actual distance towards the destination.

Table II.8: Qualitative summary of a representative subset of DCLC-compatible algorithms showcasing their practicality, exactitude, and performance. In the *Practical Features* column, the green check-mark indicates whether the algorithm supports the corresponding feature (while the red cross denotes the opposite). In the *Exactitude vs Performance* column, the two subcolumns associated which each three scenarios show how the latter impact (*i*) the exactitude (exact, strong guarantees, no guarantees) and (*ii*) the performance of the algorithm (polynomial time or not). While the orange tilde denotes strong guarantees in terms of exactitude, green check-marks (and red crosses respectively) either indicate exact results (no guarantees resp.) or polynomial-time execution (exponential at worst resp.) for performance. For both subcolumns *Bounded Pareto Front* and *Coarse Metric*, we consider the case where their spreading is polynomial with respect to the number of vertices in the input graph (and as such predictable in the design/calibration of the algorithm).

Algorithms	Practical Features			Exactitude vs Performance					
	Multi-Dest Single Run	SR Ready	Multi-thread Ready	Bounded Pareto Front		Coarse Metric		All Cases	
LARAC [183]	×	×	×	×	✓	×	✓	×	✓
LPH [356]	✓	×	×	✓	✓	∼	✓	×	✓
HMCOP [194]	×	×	×	×	✓	×	✓	×	✓
HIPH [318]	✓	×	×	✓	✓	✓	✓	∼	✓
Hassin [161]	×	×	×	∼	✓	✓	✓	∼	✓
Tsaggouris et al. [331]	✓	×	×	∼	✓	✓	✓	∼	✓
Raith et al. [277]	×	×	×	✓	✓	✓	✓	✓	×
A* Prune. [210]	×	×	×	✓	✓	✓	✓	✓	×
SAMCRA [336]	✓	×	×	✓	✓	✓	✓	✓	×
Best2Cop	✓	✓	✓	✓	✓	✓	✓	∼	✓

While many solutions exist, most possess certain drawbacks or lack certain features to reconcile both the practice and the theory. Heuristics do not always allow to retrieve the existing paths enforcing strict SLAs, while exact solutions are not able to guarantee a reasonable maximum running time when difficult instances arise, although both features are essential for real-life deployment. On the other hand, FPTAS can provide both strong guarantees and a polynomial execution time. However, they are often found in the field of operational research where, at best, possible networking applications and assumptions are discussed, but the deployment of the computed paths, with SR and its MSD constraint in particular, is not investigated. The number of segments is not a standard metric as it is not simply a weight assigned to each edge in the original graph (that is, without a specific construct, it requires to be computed on the fly for each visited path). Considering the latter can have a drastic impact on the performance of the algorithms not designed with this additional metric in mind. In addition, not all the algorithms presented here and in Table II.8 are single-source multiple-destinations. Finally, none of these algorithms evoke the possibility to leverage multi-threaded architectures, an increasingly important feature as such computations now tend to be performed by dedicated Path Computation Elements or even in the cloud.

Our contribution, Best2Cop, aims to close this gap by mixing the best existing features (such as providing both a limited execution time and strong guarantees in terms of precision in any cases) and adapt them for a practical modern usage in IP networks deploying SR. Table II.8 summarizes some key features of a representative subset of the related work. Similarly to FPTAS, Best2Cop rounds one of the metrics of the graph. However, conversely to most algorithms, Best2Cop does not sacrifice accuracy of the cost metric, but of the measured delay.

II.4.a.2 DCLC-SR, the multi-metric SR Graph and 2COP

This section introduces and defines preliminary notations and concepts used to design Best2Cop, as well as data structures at play. We aim to solve DCLC in the context of an ISP deploying SR, leading to the DCLC-SR problem that considers the IGP cost, the propagation delay, and the number of segments.

For readability purposes, we denote:

- M_0 the metric referring to the number of segments, with the constraint $c_0 = \text{MSD}$ applied to it;
- M_1 the delay metric, with a constraint c_1;
- M_2 the IGP metric being optimized.

With such notations, let us introduce the first problem we aim to solve.

Problem 6. *The Delay Constrained Least Cost problem with Segment Routing (DCLC-SR)* *Given a source s, **DCLC-SR** consists in finding, for all destinations, a segment path verifying two constraints, c_0 and c_1, respectively on the number of segments (M_0) and the delay (M_1), while optimizing the IGP distance (M_2). We denote this problem DCLC-SR(s, c_0, c_1).*

On Fig. II.10, we would have DCLC-SR$(Frankfurt, 3, 8) \supset Frankfurt - Budapest - Vienna$. This DCLC path (shown in yellow in Fig II.10), is indeed the best option to reach Vienna when considering an arbitrary delay constraint of 8ms. Since the best IGP path from Frankfurt to Vienna (the green one) does not go through Budapest, encoding this DCLC path requires at least one detour. The segment path, or the *segment list*[54], contains only one segment: here, a node segment instructing the packet to go through Budapest first.

DCLC and True Measured Delays: Leveraging Measurement Inaccuracy DCLC is weakly NP-Hard, and can be solved exactly in pseudo-polynomial time. In other words, as long as either the cost of the delay possesses only a limited number of distinct values (i.e., paths can only take a limited number of distinct distances), the Pareto front of the paths' distances is naturally bounded in size as well, making DCLC tractable and efficiently solvable[55]. Such a metric thus has to be bounded and possess a coarse accuracy (i.e., be discrete). Although this has little impact when solving DCLC in a theoretical context, it can be strongly leveraged to solve DCLC efficiently thanks to the characteristics of real ISP networks.

We argue that the metrics of real ISP networks do indeed possess a limited number of distinct values. Although Best2Cop can be adapted to fit any metric, we argue that M_1, the propagation delay, is the most appropriate one. Indeed, IGP costs depend on each operators' configurations. For example, while some may rely on few spaced weights, others may possess intricate weight systems where small differences in weights may have an impact. Thus, bounding the size of the Pareto front based on the IGP costs is not only operator-dependant, but might still result in a very large front.

On the other hand, the delay (i) is likely strongly bounded, and (ii) can be handled as if having a coarse accuracy in practice. For TE paths, the delay constraint is likely to be very strict (10ms or less). Second, while the delay of a path is generally represented by a precise number in memory, the actual accuracy, i.e. the trueness t of the measured delay is much coarser due to technical challenges [22, 21]. In addition, delay constraints are usually formulated at the millisecond granularity with a tolerance margin, meaning that some loss of information is acceptable.

Thus, floating numbers representing the delays can be truncated to integers, e.g., taking 0.1ms as unit. This allows to easily bound the number of possible non-dominated distances to $c_1 \times \gamma$, with γ being the desired level of accuracy of M_1 (the inverse of the unit of the delay grain, here 0.1ms). For example, with $c_1 = 100$ms and a delay grain of 0.1ms ($\gamma = \frac{1}{0.1} = 10$), we have only 1000 distinct (truncated) non-dominated pairs of distances to track at worst. This leads to a predictable and bounded Pareto front. One can then store non-dominated distances within a static array, indexed on the M_1-distance (as there can only be one non-dominated couple of distances (M_1, M_2) for a given M_1-distance).

The variable Γ denotes the size allocated in memory for this Pareto front array (i.e., $\Gamma = c_1 \times \gamma$). When t, i.e., the real level of accuracy, is lower (or equal) than γ, the stored delay can be considered to be exact. More precisely, it is discretized but with no loss of relevant information. When t is too high, one can choose γ such that $\gamma < t$, to keep Γ at a manageable value. In this case, some relevant information can be lost, as the discretization is too coarse. While this sacrifices the exactitude of the solution (to

[54] We use the two terms interchangeably but we more often rely on this second term to emphasize the final outcome of our algorithm that returns logical sequence of segments rather than physical paths (although such an underlying path can be indirectly retrieved from the segment list).

[55] Metric M_0 is omitted for now as this trivial distance is only required for SR and discussed in details later. While dealing with a three-dimensional Pareto front seems more complex at first glance, we will show that SR eventually reduces the exploration space because its operational constraint is very tight in practice and easy to handle efficiently.

the advantage of computation time), our algorithm is still able to provide predictable guarantees in such cases (i.e. a bounded error margin on the delay constraint).

Propagation delays are stable Referring to a path's delay may be ambiguous. Indeed, this characteristic is not monolithic. The overall delay is mainly composed of the propagation delay and the queuing delay. Both delays may play an important part in the overall latency, though none can be stated to be the main factor [297]. Although the propagation delay is stable, the queuing delay may vary depending on the traffic load. However, in order to compute TE paths, the delay metric must be advertised (usually within the IGP itself). For this reason, it is strongly recommended to use a stable estimate of the delay, as varying delay estimations may lead to frequent re-computations, control-plane message exchanges, and fluctuating traffic distribution [139, 141].

For this reason, we use the propagation delay, as recommended in [139, 141]. The latter is usually measured through the use of a priority queue, ignoring the queuing delay. Its value is deduced as a minimum from a sampling window, to increase its stability [78]. Using this delay not only makes our solution practical (as we rely on existing measurements and respect protocol-related constraints), but is actually pertinent in our case. In practice, flows benefiting from DCLC paths have a queue with high priority and experience negligible queuing delays. Consequently, we use the discretized propagation delay, enabling both practical deployment and the limitation of the number of non-dominated distances, within our structure used to encompass Segment Routing natively, the SR graph.

To solve DCLC-SR efficiently, as well as its comprehensive generalization, 2COP, we rely on a specific construct used to encompass SR, the delay, and the IGP cost: the *multi-metric SR graph*.

Turning the Physical Graph into a Native SR Representation This construct represents the segments as edges to natively deal with the M_0 metric and its constraint, $c_0 = \text{MSD}$. The valuation of each edge depends on the distance of the path encoded by each segment. While the weights of an adjacency segment are the weights of its associated local link, the weights of a node segment are the distances of the ECMP paths it encodes: the (equal) IGP cost (i.e., M2-distance), and the lowest guaranteed delay (i.e., the M1-distance), i.e. the worst delay among all ECMP paths. Hence, computing paths on the SR graph is equivalent to combining stacks of segments (and the physical paths they encode), as stacks requiring x segments are represented as paths of x edges in the SR graph (agnostically to its actual length in the raw graph). The SR graph can be built for all sources and destinations thanks to an All Pair Shortest Path (APSP) algorithm. Note that this transformation is inherent to SR and leads to a complexity of $O(n(n\log(n)+m))$, for a raw graph having n nodes and m edges, with the best-known algorithms and data structures.

This transformation is shown in Fig. II.11, which shows the SR counterpart of the raw graph provided in Fig. II.10. To describe this transformation more formally, let us denote $G = (V, E, (w_1, w_2))$ the original two weighted graph. As G can have multiple parallel links between a pair of nodes (u, v), we use $E(u, v)$ to denote all the direct links between nodes u and v. Each link (u, v) possesses two weights, its delay $w_1^G((u, v))$ and its IGP cost $w_2^G((u, v))$. The delay and the IGP cost being additive metrics, the M_1 and M_2 distances of a path p (denoted $d_1^G(p)$ and $d_2^G(p)$ respectively) are the sums of the weights of its edges.

From G, we create a transformed multi-metric graph, the SR graph denoted $G' = (V, E', (w_1^{G'}, w_2^{G'}))$. While the set of nodes in G' is the same as in G, the set of edges differs because E' encodes segments as edges representing either adjacency or node segments encoding respectively local physical link or sets of best IGP paths (with ECMP). The M_i-weight of an edge in G' is denoted $w_i^{G'}((u, v))$. In particular, we have $w_2^{G'}((u, v)) = c_1((u, v))$ with our previous notations ($c_1(.,.)$ denoting a cost function and not a given contraint). Since we use here c_j to denote constraints and not the cost function returning the (best) distance(s) of a path, and because distances become more subtle and now threefold, we here rather rely on a set of functions $d_i(.),\ i < 3$ defined on both graphs to return the distances of a (segment) path in each of its dimension[56]. Ranking paths and their costs for a given couple of nodes has no sense here so we rather consider distances associated to a given non dominated segment path. Moreover, to alleviate

[56] While $d_0^{G'}(.)$ returns the minimal number of segment of a given segment path, $d_1^{G'}(.)$ and $d_2^{G'}(.)$ respectively return the worst experienced delay and the best IGP cost of the segment path. Note that $d_0^{(}.)$ is not defined on G (or simply as the hop count) as the notion of segment path comes with G' only. Finally, while each function $d_i^G(.)$ takes only a given unique path as input, functions $d_i^{G'}(.)$ consider a segment path as input which may include natively several underlying ECMP paths.

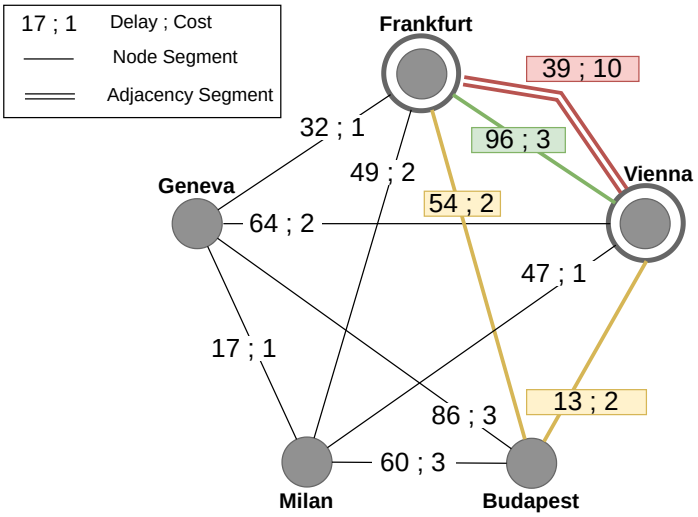


Figure II.11: This figure shows the network from Fig. II.10 translated into an SR graph. The SR graph encodes segments as edges. Plain edges represent node segments, i.e. sets of ECMP paths. Double-lines are adjacency segments, here only (*Frankfurt*, *Vienna*), and are visible only if they are not dominated by other segments. Colored edges refer to the paths highlighted in Fig. II.10 .

further notations, we denote simply $d_i(p)$ the M_i distance of a path in G' instead of $d_i^{G'}(p)$. Note that if G is connected, then G' is a complete graph thanks to node segments.

SR graph: Node segment encoding. A node segment, encoding the whole set $P_G(u,v)$ of ECMP best paths between two nodes u and v, is represented by exactly one edge in $E'(u,v)$. The M_2-weight $w_2^{G'}((u,v))$ of a node segment is the (equal) M_2-distance of $P_G(u,v)$. Since, when using a node segment, packets may follow any of the ECMP paths, we can only guarantee that the delay of the path will not exceed the maximal delay out of all ECMP paths. Consequently, its M_1-weight $w_1^{G'}((u,v))$ is defined as the maximum M_1-distance among all the paths in $P_G(u,v)$. Links representing node segments in G' thus verify the following:

$$w_1^{G'}((u,v)) = \max_{p \in P_G(u,v)} d_1^G(p) \qquad \text{(Eq. II.12)}$$

$$w_2^{G'}((u,v)) = d_2^G(p) \quad \text{for any } p \in P_G(u,v) \qquad \text{(Eq. II.13)}$$

SR graph: Adjacency segment encoding. An adjacency segment corresponds to a link in the graph G and is represented by an edge (u_x, v) in $E'(u,v)$, whose weights are the ones of its corresponding link in G, only if it is not dominated by the node segment $(u,v)_{G'}$ for the same pair of nodes, i.e. if $w_1^{G'}((u,v)) > w_1^G((u_x,v))$, or by any other non-dominated adjacency segments (u_y,v), i.e. if $w_1^G((u_y,v)) > w_1^G((u_x,v))$ or $w_2^G((u_y,v)) > w_2^G((u_x,v))$, where (u_x,v) and (u_y,v) are two different outgoing links of u in $E(u,v)$[57].

Fig. II.11 illustrates the result of such a transformation: one can easily identify the three non-dominated paths between Frankfurt and Vienna, bearing the same colors as in Fig. II.10. The green path (i.e. the best M_2 path) is encoded by a single node segment. The pink, direct path (i.e. the best M_1 path) is encoded by an adjacency segment (the double line in Fig. II.11). The yellow paths (the solution of DCLC-SR(Frankfurt, 3, 8) and an interesting tradeoff between M_1 and M_2) requires an additional segment, in order to be routed through Budapest. Note that in practice, the last segment is unnecessary if it is a node segment, as the packet will be routed towards its final IP destination through the best M_2 paths natively.

Our multi-metric SR graph (or equivalent constructs gathering the multi-metric all-pair shortest path data) is mandatory to easily consider the number of segments necessary to encode the paths being explored. However, its usage can differ in practice. We envision two modes which allow to consider this additional "off the graph" metric, using our SR Graph.

[57] If two links have exactly the same weights, we only add one adjacency segment in G'

Using the SR Graph to perform path conversions. One of the two options is to run the path computation algorithm on the original topology, and convert the paths being explored to segment lists. Performing this conversion is however not trivial. One must return the minimal encoding of the given path (with respect to the number of segments) while correctly managing the (forced) path diversity brought by ECMP, which may exhibit heterogeneous delays. However, one can efficiently perform such conversion when relying on our SR graph. By summarizing the relevant information (i.e. the worst-case delay within ECMP paths), the SR Graph allows to easily consider the ECMP nature of SR within a multi-metric context. However, the segment metric M_0 is peculiar. Extending a path does not always imply an increase in the number of necessary segments. Furthermore, the number of segments required to encode two distinct paths may evolve differently, even when the latter are extended from the same node with the same edge. Because of these properties, the way to check paths for dominancy must be revised. This extended dominance check may lead to an increased number of paths to extend, and thus to a higher worst-case complexity.

Using the SR Graph natively. Another method is to run the path computation algorithm directly on the SR graph we described. Note that this forces the algorithm to run on a complete graph, which may significantly increase the overall complexity. However, the segment metric M_0, originally an "off the graph" metric with singular properties, becomes a standard graph metric, as it is now expressed by the number of edges that compose the paths (a path encoded by x segments has x edges within the SR Graph). This method also allows using standard, known algorithms as-is to solve the DCLC-SR problem.

When designing our algorithm, Best2Cop, we use the second approach. Indeed, by using a bellman-ford-like exploration of the SR Graph, not only one can easily prune paths requiring more than MSD segments, but also benefit from efficient Pareto front management and multi-threading. These various features allow Best2Cop to efficiently solve not only DCLC-SR, but also 2COP, a more general and practically relevant problem regarding the computation of constrained paths within an SR domain. Note, however, that we will provide our competitor with both approaches to make the evaluation as fair as possible.

The 2COP Problem(s) Solving DCLC-SR exactly requires, by definition, to visit the entirety of the Pareto front for all destinations. However, although only some of these paths are DCLC-SR solutions for a given delay constraint, all paths visited during this exploration may be of some practical interest. In particular, some of them solve problems similar to DCLC but with different optimization strategies and constraints. By simply memorizing the explored paths (i.e. storing the whole Pareto front within an efficient structure), one can solve a collection of practically relevant problems. For instance, one may want to obtain a segment path that minimizes the delay, another the IGP-cost, or the number of segments. Solving 2COP consists in finding, for all destinations, paths optimizing all three metrics independently, and respecting the given constraints. We formalize this collection of problems as 2COP. Solving 2COP enables more versatility in terms of optimization strategies and handles heterogeneous constraints for different destinations. Simply put, while DCLC-SR is a one-to-many DCLC variant taking MSD into account, 2COP is more general as it includes all optimization variants.

With initial constraints c_0, c_1, c_2, Best2Cop solves 2COP, i.e. returns in a single run paths that satisfy smaller constraints c'_0, c'_1, c'_2 for any $c'_i < c_i$, $i = 0, 1, 2$, offering more flexibility than simply returning the DCLC-SR solution. More precisely, the problem can be formulated as follows.

Problem 7. *The 2-Constrained Optimal Paths problem (2COP)*
Let $f(M_j, c_0, c_1, c_2, s, d)$ be a function that returns all non dominated feasible segment paths from s to d (if one exists), verifying all constraints $c_i, 0 \leq i \leq 2$ and optimizing $M_j, j \in 0, 1, 2$. For a given source s and given upper constraints c_0, c_1, c_2, 2COP consists in computing

$$2COP(s, c_0, c_1, c_2) = \bigcup_{\substack{\forall d \in V, \\ \forall j \in \{0,1,2\}, \\ \forall c'_j \leq c_j}} f(M_j, c'_0, c'_1, c'_2, s, d)$$

Observe that, for any $s \in V$, DCLC-SR(s, c_0, c_1) consists of the paths in $2COP(s, c_0, c_1, \infty)$ minimizing M_2. Looking at Fig. II.11, we have two interesting examples (we rely on the first capital letter of the

cities):

$$f(M_2, 3, 70, \infty, F, V) = (F, B)|(B, V)\ (67, 4)$$
$$f(M_1, 3, \Gamma, \infty, G, B) = (G, M)|(M, B)\ (77, 4)$$

In the second example, recall that the M1-distances are truncated to obtain integer values and Γ is the maximum c_1 constraint we consider (multiplied by γ). When the delay accuracy allows to reduce the problem's complexity sufficiently, Best2Cop can solve exactly any of the variants within 2COP and return any desired output of the image of f.

Finally, while at first glance, a function returning *all* segment paths, like f, may look impossible to construct as the number of equal cost segment paths may not be polynomial, one only needs to store the DAG of such paths like for basic IP ECMP. On the one hand, within each segment, SR allows the native us of ECMP. On the other hand, by storing locally all equal options, the source can reconstruct the meta DAG for implementing ECMP on the top of SR (and not only the opposite). This is out of the scope of this contribution and we will only consider one arbitrary choice (e.g. the first one or a lexicographical one) to simplify the following. We will discuss this opportunity in the perspectives of the section.

II.4.b Towards an Efficient Exact Solution for Massive Scale Networks (BEST2COPe)

In this section, I will briefly describe Best2Cop, our algorithm efficiently solving 2COP (and so DCLC-SR), and then move quickly on its extension for multi-area networks. Its implementation is available online[58].

II.4.b.1 A Core Algorithm for Flat Networks

Akin to the SR graph computation, Best2Cop can be run on a centralized controller but also by each router. Its design is centered around two properties illustrated in Fig. II.12. First, the graph exploration is performed so that paths requiring i node segments are found at the $i^{th} + 1$ iteration[59], to natively tackle the MSD constraint. Second, BEST2COP's structure is easily parallelizable, allowing to benefit from multi-core architectures with low overhead.

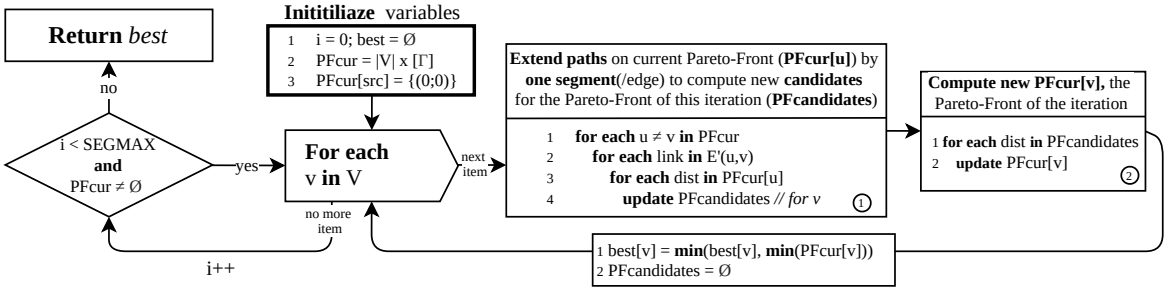


Figure II.12: Best2Cop works by exploring paths of increasing length on G'. Non-dominated paths are The algorithm ends at the MSDth iteration or when progress stops.

Simply put, at each iteration, Best2Cop starts by extending the known paths by one segment (one edge in the SR graph) in a Bellman-Ford fashion (a not-in-place version to be accurate). Paths found during a given iteration are only checked loosely (and efficiently) for dominancy at first. This extension is performed in a parallel-friendly fashion that prevents data-races, allowing to easily parallelize our algorithm. Only once at the end of an iteration are the newly found paths filtered and thoroughly checked for dominancy, to reflect the new Pareto front. The remaining non-dominated paths are in turn extended at the next iteration. These steps only need to be performed $MSD \approx 10$ times, ignoring so all paths that are not deployable through SR. When our algorithm terminates, the results structure

[58] https://github.com/talfroy/BEST2COP

[59] Note that each adjacency segment translates to at least one necessary segment, two if they are not globally advertised and not subsequent.

contains, for each segment number, all the distances of non dominated paths from the source towards all destinations.

The good performance of Best2Cop comes from several aspects. First, the fact that paths requiring more than MSD segments are natively excluded from the exploration space. Second, well-chosen data structures benefiting from the limited accuracy of the delay measurements to limit the number of paths to extend. This allows to manipulate arrays of fixed size, because the Pareto front of distances towards each node is limited to Γ at each step (enabling very efficient read/write operations). Third, using a Bellman-Ford approach allows not only to easily parallelize our algorithm but also to perform lazy efficient update of the Pareto front. Indeed, a newly found path may only be extended at the *next* iteration. Thus, we can efficiently extract the non-dominated paths from all paths discovered during the current iteration in a single pass, once at the end of the iteration. Conversely, other algorithms tend to either check for dominancy whenever a path is discovered (as the later may be re-extended immediately), or not bother to check for dominancy at all, e.g. by relying solely on interval partitioning to limit the number of paths to extend.

II.4.b.2 Scalabity in Massive Scale Network & Area Decomposition

As shown in [LAM+20], this algorithms exhibits great performance on large-scale networks of up to 1000 nodes ($\approx$ 15ms). However, since the design of Best2Cop implies a dominant factor of $|V|^2$ in term of time complexity, recent SR deployments with more than 10000 nodes would not scale well enough. The sheer scale of such networks, coupled with the inherent complexity of TE-related problems, makes 2COP very challenging if not impossible to practically compute at first glance. In fact, even Best2Cop originally exceeds 20s when dealing with $\approx$15000 nodes. As we will see in the evaluations, this is much worse with concurrent options.

We need to extend Best2Cop in order to deal efficiently with massive scale networks. By leveraging the physical and logical partitioning usually performed in such networks, we manage to solve 2COP in $\approx$ 1s even in networks of 100000 nodes. The scalability issues in large-scale networks do not arise solely when dealing with TE-related problems. Standard intra-domain routing protocols encounter issues past several thousands of nodes. Naive network design creates a large, unique failure domain resulting in numerous computations and message exchanges, as well as tedious management. Consequently, networks are usually divided, both logically and physically, in *areas*. This notion exists in both major intra-domain routing protocols (e.g., with OSPF and IS-IS). In the following, we consider the standard OSPF architecture and terminology but our solution can be adapted to fit any one of them.

Areas can be seen as small, independent sub-networks (usually of around 100 - 1000 nodes at most). Within OSPF, routers within an area maintain a comprehensive topological database of their own area only. Stub-areas are centered around the *backbone*, or area 0. *Area Border Routers*, or ABRs, possess an interface in both the backbone area and a stub area. Being at the intersection of two areas, they are in charge of sending a summary of the topological database (the best distance to each node) of one area to the other. There are usually at least two ABRs between two areas. We here (and in the evaluation) consider two ABRs, but the computations performed can be easily extended to manage more ABRs. Summaries of a non-backbone area are sent through the backbone. Upon reception, ABRs inject the summary within their own area. In the end, all routers possess a detailed topological database of their own area and the best distances towards destinations outside of their own area.

Leveraging Area Decomposition This partitioning creates obvious separators within the graph, the ABRs. Thanks to the latter, we can leverage this native partition in a similar divide-and-conquer approach, adapted to the computation of 2COP paths, by running Best2Cop at the scale of the areas before exchanging and combining the results. We do not only aim to reduce computation time, but also to keep the number and size of the exchanged messages manageable.

For readability purposes, we rely on the following notations: $\mathcal{A}_x$ denotes area x. A_x denotes the ABR between the backbone and $\mathcal{A}_x$. When necessary, we may distinguish the two ABR $A1_x$ and $A2_x$. Finally, b2cop$(\mathcal{A}_x, s, d)$ denotes the results (the non-dominated paths) from s to d within $\mathcal{A}_x$. When d is omitted, we consider all routers within $\mathcal{A}_x$ as destination.

We here detail a simple distributed and router-centric variant of our solution. However, our solution

may well be deployed in other ways, e.g. relying on controllers, or even a single one. In such cases, the computation could be parallelized per area if needed. Such discussion is left for future work.

Working at area scale. Due to the area decomposition, routers do not possess the topological information to compute a full, complete SR graph of the whole network. Thus, we make routers only compute the SR graph of their own area(s). Because exchanging the SR graphs themselves implies a large volume of information to share, we instead make the ABRs exchange their 2COP paths (i.e., the non dominated paths to all destinations of their areas) since we limit their numbers to Γ at worst. This exchange still provides enough information for all routers to compute all 2COP paths for every destination. Exchanging the computed 3D Pareto front has a message complexity of $|V| \times c_0 \times \Gamma$ at worst in theory. In practice, we expect both the size of Pareto fronts and the number of relevant destinations to consider to be fairly low ($<< \Gamma$ and $<< |V|$ resp.). In the case of non-scalable Pareto fronts, one can opt for sending only part of them but at the cost of relaxing the guarantees brought by Best2Cop.

After exchanging messages, any ABR A_x should know the non-dominated paths from itself to $A_y, \forall y \neq x$, and the non-dominated paths from A_y to all nodes within $\mathcal{A}_y$. By combining this information, we can compute the non-dominated paths from A_x to all nodes within $\mathcal{A}_y$, as we will now detail.

Cartesian product. Since ABRs act as separators within the graph, to reach a node within a given area $\mathcal{A}_y$, it is necessary to go through one of the corresponding ABRs A_y. It thus implies that non-dominated paths to nodes within $\mathcal{A}_y$ from A_x can be found by combining $\text{bcop}(\mathcal{A}_0, A_x, A_y)$ with $\text{bcop}(\mathcal{A}_y, A_y)$. In other words, by combining, with a simple cartesian product, the local non-dominated paths towards the ABRs of a given zone with the non-dominated paths from said ABRs to nodes within the corresponding distant areas, one obtains a superset of the non-dominated paths towards the destinations of the distant area. In practice, since several ABR can co-exist, it is necessary to handle the respective non-dominated paths ($\text{bcop}(\mathcal{A}_y, A1_y)$ and $\text{bcop}(\mathcal{A}_y, A2_y)$) with careful comparisons to avoid incorrect combinations.

Post-processing and merging. To ensure that the results obtained through the Cartesian product aforementioned are correct, some post-processing is required. When combining segment lists, the latter are simply concatenated. More precisely, the resulting segment list necessarily possesses the following structure: $(u_0, u_1)|\dots|(u_i, A)|(A, v_0)|\dots|(v_{j-1}, v_j)$, with A denoting an ABR. However, A being a separator, it is likely that the best IGP path from u_i to v_0 natively goes through A without the need of an intermediary segment. Thus, segments of the form $(u_i, A)|(A, v_0)$ can often be replaced by a single segment (u_i, v_0). Such anomalies can be easily corrected. should be corrected, as an additional useless segment may render the path falsely unfeasible, even though it actually fits the MSD constraint. This correction can be easily performed as detailed in [LAM+20].

Once performed for all areas, an ABR A_x now possesses all 2COP paths to all considered destinations within the network. These can then be sent to routers within $\mathcal{A}_x$, who will need to perform similar computations to compute non-dominated paths to all routers within a different area. Note that the 2COP paths for each destination can be sent as things progress, so that routers can process such paths progressively (and in parallel) if needed.

II.4.b.3 Complexity and Guarantees

An Efficient Polynomial-Time Algorithm We now provide the complexity of the underlying routines of BEST2COPe.

The flat Best2Cop. In the worst-case, for a given node v, there are up to $degree(v) \times \Gamma$ paths that can be extended towards it. Observe that $degree(v)$ is at least $|V|$ (because G' is complete) and depends on how many parallel links v has with its neighbors. With L being the average number of links between two nodes in G', on average we thus have $degree(v) = |V| \times L \times \Gamma$ paths to extend to a given node, at worst. These extensions are performed for each node v and up to MSD times, leading to a complexity of

$$O(c_0 \cdot \Gamma \cdot |V|^2 \cdot L)$$

Using up to $|V|$ threads, one can greatly decrease the associated computation time.

This yields to the following theorem:

Theorem 12. *BESTCOP solves the 2COP problem in polynomial time*
Given a graph $G = (V, E)$ and a source $s \in V$, Best2Cop *allows s to compute segment lists answering the 2COP problem for any destination $d \in V$ in $c_0 \cdot \Gamma \cdot |V|^2 \cdot L$ operations at worst.*

The Cartesian Product. Its complexity is simply the size of the 2COP solution space squared, for each destination, thus at worst $O((c_0 \cdot \Gamma)^2 \cdot |V|)$. Note that we can reach a complexity of $O(c_0^2 \cdot \Gamma^2)$, again with the use of $|V|$ threads since each product is independent. This worst case is not expected in practice as metrics are usually mostly aligned to result in Pareto fronts whose maximal size is much smaller than $c_0 \cdot \Gamma$.

Overall, `BEST2COPE` (multi-area) exhibits a complexity of

$$O\left(c_0 \cdot \Gamma \cdot \left(c_0 \cdot \Gamma + L \cdot max_{\forall i \in [1..m]}(|V_i|)\right)\right)$$

with V_i denoting the set of nodes in each area i (m being their number) and the use of $|V|$ threads and sufficient CPU resources (this bound is achievable ideally because the load is perfectly balanced and bottlenecks negligible). Note that the cartesian product dominates this worst-case analysis as long as the product $V_i \cdot L$ remains small enough. However, with realistic weighted networks, we argue that the contribution of the Cartesian product is negligible in practice, so `BEST2COPE` is very scalable for real networking cases.

What are the Guarantees ***when the Trueness exceeds the Accuracy****, i.e. if* $t > \gamma$ *?* If propagation delays are measured with a really high trueness (e.g. with a delay grain of 1 μs or less), Best2Cop (and so, BEST2COPE) can either remain exact but slower, or, on the contrary, rapidly produce approximated results. In practice, if one prefers to favor performance by choosing a fixed discretization of the propagation delay (to keep the computing time reasonable rather than returning truly exact solutions), this may result in an array not accurate enough to store all non dominated delay values, i.e. two solutions might end up in the same cell of such an array even though they are truly distinguishable. Nevertheless, we can still bound the margin errors, relatively or in absolute, regarding constraints or the optimization objective of the 2COP variant one aims to solve.

In theory, note that while no exact solutions remain tractable if the trueness of measured delays is arbitrarily high (for worst-case DCLC instances), it is possible to set these error margins to extremely small values with enough CPU power. If $t < \gamma$, each iteration of our algorithm introduces an absolute error of at most $\frac{1}{\gamma}$ for the M_1 metric, i.e. the size of one cell in our array (recall that $\gamma = \frac{\Gamma}{c_1}$ is the accuracy level and is the inverse of the delay grain of the static array used by Best2Cop). So our algorithm may miss an optimal constrained solution p^* only if there exists another solution for the same destination p such that $d_1(p) \geq d_1(p^*)$ but the M_1 distance of both solutions associated to the same integer i.e. only if $d_1(p) \leq d_1(p^*) + \frac{c_0}{\gamma}$. In this case, we have $d_2(p) \leq d_2(p^*)$ because otherwise, p^* would have been stored instead of p. From this observation, depending on the minimized metric, BEST2COP ensures the following guarantees.

If one aims to minimize M_0 or M_2 (e.g. when solving DCLC), then Best2Cop guarantees a solution p that optimizes the given metric, but this solution might not satisfy the given delay constraint $c \leq c_1$. As an example, for DCLC-SR (optimizing M_2), we have:

$$d_0(p) \leq c_0 \qquad \text{(Eq. II.14)}$$

$$d_1(p) < c + \frac{c_0}{\gamma} \qquad \text{(Eq. II.15)}$$

$$d_2(p) \leq d_2(p^*) \qquad \text{(Eq. II.16)}$$

With p^* denoting the optimal constrained solution for a given destination. When minimizing M_1, the solution returned by Best2Cop for the same destination, p, will indeed verify the constraints on M_0 and M_2, and we have $d_1(p) < d_1(p^*) + \frac{c_0}{\gamma}$. The induced absolute error of c_0/γ regarding the delay of paths becomes negligible as the delay constraint increases. If $c \approx c_1$, the latter translates to a small relative error of c_0/Γ. Conversely, it becomes significant if $c << c_1$. When minimizing M_0 or M_2, it is thus recommended to set c_1 as low as possible regarding the relevant sub-constraint(s) $c \leq c_1$ if necessary. Similarly, to guarantee a limited relative error when minimizing M_1, it is worth running our algorithm with a small c_1 as we can have $d_1(p^*) << c_1$. However, note that this later and specific objective (in practice less interesting than DCLC in particular) requires some a priori knowledge, either considering the best delay path without any c_2 and c_0 constraints, or running twice Best2Cop to get $d_1(p)$ as a first

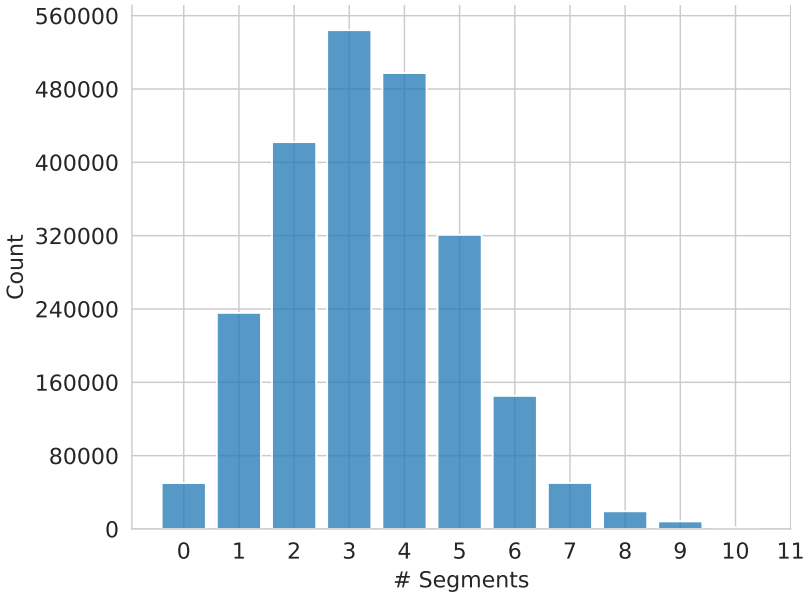


Figure II.13: Required number of segments for all DCLC solutions, in a network of 45000 nodes generated by YARGG, with delay constraints of up to 100ms.

approximation to avoid set up c_1 blindly initially (here c_1 is not a real constraint, only c_2 and c_0 apply as bounds of the problem, c_1 just represents the absolute size of our array and, as such, the accuracy one can achieve).

Even though Best2Cop exhibits strong and tunable guarantees, it may not return exact solutions once two paths end up in the same delay cell, which may happen even with simple instances exhibiting a limited Pareto front. Fortunately, a slight tweak in the implementation is sufficient to ensure exact solutions for such instances [LAM+22] as in [318]. In summary, Best2Cop is efficient and exact to deal with simple instances and/or when $t \geq \gamma$, while it provides approximated but bounded solutions for difficult instances if $t < \gamma$ to remain efficient and so scalable even with massive scale IP networks.

II.4.c Few Segments Required & Computing Time Performance

II.4.c.1 SR is Practically Relevant for Solving DCLC

Given the MSD constraint, one may question the choice of SR for deploying DCLC paths in practice. Indeed, in some cases, in particular if the metrics are not aligned[60], constrained paths may required more than MSD detours to satisfy a stringent latency constraint.

While it has been shown that few segments are required for most current SR usages (e.g. for TI-LFA [61] or when considering only one metric) [108, 31], to the best of our knowledge, there is no similar study for our specific use-case, i.e. massive scale networks with two valuation functions (delay and IGP cost). This is probably one of the most exciting challenge for SR as DCLC is a complex application. However, since such massive-scale computer network topologies are not available publicly, we rely on our own topology generator whose description is available in [LAM+20]. These topologies follow a standard OSPF-like area division. and both metrics (delay and IGP cost) follow a realistic pattern. For this analysis, we opt for a worst-case graph having $\approx$ 45000 nodes and $\approx$ 92000 edges scattered in 140 areas.

For this analysis, we keep track, for each destination, of all the solutions solving DCLC for all delay constraints up to 100ms, and extract the necessary number of segments. In other words, we show the

[60] The delay and the IGP costs in particular. Since node segments represent best IGP paths, the IGP cost and the number of segments will most likely be aligned by design

[61] Topology Independent Loop Free Alternate

number of segments required to encode all non-dominated (and thus practically relevant for some given constraint) paths, considering all delay constraints up to 100ms. The results are shown in Fig. II.13.

One can see that most paths require less than 10 segments, meaning that performant hardware should be able to deploy most DCLC paths. However, some corner-cases requiring more than 10 segments do exist, probably arising from stringent delay constraints. In addition, less performant hardware (e.g. with MSD ≈ 5), while able to deploy the majority of DCLC paths, can not deploy *any* DCLC path. Note that several mechanisms exist to bypass this limit. Flexible Algorithms [275] allows to compute shortest paths and create segments with other metrics (e.g. the delay). Binding segments [107] allows to "compress" a segment list in a single segment, which is uncompressed when popped from the list. However, both techniques increases the message exchange, number of states to maintain, and overall complexity. Their usage should thus be limited to a few corner cases.

Consequently, our analysis exhibits two main points. First, SR is appropriate to deploy TE paths. Indeed, the majority of DCLC paths should be deployable within the MSD constraint, if not all when using performant hardware. Second, since there may however exist DCLC paths requiring more than MSD segments, this limit must be considered to compute feasible paths correctly. Otherwise, a single non-feasible path dominating feasible ones is enough to lead to an incorrect algorithm. The underlying path computation algorithm must then efficiently consider delay, IGP cost and the number of segments to ensure its correctness.

II.4.c.2 Best2Cop exhibits Great Performance

To conclude, we now evaluate Best2Cop on various flat network instances, ranging from worst-case scenario to real topologies, and compare it to another existing approach based on the *Dijkstra* algorithm, SAMCRA [336]. More details can be found in... In this analysis we consider our discretization to be exact (i.e. Γ is high enough to prevent loss of relevant information) for both approaches.

Our Experimental Setup is the following

- $c_0 = MSD = 10$, as it is close to the best hardware limit;
- $L = 2$: while some pairs of nodes may have more than two parallel links connecting them in G, we argue that, on average in G', one can expect that the total number of links in E' is lower than $2|V|^2$.
- $\Gamma = 1000$, although this value is tunable to reflect the expected product trueness-constraint on M_1, we consider here a fixed delay grain of 0.1ms (so an accuracy level of $\gamma = 10$) regarding a maximal constraint $c_1 = 100$ms. This Γ limitation is realistic in practice and guarantees the efficiency of Best2Cop even for large complex networks as it becomes negligible considering large $|V|$.

Note that the delays fed to SAMCRA are *also* discretized in the same fashion as for Best2Cop, allowing the number of non-dominated paths that SAMCRA has to consider to be bounded and reduced. In addition, as SAMCRA is not designed with the SR Graph in mind, it is difficult to know which of the two methods is the most suited to consider the segment metric. Thus, we compare ourselves to both variants. First, we run SAMCRA on the fully-meshed SR Graph, which allows to use the SAMCRA algorithm nearly as-is. We call this variant SAMCRA-srg. Second, we implement our conversion algorithm, which allows to efficiently convert multi-metric paths to segment lists. This method requires however further modification of the SAMCRA algorithm, not only by adding the conversion algorithm but also by extending its dominancy checks. We refer to this variant as SAMCRA-lca. All our experiments are performed on an Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz $\times$ 8.

Computing Time & Comparisons for Flat Networks We evaluate the performance of our algorithm using three flat network scenarios. In particular, we do not take advantage of any area decomposition to mitigate the computing time.

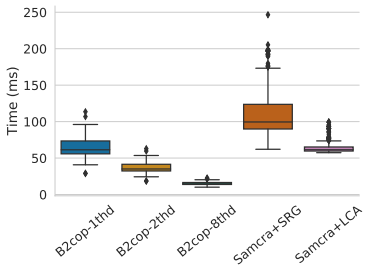
As shown in [LAM+20] by forcing Best2Cop to explore its full iteration space, our algorithm *cannot* exceed 80s at worst on topologies of 1000 nodes. This upper bound can however be drastically reduced

through the use of multi-threading, reaching a worst-case of $\approx 10s$ when relying on 8 threads, highlighting the parallel nature of our algorithm.

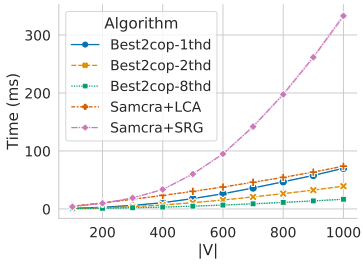
In practice, Best2Cop is far from reaching these upper bounds, even on random networks. Let us evaluate Best2Cop and compare it to SAMCRA in two main scenarios: a real network with real link valuations, and random networks of up to 10000 nodes.

Figure II.14: Computation time of Best2Cop and SAMCRA on various experiments. Although the results can be close when considering mono-threaded Best2Cop and SAMCRA, our algorithm always outperforms its competitor when using multi-threading. In some cases, multi-threading is not even necessary

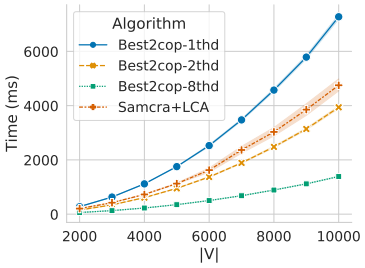
(a) Computation time of Best2Cop (1, 2, 8 threads) & SAMCRA on a real network with real link valuation.



(b) Computation time of Best2Cop & SAMCRA on random networks with random valuation for $|V| = 100$ to 1000



(c) Computation time of Best2Cop & SAMCRA on random networks with random valuation for $|V| = 1000$ to 10000



Real network. We start by considering a real IP network topology. We use our largest available ISP topology, consisting of more than 1100 nodes and 4000 edges. This topology describes the network of a Tier-1 operator and is not available to the public. While the IGP costs of each link were available, we do not have their respective delays. We thus infer delays thanks to the available geographical locations we do possess: we set the propagation delays as the orthodromic distances between the connected nodes divided by the speed of light, and run both algorithms on the obtained topology. The execution times are then shown in Fig. II.14: Best2Cop (1, 2, 8 threads as mentioned in the caption of the figures) and SAMCRA (with LCA and SRG respectively) are run for every node as source, resulting in the distributions showcased.

One can see that SAMCRA-srg (i.e. SAMCRA run directly on the SR Graph) exhibits the worst execution times out of all the algorithms and variants presented, averaging at 100ms, and reaching 250ms at worst. Interestingly, this shows that exploring the SR Graph itself may be detrimental to some algorithms (in particular priority-queue-based ones) due to its high density. Hence, algorithms not designed to take advantage of its features may fare better by exploring the original, sparser topology, and using the information within the SR Graph to compute the number of necessary segments to encode the paths being explored. This is visible on SAMCRA-lca computation times. Our construct, coupled with our conversion algorithm, allowed SAMCRA-lca to reach computation times very similar to the mono-threaded variant of Best2Cop, with an average execution time of $\approx$ 60ms. Note that Best2Cop, which runs on the SR Graph itself, shows equivalent execution time when relying on a single thread. However, when relying on multiple threads, Best2Cop outperforms its competitor in all runs, reaching a computation time of $\approx$ 25ms at worst when using 8 threads, i.e. three times faster than its competitor. These low execution times are not only due to the efficiency of the algorithms presented, but also to the realistic link valuations, which tend to be correlated in practice. In realistic cases, Best2Cop can thus work with $\Gamma > 1000$ and so with a supported accuracy $t >> 0.1$ms (to deal with a micro-second grain) for small enough delay constraint (i.e., $<<$ 100ms), while keeping the execution time in the hundreds of milliseconds. One may notice that (almost) perfectly aligned metrics reduce the usefulness of any DCLC-like algorithm, but such metrics are not always aligned for all couples in practice (even with realistic cases, we observe that the average size of the 3D Pareto front is strictly greater than 1, typically ≈ 4). Our algorithm deals efficiently with easy cases and remains exact and efficient for more complex cases, e.g. with random graphs.

Random networks. The number of publicly available large topologies being limited, we continue our

evaluation with random scenarios to assess the computation time of the aforementioned algorithms on a larger number of scenarios.

We generate raw connected graphs of $|V|$ nodes by using the Erdos-Rényi model. The generated topologies have a degree of $log(|V|)$. Both the delays and the IGP weights are picked uniformly at random. IGP weights are chosen within the interval $[1, 2^{32}/|V|/10]$, to ensure that no paths possesses a cost higher than 2^{32}. Delays are chosen within the interval $[0, 0.01 \times c_1]$, with $c_1 = 100ms$, to ensure that a high number of feasible paths exist.

We start by running Best2Cop and SAMCRA for $|V|$ ranging from 100 to 1000 (with steps of 100). To account for the randomness of both valuation functions, we generate 30 differently weighted distinct topologies for each value of $|V|$. We run Best2Cop and SAMCRA for 30 nodes selected as representative sources (randomly picked uniformly). Computing times are shown in Fig II.14.

While the computation times are slightly higher (due to the random valuations which lead to a higher number of non-dominated paths), the results are similar to the previous experiment. These results display more clearly that SAMCRA does not benefit from exploring the SR Graph. Indeed, on random networks, SAMCRAsrg is about 7 times slower than the other algorithms displayed. However, as on real networks, SAMCRA-lca shows results close (if not equal) to Best2Cop execution time. Nevertheless, even on random networks, Best2Cop remains three times faster than its competitor when relying on 8 threads.

Interestingly, Best2Cop mono-threaded and SAMCRAlca computation times get closer as $|V|$ increases. Thus, we continue our comparison on networks of 2000 to 10000 nodes. Given the long computation times of SAMCRA-srg, we here only consider SAMCRA-lca. The results are shown in Fig. II.14. On such networks, Best2Cop (mono-threaded) exhibits an execution time of $7s$, while SAMCRA-lca remains under $5s$. The quadratic complexity of Best2Cop (whose main factor is $|V|^2$) is here clearly visible. SAMCRA-lca exhibits a less steep growth. However, when relying on multiple-thread, Best2Cop remains far more efficient. While two threads already allow to reach an execution time slightly lower than SAMCRA (4s), 8 threads allow Best2Cop to remain ≈ 3.3 times faster than its competitor.

Title of the publication	Name of the venue	Year	Reference
Computing Delay-Constrained Least-Cost Paths for Segment Routing is Easier Than You Think	Network Computing and Applications (**NCA**)	2020	[LAM+20]
Deploying Near Optimal Delay Constrained Paths with Segment-Routing in Massive Scale Networks	Computer Networks (**COMNET**)	2022	[LAM+22]

Table II.9: Summary of my publications related to multi-metric routing with SR

Table II.9 lists the two publications related to this topic.

BEST2COP: Conclusions and Perspectives

While the overhead of MPLS-based solutions lead to a TE winter in the past decade, Segment Routing marked its rebirth. In particular, SR enables the deployment of a practical solution to the well-known DCLC problem. Our algorithm, Best2Cop [LAM+20] (Best Exact Segment Track for 2-Constrained Optimal Paths), iterates on the SR Graph to natively solve DCLC in SR domains with strong guarantees, through simple and efficient data structures and concepts. Best2Cop performs very well with multi-threading and it can also be used in several other contexts thanks to its versatility in handling many variants of the initial problem with 2COP.

With our ongoing work, we went several steps further with the following achievements:

- experimentally demonstrating that SR is a relevant technology to deploy DCLC paths;
- we extend Best2Cop to BEST2COPe for massive scale ISPs relying on area-subdivision, partitioning 2COP into smaller sub-problems to limit its overall complexity (time, memory and churn);
- through extensive evaluations, relying on multi-threading and our own multi-metric/multi-areas network generator, we have shown that BEST2COPe is very efficient in practice. This was confirmed through a comparison with a relevant state-of-the-art algorithm, which benefited from a novel path to segment multi-metric conversion algorithm that we designed.

To the best of our knowledge, Best2Cop is the first practically exact and efficient solution for 2COP within SR domains, making it the most practical candidate to be deployed for such a TE flavor in today ISPs. It is able to solve 2COP on massive scale realistic networks having 100000 nodes in less than a second. For large areas having thousands of routing devices, we have shown that Best2Cop can easily deal with random topologies while its competitors do not scale. We envision to improve our evaluations by considering more graph diversity (e.g. generating large Pareto Fronts). More advanced and flexible structures can be envisioned to deal with high trueness requirements, while deploying novel flex-algo strategies can help to mitigate the rare SR limit drawbacks.

We also currently refine our two options to encode paths into segment lists, either natively considering SR virtual edge (relying directly on the SR graph), or converting newly found paths to check whether they should be extended regarding this extra metric (outside the raw graph). We are currently looking at an interesting question: in which cases, adding an additional dimension to the problem (with a hop count kind of metric) can ease its resolution with Best2Cop? In any cases, a structure similar to a SR graph is required as any path in the APSP may be visited to build a (long) segment list. I envision to continue working on these conversion models to compare them considering diverse graph properties and implement both earliest and exit approaches in the multi-metric case.

With SR, we have made the choice to consider a source routing architecture, where sources are edge routers sharing BGP knowledge and core routers not aware of the underlying constrained paths and user requirements. However, Best2Cop may also work without considering this source routing model and can be also one of the most existing efficient option regarding a fully distributed approach. With this hop by hop paradigm, the (final) destination is still not enough to forward a packet as the contraint has to be updated along the route to select the right path(s). Several intermediary options are possible to distribute the knowledge and routing states within the network: how to build sub-areas logically rather than rely on existing ones?

Finally, I aim to study the opportunity to load balance the flows among *all the ECMP aware* Pareto Front considering source routing. Indeed, with source routing the load balancing looks complex at first glance: how to control the possibly exponential number of ECMP routes from the source? While the Pareto Front is limited to Γ thanks to the delay approximation, ECMP can still lead in an overall DAG of segments with many combinations. One interesting option is to let the source router choose randomly, among all routes within the DAG, which packet are forwarded for which route. One needs to ensure the path congruence to preserve TCP flows: all packets of a given (sub)flow should be forwarded via the same route (as in MP-TCP[62]). A route is a succession of next-hops in the DAG that the source can easily store. Hence, hashing a given set of fields in the header of each packet to return $h = hash(< fields... >)$, the source can then interpret h in many manners and derive from its content an arbitrary random list of next-hops (in the SR graph), and thus encode them as segments. Instead of only applying ECMP locally in each segment, the source (typically a Provider Edge Router, PER) can thus also control the meta-ECMP DAG of SR routes (they indeed form a overlay DAG whose links are one hop segment). Manipulating ECMP at the sources, i.e. on ingress PER, come with several advantages, in particular it eases the traffic control to avoid inconsistent decisions. Such sources can take their own decisions locally, or exchanging/sharing traffic information among them. I envision to explore this architecture to assess its performance in a distributed congestion control system.

[62] https://www.multipath-tcp.org/

II.5 Other Works, General Discussions and Perspectives

Selecting and Controlling Transport Paths

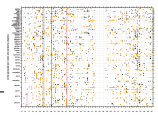
Multipath routing (with ECMP or more relaxed rules) not only enables fast rerouting and persistent loop-free convergence, it also allows for distributing the traffic among the ressource available in the network. When looking at load balancing, several options exist to control the load balanced paths. In [DPvdL+13], we have explored several possible ways to encode such multiple routes in data center networks and show one can finely control the mapping between its set of flows and the deployed routes. Our solution relies on a small label contained in the header of packets (e.g. their transport ports) that routers use to determines their best next-hop (although the load balancing is performed randomly with current ECMP implementation). More generally speaking, this is interesting to look at interactions between MP-TCP, with their underlying congestion control mechanisms, and load balancing or simply routing changes. Because of the poor performance of TCP when using per packet load balancing [Otto, 2016] on routes with distinct characteristics (e.g. the propagation delay), this is not trivial to implement correctly. The interactions between TCP and the routing has to be look in details to avoid interferences and performance degradation. I think there exists many opportunities to improve the state of the art between these two layers. In particular, the transport layer should rely on more routing signals than the ones currently exploited and may benefit from more options and diversity like with Best2Cop.

Overall Conclusion and Future Works

In this chapter, we have reviewed diverse techniques that I have co-designed to enhance and improve IGP routing with new features in general. With my co-authors I have addressed several challenges in the field, ranging from re-routing, loop-free re-configurations and convergence in general to multi-metric problems. We have detailed the motivations and the objectives (robustness and diversity) and their related problems by developing properties of shortest paths considering several assumptions (on the context and the link valuation in particular). Using similar graph transformation and constructs, we have proposed efficient algorithms to solve such problems and position them regarding existing works. Our visible publications in these topics (II.3, II.6, and II.9) show the relevance and the quality of the work accomplished in this area of networking, in particular our will to provide a technically detailed analysis of our algorithms and their properties. We also discussed each specific extension considered for each topic independently.

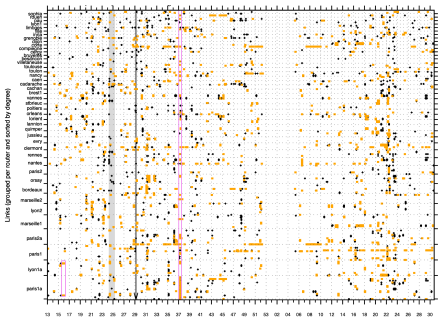
I now aim to develop them as a whole to improve their strengths. In particular, considering symmetric weights, we will see in Chapter V how it eases the construction of a safe but very fast convergence architecture at the data-plane. We will continue into the direction considered in section II.1: using SR as baseline, I will show that it is possible to both avoid any anomalies (transient forwarding loops) and provide an almost instantaneous re-convergence. Generally, these two objectives are handled in an orthogonal way but it is possible to conciliate them with a careful design. To go one step further than basic link state changes, we will explore the opportunity to protect an entire router (failure and not only maintenance) with an ideal architecture at the data-plane. We will do the same for *OPTIC* and the transit BGP traffic. Both options are good candidates to fit within current flexible hardware.

To reach these objectives, I envision to build a prototype. The goal is to emulate the obtained behavior in a controllable environment. We will have to develop the underlying detection triggers with passive measurements and propose efficient primitives to switch from pre- to post-convergence paths. Such a technical implementation will require large efforts but will be a great source of knowledge to understand missing pieces. From a more theoretical perspective, relaxing our assumptions and looking for more complex failure cases and models in massive scale networks, should be an interesting playground. How to construct updates with segment lists longer than two and more complex conditions (not only local)? How balancing the traffic from a source towards a possibly exponential number of paths with SR? Many challenges still need to be addressed. Finally, I envision to not only rely on graph theoretical constructs but also a more general distributed analysis to guarantee self stabilization with challenging adversarial models introducing errors more complex than simple failures to deal with the link flapping issue in particular (as discovered in section III.2).



Chapter III

IP Measurements or the Art of Characterizing Computer Networks



Measuring and monitoring IP networks is a critical task. Not only in itself to help to control, fix and reconfigure the network, but also, with a longer term perspective, to better **understand inherent limitations of deployed protocols and architectures**. As such, measurements are helpful to troubleshoot and guide routing proposals towards better models.

In this chapter, I present three of my main contributions in the field. That is first studying MPLS, the de facto technology in use for deploying TE in nowadays IP networks. In particular we develop an efficient probing tool, TNT, able to reveal some hidden parts of the Internet due to **invisible MPLS tunnels**. Second, with DCART, we deploy a multi-source monitoring infrastructure in RENATER, the french National Research and Educational Network (NREN), whose main goal is to exhibit the nature of **correlations between routing transitions, packet losses and forwarding loops**. Third, we study the existence of FD in the wild, that is forwarding routes not being optimal regarding the IGP metric in use (while their internal selection should be *isotonic*). Finally, we will briefly fly over contributions related to topology discovery and related analysis in a last section.

Contents

III.1 A Close Look at MPLS Networks

This work started with a joint collaboration with Benoit Donnet (University of Liège), Jean-Jacques Pansiot (my former Ph. D. director) and Matthew Luckie (CAIDA) with the paper [DLMP12]. Then, with the Ph. D. thesis of Yves Vanaubel [Vanaubel, 2018] on this topic (under the guidance of Benoit Donnet), we extend the project in many directions and other collaborations that were fruitful in term of publications (including 3 IMC venues, the best conference in this field). Notably, we study MPLS in IPv6 [VMPD16], reveal hidden tunnels in [VMPD17] and study the TE usage [VMPD15] in ISP networks. Thanks to those probing campaigns we also develop router fingerpinting techniques [VPMD13]. Here we will mainly develop the last work presented in TMA and its extended journal version TNSM in 2019: [VLM+19] (we won multiples awards[1]) and [LVM+19]. Our aim is to reveal invisible MPLS tunnels with efficient probing tools. The table of content of this section is the following:

III.1.a Classifying MPLS Tunnels

Despite the numerous attempts to map the Internet and the work done so far in general in this area [246, 247], a lot of issues still remain [19], especially in data collection processes based on `traceroute` (the usual basis to obtain either router-level or AS-level maps). For instance, collecting data about Layer-2 devices inter-connecting Layer-3 routers and so clean the bias and interferences they generate in the Layer-3 maps is still an open question (in particular to collect physical Layer-1 graphs), although it has been already and partially addressed previously with the use of IGMP-based probing [MDBP10]. Nowadays, `mrinfo`, the underlying tool enabling this specific study, is largely deprecated (although we have tried to extend it as explained in Sec. III.4 where we also provide a summary of its advantages along with main results we extract thanks to them). Another example that complexify the original Internet paradigm (all layers of the TCP-IP pile or the OSI model are supposed to be independent and almost hermetic between themselves) is the relationship between traditional network hardware and the so-called middleboxes [91, 99]. Last but not least, MPLS tunnels [287] also have an impact on topology discovery as they allow operators to hide internal hops, as highlighted by our preliminary works in that field [DLMP12, VMPD17]. This is true both with the SR control-plane (when SR relies on MPLS instead of SRv6) and LDP or with the RSVP-TE management.

This leads us with the following questions and challenges:

Research Question

How Measure and Analyze MPLS in the Wild, and What is its Impact on Topology Discovery? How to Reveal Hidden Hops Lying in Invisible Tunnels?

In this work, we go further than previous studies and focus on the detailed interactions between `traceroute` and MPLS, including relevant aspects that were omitted or just neglected (underestimate) in our previous work [VMPD17].

[1] Including the best paper award that leads to a fast-track publication in TNSM and we also have been selected for the best data set runner and finally received the best paper award of the year in 2019 within the IEEE ComSoc Internet TC (ITC) community.

In a nutshell, MPLS has been designed to reduce the time required to make forwarding decisions thanks to the insertion of *labels* (called *Label Stack Entries*, or LSE) before the IP header. In an MPLS network, packets are forwarded using an exact match lookup of a 20-bit value found in the LSE. At each MPLS hop, the label of the incoming packet is swapped with its associated outgoing label (such a mapping being defined in a specific MPLS switching table). The MPLS forwarding engine is lighter than the IP one, as performing an exact match for a label is simpler than retrieving the longest matching prefix for an IP address.

Some MPLS tunnels may be visible to `traceroute` because MPLS routers are able to generate ICMP `time-exceeded` messages when the MPLS TTL expires. Since the ICMP message embeds the LSE, the presence of the tunnel is then obvious as pointed in [317] and [DLMP12]. However, MPLS supports optional features that make tunnels more or less invisible to `traceroute`. Such features modify the way routers process the IP and MPLS TTL of a packet. By carefully analyzing MPLS related patterns based on TTL values (e.g., the quoted TTL or the returned TTL of both error and standard replies), one can identify and possibly discover L3-hops hidden within an MPLS cloud. In [VMPD17], we have already proposed a first attempt for revealing so-called Invisible tunnels and then improve, correct and extend it with [VLM+19] and finally [LVM+19] (the final journal long version).

III.1.a.1 Background: MPLS Basics and Control Plane Operations

This section provides the minimal technical background to enter the topic and understand our main contributions. MPLS routers, i.e., *Label Switching Routers* (LSRs), exchange labeled packets over *Label Switched Paths* (LSPs). In practice, those packets are tagged with one or more *label stack entries* (LSE) inserted between the frame header (data-link layer) and the IP packet (network layer). Each LSE is made of four fields: an MPLS label used to forward the packet, a Traffic Class field (for quality of service, priority, and Explicit Congestion Notification [28]), a bottom of stack flag bit (to indicate whether the current LSE is the last in the stack [286])[2], and a time-to-live field (*LSE-TTL*) having the same purpose as the IP-TTL field [16] (i.e., avoiding routing loops).

Labels may be allocated through the *Label Distribution Protocol* (LDP) [29]. Each LSR announces to its neighbors the association between a prefix in its routing table and a label it has chosen for a given *Forwarding Equivalent Class* (a FEC is a destination prefix by default), populating so a *Label Forwarding Information Table* (LFIB) in each LSR. LDP is mainly used for scalability reasons (e.g., to limit BGP-IGP interactions to edge routers) and to avoid anomalies for the transit traffic such as iBGP deflection issues. Indeed, LDP deploys tunnels following the same routes as the IGP. Labels can also be distributed through RSVP-TE [35] when MPLS is used for Traffic Engineering (TE) purposes. In practice, most operators deploying RSVP-TE tunnels also use LDP [VMPD17] as an underlying default labeling protocol.

With LDP, MPLS has two ways of binding labels to destination prefixes: (*i*) through ordered LSP control (default configuration of Juniper routers [36]) and, (*ii*) through independent LSP control (default configuration of Cisco routers [86, Chap. 4]). In the former mode, an LSR only binds a label to a prefix if it is local (the LSR is the exit point of the LSP), or if it has received a label binding proposal from the IGP next hop towards this prefix. This mode is thus iterative as each intermediate upstream LSR waits for a proposal from its downstream LSR, building thus the LSP from the exit to the entry point. Juniper routers use this mode as default and only propose labels for loopback IP addresses.

In the second mode, the Cisco default one, an LSR creates a label binding for each prefix it has in its RIB, even if it is not directly connected to it. This label binding is then distributed to its neighbors. This mode does not require any proposal from downstream LSRs. Consequently, a label proposal is sent to all neighbors without ensuring that the LSP is enabled up to the wanted exit point. LSP setup takes less time but may lead to uncommon situations in which an LSP can end abruptly before the supposed exit point of the tunnel.

The last LSR towards an FEC is the *Egress Label Edge Router* (the Egress LER – PE_2 in Fig. III.1). Depending on its configuration, two labeling modes may be performed. The default mode [VMPD17] is *Penultimate Hop Popping* (PHP), where the Egress advertises an Implicit NULL label (label value of 3 [286]). In this case, the previous LSR (*Penultimate Hop LSR*, PH LSR – P_3 in Fig. III.1) – is in

[2]To simplify the presentation we will assume only one LSE in this section. This simplification is reasonable as the vast majority of collected tunnels only carry one label (i.e. more than 95% of the cases excluding VPRN usages).

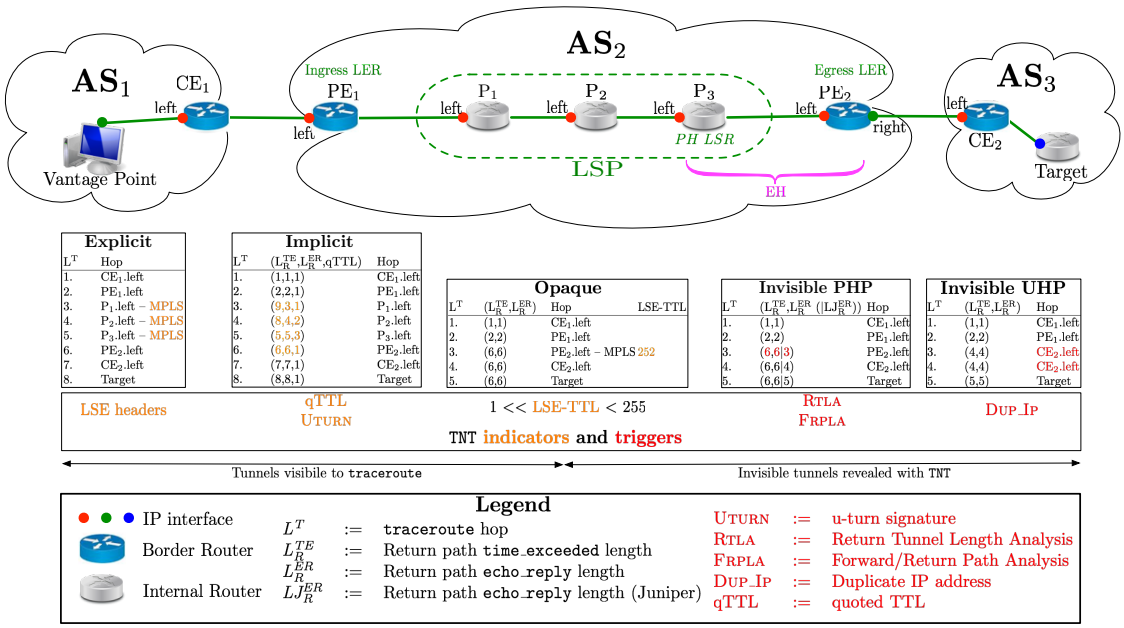


Figure III.1: Illustration of MPLS vocabulary and relationship between MPLS and `traceroute`. The figure is made of three parts. The upper part represents the network topology used throughout the section to illustrate MPLS and `TNT` concepts. In particular, with respect to MPLS, P_1 is the LSP First Hop (FH), while P_3 is the Penultimate Hop LSR (PH LSR). In case of PHP, P_3 is the Ending Hop (EH) responsible for removing the LSE, while, with UHP, it is the Egress LER (PE_2). The middle part of the figure presents our MPLS classification. Finally, the bottom part of the figure provides triggers and indicators of an MPLS tunnel presence when probing with `TNT`. The relationship between the trigger/indicator and the observation made with probing is provided in red. Additional information (e.g. `time-exceeded` path length) are provided to illustrate `TNT`.

charge of removing the LSE to reduce the load on the Egress. In the *Ultimate Hop Popping* (UHP) mode, the Egress LER advertises an Explicit NULL label (label value of 0 [286]). In this case, The PH LSR will swap the current label with an Explicit NULL label and the Egress LER will be responsible for its removal. Labels assigned by LSRs other than the Egress LER are distinct from Implicit or Explicit NULL labels. The *Ending Hop LSR* (EH) is the LSR in charge of removing the LSE, it can be the PH LSR in case of PHP, or the Egress LER in case of UHP[3].

III.1.a.2 Label Switching in a Nutshell: MPLS Data Plane and TTL processing

Depending on its location along the LSP, an LSR applies one of the three following operations:

- Push: the first MPLS router (the tunnel entry point) pushes one or several LSEs in the IP packet, turning it into an MPLS one. The *Ingress Label Edge Router* (Ingress LER) associates the packet FEC to its LSP;
- Swap: within the LSP, each LSR makes a label lookup in the LFIB, swaps the incoming label with its corresponding outgoing label, and sends the MPLS packet further along the LSP;
- Pop: the EH, the last LSR of the LSP, deletes the LSE, and converts the MPLS packet back into an IP one. The EH can be the *Egress Label Edge Router* (the Egress LER) when UHP is enabled or the PH LSR otherwise.

*LSP Entry Behavior (*Push*)* When an IP packet enters an MPLS cloud, the Ingress LER binds a label to the packet thanks to a lookup into its LFIB, depending on the packet FEC, e.g., its IP destination prefix. Before pushing the LSE into the packet, the Ingress LER has to initialize the LSE-TTL. Two

[3]Note that, in the case of independent LSP control, any LSR can be in charge, despite itself, of the popping operation when the tunnel ends abruptly.

behaviors can then be configured: either the Ingress LER sets the LSE-TTL to an arbitrary value (255, `no-ttl-propagate`) or it copies the current IP-TTL value into the LSE-TTL (`ttl-propagate`, the default behavior). Operators can configure this operation using the `no-ttl-propagate` option provided by the router manufacturer [16]. In the former case, the LSP is called a *pipe LSP*, while, in the latter case, a *uniform* one.

Once the LSE-TTL has been initialized, the LSE is pushed on the packet that is sent to an outgoing interface of the Ingress LER. In most cases, except for a given Juniper OS (i.e., Olive), the IP-TTL is decremented before being encapsulated into the MPLS header.

*LSP Internal Behavior (*Swap*)* Upon an MPLS packet arrival, an LSR decrements its LSE-TTL. If it does not expire, the LSR looks up the label in its LFIB. It then swaps the top LSE with the one provided by the LFIB. The operation is actually a swap only if the outgoing label returned by the LFIB is neither Implicit NULL nor empty[4]. Otherwise, it is a Pop operation as described in the next subsection. Finally, the packet is sent to the outgoing interface of the LSR with a new label, both according to the LFIB.

If the LSE-TTL expires, the LSR, in the fashion of any IP router, forges an ICMP `time-exceeded` that is sent back to the packet originator. It is worth to notice that an LSR may implement RFC 4950 [56] (as should be the case in all recent OSes). If so, the LSR will quote the full MPLS LSE stack of the expired packet in the ICMP `time-exceeded` message.

ICMP processing in MPLS tunnels varies according to the ICMP type of message. ICMP *Information messages* (e.g., `echo-reply`) are directly sent to the destination (e.g., the originator of the `echo-request`) if the IP FIB allows for it (otherwise no replies are generated). On the contrary, ICMP *Error messages* (e.g., `time-exceeded`) are generally forwarded to the Egress LER that will be in charge of forwarding the packet through its IP plane [DLMP12].

*LSP Exit Behavior (*Pop*)* Upon the MPLS packet arrival, the EH decrements the LSE-TTL. If this TTL does not expire, the EH then pops the LSE stack after having determined the new IP-TTL.

Using PHP comes with the advantage of reducing the load on the Egress LER, especially if it is the root of a large reverse LSP-tree. Indeed, when using PHP, the last MPLS operation (i.e., Pop) is performed one hop before the Egress LER, on the PH LSR. On the contrary, UHP[5] is generally used only when the ISP implements more sophisticated traffic engineering operations or wants to make the tunnel content and semantics more transparent to the customers (e.g., for VPRN purposes).

When a packet exits the tunnel, the router is left with a packet containing two TTLs: the IP-TTL, and the LSE-TTL. It thus has to decide which TTL should be kept and copied in the IP header before forwarding the packet as a standard IP packet. To ensure that the outgoing TTL cannot be greater than the incoming one, the EH would theoretically have to consider the configuration of the Ingress LER. If the Ingress LER has activated the `no-ttl-propagate` option, the EH should pick the IP-TTL of the incoming packet while the LSE-TTL should be selected otherwise. Indeed, in the former case, because the tunnel is hidden, the LSE-TTL was initialized at 255 and is likely superior to its IP counterpart. Consequently, the EH should select the IP-TTL to ensure a monotonic decrement. In the latter case, the LSE-TTL was initialized at the value held by the IP-TTL, and is thus necessarily smaller than the IP-TTL upon exiting the tunnel as it now takes into account the MPLS hops. Consequently, the EH should here select the LST-TTL to ensure a monotonic decrement. In both cases, the TTL behavior remains monotonic. In order to synchronize both ends of the tunnel without any message exchange, two mechanisms might be used to select the IP-TTL at the EH:

1. applying a **Min(IP-TTL, LSE-TTL) operation**, i.e., selecting the TTL which holds the smallest value. This is the solution implemented for Cisco PHP configurations [86];

2. assuming that the Ingress configuration (`ttl-propagate` or not) is the same as the local configuration. This is the solution implemented by some JunOS and also in some Cisco UHP configuration.

[4] In practice the actual label used for the forwarding is then greater than or equal to 0 (this specific value being reserved for Explicit NULL tunnel ending, i.e. for UHP) but excluding by design the reserved value 3 that is dedicated for Implicit NULL.

[5] The UHP feature has been recently made available on Juniper routers when LSPs are set with LDP. However, PHP remains the rule on Juniper [112, Chap. 1].

Applying the Min(IP-TTL, LSE-TTL) seems to be the best option, as it correctly supports heterogeneous `ttl-propagate` configurations while mitigating forwarding loops without exchanging signalization messages. This Min(IP-TTL, LSE-TTL) operation might be used to detect the presence of hidden MPLS tunnels [VMPD17]. Indeed, it is likely that the ICMP `time-exceeded` message generated by the EH will enter the same MPLS cloud immediately to reach the vantage point.

In that case, when the reply leaves the MPLS cloud, its IP-TTL will not have been decremented, while the LSE-TTL will take the number of hops within the MPLS tunnel into account. Consequently, the EH of the return path (P_1 in Fig. III.1) will choose to copy the LSE-TTL in the IP-TTL, as the IP-TTL of the reply still holds its maximum value. Thus, while the forward path through the hidden MPLS cloud has no effect on the IP-TTL of the packet, the return path is taken into account, as the PH LSR of the return path (P_1), copies the LSE-TTL within the IP-TTL.

This MPLS behavior strongly depends on the implementation and configuration. For instance, on some Juniper OS routers or when the UHP option is activated on some Cisco IOS, the Min(IP-TTL, LSE-TTL) operation is not systematically applied. The EH assumes an homogeneous propagation configuration among LERs. When it is not the case (`ttl-propagate` at one end of the tunnel and `no-ttl-propagate` at the other end), the EH will use the IP-TTL instead of the LSE-TTL, leading to a so-called *jump* effect with `traceroute`. In other words, as many hops as the LSP length are skipped after the tunnel by `traceroute`, the TTL of the packet is brought back to the value it held before going through the LSP. Except when explicitly stated, we will consider homogeneous configurations (e.g., `ttl-propagate` on the whole tunnel).

III.1.a.3 Measure MPLS: from Visible Tunnels to Hidden Ones

According to whether LSRs implement RFC4950 (i.e., ICMP `time-exceeded` quoting MPLS LSE) or not and whether they activate the `ttl-propagate` option or not, MPLS tunnels are more or less visible to `traceroute` [DLMP12].

Explicit tunnels are tunnels with RFC4950 and the `ttl-propagate` option enabled. As such, they are fully visible with `traceroute`, including the labels used along the LSP. ***Implicit*** tunnels also enable the `ttl-propagate` option but do not implement the RFC4950. IP level information is not missing but LSRs are seen as ordinary routers; leading to a lack of "semantic" in the `traceroute` output. ***Opaque*** tunnels are partially obscured from `traceroute` as the `ttl-propagate` option is disabled while the RFC4950 is implemented. Moreover, an Opaque LSP ends at its EH with a non-terminating label. Consequently, the EH is the only hop being seen as an MPLS one while the internal content of the LSP is totally hidden. Finally, ***Invisible*** tunnels are fully hidden as the `no-ttl-propagate` option is enabled and the LSP ends properly (RFC4950 being implemented or not).

As illustrated in Fig. III.1, **Explicit tunnels** constitute the ideal case as all the MPLS information comes natively with `traceroute`. For **Implicit tunnels**, Donnet et al. [DLMP12] have proposed techniques to identify their LSRs based on the way they process ICMP messages and the quotation of the IP-TTL in the `time-exceeded` reply (qTTL and Uturn in Fig. III.1).

Opaque tunnels are only encountered with Cisco LSPs and are due to LSPs ending abruptly, in an improper fashion. In other words, the MPLS packet reaches the exit point of the tunnel without a terminating label (Implicit or Explicit NULL) within its LSE to properly signal the end of the LSP, causing the LSP to *break*.

Thanks to our large scale campaign and experiments with our emulation platform, we conclude that the vast majority of Opaque tunnels are caused by Carrier-of-Carriers VPN [285] or similar technologies. Indeed, such technologies provoke an abrupt tunnel ending as the LSP ends with the LSE containing the label used to identify the VPN instead of a standard terminating label. As we will show later in details, they lead to non-revealable tunnels.

The `traceroute` behavior for Invisible tunnels differs according to the popping scheme (i.e., PHP or UHP) and the OS, as illustrated in Fig. III.1. While **Invisible PHP** tunnels are identified through path length asymmetry [VMPD17], Invisible UHP tunnels provoke a duplicated IP (at least with the IOS 15.2). More precisely, upon the reception of a packet having an IP-TTL of 1, the Egress LER (PE_2 in Fig. III.1) does not decrement this TTL, but rather forwards the packet to the next hop (CE_2 in the example), leading so to the Egress being hidden in the trace. In contrast, the next hop will appear twice: once for the probe that should have expired at the Egress and once at the next probe. This surprising pattern, a

duplicated IP at two successive hops, illustrated as **Invisible UHP** in Fig. III.1 might be misunderstood as a forwarding loop.

Our goal is to reveal such Invisible tunnels and our problem can be technically formulated as follows:

Problem 8. *The problem of Revealing Hidden Tunnels (RHT)*
Given a trace made of subsequent hops h_i, h_{i+1} from s to d, determine efficiently whether there is an invisible tunnel between h_i and h_{i+1}.

III.1.b TNT, a Tool to Reveal all MPLS Tunnels

III.1.b.1 Path Revelation Techniques

Techniques for revealing the content of Invisible PHP and UHP tunnels are similar. Solving the RHT problem can be done relying on an unified tool. In the case of an Invisible PHP tunnel, they can be applied directly as we know both ends of the tunnel (Ingress and Egress LER as in Fig. III.1). However, for Invisible UHP, the Egress LER is missing from the `traceroute` output (look at middle part of Fig. III.1).

It is nevertheless possible with Invisible UHP to infer the outgoing IP interface of the Egress LER (the right interface, in green, on PE_2 in Fig. III.1). Thanks to its retrieval, `TNT` can force replies from the Egress LER incoming interface (the left one, in red, on PE_2 in Fig. III.1). This technique, called *buddy*, assumes a simple point-to-point connection between the Egress LER and its next-hop (this naive assumption comes for the sake of simplicity, but the technique can be extended to deal with point-to-multipoint subnet [MDBP10][148, 147]). The IP addresses belonging to the same /31 or /30 prefix are called **buddies** and `TNT` just needs to infer the correct prefix length to guess the address of CE_2's buddy (i.e., PE_2.right in Fig. III.1).

With a /30, four IP addresses are available: addresses 0 and 3 are the network and broadcast addresses while addresses 1 and 2 are used for numbering interfaces. If CE_2.left corresponds to address 0 (resp. address 3) in a /30, it means that PE_2 and CE_2 share a /31 and PE_2.right is address 1 (resp. address 2) of the /30. However, if CE_2.left corresponds to address 1 (resp. address 2), we launch a `ping` towards address 0 within the /30. If an `echo-reply` is received, both interfaces are on a /31 and PE_2.right corresponds to address 0 (resp. address 3). Otherwise, both interfaces are on a /30 and PE_2.right corresponds to address 2 (resp. address 3 if CE_2.left corresponds to address 2). Note that the buddy identification process can be further improved by considering more advanced techniques [146] whose probing overhead can be mitigated.

As ICMP `time-exceeded` typically contains the IP address of the incoming interface having received the expiring probe, running a `traceroute` towards the inferred address of PE_2.right allows to obtain PE_2.left. Once the potential Ingress and Egress LERs are known, we can launch a hidden tunnel revelation technique, i.e., DPR or BRPR [VMPD17]. The choice of technique depends on the way labels have been bound to destination prefixes. It is worth recalling that one can easily discriminate Cisco and Juniper devices using network fingerprinting [VPMD13].

On the one hand, with ordered LSP control used with Juniper by default on loopback addresses, all the external BGP transit traffic goes through MPLS tunnels while the traffic destined to internal prefixes relies on IP forwarding. Thus, a single `traceroute` targeting the internal Egress LER is enough to reveal all LSRs along the LSP. This technique is called ***Direct Path Revelation*** (DPR). Applying DPR on Fig. III.1, `TNT` simply sends probes targeting PE_2 revealing P_1, P_2, and P_3 in a single shot (without labels, as the probe targeting PE_2 follows the same path as a transiting probe, but without entering the MPLS cloud).

On the other hand, with independent LSP control used by Cisco by default on all IP addresses, LDP is entirely enabled for all the network such that each LSR binds labels for each prefix in its IGP RIB. Thus, as all traffic goes through the MPLS cloud, DPR can not be used.

Our other revelation technique, ***Backward Recursive Path Revelation*** (BRPR) takes benefit from the prefix locality: the targeted incoming interface of the Egress LER is in the same prefix as the outgoing interface of the PH LSR. Thus, since the PH LSR is directly connected to the targeted prefix, it acts as the Egress LER for it and consequently becomes visible to `traceroute`. Applying this method iteratively in a backward fashion up until the Ingress LER, we can reveal each hidden LSR. Applying BRPR on

Fig. III.1, we first send a `traceroute` towards PE_2 and discover P_3. We next send a `traceroute` towards P_3 and discover P_2 and so on until the Ingress LER is met again.

As the targeted IP changes at each iteration of BRPR, its outcome may be affected by load balancing. Revealed links may not belong to the same consistent path. Conversely, DPR works in a single shot and does not suffer from this limit (as `TNT` is built upon Paris Traceroute which relies on constant five tuples in each probe of the same trace).

III.1.b.2 An Efficient Signature (Indicator-Trigger) for each Class

Our tool, `TNT` (**T**race the **N**aughty **T**unnels), is able to efficiently reveal most of MPLS tunnels hidden along a path to solve RHT problem. `TNT` is an open-source scamper [220] plugin extension built upon Paris Traceroute [33], in order to mitigate load balancing issues.

`TNT` consists in collecting, in a hop-limited fashion, intermediate IP addresses between the vantage point and a given target. The tracing phase ends when the target has been reached or a gap has been encountered (e.g., five consecutive non-responding hops). `TNT` uses a moving window of two hops such that, at each iteration, it looks for <Ingress/Egress> pairs of candidates, possibly hiding Invisible tunnels.

For each pair of collected IP addresses, `TNT` checks for the presence of tunnels through so-called *indicators* and *triggers*. The former provides reliable indications about the presence of an MPLS tunnel without requiring additional probing. **Indicators** suggest uniform tunnels, and are basic evidence of visible MPLS presence such as LSEs quoted in the ICMP `time-exceeded` packet. **Triggers**, except DUP_IP, consider unsigned values suggesting the presence of Invisible tunnels through a large shifting in path length. When exceeding a given threshold $\mathcal{T}$, a revelation is attempted. `TNT` is cautious by design: it does not conclude anything from revelations or detections hindered by network anomalies. In addition, while `TNT` is, as other active probing tools, subject to network anomalies, we designed it to be fairly resilient to load balancing and rate limiting thanks to its Paris Traceroute base and inherent lightweight nature respectively.

Fig. III.1 highlights the main patterns `TNT` looks for in a simple scenario where forward and return paths are symmetrical. Indicators and Triggers are the keys we use to unveil the MPLS ecosystem.

Visible Tunnel Indicators They are obvious pieces of evidence of an MPLS tunnel presence: tunnels (or parts of them) can be directly retrieved from the `traceroute` output. Explicit tunnels are indicated through LSEs directly quoted in the ICMP `time-exceeded` message – See `traceroute` output on Fig. III.1.

The indicator for Opaque tunnels consists of a single hop LSP with a quoted LSE-TTL not being equal to an expired value. This abnormal behavior is due to the way labels are handled with Cisco routers, in particular with VPRN tunnel ending. This is illustrated in Fig. III.1 with a value of 252 because the LSP is actually 3 hops long. This surprising quoted LSE-TTL is evidence in itself. A single hop is tagged as Opaque if the quoted LSE-TTL is between a minimum threshold, $\mathcal{T}_{LSE\_TTL}$ and 254 (the LSE-TTL being initialized to 255). This is the only indicator that can fire additional probing in order to reveal the content of the tunnel. However, in practice, it does not perform well as a trigger as we will see later.

Implicit tunnels are detected through qTTL and/or URTURN indicators [DLMP12]. First, if the IP-TTL quoted in an ICMP `time-exceeded` message (qTTL) is greater than one, it likely reveals the `ttl-propagate` option at the Ingress LER of an LSP. As the LSE-TTL was initialized at the IP-TTL value, the packet can expire within the LSP. However, as the IP-TTL is not decremented within the tunnel, the qTTL is greater than one. For each subsequent `traceroute` probe within the LSP, the qTTL will be one greater, resulting in an increasing sequence of qTTL values. Second and by default, the URTURN indicator relies on the fact that LSRs send ICMP `time-exceeded` messages to the Egress LER which, in its turn, forwards the packets to the probing source. However, such LSR reply directly to other kinds of probes (e.g., `echo-request`) using their own IP forwarding table, if available. As a result, return paths are generally shorter considering `echo-reply` messages than regarding `time-exceeded` replies and the URTURN indicator reflects this difference in these lengths. Note that while the URTURN and RTLA computations are identical, Juniper routers do not exhibit, by default, any implicit URTURN pattern and `TNT` does not consider this case.

Triggers for Revealing Invisible Tunnels There exists patterns suggesting their presence (both for Invisible PHP and UHP) and so firing additional probing. `TNT` looks first for potential Invisible UHP tunnels. They occur with Cisco routers using IOS 15.2 and result in a duplicate IP address in the trace output (CE_2 in Fig. III.1).

The two remaining triggers, RTLA (Return Tunnel Length Analysis) and FRPLA (Forward/Return Path Analysis) [VMPD17], rely on path lengths. More precisely, RTLA is the difference between the `time-exceeded` and the `echo-reply` return path lengths, while FRPLA is the difference between the forward and the return path lengths of `traceroute` probes and associated replies. Both triggers are based on the idea that replies sent back to the vantage point are also likely to cross back the MPLS cloud, which will lead to the application of the MIN(IP-TTL, LSE-TTL) operation at the EH of the return tunnel. In the absence of Invisible tunnels, one can expect to find length differences equal or close to 0. Therefore, any significant deviation[6] from this value is interpreted as the potential presence of an Invisible MPLS cloud.

To check for those triggers, we first extract the key distances thanks to the IP-TTLs in replies received by the vantage point. Since RTLA only works with JunOS routers [VMPD17], prior to estimating the triggers, `TNT` uses network fingerprinting [VPMD13] to determine the router brand of the potential Egress LER.

In the presence of a JunOS hardware, `time-exceeded` and `echo-reply` packets have different initial TTL values [VPMD13], and the RTLA trigger can exploit the TTL gap between those two kinds of messages caused by the MIN(IP-TTL, LSE-TTL) behavior at the Egress LER. Indeed, the L_R^{ER} is longer than the LJ_R^{TE} as the MIN operation considers a differentiated pick. This difference represents the number of LSRs in the return LSP, and is compared to a pre-defined threshold $\mathcal{T}_{\text{RTLA}}$. This threshold filters out very short LSPs. Finally, if the signature does not correspond to JunOS, `TNT` falls back to the less reliable URTURN indicator.

FRPLA however applies to any configuration. FRPLA compares the lengths of the forward (i.e., L^T) and return paths (i.e., L_R^{TE}). In the presence of MPLS tunnels, return paths are expected to be seen as longer than forward ones. Indeed, LSRs are not counted in the forward path while they are taken into account in the return paths due to the MIN(IP-TTL, LSE-TTL) behavior at the return Egress LER. Then, we can analyze their length difference and check whether a shift appears. This is illustrated in Fig. III.1 ("Invisible PHP") in which L^T is 3 while L_R^{TE} is equal to 6, leading so to an estimation of the return tunnel length of 3. At the AS granularity, when no IP hop is hidden, we expect that the values associated to FRPLA will look like a normal distribution centered in 0 (i.e., forward and return paths have, on average, a similar length). If we rather observe a significant and generalized shift towards positive values, it means the AS probably makes use of the `no-ttl-propagate` option. FRPLA, on the contrary to RTLA, is sensible to the path asymmetry as it relies on the difference between the forward and return paths. To handle this path asymmetry at the trace granularity and so avoid generating numerous false positives, `TNT` uses a threshold, $\mathcal{T}_{\text{FRPLA}} > 0$.

The main purpose of triggers is to limit the overhead generated by `TNT`. Revelations launched at each hop in a brute force fashion may reveal nearly all MPLS tunnels. However, by first checking for triggers, we limit the amount of unnecessary probes (i.e., leading to no revelation).

`TNT` Limits and Opaque Tunnels By using GNS3, we aimed first at verifying that the inference assumptions considered in the wild are correct and reproducible under a controlled environment, validating so the triggers, indicators, and revelation methods used by `TNT`. Second, some of the phenomena we exploit to reveal tunnels in the wild have been directly discovered in our testbed by reverse-engineering the TTL processing of some common OSes used by many real routers. Indeed, our emulated testbed allowed us to run several OSes and numerous configurations in a controlled environment, similarly to a physical testbed. Thus, we could link each triggers and indicators to specific kinds of tunnels as well as establish the limits of `TNT`.

Table III.1 provides a summary of `TNT` revelation and discrimination capacities considering several MPLS usages in standard configurations. In particular, it shows that `TNT` is able to discriminate between Cisco Invisible UHP and PHP tunnels while it is not the case for Juniper routers. Indeed, for both

[6] In practice, we do not consider negative values. Indeed, they do not suggest the presence of MPLS tunnels but rather path asymmetry evidences (for FRPLA) or load balancing practices on the return path (for RTLA).

Configurations	Pop	Cisco iOS15.2	Juniper VMX
P2P circuits (e.g., LDP or RSVP-TE tunnels)	PHP	FRPLA, BRPR ☑⊠	RTLA, DPR ☑
	UHP	DUP_IP, BRPR++ ☑⊠	RTLA, DPR ☑
P2MP overlays (e.g., VPRN: CsC or VPN BGP-MPLS)	PHP	LSE-TTL, - ⊠	RTLA++, - ⊠
	UHP	LSE-TTL++, - ⊠	N/A

Table III.1: `TNT` revelation (☑) and classification (⊠) capacities according to the OS and the MPLS tunneling technologies (P2P or P2MP). This table also provides the default indicator/trigger and its associated path revelation method.

UHP/PHP Juniper configurations, the trigger and the revelation methods are the same (RTLA and DPR respectively). Moreover, we also show for which cases our basic set of techniques needs to be extended for enabling revelation and distinction among different classes. We use the symbol ++ to highlight these new requirements. For example, revealing UHP Cisco tunnels requires to extend BRPR with the additional buddy functionality and UDP[7] probing in order to extract the incoming Egress IP address that, in turn, allows `TNT` to reveal the tunnel. LSE-TTL++ refers to a way of discriminating UHP VPRN from PHP ones, both resulting in Opaque tunnels (with UHP, the quoted LSE-TTL is equal to 255 instead of reflecting the length of the tunnel). Finally, RTLA++ is a way to distinguish VPRN configuration from basic tunneling on Juniper devices. We discuss this specific situation at the end of the section as it is more complex.

Opaque tunnels may arise for different reasons, such as routing devices heterogeneity, BGP edge configurations, or VPRN. Our GNS3 platform shows that VPRN content cannot be revealed with `TNT`, while other Opaque tunnels can. However, both arise from a non-standard terminating label. Indeed, upon its arrival at the Egress, at least one label is still present in the MPLS header. This surviving inner label is used to identify the VPN and the associated VRF[8]. As the VPN label value is neither Explicit NULL nor Implicit NULL, the Egress behaves as if the tunnel did not end in a controlled fashion.

This absence of content revelation can be explained by the IP address collected by `TNT` from the ICMP reply. Usually, this address is the one of the incoming interface of the Egress PE. In the Cisco VPRN case, the collected IP address is the one assigned to the interface onto which the VRF is attached which usually is the outgoing interface, towards the VPN at the customer's side. Because the incoming address is the only one that enables a successful revelation, this type of Opaque tunnels cannot be revealed. While the outgoing address usually allows `TNT` to get the incoming one, it turns out to be impossible within a VPRN, as all probes are pushed to the VRF of the VPN and its associated interface before the error message is generated.

Juniper VPRNs behave in a slightly different fashion. For such tunnels, no Opaque indicator can be seen. Instead, similarly to Cisco Invisible UHP tunnels, the packets destined to the VPN are IP forwarded directly to the next-hop without manipulating or looking at the IP-TTL whatever its value.

Thus, when performing a direct trace targeting the IP interface of the Egress LER belonging to the VPN, this address and its buddy appear in the wrong order. The two addresses are switched, meaning that the CE IP address appears before the Egress one. Indeed, being forwarded without inspecting the IP-TTL, the probes targeting an IP belonging to the VPN are automatically forwarded to the corresponding CE router where they expire. The next probe, having a greater initial TTL, follows the same path, but can be forwarded back to the Egress, its destination, by the CE router. This about-turn can be inferred as the two IP addresses are switched regarding their actual position, and the TTL of the ICMP `time-exceeded` deviates from its strict monotony (as the first probe went further than the second one). These two artifacts are reflected by the RTLA++ method in Table III.1.

While RTLA++ can theoretically discriminate Juniper VPRN from basic P2P circuits, this extended

[7] With ICMP probes, the target will not answer with its incoming IP address as the probe does not result in a error reply when reaching the target.

[8] In case of VPRN, a router contains a Virtual Routing and Forwarding table (VRF) for each virtual network.

Tunnel Type	Indicator/Trigger	# LSP Revealed per Category				# LSPs	# LSRs	# LSRs per LSP
		DPR	BRPR	1Hop_Lsp	Mix			
Explicit	LSE headers	-	-	-	-	150,036	31,749	2
Implicit	qTTL	-	-	-	-	2,689	1,766	2
	Uturn	-	-	-	-	7,216	7,155	2
Opaque	LSE-TTL	22	17	43	-	3,346	52	2
Invisible UHP	Dup_Ip	1,609	1,531	686	296	4,122	862	2
Invisible PHP	Rtla	11,268	1,191	2,595	279	15,333	3,008	4
	Frpla	5,903	2,555	3,260	1,012	12,730	2,897	3
Total		18,802	5,294	6,584	1,587	195,525	47,489	3

Table III.2: Raw number of tunnels discovered by `TNT` per tunnel category and class. No additional revelation technique is necessary for Explicit and Implicit tunnels.

trigger would be fairly unreliable in practice, as the artifacts it tries to detect are minute compared to the Opaque indicator. However, Rtla being itself a pretty reliable trigger for Juniper devices, it should consequently always result in the revelation of internal LSRs. Thus, following an Rtla trigger, if no new content is revealed while the Ingress was reached, one can conclude at a Juniper VPRN.

III.1.b.3 Results and Analysis: Numerous Hidden Tunnels

We deployed `TNT` on the Archipelago infrastructure [79] on April 23th, 2018 with parameters $\mathcal{T}_{\text{Frpla}}$ fixed to 3 and $\mathcal{T}_{\text{Rtla}}$ to 1.

`TNT` has been deployed over 28 vantage points, scattered all around the world: Europe (9), North America (11), South America (1), Asia (4), and Australia (3). The overall set of destinations, nearly 2,800,000 IP addresses, is inherited from the Archipelago dataset and spread over the 28 vantage points to speed up the probing process.

A total of 522,049 distinct unique IP addresses (excluding `traceroute` targets) have been collected, with 28,350 being non-publicly routable addresses (and thus excluded from our dataset).

Overall Results Table III.2 provides the number of MPLS tunnels discovered by `TNT`, per tunnel class as indicated in the first column. The indicators/triggers are provided, as well as the additional revelation technique used. Explicit tunnels are the most prevalent class (76% of tunnels discovered): most operators do not seem to hide their MPLS infrastructure.

Implicit tunnels represent 5% of the whole dataset, with the Uturn indicator being more present than the qTTL one. Compared to previous works, it is clear that this class is not as prevalent as expected at the time, both because we corrected and improved our methodology by defining Rtla for Juniper routers, and also because the RFC4950 is likely to be more and more deployed.

Opaque tunnels are less prevalent (1.7% of tunnels discovered). Additional revelation techniques (Dpr or Brpr) do not perform well with such tunnels. The content of 98% of Opaque tunnels cannot be revealed, suggesting that the vast majority of Opaque tunnels arise due to Cisco VPRNs. Note that, since Juniper devices do not generate Opaque tunnels, this distribution reflects the way Cisco VPRN affects the trace's output. Indeed, due to those kinds of configurations, we gather the outgoing IP address of the Egress LER, followed by the incoming IP address of the next-IP. It is, thus, likely for these two addresses to share the same /30 or /31 prefix. The fact that the majority of (Egress LER, next-IP) couples share a /30 or /31 prefix is in adequacy with the fact that most Opaque tunnels seem to arise due to VPRN configurations, as can be seen in Table III.2

The proportion of Invisible tunnels is not negligible: 16% of tunnels in our dataset. These measurements clearly contradict our previous work suggesting that Invisible tunnels were probably 40 to 50 times less numerous than Explicit ones [DLMP12]. More precisely, Invisible PHP is the most prominent configuration (87% of Invisible tunnels belongs to the Invisible PHP class), confirming so our last survey [VMPD17]. Rtla appears as being the most efficient trigger. This is partially due to the order[9] of

[9]In case several triggers apply, we prefer to use the most reliable, i.e., the less subject to any interference like BGP

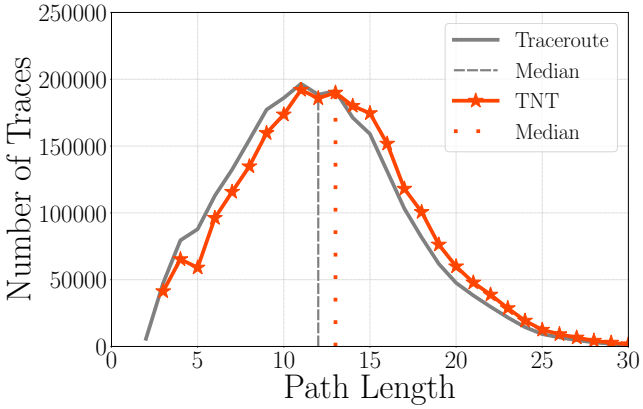


Figure III.2: Path length distribution correction with `TNT`. "Median" corresponds to the median path length for classic `traceroute` exploration (dashed grey) and when additional paths are revealed through `TNT` (dotted red).

triggers in the `TNT` code as it favors a high ranked trigger (RTLA) compared to low ranked one (FRPLA). Moreover, DPR works better than BRPR in practice, showing that both Juniper routers are popular for MPLS configurations and the ordered mode applied only on loopback addresses seems a common practice. It is worth noticing that in 1,784 cases (not shown in the table), RTLA was triggered but no content could be revealed. This number could represent an upper bound of Juniper VPRN that were encountered during the campaign. Those cases are not counted within the 15,333 LSPs shown in Table III.2. In comparison, FRPLA is responsible for 11,590 unsuccessful revelation attempts. For Invisible UHP, less numerous than PHP ones ($\approx$ 2% of all LSPs), it is worth noticing (although not shown in the table) that the buddy extension was required in only 25% of the cases.

The column labeled "mix" corresponds to tunnels partially revealed thanks to BRPR and partially with DPR. Typically, it comes from *heterogeneous MPLS clouds*. For instance, an ISP may deploy both Juniper and Cisco hardware without any homogeneous prefixes distribution. Note that it is also possible that the UHP and PHP label popping techniques co-exist when using BRPR. `TNT` can deal with such complex situations, making the tool robust to pitfalls encountered in the wild (5% of the Invisible tunnels encountered). The column labeled "1HOP_LSP" corresponds to single LSR tunnels where DPR and BRPR cannot be distinguished.

It is also worth noting that some tunnels may belong to multiple classes. We have indeed encountered situations in which an Explicit tunnel contains a few LSRs without RFC4950 enabled (i.e., being so Implicit LSR). Those tunnels and their respective LSRs are not counted in Table III.2 and represent less than 5% of all tunnels founds.

While the column "# LSPs" provides the total amount of MPLS tunnels detected or revealed per tunnel class, the column "# LSRs" gives the contribution of each class in terms of unique IP addresses detected (with indicators) or revealed (with triggers). In both cases, the share of new MPLS data (i.e., non-explicit) that was detected (for Implicit and most Opaques) or revealed (for Invisible and some Opaques) is significant, representing more than 20% of the overall quantity of MPLS information.

What Impact on Topology Discovery? Finally, Fig. III.2 provides the distribution of path length with standard `traceroute` and with `TNT`. As `TNT` reveals intermediary hop within each tunnel we wan extend the initial topology to correct it. We clearly see that `TNT` leads to a shift of the distribution towards the right (longer paths). This shift is lower than the median length of tunnels given in the last column of Table III.2 because all traces are taken into account, even the ones with no tunnels. We have performed more in-depth analysis on other Internet metrics available in [VMPD17] as tunnels modify its structure. Revealing hidden tunnels indeed impacts most standard graph metrics, like its node degree distribution, its diameter or other centrality or transitivity related metrics.

asymmetry.

Title of the publication	Name of the venue	Year	Reference
Revealing MPLS Tunnels Obscured from Traceroute	Computer Communications Review (**CCR**)	2012	[DLMP12]
Network Fingerprinting TTL-based Router Signatures	Internet Measurement Conference (**IMC**)	2013	[VPMD13]
MPLS Under the Microscope Revealing Actual Transit Path Diversity	Internet Measurement Conference (**IMC**)	2015	[VMPD15]
A Brief History of MPLS Usage in IPv6	Passive & Active Measurements Conference (**PAM**)	2016	[VMPD16]
Through the Wormhole Tracking Invisible MPLS Tunnels	Internet Measurement Conference (**IMC**)	2017	[VMPD17]
TNT, Watch me Explode A Light in the Dark for Revealing MPLS Tunnels	Traffic & Measurement Analysis (**TMA**)	2019	[VLM+19]
Let There Be Light Revealing Hidden MPLS Tunnels with TNT	Transactions on Network & Service Management (**TNSM**)	2019	[LVM+19]

Table III.3: Summary of my publications related to MPLS

Before concluding the section, Table III.3 summarizes the list of publications related to my work on this topic.

MPLS: Conclusions and Perspectives

Although MPLS is heavily used on the Internet (being the de facto technology to deploy TE, in particular used as one of the two data-planes of SR – with SRv6), it was not really investigated by the research community prior to the thesis of Yves Vanaubel. From a measurement perspective, our contributions can be summarized in two subsequent objectives:

(i) priori to our analysis, only one work focused on the identification of MPLS tunnels in `traceroute` data [317]. Unfortunately, they lack of precision in some of their models or do not take into account all possible configurations of the protocol. Consequently, the knowledge on its deployment and impact on the Internet was incomplete. In addition, its use in IP networks, as well as the architectural principles and guidelines followed by operators were never evaluated. Our first objective was to provide a general analysis of the use of MPLS on the Internet (e.g. its quantification and prevalence [DLMP12] and its TE usages [VMPD15]);

(ii) since network operators may configure their MPLS tunnels in order to hide their content to `traceroute` exploration, it can lead to incomplete and inaccurate Internet topology data. This incompleteness and link virtualization may cause artifacts when it comes to modeling the graph properties. Indeed, the resulting graphs may contain errors, such as false IP links, or nodes with an artificially high degree. Consequently, the graph property inferred with these biased Internet characteristics, such as path length or node degree distributions, are also biased. Even if hidden routers were already investigated by a few researchers [230, 229, 307, 308], their revelation from `traceroute` measurements was still an open question, in particular regarding MPLS. Our second goal is then about revealing missing hops between tunnels endpoints.

In our MPLS work, we tackled these two objectives with efficient probing tools and measurement campaigns designed according to our findings validated and some obtained with the emulation of several router OSes. We improve both the understanding and knowledge of the deployment and use of MPLS, and the ability to reveal most routers hidden by MPLS. In particular with `TNT`, we have presented new techniques and filters (implemented in an integrated tool deployed in a large scale platform) for classifying and analyzing tunnels. `TNT` relies on triggers to launch additional measurements (almost only) when necessary and we have calibrated it for achieving a performant tradeoff with false negatives. It is now part of a CAIDA project[10] and we continue to rely on its data for other proposals [MMD22].

I think our methods, their implementations and the data they result in are valuable for many kinds of analysis. Not only for topology discovery and analysis but for troubleshooting networks and their

[10] https://www.caida.org/catalog/datasets/ipv4_tnt_dataset/

anomalies. The features we have developed may be integrated in an unified probing tool annotating and augmenting `traceroute` data along with load balancing discovery and fingerprinting. `TNT` can be improved to better support the VPRN identification and to provide more inputs for alias resolution techniques.

Moreover, we now aim to better understand the part of the SR ecosystem relying on MPLS as developed in V.4. MPLS and similar or related technologies adds a layer of complexity to the Internet that we will continue to study in the future. Besides, in the perspective of an unified tool inferring the IGP valuation, discriminating TE from LB allows for cleaning and refining the data used in the inference model. MPLS tunnels not always follow the same paths as the best IGP ones and, when observing such tunnels having specific patterns (TE tunnels with specific sequences of labels), one can discard them from the equation system modeling the weights of the links with basic IP constraints to focus on the ones relying on simple LB as with ECMP.

III.2 DCART: Dynamic Change Analysis with Routing Traces

This work has been possible thanks to two distinct but related projects: the first within the context of a national collaboration with the french educational and research network, RENATER, while the second took place within an European Project (GN4-2) with GEANT, the European research and educational network (I was the local responsible of the measurement work-package). The national project led, thanks to a joint effort with current and former colleagues (Pierre David and Jean-Jacques Pansiot at my university and François Clad and Stefano Vissicchio respectively at Cisco Systems and University College London), to a common publication [MDP+18] whose I will detailed most relevant aspects here. The second project also led to an international journal publication [VBD+20] that I will briefly introduce at the end of this section. Not only I was involved in this later project with many European researchers from various countries but also with a local Master student, Thibault Ehlinger, who did his Master Thesis ([Ehlinger, 2017]) under my supervision. We will rely on the following table of content to illustrate my work on network monitoring, in particular on the correlation of routing changes and outages:

In the study provided here, RENATER, the French National Research & Education Network, is the object under scrutiny: it is a transit network made of 79 routers located in all the largest French cities, connected by 234 IP links, including parallel physical links.

In the following we will answer these questions and address their related challenges:

Research Question

How Resilient is an IGP Network During Routing Changes? What are their Impact on the Traffic Performance (Losses and Forwarding Loops in particular)?

Compared to works mentioned in the introduction and existing monitoring platforms in general (e.g., [37]) and more recent propositions of the literature [92, 133], the originality of *DCART* lies in its ability to finely look for multiple correlated events. In particular, we obtain interesting novel results about forwarding loops regarding related works [257, 163, 214, 320].

I thank RENATER for this great opportunity that help us to extract relevant results that I believe to be general enough as more related to network devices and protocols than to the specifics of the NREN under scrutiny. Let us first provide the technical context of this study.

The routing IGP in use is IS-IS [259], all routers are level 1 routers in the same area. Most routers enable the ECMP feature to load balance the traffic across several links [20]. However, some of them only apply the hashing function to the IP header (e.g. Paris and Lyon's Cisco CRS routers deploying IOS XR). In addition to the IP routing protocol, RENATER uses MPLS switching along tunnels built with LDP[11].

The control traffic of *DCART* is tunneled through LDP tunnels following standard IGP paths. Among other information useful to attach loops to loss sequences, we capture the MPLS header quoted in "TTL Exceeded" error messages. Indeed, error messages resulting from forwarding loops contain a quotation of the original faulty message in transit. We verify that our probes are tagged with MPLS labels. Most

[11] Label Distribution Protocol: the protocol used to distribute MPLS labels.

RENATER routers send many "TTL Exceeded" error replies, thanks to a loose rate limiter, while we did not receive any "Destination Unreachable" reply during our campaign.

To build our platform, we have been granted the possibility to deploy monitoring nodes, our set of probers, in the operational network. We deploy 16 probers directly connected to routers located on several french cities. Notably, Paris sub-network includes about 20 routers and 4 probers.

The location of probers result from two considerations: the operational deployability cost and the link coverage optimization. We use a greedy heuristic where, at each iteration, we look for the sender location able to theoretically cover the most forwarding loops among any possible feasible locations. For each candidate, we compute the total number of not yet covered potential loops from which it can collect evidences: this is achieved by considering all cycles in the graphs resulting from the merging of all outgoing paths before and after the failure of any component.

With the resulting configuration, *DCART* has the ability to discover most loops and covers more than 80 % of the links used in normal operational mode, i.e. when no failures occur. Our probers placement enables us to send probing packets over most active network paths. DCART monitors almost all links and routers except unused backup links that cannot be monitored when there is no failure and leaf parts which do not have an attached *DCART* prober.

III.2.a Correlating Routing Events with their Impacts, i.e. Losses and Loops

DCART is able to collect routing states and forwarding phenomenons related to ECMP load balanced paths. However, for practical reasons, our deployment does not monitor all these paths. The major limitation of the adopted hardware is that it does not support a high probing rate. Namely, we experimentally verified that sending probes to 15 destinations every 20 ms saturates a R-PI 2 CPU[12]. Indeed, in such a case, our probers only send (and thus receive) approximatively 50 KB/s each (probe packets are the smallest possible, i.e., 64 bytes), while the capacity of the Ethernet interface of a Raspberry Pi should theoretically support 100 Mb/s and the typical link capacity of RENATER is greater than 10 Gb/s. If the traffic load is controlled thanks to small packets, their processing is very time consuming for the CPU of the Raspberry Pi that clearly exhibits an overload in such a situation (i.e. it is totally saturated in practice). One can easily overcome this limitation by carefully deciding which prober monitors which paths at which frequency. Exploiting the high coverage of our probers enables us to continuously monitor most transit links at a high probing rate while using a low rate at leaf links. Precisely, active measurements are performed by default every 40 ms between every pair of probers. As a result, each monitored pair from source a to destination b is probed every $40 \times k$ ms, where k is the number of flows used between a and b. A flow consists in a single forwarding path from a to b. In practice, we use $k = 5$ such that if several ECMP paths exist between a given pair of probers, we have almost 94 % of chances to explore distinct physical paths with our 5 flows: $(1 - \frac{1}{2^4}) \times 100 = 93.75$. On RENATER, our listener ECMP feature shows that some specific degraded topological situations may lead up to $18,000$ ECMP distinct paths for a single pair of probers. Our 5 flows are only a way to actively confirm the presence of some path diversity and so, in some favorable cases, measure new indicators (e.g. a loss sequence that occurs on another path) that we may miss otherwise with $k = 1$. On the whole listener collecting period, *DCART* records more than $4,000$ distinct global network states, to compare with almost $70,000$ global state transitions. Most states are degraded states resulting from - sometimes multiple - failures occurring on several long term standard topologies.

Logs are temporarily stored on probers[13] and periodically sent to the database via the RENATER network itself. Thanks to the high redundancy of RENATER, data collection was not impacted during transit link failures. We also engineer logs to improve scalability. To reduce measurement overhead and limit the load on probers, the monitoring controller fine-tunes the events that are logged by each prober. Such fine-tuning globally reduces the size of a 1 hour active measurement logs from 1 GB using raw reporting to 72 MB. A heartbeat mechanism at a rate of 2 packets per second is used to track probers status and accounts for a large part of this data.

[12] Central Processing Unit of Raspberry Pi 2. New generations of Raspberry Pi hardware are more powerful and are likely to support higher probing rates.

[13] The main weakness of the Raspberry hardware seems to be the SD card that fails when too much logs are written in a row.

Finally, note that timestamps for both active and listener events are expressed as microsecond numbers. Although their accuracy is certainly closer to the millisecond scale for multiple reasons – and synchronization limitations in particular, DCART does not require such an accuracy. It can operate at the order of a tenth of a second.

III.2.a.1 Main Objectives

DCART collects informations from multiple data sources in order to analyze the causes and the effects of the dynamicity of a real ISP network infrastructure. The main feature of *DCART* is its ability to perform fine grain temporal correlations between these distinct sources of informations. More specifically, our objectives in designing the architecture of *DCART* are:

- Monitoring simultaneously the data, control and management planes. Our aim is to provide as much informations as possible while keeping a limited impact on the network under scrutiny;
- Allowing and performing correlations between distinct measurement sources to provide many insights into network dynamics;
- Designing specific one way active probes: we aim to uncover generally neglected details about the network data plane, such as forwarding loop occurrences and their location at the data flow level;
- Gather data for later analysis: since this is an exploratory work, data must be kept in a database with a powerful querying facility. We choose a relational database with the SQL language;
- Be vendor independent, scalable and flexible: *DCART* does not rely on specific hardware or software characteristics, whether for the ISP network devices or for internal *DCART* components. Probers are low cost commodity hardware and new components can be easily added to increase coverage, accuracy or reliability.

III.2.a.2 Our Basic Setup

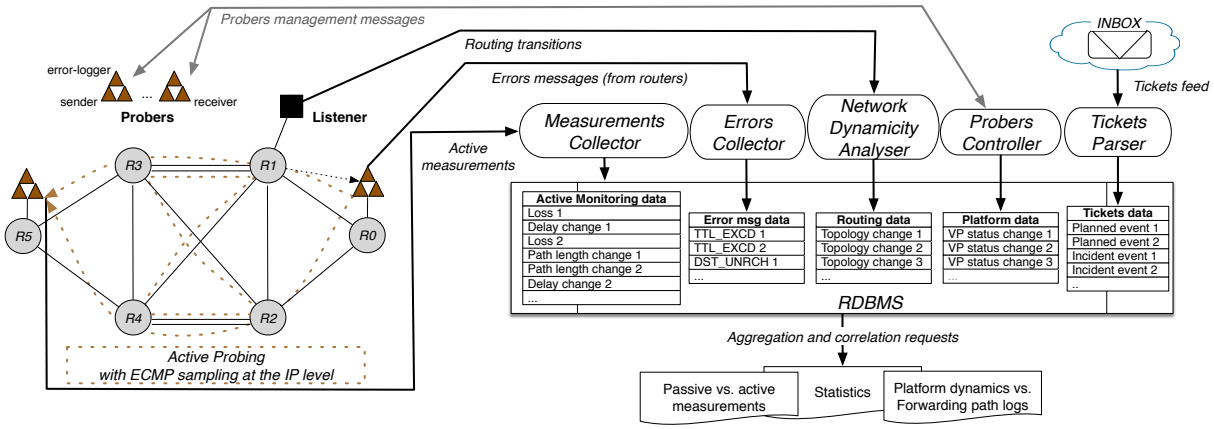


Figure III.3: The architectural overview of *DCART*.

Fig. III.3 describes the main components of the *DCART* architecture. The first high level component consists in a set of active probers. They are distributed on the monitored network and are made of three sub-components: the sender, the receiver and the error-logger. The sender emits ICMP datagrams to all other probers in order to form an active probing full mesh at the global scale. A receiver records ICMP datagrams emitted from a given sender of another prober while the error-logger sub-component records ICMP error messages sent from the intermediate routers. Probers are then able to detect packet losses for multiple data flows independently as well as other informations such as the variations in terms of

delay and number of hops of one way load-balanced paths. In addition error-loggers allow *DCART* to obtain information on forwarding loops.

The second component is the routing listener. It passively collects IGP routing messages. These informations provide a comprehensive knowledge of each logical signalization event as well as the ability to understand the topology, its evolution including details on all load balanced forwarding paths at a given time.

The third component parses and records operational tickets issued by the NOC: this provides valuable information supplied by network engineers to validate some hypothesis or understand the root cause of detected problems.

Collected informations from all these sources are stored in a relational database in order to perform a-posteriori analysis, statistics and correlations.

Active Probers: Monitor the Data Plane Active measurements are performed by probers located at carefully selected points on the network in order to maximize the coverage of monitored links. They are directly connected to the routers or L2 devices within the ISP and synchronized using the Network Time Protocol (NTP).

Each sender periodically emits ICMP Hello Request datagrams to its associated receivers: this <source, destination> couple forms a one way flow. We use ICMP rather than UDP or TCP since we observed that, at least on RENATER, some routers do not send back error messages when receiving UDP or TCP packets while they always return such messages when receiving ICMP probes. The sender inserts a timestamp and a sequence number in each probe packet sent. This way, the receiver is able to handle the probe packet on its own: nothing is recorded at the sender side and no other information exchange is required between the two endpoints. A receiver only logs what it interprets as an anomaly at the flow granularity, e.g. out of order packets. To minimize the churn of our measurements on the user traffic, replies are never sent to the sender.

To provide additional information about the occurrence of a forwarding loop, the error-logger records ICMP "TTL Exceeded" messages which are sent back by intermediate routers when a probing datagram is in fault, i.e. when its TTL is strictly lower than 1. To force the detection of such network anomalies, we configure senders to use a value just slightly higher than the network diameter as the initial TTL of probes. This way, we both ensure that collected ICMP "TTL Exceeded" messages are actually loop indications and, mostly, minimize the likelihood to miss short-lived loops. More precisely, for subsequent messages, we alternate the initial TTL to two consecutive values. This guarantees that we are also able to unambiguously infer the actual location of any transient loop. Indeed, only two-node loops can occur if all link weights are symmetric [121] (I provide this proof in the last Chapter) as it is the case in RENATER so that it is easy to show that two consecutive TTL initial values will trigger error messages from the two routers involved in the loop.

Probers are centrally monitored by a controller which periodically checks their states in order to automatically refine the probing at the flow granularity. That is, when the controller detects the failure of a prober p, it asks other probers to turn off their useless active sub-components related to p. When the controller observes that p comes up it notifies other probers to relaunch their related sub-components. In case of software crash, note that probers also periodically try to restart their own crashed subcomponents as long as the controller does not explicitly notify probers to turn them off.

IS-IS Listener: Monitor the Control Plane Active measurements are complemented with the passive listening of messages carried by the link-state IGP in use, IS-IS in our implementation. Upon each local change on its state, a router floods a LSP (Link State PDU, the elementary topological message of IS-IS routers) to all other participating routers. We built a program based upon an open-source routing daemon [245] which listens to IS-IS messages, parses and dumps them in the standard MRT format [55]. This program runs on a Linux server directly connected to an ISP router. Furthermore, several extensions have been designed to build and manipulate all distinct transient topologies resulting from routing changes. Those network states are associated to the set of time intervals during which the network has undergone the corresponding routing modification. Such time-based topologies enables spatiotemporal correlations that require to check the forwarding path in use at a given moment.

ISP Tickets: Monitor the Management Plane For most ISPs, a NOC sends informational tickets to its clients or partners to report maintenance periods or network outages. Their content is supplied by network engineers and provide human-readable informations on the reported event. We developed a ticket parser specific to our deployment network. It both classifies reported events into a type (incident, maintenance) and a sub-type (e.g. client, router, link) and extracts spatial and temporal informations available from the original tickets.

Database: Perform Statistical Analysis Raw data resulting from the previous sources are a-posteriori processed and stored in a relational databas. Using such a relational database provides a structured view with integrity constraints which helps to strengthen data and their relationships. The SQL language allows both easy querying and good request performances.

All analysis described in this section has been performed on refined data stored in the relational database after the measurement campaign has been completed. A future work would be to incrementally process data and perform analysis in real-time in order to detect network glitches long before they become problems.

III.2.a.3 How to Measure Relations between Sources with an Adaptive Design?

To understand the origins of long outages and losses in general, looking at each individual data source is not enough. Correlating them becomes a necessity. Indeed, even if individual data sources already provide valuable information on their own, we expect to improve our understanding of dependencies among them from an analysis combining information at multiple planes. In this section, we correlate three sources of data provided by *DCART*. Our aim is to explain the root causes of losses and loops, in particular the share of them related to routing transitions. For this we design an efficient method to solve the following problem:

Problem 9. *The problem of Correlating Network Traces (CNT)*
Given a set of heterogeneous sources of data (e.g., active, passive, manual or protocol-based), how to cross them to analyze and understand their correlations?

To solve CNT, we perform such correlations on a time basis possibly looking at spatial crosscheck and other ground evidences (e.g. error messages). In practice, we simply start by considering a maximal time distance. Then we attach each loss sequence to its closest routing event in the defined time range. In the following, we will explain how we calibrate such a time distance to mitigate false positives and negatives. Convergence related losses are generally very close in time to the routing event and their durations are longer than the ones of other (random) losses. While routing transitions may cause packet loss, it is also possible that a high loss rate on a link may induce a routing transition: the link goes down. In this case, the high loss rate will disappear shortly after the link goes down, and appear back when the link goes up if the problem has not been solved: the link may flap. Finally a high loss rate due to congestion (e.g. DDoS) may occur without any routing transition. In the following we will try to distinguish these situations.

In theory, the temporal correlation could be reinforced by additional spatial information provided by the listener. Indeed, *DCART* has the ability to check whether it is likely that a given loss can be attached to a given link state change considering the possible paths for a given prober pair (this set of paths forms an acyclic graph). However, exactly assigning a given data flow to a given physical path requires to know the hash function in use in ECMP routers. Besides, performing extra topological tests may also lead to produce false negatives. A change on a given router may impact flows that are not directly concerned by the link where the change occurs — we will provide such an example in the next section. These two limitations can either produce false negatives or turn the test in a shape that is not restrictive enough (if the acyclic graph is a large partial graph of the initial network).

Losses and Routing: Some Background Losses may have many causes. Some may occur independently of any problem on the data plane or control plane. This is the case of losses induced by congestion, and among them losses due to DDoS attacks: packets are dropped when router buffers are full. Some other losses may be due to a failing network component. This is the case when packets are dropped because

of transmission errors, or because a router has no working route to the packet destination. In the latter case, if the network redundancy can provide alternative routes, this should last only until the alternative route is computed and installed. Finally, any transition in the control plane, be it the consequence of an unplanned state change of a given device or of a new configuration operated by the NOC, may also induce forwarding loops or routing blackhole both provoking losses. Our aim is to understand the origin of such data plane interruptions: what is the share of losses that result from routing transitions?

To illustrate this, consider first a link that goes down. The neighboring router must first detect the failure: this can be achieved either by the routing protocol itself (e.g. with Hello messages or the use of an extra protocol such as BFD) or lower networking layers (e.g. physical alarms). The logical detection delay depends on various timers, such as the IS-IS holding timer. Then the routing protocol must inform all routers of this event by generating a LSP, which may be delayed by various timers (e.g. LSP gen interval) in order to limit the number of LSP generated. These LSP are then flooded to all routers. This is the information we collect with our listener. Then, each router schedules the computation of a new routing table using a SPC algorithm. This computation may again be delayed by the SPC interval timer. Finally, the newly computed routes must be inserted into the FIB[14] and LFIB[15]. A blackhole period starts when the link fails and ends when the FIB is updated, while the associated listener event we collect is somewhere in between. A forwarding loop may occur while FIBs of routers along the path are not consistent. Since the whole updating process is time consuming, it is expected that loops will start a while after the LSP flooding, hence the listener event. So, at the link down, we expect that a loss sequence will start before the timestamp of the listener event and finish after this event, possibly ending with a loop.

By contrast, when a link goes up, alternative paths are used until the update of the FIB, so there is no blackhole. However, losses can be induced by forwarding loops while FIBs are being updated. Therefore, we expect loss sequences to appear after the corresponding listener event, and mostly in the case of loops. In order to associate these loss sequences to the corresponding listener event, we introduce a *loop indicator* condition: a loss sequence associated to a loop occurring after a listener event is still associated to this listener event even if our time distance threshold is not verified (in practice, *DCART* did not collect any loop indicator after a delay of 2 seconds). Finally, note that the blackhole occurring at a link down will create losses on all flows going through it, while a loop generally only impacts a subset of these flows (i.e. they appear only for a subset of destinations that are not updated in a consistent order among routers).

Which Time Range for the Correlation? In order to determine the time distance we should use to correlate routing events and losses, we analyzed the cumulative density function of the delay between all listener events and the closest (before or after) losses using a relatively large time window of $[-1$ min, $+1$ min$]$. A loss is actually a sequence, i.e. we define it as a time interval counting the number of loss packets in a row, while a listener event is just a single date. By closest distance, we mean the one minimizing the time between the listener event and the beginning of the loss interval. It is important to understand that while we are particularly interested in losses that follow the actual link state changes, we may face many time shifting between the actual physical change, its recording by the listener and the data plane impact trace (typically a packet loss) it may result in. Several limitations make the challenge more complex than expected: NTP synchronization accuracy, the delay between the real physical change and the signal it triggers towards the control plane, LSP processing durations, transmitting timers and, finally, the sampling frequency tradeoff of our probers. Therefore, and in practice, ***DCART* may record a loss sequence before its associated listener event even if it is its consequence**.

We notice that an impressive quantity of losses are relatively close in time to routing events but no clear threshold arises. It means that many losses do not really result from the convergence of the routing control planes, but are due to lossy links because of either bad transmission quality that may trigger link flapping, high congestion rates as DDoS or random traffic congestion. The symmetry between up and down events we have observed also indicate that those losses occur at the same level of frequency after up events and before down events. This is to be expected: on a lossy link, losses appear when the link goes up, and disappear when the link goes down again.

Even on a flapping link, the delay between an up and the next down (inter-outage) is greater than

[14]Forwarding Information Base
[15]Label Forwarding Information Base, used in MPLS networks

5 s in more than 90 % of the cases. It is thus unlikely (although always possible) that a loss caused by an up event be closer to the next down event on the same component. Nevertheless, we observe a very tight transition arises around 0 second: we see that a vast majority of loss sequences start in the range [−0.3 s, +0.3 s]. Experimental bounds to use clearly appear as no flapping noise disturbs data. We use them to extract losses that we consider following the routing change even if the routing event is included in a flap. This tight range helps us to remove most bursty losses around flapping periods from the one following routing changes and typically lasting longer. Most of other packet losses are short sequences, i.e. a single packet, but result in a significant loss rate on long periods of time. Clearly, down isolated events are correlated with more losses than up events.

The losses at down events increase between −0.3 s and +0.3 s, while the losses at up events starts to grow a little later, and continues to grow slowly after the +0.3 s point as expected. A closer look into the *DCART* database shows that a significant share of those late losses sequences are actually associated with "TTL Exceeded" error messages. This indicates that these losses are associated with a forwarding loop associated in its turn with the last up event. This is why, in the following, we also consider the *loop indicator* condition to correlate listener events and losses, even if they occur one second after the +0.3 s threshold.

III.2.b Results and Analysis: Flapping is the Rule and Forwarding Loops a Reality

III.2.b.1 A First Analysis for Each Source

This section describes the main results obtained from each data source separately. We start with the listener that reveals numerous cases of more or less severe link flapping.

Events reported by each source frequently occur in bursts resulting from a recurring cause such as an unstable link. In order to aggregate related raw events, we define temporal and spatial thresholds under which raw events are grouped into higher level events.

Listener: Link Flapping Is The Rule The listener is certainly the most efficient component of our infrastructure as its deployment and maintenance costs are almost negligible regarding the benefits it offers. We are able to follow the entire knowledge of the network routing dynamics, i.e. the control plane information. The listener captures and filters all link state changes in order to classify them according to their high level nature. Since the RENATER network has only one IS-IS area, we do not need to merge information from several routers. Moreover, lost LSPs can be easily detected because they are numbered. Fortunately, both the listener and IS-IS in general were reliable enough to provide consistent informations (i.e., very rare loss).

We dig into more than one year of topological changes (453 days), i.e. looking at the transitions that imply modifications in the network topology graph. We group them according to their nature and impact (link up or down, or weight change) and their scale (link or router). We first observed that routing changes are frequent in general: several dozens on average per day, and more than 2,000 in the worst day. Weight and router wide changes are very rare events while link up/down events are, by far, the most frequent. We also observe a significant gap between the median and the average of link events. It is due to severe periods of link flapping: in practice, we observe that a few periods, that last several days each, account for more than half of routing changes. In the remainder of the section, we will generally consider both raw events and several forms of temporal aggregations but only for link events (no spatial aggregation).

Most of the collected raw events result from flapping issues: a temporal aggregation is unavoidable to understand and correctly handle the actual nature of underlying events. Formally, we define a flapping period (*flap*) as a time period where each 2 hours time window or less contains at least 3 events for the same directed IP link. We compute all distinct longest periods for each link. Note that a simple outage (a "down" then "up" pair) is not classified as a flap because the four changes occurring on the two directions of the same link account for only two raw events for each direction.

Using this aggregation, we are able to distinguish isolated outages from flaps. An *isolated event* is de facto an event not included in a flap, and we also consider it as a high level event. **The result is impressive: almost 95 % of routing transitions fall in the flap category**, meaning that the frequency of routing events (isolated or flap period) drops to less than 13 events per day instead of ≈ 153 raw events

without any aggregation. We also observe that isolated events occur during very short down-up periods. Note that most of our statistical indicators are provided at the day scale in order to manipulate tangible data. The notion of *busy day* then follows naturally as a day with at least one event. For example, we observed 74 % of busy days for isolated events, it means that, on average, less than 2 days per week are free of any isolated changes. It is worth noticing that this value of 74 % remains the same considering globally all of routing events.

While isolated events produce 8.47 state changes in average (and so ≈ 11.4 per busy day), flap periods lead to about 200 state changes per busy day. We notably observe that only ≈ 6 % of raw events fall in the category of isolated events. As the worst example collected a given IP link, we have observed that during three subsequent days, at least $4,000$ events have been reported per day to the listener with a maximum peak of $8,070$ in a single day.

With this first analysis, we can conclude that ***link flapping is the rule***. Put in more networking words, the failure of a link does not look to predominantly result from a physical fail-stop model but rather from intermittent failures due, for example to software problems or degradation of hardware electronics or link quality. Despite the prevalence of flapping periods that sometimes last so long that they cumulate, and so interfere, we did not observe many raw event interferences among independent network locations. More precisely, it is very rare that down/up state transitions of a given device is interlaced with transitions of another device.

Considering per outage counters, the median of the number of isolated network outages occurring per day is only slightly larger than one. Eventually, we observe that the intensity of flaps is quite impressive even when considering the median, i.e. they account for 9 network outages per busy day.

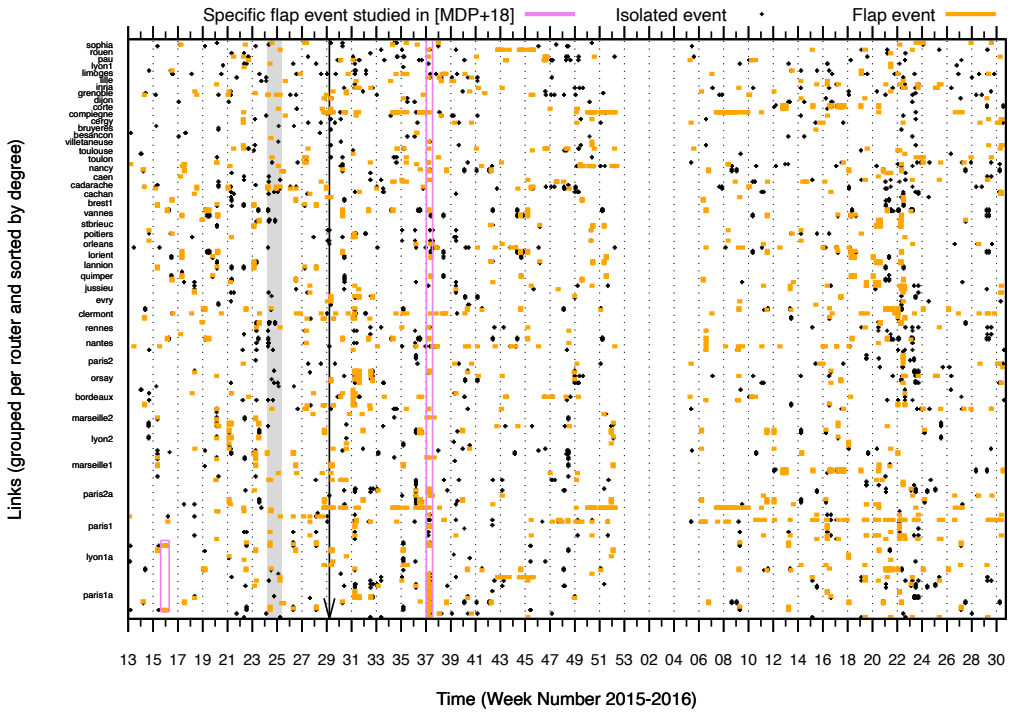


Figure III.4: Spatiotemporal repartition of listener events sorted per router degree on the y-axis. Isolated events are given as black points (whatever they are down or up events) while flap events are orange intervals whose length depends on their durations. The light gray rectangle on weeks 24-25 (2015) means that DCART was down during that period except the listener while the black arrow (on week 29) means that all probers were definitively down from this point. Note that the listener was also down during almost one month between weeks 52 (2015) and 4 (2016): the figure is empty during this period.

Fig. III.4 shows a different perspective that also highlights these first surprising results. This figure aims to first show the coarse grain big picture that introduces the general analysis before digging statistically into it. In particular, this first spatiotemporal illustration comes with several advantages: it offers a

first look at our different collecting periods showing that there are many control events, mostly included in flaps.

For consistency and readability reasons, we distinguish several time periods on the x-axis: the first one, on the left of the black vertical arrow is the one that we will latter cross with active probing (except the light gray period in background where *DCART* probers were down), while the second, on the right, after week 29, only offers passive monitoring informations with the listener and tickets. Statistics about passive measurements are computed on the whole period including the light gray period but not the empty one (from the end of 2015 until end of January 2016) where the listener was down during a bit more than a month. On the y-axis, we sort routers according to their degree and plot all events (both isolated and flapping ones) related to each of their outgoing links. Note that we plot each topological change at the IP link scale.

We can observe several remarkable patterns revealing distinct network outages. An orange horizontal line implies link flapping while a black column reveals a dependent group of links that can range from a line-card up to the more general case of SRLG including router-wide events such as the Paris2a router fail-stop at week 36. An orange column then depicts a flapping at the scale of a group of links, typically a router flapping (as highlighted with the second violet frame at week 37 – a more detailed analysis of such frames is given in [MDP+18]).

Generally speaking, the most visible patterns are long orange twin-lines, such as the ones that range from week 7 to 10 of 2016 between Paris2a and Compiegne or the ones between marseille-1 and marseille-2 at week 37 that consist in the most intense flapping of our passive dataset (more than 15,000 events). These patterns illustrate very long periods of link flapping in both directions of a given physical link. In particular, one remarkable event that we are able to correlate with active measurements is recognizable through the two small lines in the left down corner. This aggregated event results from a 3 days period of severe link flapping between Paris1 and Lyon1 during the second week of April.

Instead of just aggregating the listener raw data into orange flaps, we also try to get insights about the internal patterns inherent to such an aggregation. First, we observe that, most of the time, up and down events occur within the same minute or even less and that the flap intensity is high, i.e. dozens of state changes can occur in less than one hour. Moreover, it seems that as soon as the link or the router goes down, it comes back just after and then goes down again but generally a while after.

We then also study the link outage durations, i.e. delays between down and up state changes, and their inter-arrival periods, i.e. delays between up and down state changes. Analyzing the cumulative distribution function for outage durations considering only events belonging to flaps, we discover that 90 % of them last less than 2 minutes. **Most outages are not really long, i.e. 99 % of them are shorter than 10 minutes.** Note that on RENATER, there generally exists enough redundancy for most <source, destination> pairs, so link outages are much longer than data outages. Looking at the inter-event time distances on a per component basis, we observe that in about 75 % of the cases, the delay between an up event and the next down event in the same flap period is less than 2 minutes. However, we note that **inter-outage duration remains significantly longer than (intra) outage duration despite of the prevalence of flaps**.

Since isolated events are far less numerous than flap ones, we observed that events are much more spaced out in average than events belonging to flaps. Regular outages last longer, the median is almost 500 seconds and their inter-occurrence last several days, the median is more than 10 days. We observe some sudden increases in this distribution, they probably correspond to various protocol timers (lower level link-state triggers, various IS-IS timers such as Hello hold time, LSP generation interval, LSP transmission interval, etc) that are stressed by flaps.

ISP Tickets: Manual Reporting is not Enough Collecting this second passive data source is more lightweight than the former but the information we get is less accurate. Tickets are filled in manually, and time unit is often the minute. There exists a risk of errors, approximations and omissions. However, tickets have the strong advantage to provide information on the management plane, notably about planned operations such as maintenance. Ticket parsing is performed by a carefully crafted program, coping with irregularly shaped messages. For a given ticket, only the closing message is taken into account: it provides the temporal period of the operation.

We observe that incident and maintenance tickets occur roughly at the same frequency, but mainte-

Loss Counter \ Loss Seq.	≤ 1 sec	> 1 sec
Sequences (Seq.)	99.4 %	0.6 %
Packets in Seq.	84 %	16 %

Table III.4: Loss Sequence Duration and Packet Loss

nance events have a longer duration, possibly because maintenance windows are overestimated and their bounds are often rounded to the nearest hour. Tickets cannot be directly compared to listener events. First, for isolated events, a single outage such as a link down then up generates 4 raw listener events for a single corresponding ticket. Second, a long flapping sequence may generate hundreds of listener events which can be summarized by the NOC in only one ticket. That said, it seems that many high level routing events are not reported into any ticket. For example regarding only isolated events: even the average of ≈ 2.1 isolated outages per day is by itself greater than the average of ≈ 1.8 tickets we collect per day. It is approximatively the same for flap events considering link flapping as bidirectional events: ≈ 2.2.

Many routing events (and their corresponding downtimes) are simply unreported. Most of them do not imply a sufficiently long connectivity loss, thus are unnoticeable with basic tools and not subject to a notification to network users. Tickets often report problems which are detected sooner by the *DCART* listener, this suggests that *DCART* could help to diagnose problems before they provoke a network outage resulting in a noticeable downtime. Our preliminary analysis shows that an ISP may benefit from *DCART* with automatically pre-fill incident tickets to improve their accuracy.

Active Probing: About Packet Loss Active probing is precious since it gives information on the data plane, i.e. the quality of service experienced by customers. The lifespan of our probers was limited to 16 weeks of data. Nevertheless, on this short period, we collected really useful knowledge about real downtime periods in term of data plane convergence. Indeed, the listener is not able to show the impact of convergence delay on user traffic as it does not take into account data plane related issues but only control plane informations. We will use here the term of *downtime* instead of *outage* to depict the loss sequence period and duration.

At a global scale, the overall loss rate is very low, i.e. a ratio of 7.6×10^{-5}. We consider here the cumulated loss periods divided by the period where our probers where effectively able to analyze their own 15×5 directed flows at maximum. This ratio includes both downtime periods provoked by forwarding changes and congestions that can be due to DDoS or simply traffic peaks. RENATER is robust enough to support most of the failures that occurred. For instance, we observe that the vast majority of downtime periods consists in a single packet loss while very few events last more than 10 seconds. The median is of 200 ms as probes packets are spaced by this constant time and we observe a downtime longer than 5 hours at worst, i.e. the network was partitioned due to a single point of failure, typically a stub router.

We look at the detail of the duration distribution of loss sequences observed with active probing, i.e. loss of consecutive packets in the same flow. Some of them are quite long but generally the network seems to converge in less than a few seconds. The vast majority consists in just one packet loss and do not necessarily result from a state change but rather from a congestion (e.g. due to a traffic peak, a faulty link or a DDoS attack). The underlying network physical graph looks redundant enough as it is very rarely partitioned: while link outage duration often lasts some minutes, the downtime is almost always less than 10 seconds. While the average duration is about one second, with no surprise the median of the flow downtimes is only 200 ms since it is the period we calibrate for sampling. When removing outliers, i.e. single packet loss or long sequences greater than 15 minutes due to a network partition that affects a given prober, the downtime average is about 0.8 second.

On average and for each prober pair, more than 5 days and half per week are free of any loss: only 18.34 % of days come with at least one loss sequence (busy day). At a global scale, however, **losses occur every day and still more than 4 days per week considering only loss sequences longer than a second**. This ratio is comparable to the number of busy days with routing changes. Durations of collected losses follow a power-law distribution, the vast majority of them are very short, probably due to congestion, only 0.6 % last strictly longer than one second. More surprising, given the robustness of the underlying

network, is the significant impact of longer losses. Table III.4 shows that **while long loss sequences seem to be marginal, they account for 16 % of all lost packets and have a significant impact**.

Routing Error Messages: About Loops Finally, we also collect forwarding error messages sent back by routers that are triggered by our specifically crafted probes. In particular, *DCART* records more than 4,000 ICMP "TTL Exceeded" messages indicating a significant amount of forwarding loops. Some of them last more than several seconds. However, we do not observe any ICMP "Destination Unreachable" messages. Rate limiters are not configured in the same way for these two kinds of error messages. While replies of the former kind are typically filtered in the shape of burst (token bucket) the second kind of filter is more severe, e.g. 1 message per second per prefix for all traffic [281]. During the convergence, if IGP weights are symmetric, forwarding loops may only occur between two neighboring routers at a given time, each one sending the traffic to the other for a given prefix. The TTL included in the IP header of the probe is decreased by 2 at each round in the loop. When the TTL reaches 0, the router sends an "TTL Exceeded" to the sender. If the loop occurs inside an MPLS tunnel (which is an usual case in RENATER), the error message is first sent to the tunnel end, the Egress edge router [VMPD17] before it is returned back to the prober. Therefore, the error message may itself be caught in the forwarding loop or in a black hole and may never reach the error logger of the sender. Due to both this phenomenon and rate limiting, loop detection is only partial, while all packet losses are detected and all control plane events are received. Note that loops of short duration may also remain undetected because either no packet probe enters the loop or the loop ends before the probe TTL expires. Since probe packets are sent with a small initial TTL (e.g. alternatively 20 and 21), only very short loops are missed. In this case, no packet is lost during the loop. In our analysis, we mainly focus on long loops triggering packet loss.

We aggregate "TTL Exceeded" messages to mitigate such limitations and so produce loop sequences. First, we do not take into account the reporter of the error messages (the router where the packet expires) but only the flow identifier, i.e. the couple of probers – sender and receiver – and the flow number. Second, we attach loop indications to the loss sequences they belong to. *DCART* can easily perform this attachment between losses and loops because it parses error messages to extract the quotation of the faulty packet containing the flow identifier. Thus, we do not rely on any time based correlation to perform this deterministic attachment. The duration of a loop is computed as the largest time distance between error messages belonging to the same loss sequence. In practice, it is worth to notice that we never observe long loop sequences where collected error messages are separated by 2 second or more.

The first observation is that loops seem to be rare side effects of routing changes although some of them last longer than expected. The average is about 400 ms and we observe several loops lasting more than one second (with a few lasting more than 10 seconds). Analyzing the data at the day granularity offers another perspective on loop prevalence. For a given probing pair, still less than 4 % of days are subject to loops. However **considering all flows together, almost one day over two is subject to loops, and one day per week is subject to loops lasting more than a second**. Since some loops are not detected, these statistics show that their impact should not be neglected. While all loops are attached to a given loss sequence, the vast majority of loss sequences are not associated to loops except when they last long enough, 25 % of those lasting more than a second are attached to loops.

Although the vast majority of loops last clearly less than 2 seconds, some of them last up to 10 seconds in very rare occasions. As already stated, their number is far lower than loss sequences, in particular for short ones. Moreover, actual black hole periods reporting the absence of any route for a given prefix are not captured (no "Destination Unreachable" message) but may exist when a failure occurs. In the next section, we will study in depth some possible relations between control, data and management planes to understand, for instance, why loops explain all long downtime periods triggered by up events of the control plane.

III.2.b.2 Losses vs. Routing Transitions

We first observe that longer losses are much more directly associated with routing events than short losses. **Long loss pattern is a first hint that a link is faulty: the error rate is under the radar and is not correlated with routing events contrary to higher error rates.** Obviously, short losses may also be due to congestion independently of any routing event. A lossy link seems then to be one of the main root causes

of listener events (and flaps in particular). A low error rate yields a random packet loss pattern with short loss sequences. The control plane reacts above a certain loss threshold, either at the routing layer (Hello or BFD Messages) or at lower layers (e.g. SDH alarms or L2 triggers). These long loss periods do not consist in long loss sequences but rather in a great number of one or two packet loss sequences: the threshold of three packet losses in a row is not reached. We sometimes observe that non correlated long loss sequences occur several times with the same set of destinations. Such losses are severe congestions due to DDoS attacks[16]: in practice, some part of the network has been regularly attacked during the period of monitoring and provokes many rather short losses, from many one packet losses up to quite rare four packet losses.

Using our simple attachment model for correlating routing changes and losses that follow, we can dig into real networking questions. *First, what about the share of listener events correlated with at least one loss sequence?* On the active collection period, we determined that about 64 % of the more than 1900 down events are closely related to losses and, more surprisingly, about 55 % of the same number of up events also result in losses. Without the loop indicator condition, this percentage falls to 40 %. Note that these numbers are lower bounds on the number of listener events correlated with losses. *DCART* records all listener events but does not collect losses on all possible paths: probers cannot cover all existing paths in the network in particular because of the path diversity due to ECMP and parallel links. Moreover, since we do not detect all forwarding loops, the loop indicator also gives a lower bound. So, as a first conclusion, we can say that a vast majority of listener events come before losses, and this both when links are going up or down.

Let us now focus on the severity of losses related to listener events to better understand the kind of root causes that may explain such surprising results. We analyze the longest correlated loss sequence for each down and up listener event (using the loop indicator). Most loss periods are rather short, lower than 2 s in average when removing outliers, especially for up events. However, we observe that down or up events provoking loss periods of several seconds are not negligible and can impact critical traffic. Moreover, we notice that there exists a small amount of up events producing very long loss periods resulting from long forwarding loops.

Routing-caused losses are longer on average. While we compute an average of 14.4 s (to be compared to the mean of only 1 s observed overall) when considering outliers[17], the mean is about 1.5 second without (to be compared to 0.8 s overall). The comparison between the event type (up/down) is rather clear on the figure. Down events generate much more longer losses than up events. They are worst both in terms of average duration and quantity. We compute the following respective means for each kind of events (including average and medians with outliers in parenthesis): 1.2 s (0.7 s, 0.4 s) for up events and 1.8 s (18 s, 0.4 s) for down ones. The existence of a blackhole period at down events explains these not really surprising differences.

After looking at the consequence of routing changes, we will take a look at the opposite question: *what is the share of loss sequences correlated with listener events?* For this, we focus on the complete set of losses shorter than 11 seconds (since longer loss sequences are really rare) and the subset of losses correlated with a listener event. We do not consider here the number of listener events but the number of losses which is two orders of magnitude higher. Thus, the same listener event may be associated to loss sequences on many prober flows. We look at the distribution of the duration of all losses, those correlated with an up or to a down event. We first observe that very short loss sequences, including only one or two packets in a row, usually do not result from routing changes, they are rather due to short congestions or faulty links.

In contrast, we observe that a majority of longer losses from 1 to 3 seconds, seems to be strongly related to routing changes. For even longer losses, correlation is not clear to the same extent because the number of such events is very low anyway. Note that while listener events give a precise fault location, losses are less explicit since they may occur anywhere along one of the paths between a couple of probers. As a direction for further investigations, when losses occur at the same time for distinct prober pairs, but are not correlated to listener events, we aim to improve the localization of the fault by intersecting the paths in use for all concerned flows.

[16] As it has been confirmed by an administrator or RENATER.

[17] That is either single packet losses that are difficult to interpret since it is directly related to our sampling rate or losses longer than 15 min that are due to the disconnection of a given prober.

We compute that only about 28 % of long loss sequences are not (closely) correlated with listener events. A long loss sequence being defined as strictly longer than 1 s in the whole section. A closer look in our database indicates that a majority of them occur on flows going through Paris, Rouen, then Caen. RENATER confirmed us that these links experienced several strong congestions, with throughput above 9,95 Gb/s on 10 Gb/s links, due to recurrent DDoS attacks on this part of the network. We can conclude that in our dataset, only ≈10 % of long losses do not seem to be closely correlated with a routing event or a DDoS. They may result from very rare long congestions, system anomalies of the platform or just being the result of a missed correlation due to a too tight correlation time margin. Indeed, the vast majority of remaining non associated long losses occur in flappy periods. As a final observation, note that the localization of the DDoS attack was possible thanks to the use of several flows between probers: between the same pair, some flows were subject to losses while others, using another path, were not. With a single flow, we could not have localized the problem, or possibly it could have been undetected altogether.

III.2.b.3 Losses and Forwarding Loops vs. Routing Transitions

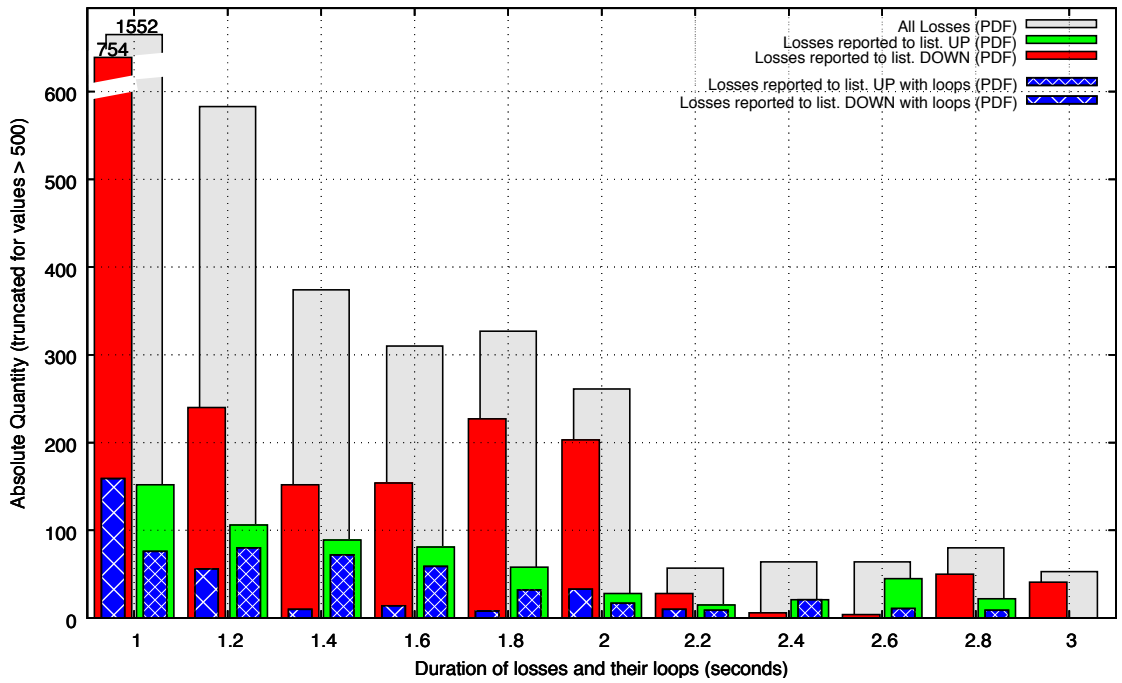


Figure III.5: Distribution of loss durations and share of losses and loops triggered by listener events according to their durations. Loop sequences are attached to their respective loss sequences and this association is correlated with listener events.

Let us now quantify the amount of forwarding loops when routing events and/or long losses occur. We first look at the spatiotemporal distribution of error messages, attached to long loss sequences. While most short loss sequences are not correlated with loops (those losses are not shown here), looking only at longer sequences, i.e. at least 3 packets lost in a row, a significant share of loss sequences seems to result from loops. However, most of them seem to be loop-free. While some routing changes may appear totally loss free because of *DCART* sampling limitations to capture very short convergence periods, it is clear that a large share of routing changes do not trigger forwarding loops. Also, one can see on this spatiotemporal plot, that many flows incur both long losses and loops at the same period, grouped in the same column of the graph.

Fig. III.5 shows the triple correlation between losses, listener events (either up or down) and, among them, those correlated with loops. Short sequences (less or equal to 1 s) are very frequent and are truncated on the figure. **One can see that a large fraction of long losses are correlated with listener events, with more downs than ups. More interestingly, up events are more often correlated with loops**

Cause of Loss	Loss Seq. Dur. (%)	
	≤ 1 s	> 1 s
Flappy links	22.9	11.5
Blackholes: down / up	3.0 / 0.5	48.2 / 7.2
Loops: down / up	0.1 / 0.2	5.2 / 11.9
Other (incl. DDoS)	73.3	16.0

Table III.5: Causes of short and long losses

than down events. There are many possible reasons for loops to occur more frequently at up than at down. Globally, and assuming that loops occur mostly on the last link before the link going down or up, if the router R adjacent to the routing change takes more time to update its FIB than its predecessor P, it turns in two options. Either, at down, there will be no loop but a blackhole on R that lasts a little longer or, when the link goes up, a loop occurs between R and P. There are many reasons for the router R to be slower than P: it has more control plane operations to achieve, such has establishing LDP sessions, and more data plane operations to perform because the closer a router is from the failure point, the more prefixes it has to update. Finally, a link up may be part of a router up that may stress its CPU. We observe that many losses due to an up event appear up to 1 second after the link addition while losses correlated with a down event are much closer in time around the failure and frequently just before rather than long after.

Performing control vs. data plane correlations It provides the ability to classify losses according to their causes: although long loss sequences of more than 1s are quite rare (around 16 %), most of them can be explained by correlated routing transitions. A summary of the origin of losses is given in Table III.5. Loss sequences are classified as *Flappy links* if they occur during a flapping period, and if they are not associated to a link transition. Loss sequences associated to a transition are classified into *loops* if a loop has been detected or *blackholes* otherwise. Note that the tight time margin we use for validating a correlation between a loss and a routing transition may result in an overestimation of losses associated to flappy links in particular for long losses. On the contrary, we observe that many short losses may occur before the actual detection of a flappy link resulting in an underestimation of the share of them actually correlated to flappy links.

Flappy links account for a significant share of losses in general but long losses are mainly the result of routing transitions. Almost half of long losses results from blackholes occurring at link down and a significant share of them (more than 10 %) results from loops occurring at link up. A vast majority of short loss sequences are not correlated with routing events (e.g. random errors, DDoS attacks), whereas most long loss sequences are indeed correlated.

From these results, one can envision several more or less automatic solutions. First, about forwarding loops, there already exists solutions to mitigate or prevent them [Clad, 2014] (and as already detailed in this report). They do not require any manual intervention nor significant protocol change: it is up to the ISP to decide to deploy them with a very limited overhead. Second, about blackholes, it is also possible to automatically limit such periods by speeding up the convergence time using existing mechanisms such as PIC, LFA, incremental SPC algorithms or equivalents [123]. One can also mitigate such periods by configuring several timers that may slow down the convergence or by using more appropriate hardware to quickly detect failures. Eventually, for flap related issues, we do not believe in a fully automatic solution. Indeed, while it is possible to envision some vendor dampening solutions (i.e. shutting down a link that changes its state too frequently), it is a bit risky to automatically remove a link as it can put the network in a degraded state (e.g. partially disconnected in the worst case). Here we prefer to consider the deployment of a support for triggering human actions. That is triggering alarms, that may automatically pre-fill NOC tickets, intended to network administrators. Indeed, before removing – even transiently – a flappy link, it is preferable to let the NOC decide on its own which is the most suitable answer: letting some loss occur or taking the risk to disconnect some part of the network. The decision may vary according to several factors e.g., the loss rate, the flap density and the link position in particular.

Title of the publication	Name of the venue	Year	Reference
A Fine-grained Multi-source Measurement Platform Correlating Routing Transitions with Packet Losses	Computer Communications (**COMCOM**)	2018	[MDP+18]
Localization of Network Service Performance Degradation in Multi-tenant Networks	Computer Networks (**COMNET**)	2020	[VBD+20]

Table III.6: Summary of my publications related to IP network monitoring

Table III.6 summarizes my work in the field of IP network monitoring: I envision to develop my activities in this area considering embedded data-plane measurements.

DCART: Conclusions and Perspectives

In this section we have shown how *DCART* efficiently collects and combines multiple sources of information about the network state. Strong temporal correlation can be observed between listener events, packet losses and forwarding loops such as the ones seen in the last chapter. As a first result, we have verified that forwarding loops actually occur during routing transitions. In worst cases, such loops last several seconds. Not only at link down but even, and mostly, at link up operations (observation *i*). Another interesting result is that the vast majority of routing events and most loss sequences are related to flappy links (observation *ii*). In a network with a redundant topology, flapping may go unnoticed. With the ECMP fine-grained loss monitoring enabled by *DCART*, link flapping may not only be detected, but possibly be anticipated with the rise of loss frequency. Finally, blackholes, i.e. periods where there is no active FIB entry for a given set of prefixes, also produce a significant share of loss sequences. From its detection to its resolution, the recovery of a FIB entry may last several seconds after a link down operation (observation *iii*).

From this diagnostic, several improvements are possible for both router vendors and ISP. First, there already exists several solutions for dealing with (*i*) at the vendor side but they are rarely implemented or deployed. Dampening flappy interface at the up may be a feasible solution for solving (*ii*) at the manufacturer side, while (*iii*) may be mitigated both by using a better routing architecture enabling pre-provisioned backup paths and by a better calibration from the ISP of configurable updating timers.

ISP tickets occur at another timescale. Their analysis provide more insight in the consequences of maintenance operations, and may help improve maintenance procedures. One could also envision automatic tickets pre-generation triggered by listener events and evolution of loss rates. Human ticket generation, which is often not very precise, could be assisted with data automatically retrieved from our monitoring architecture. In this topic of almost real time actions, the set of monitored flows could be adapted to locate faulty links. While monitoring all routes between all pairs of routers would be quite expensive in a large network, monitoring a set of pairs whose flows cover a maximum of links would be much more lightweight. In this case, adding some additional flows on demand would allow to get more information during a long lasting incident.

We envision that in a large network, probers could be attached to most routers, each prober generating only a small number of flows. When the loss rate on a flow grows above a predefined threshold, without any correlated routing event, additional flows could be launched to locate where losses occur, in the manner of tomography tools [222]. This could also allow to distinguish a DDoS, where we expect losses to occur along a tree towards the target from a single faulty link.

DCART focuses on revealing the intra-domain properties while I also have looked at the performance of VPN BGP-MPLS tunnels in the context of multi-tenant networks [VBD+20]. I was involved in an European project with the engineers of GEANT and RENATER, as a member of the monitoring subtask of the GN4-2 project. We have proposed an hybrid system able to passively capture probes that we actively control within the tunnels to monitor. It allows to finely understand how the inter-domain path performs from AS to AS, in particular between ingress-egress couples of each domain. The main difficulty was to design the correlation centralized functions to re-assemble the paths and define triggers to observe their evolution (and report SLA violations). Building an efficient and reactive model re-constructing global paths and their characteristics looking at each packet of each flow was challenging. Locally, we used probers marking packets and high performance captures disseminated between traversed domains.

Globally, we relied on a temporal grid to record all sequenced packets timestamps within their flow identifier. This grid allows for detecting packet losses and performance degradation on the fly; it moves over time to flush already analyzed information considering multiple slots to discretize the time. The reaction time follows the time granularity.

Monitoring is a key networking function. However it is often dedicated to a posteriori analysis and implemented aside in a management plane. I aim to explore routing solutions embedding measurements at the data-plane to take local or coordinated decisions. Verifying the forwarding performance like TCP connections may be useful to improve the routing adjustments and adapt the load distribution properly. In chapter V, I will discuss the projects I aim to follow in this area.

III.3 From Routing Inconsistencies to Forwarding Detours & BGP Lies

This research topic is related to the Ph. D thesis of Julian del. Fiore [Del Fiore, 2021], under my guidance and the direction of Cristel Pelsser, and with the collaboration of Valerio Persico and Antonio Pescape (University of Napoli). The thesis and our collaborative work led to these two common publications: [DFPM+21] and [DFMP+19]. While the later focuses on the root causes, that is BGP lies and routing inconsistencies (e.g. possibly intentional mismatches between the data and control planes), the former studies the impacts of the consequences, that is detours occurring for the transit traffic flowing in the failed or malicious ISP. In particular, in [DFPM+21] that we will develop in more detail in the following. We assume a forwarding model similar to the one used for OPTIC, or with MPLS as in the previous chapter, that is a hierarchical model whose scalability limits are now pointed here and in the thesis of Julian del. Fiore.

The full Internet feed, reaching more than ∼900K prefixes as in 2022, has been growing at ≈50K prefixes/year over the last 10 years. To counterbalance this sustained increase, Autonomous Systems (ASes) may filter prefixes, perform prefix aggregation and use default routes. Despite being effective, such workarounds may result in *routing inconsistencies*, i.e., in routers along a forwarding route mapping the same IP addresses to different IP prefixes. In turn, the exit AS border routers associated with these distinct prefixes may potentially differ. For some prefixes, *forwarding detours* (FDs) may occur, i.e., traffic may deviate from best IGP paths. In this work we investigate the phenomenon of FDs and derive a methodology to detect them. The questions we aim to address in this section are the following:

Research Question

How and Why Routing Inconsistencies and Lies Occur in the Wild? How to Reveal their Visible Share (The Forwarding Detours)? What are their Visible Prevalence in Real Traces?

The table of content is given here, it resumes our progression to answer the previous questions:

III.3.a Detecting Forwarding Detours

While there exists some workarounds that may look effective to mitigate BGP scalability issues at first glance (e.g. partial FIB and default routes mentioned in the Introduction), ASes relying on them may suffer from *routing inconsistencies*. In such cases, inside those ASes, routers along a route may map the same destination IP address to distinct (most specific) prefixes. Since these prefixes may be associated to discrepant AS border routers (ASBRs), *forwarding detours* (FDs) may occur, i.e., for some prefixes, traffic may not traverse the network through best IGP paths. Due to this, we refer to such prefixes as *prefixes subject to FDs*. In general, the simultaneous existence of prefixes subject and not subject to FDs generates *multi-path routing* patterns. However, contrary to hot-potato routing, FDs increase the IGP distance required to traverse an AS and may generate loops [125], arguably resulting in waste of resource utilization inside the network. Attempting to suppress FDs, network operators may implement tunneling

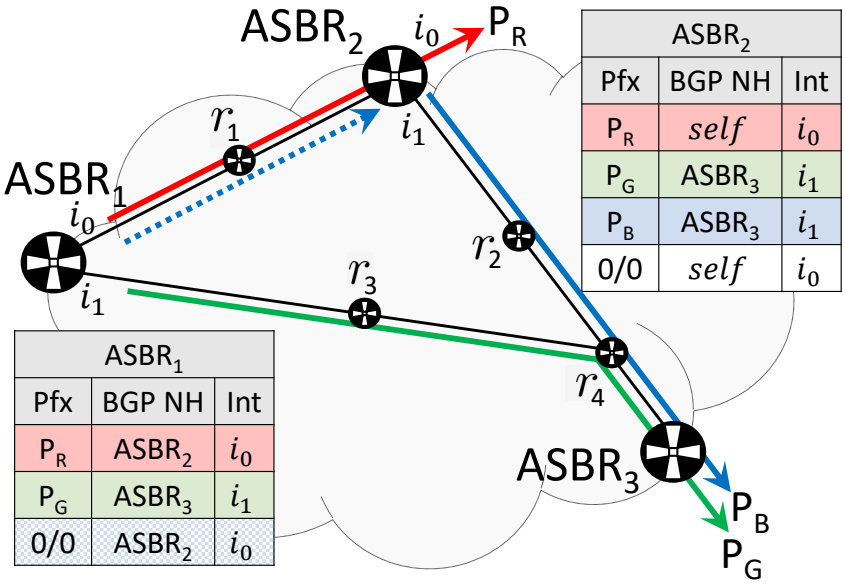


Figure III.6: From routing inconsistencies to FDs. The default route of $ASBR_1$, that has a partial-FIB, leads to a routing inconsistency between this router and $ASBR_2$ for the blue prefix P_B. Since $ASBR_2$ redirects traffic concerning P_B towards $ASBR_3$, the resulting route does not match the best IGP from $ASBR_1$ to $ASBR_3$. Hence, we say that P_B is subject to FDs. Moreover, as P_G is not subject to FDs, a multi-path routing pattern appears between $ASBR_1$ and $ASBR_3$.

techniques [LVM+19], with Label Distribution Protocol (LDP) [29] or Segment Routing (SR) [44]. However, these mechanisms only allow to avoid FDs within each tunnel/segment (for BGP-free core routers in particular) but may fail to do so between endpoints of an AS.

Fig. III.6 illustrates how routing inconsistencies may produce FDs. In this example, $ASBR_1$ has a partial-FIB and, relying on its default route, forwards traffic destined to prefix P_B towards $ASBR_2$ (blue dotted line). There exists a routing inconsistency for P_B since $ASBR_2$ disagrees with $ASBR_1$ regarding the BGP exit point; indeed, rather than itself, $ASBR_2$ considers $ASBR_3$ as the best BGP next-hop for P_B. Hence, $ASBR_2$ redirects traffic targeting P_B towards $ASBR_3$. While the best IGP path from $ASBR_1$ to $ASBR_3$ is $(ASBR_1, r_3, r_4, ASBR_3)$, and is used for P_G, the forwarding route for P_B differs, being $(ASBR_1, r_1, ASBR_2, r_2, r_4, ASBR_3)$. Consequently, P_B is subject to FDs, but P_G is not, thus generating a multi-path routing pattern between $ASBR_1$ and $ASBR_3$. Moreover, even if tunnels mechanisms were used between ASBRs, e.g. $ASBR_1$ and $ASBR_2$, after exiting the tunnel, traffic concerning P_B would still be redirected towards $ASBR_3$.

In this study we take a close look at the phenomenon of FDs. As discussed before, FDs may result as a side effect of scalability workarounds. However, misconfigurations [90] or bugs in router's software such as BGP zombies [118] may also create routing inconsistencies leading to FDs. Consequently, network operators may ignore FDs occur on their AS, and provide degraded performance to customer ASes. Prior work has focused on detecting routers relying on backup default routes [62], or identified them as a possible cause of BGP lies [DMP+19]. However, no study has focused on the impact of such techniques on the forwarding inside ASes. In that sense, to the best of our knowledge, we are the first to tackle the problem of detecting FDs, indistinctly of the underlying causes generating them. Our methodology allows network operators to check the sanity of the routing inside their own network, and customer ASes to check whether their provider ASes suffer from FDs. The detection of FDs is the first step towards the ultimate goal of systematically quantifying the effect of FDs on traffic.

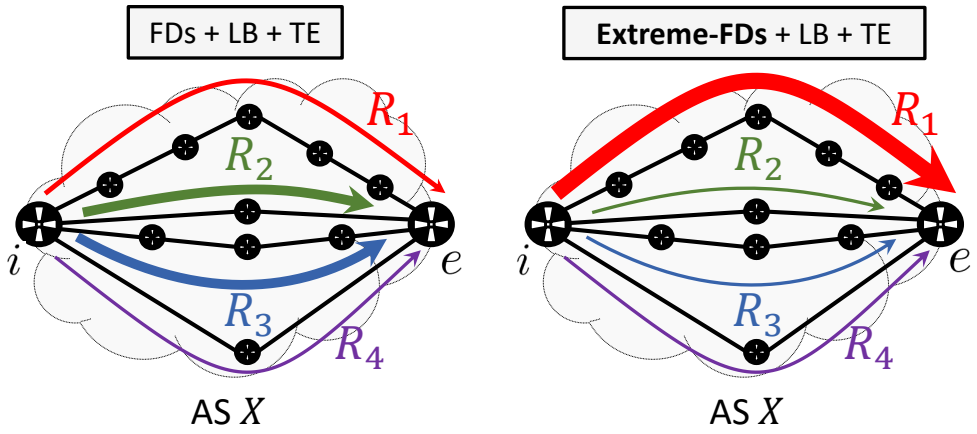


Figure III.7: Forwarding patterns when FDs ($\mathcal{R}_X^{FD}(i,e) = \{R_1\}$), LB ($\mathcal{R}_X^{LB}(i,e) = \{R_2, R_3\}$) and TE ($\mathcal{R}_X^{TE}(i,e) = \{R_4\}$) co-exist. The size of every arrow is proportional to the number of prefixes for which each route is used. While the forwarding pattern inside the AS on the left case undergoes no major change due to FDs, on the right case it is largely modified by the occurence of extreme-FDs, i.e., FDs for most prefixes.

III.3.a.1 Challenges & Problem Statement

Here we explain why detecting FDs, a problem for which there is currently no recipe, is challenging. In particular, this task is not trivial since Load Balancing (LB) and Traffic Engineering (TE) techniques also produce multi-path routing patterns.

In practice, observing a multi-path routing pattern between any two routers i and e of an AS X is not enough to declare the occurrence of FDs: the use of LB and TE can also produce the same effect. With LB methods such as equal-cost multi-path (ECMP), the strict notion of best IGP path is generalized to a set of paths $\mathcal{R}_X^{LB}(i,e)$ sharing the same IGP distance. The purpose of ECMP is to evenly spread the load across such set of best parallel IGP paths. On the other hand, TE allows to create sets of constrained paths $\mathcal{R}_X^{TE}(i,e)$ that are commonly used for specific usages regarding a limited number of external prefixes, but not for best-effort traffic. Let us for example assume a simple scenario with a unique detouring route denoted R_1 (in Fig. III.7), but where however, between the same ingress-egress couple, respectively i and e, multiple routes exist because prefixes are subject to deflections due to different underlying causes (LB and TE). Considering the left side of Fig. III.7, where $\mathcal{R}_X^{FD}(i,e) = \{R_1\}$, $\mathcal{R}_X^{LB}(i,e) = \{R_2, R_3\}$, $\mathcal{R}_X^{TE}(i,e) = \{R_4\}$, the question we aim to address is, by simply collecting routes with `traceroute`, how can we distinguish what we call FDs from other patterns?

In the following we will neither assume a simple metric as hop count nor that transit traffic traverses exactly two ASBRs inside an AS. Such naive assumptions may lead to both false positives and negatives in the detection of FDs as illustrated in our examples. To correctly detect FDs, rather than computing misleading metrics for each route and/or comparing them one at a time, we propose to analyze the *forwarding pattern* for (i,e) in AS X. In other words, we propose to closely study which routes of X, leading from i to e, are used depending on the targeted prefixes. For this, multiple traces traversing i and e need to be collected for as many prefixes and destinations as possible, and the distribution of prefixes per set of routes analyzed. On the left case of Fig III.7, few prefixes are subject to FDs, and thus differentiating them from TE and LB might be too challenging. The main bulk of prefixes evenly spreads over $\mathcal{R}_X^{LB}(i,e)$, and only a reduced number of prefixes are forwarded across $\mathcal{R}_X^{TE}(i,e)$. In contrast, in the cases involving *extreme-FDs*, i.e., scenarios where most prefixes are subject to FDs, we expect to see a remarkably distinct forwarding pattern in which a large fraction of external prefixes is aggregated on $\mathcal{R}_X^{FD}(i,e)$. This is exemplified on the right side of Fig. III.7, where traffic traversing from i to e is aggregated on $\mathcal{R}_X^{FD}(i,e) = \{R_1\}$ for multiple prefixes.

Studying forwarding patterns focusing on the detection of extreme-FDs is not enough to actually identify the existence of $\mathcal{R}_X^{FD}(i,e)$, i.e., concluding that the routes on which most prefixes are aggregated do not represent LB or TE routes requires more analysis. In addition, modeling the effects of different LB flavors is necessary. This is particularly important since there exists a specific LB flavor that defines flows

at the prefix granularity. As such, this can generate, as for FDs and TE, a forwarding pattern in which the route in use may vary depending on the prefix that is considered. Last but not least, LB and FDs can interfere with each other, since ECMP can also apply on detouring routes. Overall, to understand how FDs can be detected, having a clear understanding of the distinct forwarding patterns that LB, TE and FDs produce is imperative. The problem can be stated as follow:

Problem 10. *The problem of Detecting Forwarding Detours (DFD)*
Given an ingress-egress (i, e) couple in a AS X, infer whether there exists prefixes that are defledted accross X via the pair (i, e) due to Routing Inconsistency (and not other reasons).

Before solving such a problem let us first explain the root causes behind such FDs and what share of them one can expect to reveal in practice.

III.3.a.2 Forwarding Model and Notations

Lookup functions: prefixes, gateways and next-hops In the forwarding model considered here, each router r inside any AS X determines next-hops relying on three lookup functions, namely $\mathcal{N}_r(G)$, $\mathcal{G}_r(P)$ and $\mathcal{P}_r(d)$. Each of these functions conveys a different objective:

The function $\mathcal{P}_r()$ receives a destination IP address d, and returns the most specific prefix covering it. The result $\mathcal{P}_r(d)$ can be either an internal or external prefix, i.e.,

$$\mathcal{P}_r(d) = \begin{cases} \text{Internal prefix}, & d \in X \\ \text{External prefix}, & \text{otherwise} \end{cases}$$

where d is advertised in the IGP of X in the first case, or learned via BGP, and thus used for transit traffic, in the latter.

The function $\mathcal{G}_r(P)$ takes a prefix P as argument and outputs the IP address inside X to be reached in order to eventually access prefix P. We refer to $\mathcal{G}_r(P)$ as the *gateway* for P. The resulting gateway $\mathcal{G}_r(P)$ varies depending on whether P is internal or external, i.e.

$$\mathcal{G}_r(P) = \begin{cases} d, & P \text{ is an internal prefix} \\ BGP(P), & \text{otherwise} \end{cases}$$

where $BGP(P)$ returns the iBGP next-hop for P.[18]

Hence, the gateway $\mathcal{G}_r(P)$ should be, theoretically, the egress-ASBR for transit traffic, i.e., the last hop in X that forwards transit traffic to the eBGP next-hop that advertised P.

The function $\mathcal{N}_r(G)$ computes the next-hop towards an IP address G (a priori a gateway) inside X, i.e. the router linked via an outgoing interface of r to which a packet has to be sent in order to ultimately reach G. The value of $\mathcal{N}_r(G)$ depends on that of G, i.e.,

$$\mathcal{N}_r(G) = \begin{cases} \delta(d), & G = r \\ IGP(G), & \text{otherwise} \end{cases}$$

where $IGP(G)$ provides the IGP next-hop towards G and $\delta(d)$ is a function of d that takes different values depending on whether $d = r$ or $d \neq r$. In the first case, the forwarding stops and the packet is handled by the higher layers of the protocol stack, i.e., $\mathcal{N}_r(G) = \delta(r) = \emptyset$. On the other hand, if r is not the destination, then r acts as the egress-ASBR used to reach the external destination d, and thus $\mathcal{N}_r(G) = \delta(d)$ is the eBGP next-hop for P.

To forward packets towards a destination IP address d, routers apply the lookup functions we have defined in sequence. Like this, packets are forwarded to the next-hop $\mathcal{N}_r \circ \mathcal{G}_r \circ \mathcal{P}_r(d)$ leading to the gateway $\mathcal{G}_r \circ \mathcal{P}_r(d)$ defined for the longest matching prefix $\mathcal{P}_r(d)$ of d.

[18] When the BGP next-hop-self option is enabled, this is exactly the case. We design our model relying on this feature for convenience, to simplify the illustrations in this section that would otherwise require showing a neighboring AS, but without loss of generality.

Internal routes with respect to an AS As routers at each hop receive and forward packets based on its own routing knowledge, the chaining of these events results into a hop by hop forwarding route.

Forwarding route: *a forwarding route of an AS X towards a destination d, denoted $R_X(d)$, is a sequence of routers $R_X(d) \triangleq [r_0, r_1, ..., r_j, ..., r_n]$, where, we consider implicit that $r_j \triangleq r_j(d)$ to simplify notation, such that:*

$$\forall j \in \{0, ..., n-1\},\ r_{j+1} = \mathcal{N}_{r_j} \circ \mathcal{G}_{r_j} \circ \mathcal{P}_{r_j}(d)$$

One can extract the internal part of such a route in the AS X.

Internal route: *an internal route of an AS X towards a destination d, is a forwarding route $R_X(d)$ such that:*
i) $\forall j \in \{0, ..., n\},\ r_j \in X$
ii) r_0 *is an ingress-ASBR of* X
iii) $\mathcal{G}_{r_n} \circ \mathcal{P}_{r_n}(d) = r_n$

A priori, the strict notion of what an internal route is only requires condition i, but we narrow their scope additionally considering conditions ii and iii to align our model to our FD-detector. While condition ii is self-explanatory, according to condition iii, router r_n chooses itself as gateway, hence $\mathcal{N}_{r_n} \circ \mathcal{G}_{r_n} \circ \mathcal{P}_{r_n}(d) = \mathcal{N}_{r_n}(r_n)$. Given this holds, r_n is the last hop in X, and there exist two options:

- r_n is directly the destination d and $\mathcal{N}_{r_n}(r_n) = \mathcal{N}_{r_n}(d) = \emptyset$. We refer to these routes as **direct internal routes (DIRs)**.
- $r_n \neq d$ and acts as egress-ASBR of X, consequently $\mathcal{N}_{r_n}(r_n) \notin X$. The packets being forwarded represent transit traffic in these cases, hence we say the routes are **transit internal routes (TIRs)**.

Routing Consistency According to our definition of internal routes, indistinctly of being TIRs or DIRs, the actual gateway and best covering prefix that each router along the route chooses is not defined. The only exception is the gateway of r_n, that selects itself. In particular, when all routers along the route choose the same best covering prefix P and gateway G, we say that the resulting internal route $R_X(d)$ is consistent.

Routing consistency: *an internal route $R_X(d)$ is consistent when all routers along the route choose the same best covering prefix P and gateway G for d, i.e., when*

$$\forall\, r_j \in R_X(d),\ \mathcal{P}_{r_j}(d) = P\ \wedge\ \mathcal{G}_{r_j} \circ \mathcal{P}_{r_j}(d) = \mathcal{G}_{r_j}(P) = G$$

.

In particular, note that r_0, the first router in $R_X(d)$, imposes conditions, i.e., it must hold that $P \triangleq \mathcal{P}_{r_0}(d)$ and $G \triangleq \mathcal{G}_{r_0} \circ \mathcal{P}_{r_0}(d) = r_n = \mathcal{G}_{r_n} \circ \mathcal{P}_{r_n}(d)$.

III.3.a.3 Only Some are FD are Visibles as they imply RI but are not Equivalent to

The Two Causes of Inconsistencies An internal route $R_X(d)$ may be inconsistent at two different levels depending on the best covering prefixes and gateways that routers along the route choose for d.

Routing inconsistency at the prefix level: *there exists a routing inconsistency at the prefix level in an internal route $R_X(d)$ when an upstream router r_k of $R_X(d)$ chooses a different best covering prefix than a downstream router r_j of $R_X(d)$, i.e.,*

$$\exists\, j < k \leq n \mid \mathcal{P}_{r_j}(d) \neq \mathcal{P}_{r_k}(d)$$

Routing inconsistency at the gateway level: *there exists a routing inconsistency at the prefix gateway level in an internal route $R_X(d)$ when an upstream router r_k of $R_X(d)$ chooses a different gateway than a downstream router r_j of $R_X(d)$, i.e.,*

$$\exists\, j < k \leq n \mid \mathcal{G}_{r_j} \circ \mathcal{P}_{r_j}(d) \neq \mathcal{G}_{r_j} \circ \mathcal{P}_{r_k}(d)$$

Note that both types of RIes **are not exclusive**: the one at the prefix level may lead to another at the gateway level. As an example, if r_j and r_k are routers in $R_X(d)$, such that r_j is a partial-FIB router with a default route and r_k is full-FIB router, this will likely generate RIes at both levels. As r_j has a partial-FIB, multiple originally disaggregated prefixes present in r_k are aggregated in the FIB of r_j into a default route, which originates RIes at the prefix level. While r_k may associate different gateways to each of these prefixes, a unique arbitrarily chose, a default one, is used by r_j. Consequently, this may likely potentially leading to additional RIes at the gateway level.

In practice, without privileged knowledge, RIes are detected only at their last level: it is not possible to determine which cause actually originates them. Therefore, to simplify our notation, we will focus on detectable RIes, and **consider implicit that routing inconsistency occur in internal routes and not refer to their exact causes (gateway or prefix level).**

Forwarding Alterations When the resulting internal route $R_X(d)$ is different to the forwarding route that would have been used if all routers had chosen the same gateway, we say that the difference was generated by a FA in $R_X(d)$.

Forwarding Alteration: *there exists a forwarding alteration in an internal route $R_X(d)$ if an upstream router r_k uses different next-hops for the gateway it chooses and the gateway a downstream router r_j selects, i.e.,*

$$\exists\, j < k \leq n \mid \mathcal{N}_{r_k} \circ \mathcal{G}_{r_k} \circ \mathcal{P}_{r_k}(d) \neq \mathcal{N}_{r_k} \circ \mathcal{G}_{r_j} \circ \mathcal{P}_{r_j}(d)$$

Theorem 13. *Forwarding Alterations ⇒ Routing Inconsistencies*
If there exists a forwarding alteration in an internal route $R_X(d)$, then $R_X(d)$ is inconsistent.

FAs imply RIes, but the converse may not hold.Indeed, for $j < k$, even though routers r_j and r_k may choose different gateways, G_j and G_k respectively, when $\mathcal{N}_{r_k}(G_k) = \mathcal{N}_{r_k}(G_j)$, then no FA occurs. In these cases, we say that the existing RIes are **not visible**. This may particularly happen in networks that **lack path diversity**, e.g. if the disagreeing router r_k only has one possible next-hop, then it can never introduce a FA.

Forwarding Detours A forwarding detour occurs when the internal route in use inside an AS does not match the best IGP path between the endpoints of the route, i.e., r_0 and r_n. The illustrations in Fig. III.8 show the difference between best IGP paths and FDs. In the left case, the routing is consistent since all routers choose the same gateway o, and thus the forwarding route coincides with the best IGP path available between l and o. On the contrary, on the right case, router p introduces RIes, and a FA such that packets exit the AS via o. Since l would have used the direct link with m to forward packets towards o, instead of that via n, the FA translates into a FD.

Forwarding Detour: *an internal route $R_X(d)$ detours, i.e., a forwarding detour occurs for destination d, when an upstream router r_j in $R_X(d)$ would not have used r_{j+1} in $R_X(d)$ as next-hop to reach a downstream router r_z of $R_X(d)$, i.e.,*

$$\exists\, j < z \leq n \mid \mathcal{N}_{r_j}(r_z) \neq r_{j+1}$$

Theorem 14. *Forwarding Detours ⇒ Forwarding Alterations*
If an internal route $R_X(d)$ detours, then $R_X(d)$ is subject to forwarding alterations.

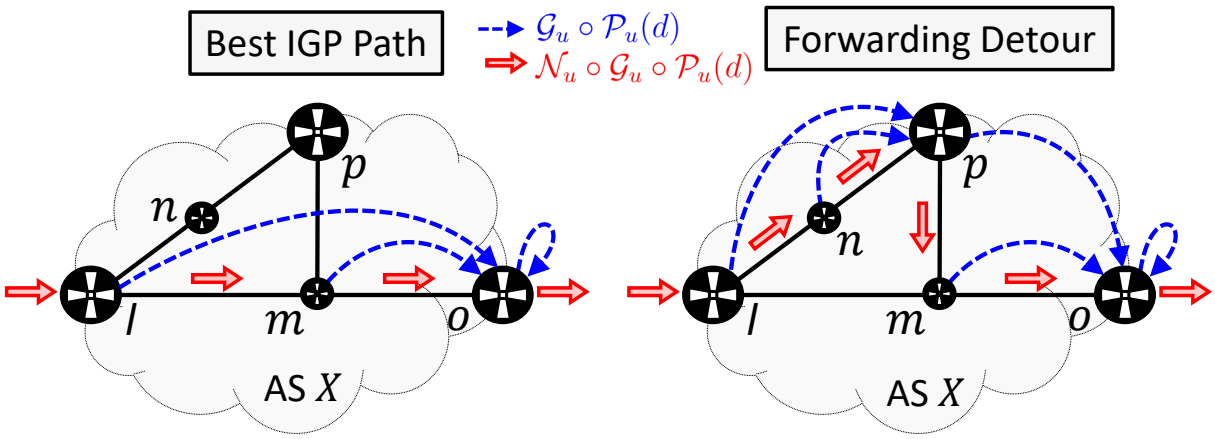


Figure III.8: Best IGP path vs Forwarding Detour. On the left, all routers choose o as gateway, hence no RIes occur and traffic flows through the best IGP path from l to o. On the contrary, on the right, router p introduces RIes choosing o instead of itself as gateway. As p should act as egress-ASBR, but instead sends traffic to m, then $p<$ introduces a FA. Since l would have straightforwardly sent traffic to m, had it selected o as gateway, then the resulting internal route is subject to FDs.

FDs imply FAs, but the converse may not hold. A router n used by an upstream router l may deviate its traffic towards a router o, but the resulting route may still be the best IGP path between routers l and o. In general, this occurs when the the best IGP path between r_0 and r_n either includes the sub-path from r_0 to r_k, the router that introduces the FA, or is a sub-path of the path between r_0 and $\mathcal{G}_{r_0} \circ \mathcal{P}_{r_0}(d)$.

By the two last theorems, we thus have **FDs** $\Rightarrow$ **FAs** $\Rightarrow$ **RIes**. When FDs occur, there is a router r_k between r_j and r_z that introduces a RI choosing a different gateway than r_j. This router r_k uses different next-hops to reach both gateways, i.e., $\mathcal{N}_{r_k} \circ \mathcal{G}_{r_k} \circ \mathcal{P}_{r_k}(d) \neq \mathcal{N}_{r_k} \circ \mathcal{G}_{r_j} \circ \mathcal{P}_{r_j}(d)$, and introduces a FA. Since r_k (re)directs traffic towards $\mathcal{G}_{r_k} \circ \mathcal{P}_{r_k}(d)$, when r_j would not have chosen r_{j+1} as next-hop for this gateway, a FD results. This is exactly the case in the right side of Fig. III.8 where $r_j = l$, $r_k = p$ and r_z could be either m or o.

Finally, note that when FDs occur, they create **multi-path routing patterns** inside ASes. This results from the fact that the best IGP path between the endpoints is used to forward traffic of prefixes either non-subject to RIes, or subject to FAs but not FDs (left side of Fig, III.8), but for those destinations and prefixes subject to FDs, the resulting internal routes differs (e.g. right side of Fig. III.8).

III.3.b A Tool to Reveal Visible Deflections

III.3.b.1 Discriminating Forwarding Patterns: Load Balancing, TE and Detours

As multiple patterns are not only related to FD, we first study current LB techniques, and their own impact on data plane information collected with `traceroute`. Different LB flavors exist and they produce distinct forwarding patterns, sometimes similar with that generated by FDs and TE.

Load Balancing in a Nutshell With the use of LB techniques, for any two routers i and e inside an AS X, multiple LB routes connecting them, denoted $\mathcal{R}_X^{LB}(i,e)$, might exist. This LB set results from the presence of load balancers, i.e., routers that may use different next-hops towards the same destination IP address. To balance packets across next-hops, these LB routers take into account either (some) packet header fields, or none at all [33, 342].

The simplest mode of LB, namely *per-packet* LB [5, 7], assigns packets to next-hops blindly, in a round-robin fashion. Consequently, with this approach, packets exchanged in a TCP connection are subject to reordering, a fact known to degrade the performance of TCP [204, 271, 47]. Moreover, faced to this LB flavor, any traceroute implementation may fail to reveal some links, and even infer false ones [33, 342]. Fortunately, per-packet LB is rarely found in practice [338, 340].

Other more sophisticated LB methods, which we call hash-based, decide next-hops relying on the use of a hash function, rather than blindly. More precisely, load balancers apply a hash on packet header values, and use the outcome of such computation to choose one among the available next-hops. As a consequence, in contrast with per-packet LB, packets belonging to the same TCP connection are always forwarded to the same next-hop. Due to this, such packets are said to belong to the same flow, and to have a similar flow-identifier (or simply flow-ID). Depending on the fields used to compute the hash, hash-based LB methods have historically been subdivided in two types: *per-destination* LB, or in short *per-dest* LB [5], and *per-flow* LB [5, 7, 9]. While the source and destination IP addresses are used as input in per-dest LB, the source and destination transport ports are additionally taken into consideration in per-flow LB.

Previous work has mainly focused on per-dest and per-flow LB, that are the two most widespread LB flavors [20], however, there exists a third hash-based LB flavor that has been systematically omitted in the literature, known as *per-prefix* LB [9, 4]. With per-prefix LB, the hash function is evaluated on the most specific prefix associated with the destination IP address of each packet. Note how this LB flavor contrasts with the other two hash-based LB methods, where the destination IP address is hashed at once. Due to this, we say that per-prefix LB is a *coarse-grained* LB type, while per-dest and per-flow LB are *fine-grained* LB types. We often indicate fine-grained LB types as per-dest/flow LB.

Finally, to mimic distinct hashing functions, load balancers also rely on additional parameters, such as the router-id or a configured seed value, to determine next-hops. These complementary inputs neither depend nor are extracted from the packets being forwarded. This allows to avoid polarization effects, that prevent the use of redundant routes [3], but has also been observed to produce next-hops re-mapping events often mistakenly attributed to routing changes [340].

Forwarding Patterns: LB Might Resemble FDs and TE We are interested in the forwarding patterns that the different hash-based LB flavors produce inside an AS, in order to be able to discriminate them from FDs[19].

For both per-dest and per-flow LB, the route that each `traceroute` reveals may vary as the destination changes. This possible variation of route also applies even when the destinations traced belong to the same prefix. As a consequence, for fine-grained LB types, exploring one prefix is enough to reveal all routes of $\mathcal{R}_X^{LB}(i, e)$.

On the other hand, per-prefix LB discriminates packets on a prefix basis and thus, for each prefix, the same next-hop is consistently chosen. Hence, each route of $\mathcal{R}_X^{LB}(i, e)$ is used only to forward traffic destined to the specific set of prefixes for which the same next-hop is chosen. With coarse-grained LB types, there is no route variation for different destinations belonging to the same prefix.

From this analysis, we can derive a critical concept: the forwarding pattern of per-prefix LB is similar to that of FDs and TE. This occurs since the three of them are *prefix-based mechanisms*. Indeed, in the same vein as the route used in per-prefix LB may change or not depending on the prefix that is considered, so does the occurrence of FDs, and the use of constrained TE paths. Hence, we say that per-prefix LB, FDs and TE produce *prefix-based forwarding patterns*.

III.3.b.2 Exploring and Partitioning the Forwarding Routes

To investigate existing forwarding patterns inside ASes, and determine whether they are prefix-based, we propose an analysis in four steps, referred to as exploration, prefix-grouping, multi-route discovery and merging phases, respectively. The exploration phase collects traces and identifies *ASBR-couples* of each AS, i.e., the ingress-ASBR and egress-ASBR of an AS that are simultaneously traversed by a trace. For these ASBR-couples, we determine their associated *internal routes*, i.e., the routes inside the AS that connect each couple. Then, the prefix-grouping phase looks for multi-path routing patterns across different ASes, i.e., whether depending on the traced prefix, the internal route revealed for an ASBR-couple varies. For each couple where such pattern is found, we continue the study with the multi-route discovery phase. This step extends the probing, aiming to reveal all internal routes that are used for each of the prefixes for which an ASBR-couple is observed. Finally, the merging phase discriminates between per-dest/flow LB and prefix-based mechanisms for each ASBR-couple. Next, we detail these steps relying

[19] Per-packet LB being rarely found in practice [340].

on the following notation: R is used to denote a route, $\mathcal{R}$ a set of routes, and $\mathbb{R}$ a set of sets of routes. The same convention is used for prefixes, i.e., we use P, $\mathcal{P}$ and $\mathbb{P}$, respectively.

Exploration Phase This step collects ASBR-couples and internal routes across ASes. For this, we perform a lightweight traceroute campaign, launching traces for some random prefixes (e.g. /24 subnets). An IP-to-AS mapping tool is used to determine ASBR-couples, and the internal routes inside each AS. According to the prefixes that are probed, it could happen that few traces traverse some couples. To enlarge the set of routes that are gathered for each of them, we collect a special internal route, that we call the *direct internal route (DIR)*. The DIR of each ASBR-couple is obtained by tracing the egress-ASBR, and is the internal route that starts in the ingress-ASBR and finishes in the egress-ASBR. As detailed later, the DIR has a key role in the detection of FDs, hence we discard those couples for which the DIR cannot be determined.

As a last step, we annotate the prefixes for which each internal route was revealed, i.e., the /24 subnet (usual longest BGP prefix [322]) covering the destination IP of the trace from which the internal route was extracted. The only exception is the DIR, which we consider associated to a /32 prefix, e.g. for a couple (i, e), then $e/32$. In the left table of Fig. III.9 we show the outcome of the exploration phase for a couple (i, e): tracing the prefixes of the left column $\{P_1, \dots P_7, e/32\}$, the routes on the right column $\{R_1, R_2, R_3, R_4\}$ are revealed.

Prefix-grouping Phase For the ASBR-couples that remain at this stage, we seek for a multi-path routing pattern by grouping the prefixes for which the same internal route was revealed. The outcome of the prefix-grouping phase for an ASBR-couple (i, e) is illustrated in the middle matrices of Fig. III.9, for both prefix-based mechanisms and per-dest/flow LB. Indeed, the prefixes for which the same route is observed, e.g. $\mathcal{P}_1 = \{P_1, e/32\}$, $\mathcal{P}_2 = \{P_3, P_7\}$ are respectively associated with R_1 and R_2, etc. As highlighted on the figure, the prefix-grouping phase may return the same result for per-dest/flow LB and prefix-based mechanisms. Thus, to be able to differentiate between both of them, further analysis is required.

Finally, note that for each ASBR-couple (i, e) of each AS X, two sets are stored: (i) a set of prefixes $\mathbb{P}_X(i, e)$ grouping the sets of prefixes for which the same internal route in X from i to e is observed; (ii) a set of corresponding internal routes $\mathcal{R}_X(i, e)$, one for each set of prefixes in $\mathbb{P}_X(i, e)$. At this stage, $\mathbb{P}_X(i, e) = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_r\}$ is a set of sets of prefixes, whereas $\mathcal{R}_X(i, e) = \{R_1, R_2, \dots, R_r\}$ is a set of routes, such that $r = |\mathbb{P}_X(i, e)| = |\mathcal{R}_X(i, e)|$. In particular, for the couples where $r = 1$, no multi-path routing pattern is observed and, therefore, there is no need to continue exploring them. On the contrary, when $r > 1$, then $\mathbb{P}_X(i, e)$ and $\mathcal{R}_X(i, e)$ are transferred to the multi-route discovery phase. This is the case in Fig. III.9, where $r = 4$.

Multi-route Discovery Phase This block extends the probing for the ASBR-couples delivered from the prefix-grouping phase. Our aim is to determine all the internal routes associated with each set of prefixes for which traces traverse an ASBR-couple. In other words, for each ASBR-couple (i, e) in any AS X, for each $\mathcal{P}_j \in \mathbb{P}_X(i, e)$, we look whether routes inside AS X other than $R_j \in \mathcal{R}_X(i, e)$ can be revealed probing destinations in $\mathcal{P}_j$. For this, we replace each route R_j with a set of routes $\mathcal{R}_j$ where we keep track of all internal routes in AS X from i to e that are found probing $\mathcal{P}_j$. As a result, note that while r remains constant, $\mathcal{R}_X(i, e)$ becomes a set of sets of routes $\mathbb{R}_X(i, e)$, i.e. $\mathbb{R}_X(i, e) = \{\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_r\}$. The unaltered set of prefixes $\mathbb{P}_X(i, e)$ and $\mathbb{R}_X(i, e)$ are then passed to the merging phase.

The right matrices of Fig. III.9 show the result of the multi-route discovery phase run for the couple (i, e) with $\mathbb{P}_X(i, e) = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4\}$ and $\mathcal{R}_X(i, e) = \{R_1, R_2, R_3, R_4\}$ as delivered from the prefix-grouping phase. Contrary to what was observed in the previous step, the outcome of the multi-route discovery phase is different for prefix-based mechanisms and per-dest/flow LB. For the first, each set of $\mathbb{R}_X(i, e)$ ends up containing a unique route, the one discovered in the exploration phase, i.e., $\forall j$, $\mathcal{R}_j = \{R_j\}$. Indeed, for prefix-based mechanisms, the route observed for any set of prefixes $\mathcal{P}_j$ remains constant indistinctly of the IP target inside $\mathcal{P}_j$ that is traced. On the other hand, for per-dest/flow LB, additional internal routes are discovered for each set of prefixes, e.g., $\mathcal{R}_1 = \{R_1, R_2, R_4\}$, $\mathcal{R}_2 = \{R_2, R_3, R_4\}$, etc. This happens because per-dest LB and per-flow LB are fined-grained LB types, meaning that the destination IP address is part of their flow-ID. Consequently, probing several IP addresses included in

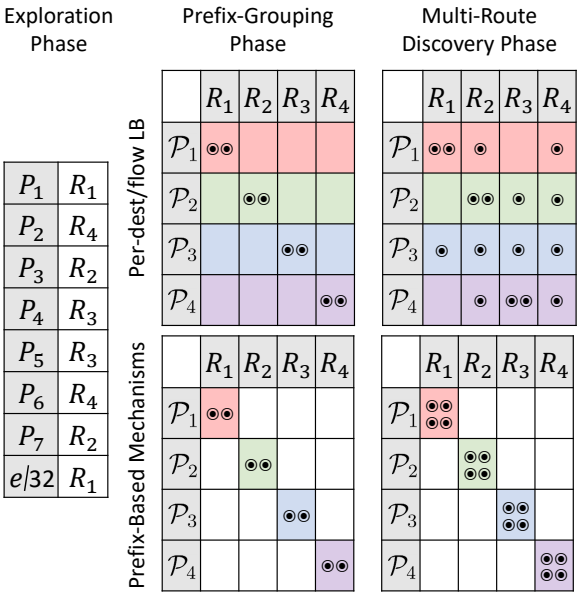


Figure III.9: Detecting the type of forwarding pattern for an ASBR-couple (i, e). While the colored cells represent the routes associated with each set of prefixes, the dots show those revealed while tracing. The exploration phase runs `traceroute` and reveals one internal route per measured prefix. The prefix-grouping phase then groups those prefixes for which the same route was revealed. At this stage, the result is the same for per-dest/flow LB and prefix-based mechanisms. The multi-route discovery phase extends the measurements to find the complete set of routes associated with each set of prefixes. For per-dest/flow LB we see that routes in common emerge across the different sets of prefixes. However this does not occur for prefix-based mechanisms. Ultimately, the merging phase will expose the nature of the forwarding pattern, merging all routes and prefixes into a unique set for fine-grained LB, but failing to do so for prefix-based mechanisms. Therefore, in the cases where more than one set remains at the final step, we can conclude that the forwarding pattern for (i, e) is prefix-based.

$\mathcal{P}_j$, it is likely that $\mathcal{R}_j$ will include more routes than just R_j. In an ideal case, for fine-grained LB types, it holds that for $\forall j \in \{1, 2, \ldots, r\}$, $\mathcal{R}_j = \mathcal{R}_X^{LB}(i, e)$, as what happens for $\mathcal{P}_3$ in Fig. III.9.

Merging Phase For each ASBR-couple (i, e), this step analyzes $\mathbb{P}_X(i, e)$ and $\mathbb{R}_X(i, e)$ to determine whether the forwarding pattern observed between i and e inside AS X corresponds to that of per-dest/flow LB or prefix-based mechanisms. During the multi-route discovery phase, while the sets composing $\mathbb{R}_X(i, e)$ do not change for prefix-based mechanisms, it is likely that they are enlarged and contain internal routes in common for fined-grained LB. Hence, we (always) proceed to convert $\mathbb{R}_X(i, e)$ into a partition, i.e., we repeatedly merge the intersecting sets of routes until no more overlaps exist among the merged sets. In this process, we also merge the subsets of $\mathbb{P}_X(i, e)$ accordingly. This operation results in $s \leq r$ sets composing $\mathbb{R}_X(i, e)$ and $\mathbb{P}_X(i, e)$.

The merging phase outputs different results for fine-grained LB flavors and prefix-based mechanisms, and thus allows to determine if a prefix-based forwarding pattern is observed for an ASBR-couple (i, e) inside AS X. [20] For per-dest/flow LB, it holds that $s = 1$, such that $\mathbb{R}_X(i, e) = \{\mathcal{R}_X^{LB}(i, e)\}$ and all prefixes in $\mathbb{P}_X(i, e)$ are also grouped into a unique set. In the example of Fig. III.9, all sets overlap[21], and thus the merging phase outputs $\mathbb{R}_X(i, e) = \{\{R_1, R_2, R_3, R_4\}\}$ and $\mathbb{P}_X(i, e) = \{\{P_1, \ldots, P_7, e/32\}\}$. On the other hand, for prefix-based mechanisms, since the sets do not overlap, as shown in the bottom-right matrix in Fig. III.9, the composition of $\mathbb{P}_X(i, e)$ and $\mathbb{R}_X(i, e)$ does not change, thus it holds that $s = r > 1$, and $s = 4$ in this particular example. [22]

[20] Note that per-dest/flow LB and prefix-based mechanisms may interfere with each other, generating more complex forwarding patterns.

[21] This condition is sufficient, but not necessary for $s = 1$ to hold.

[22] Indeed, for the multi-route discovery and merging phases to be applied on any ASBR-couple, a multi-path routing pattern must have been discovered in the prefix-grouping phase, meaning $r > 1$.

When per-prefix LB and TE are jointly present, looking at the number of sets composing $\mathbb{P}_X(i,e)$ and $\mathbb{R}_X(i,e)$ is not enough. The size and content of their merged subsets need to be analyzed in more details.

III.3.b.3 Use the Direct Internal Route to Conclude

The key to conclude is a *lonely DIR*, it enables the FD-verdict. To detect FDs for an ASBR-couple (i,e) of AS X, we propose looking at the set of prefixes associated with the DIR, the special internal route introduced in the Exploration Phase. The DIR, denoted $D_X(i,e)$, is the route inside X from i to e obtained by tracing e. This internal route is particularly important since it must hold that

$$D_X(i,e) \in \mathcal{R}_X^{LB}(i,e)$$

The networking rationale for this assumption is that, presumably, internal prefixes of ASes, such as the internal destination e of AS X, are not subject to FDs. In other words, regarding internal destinations, it is reasonable to assume that all devices are full-FIB routers.[23] Hence, $D_X(i,e)$ is not expected to detour, and always to represent a best IGP path, which by definition is included in $\mathcal{R}_X^{LB}(i,e)$.

When we conclude for a prefix-based forwarding pattern, i.e., $s \geq 2$, we then declare that extreme-FDs occur only if we see a *lonely DIR*, i.e., when:

$$D_X(i,e) \in \mathcal{R}_j \ \wedge \ |\mathcal{P}_j| < t(Z, \mathbb{P}_X(i,e)) \qquad \text{(Eq. III.1)}$$

where $t(Z, \mathbb{P}_X(i,e))$ is an adaptive threshold with $0 < Z \leq 1$ an adjustable parameter and $\frac{1}{s}\sum_{k=1}^{s}|\mathcal{P}_k|$ the number of prefixes that each set of prefixes $\mathcal{P}_m \in \mathbb{P}_X(i,e)$ should contain assuming a uniform distribution. We define this adaptive threshold with the following equation:

$$t(Z, \mathbb{P}_X(i,e)) = \frac{Z}{|\mathbb{P}_X(i,e)|} \sum_{\forall \mathcal{P}_k \in \mathbb{P}_X(i,e)} |\mathcal{P}_k| = Z \cdot \frac{1}{s}\sum_{k=1}^{s}|\mathcal{P}_k| \qquad \text{(Eq. III.2)}$$

Note that, depending on the considered ASBR-couple (i,e), the values of $\sum_{k=1}^{s}|\mathcal{P}_k|$ and s, obtained analyzing $\mathbb{P}_X(i,e)$ and $\mathbb{R}_X(i,e)$, usually change. On the other hand, the value of Z can be modified to tune the precision and recall of the FD-verdict, i.e., adjust how cautious we are to declare that FDs occur. The lower Z, the stricter the condition.

The reasoning for the threshold we compute is as follows. In the absence of FDs, while the constrained routes composing $\mathcal{R}_X^{TE}(i,e)$ may carry the traffic of a limited number of prefixes, the LB routes $\mathcal{R}_X^{LB}(i,e)$ evenly distribute the load of the main bulk of prefixes. When FDs occur, some prefixes are forwarded across the routes in $\mathcal{R}_X^{FD}(i,e)$. This can strongly modify the usual distribution of prefixes across routes: fewer prefixes are associated with LB routes. The more prefixes subject to FDs, the less the IGP routes are used to carry transit traffic. In particular, in the event of extreme-FDs, most prefixes are subject to FDs. Hence, looking at the set containing the DIR, we can infer whether the LB set is associated with few or no external prefixes, and we argue that this is a strong hint revealing the occurrence of extreme-FDs.

To illustrate the behavior of the FD-verdict, let us recall the example of Fig. III.9, and assume that while tracing a complementary set of prefixes $\mathcal{P}_5 = \{P_9, P_{10}, \ldots, P_q\}$ a new detouring route R_5 was always revealed. Note that, in the updated example, in total q prefixes are measured, 8 from Fig. III.9, and the remaining included in $\mathcal{P}_5$. Hence, the higher q, the more prefixes subject to FDs. Since R_5 was not revealed before, then s increases by one for both per-dest/flow LB and prefix-based mechanisms. Indeed, for the first, instead of $s = 1$, we would now have $s = 2$: the new set $\mathcal{P}_5$, and $\{P_1, P_2, \ldots, P_7, e/32\}$, the previously merged one. A uniform distribution would thus require finding $q/2$ prefixes in each set. Assuming $Z = 0.1$, our FD-verdict concludes for extreme-FDs if less than $0.1 \cdot q/2$ prefixes are associated with the DIR, i.e., if $q > 20 \cdot 8$. On the other hand, for the prefix-based mechanisms, we would go from $s = 4$ to $s = 5$, each set containing 2 prefixes, except for $\mathcal{P}_5$. In this case, following the same reasoning as before, the condition to declare extreme-FDs is $q > 50 \cdot 2$. In particular, these examples highlight that, for the FD-verdict to be robust, the number of prefixes analyzed per ASBR-couple needs to be high, e.g. at least 100 prefixes.

[23] Since the IGP does not suffer from similar scalability issues as BGP does, all internal prefixes are expected to be installed in all routers. In addition, IGP prefixes constitute the backbone of an AS and removing them from the FIB of any router would represent a minor scalability gain while letting BGP running on top of a flawed IGP network.

III.3.c Results: Numerous Detours and partial-Forwarding Base

We run measurements from 100 NLNOG RING's VPs, among which only 92 where able to complete the analysis. In the exploration phase, out of the 100K traces we run, we extracted on average 3 internal routes per trace distributed across 7500 ASes. From those internal routes with unambiguous borders, we see that we traverse from 1405 up to 2205 distinct ingress-ASBRs (except one VP where the value raises up to 2335), between 5662 and 8758 unique egress-ASBRs, and from 6475 to 11590 different ASBR-couples. However, our results indicate that most couples are not commonly encountered: at least 50% appear only once, and 96% are traversed at most for 30 traces. Hence, while the requirement of finding 100 prefixes per couple has a limited effect on the final dataset we analyze, it allows us to be conservative, avoiding to introduce false positives/negatives. On the other hand, when tracing the egress-ASBRs to collect DIRs, we had a success rate usually between 50% and 60%.

Our FD-detector was able to analyze 3963 ASBR-couples spanning 54 ASes. In terms of couples covered and traversed ASes, the main tendency is a linear increase with the number of VPs. However, the decreasing slope of the distribution and the plateau at its tail suggest that the gain after 70 VPs is negligible. Indeed, beyond that point, we are able to investigate only 138 additional couples. In the end, we find extreme-FDs in 25 ASes, across 168 ASBR-couples and 65 ingress-ASBRs.

III.3.c.1 Distribution of FDs per AS and ASBR-couples

A Binary Effect We discover a **binary effect** around FDs, i.e., **either all the observed transit traffic traversing a couple detours, or none does**. We observe that ∼96% of the ASBR-couples for which $s = 1$ and all prefixes are forwarded along best IGP paths. On the other hand, the ∼4% remaining are those ASBR-couples for which $s = 2$. Since the rate of prefixes associated to the DIR is always 0%, then all these couples are subject to FDs, i.e., the rate of prefixes subject to FDs is of 100% (except for the DIR, of course). This shows that our FD-detector is not sensitive to any calibration issue concerning the adaptive threshold $t(Z, \mathbb{P}_X(i, e))$ in the FD-verdict. In other words, there are no gray regions: when $s = 2$, no false negatives can occur since it always holds that 100% of the prefixes are not associated with the DIR, i.e., lonely DIRs are always completely alone.

Fig. III.10 shows the breakdown per AS of the 168 ASBR-couples subject to FDs, sorted by increasing relative fraction across ASes. We observe no general trend, indicating that the prevalence of FDs is AS-specific, e.g. depending on both router's hardware and OSes in use. This analysis is supported by the fact that, even though most ASes have few measured couples with FDs, less than 10 in general, the relative values spawn from as low as almost 0% to up to 100%. Moreover, while one could argue that the left side of the Fig. III.10 seems to be populated with ASes with a high AS Rank [2], the same holds for example for AS6762, that has all of its measured couples with FDs. In addition, it is interesting to mention the case of AS2914, with a relative value around 10%, but more than 50 couples for which traffic detours; and those of AS7473 and AS4230, both with 20 couples exhibiting FDs, but that represent 40% and 80% respectively of the total measured. These three cases emphasize the lack of a general tendency among ASes, i.e., the FD-phenomenon seems to depend on configurations specific to each AS. More in depth, considering the granularity of the ingress-ASBR, across the 168 ASBR-couples subject to FDs, we observe that they span (only) 65 ingress-ASBRs.

III.3.c.2 Speculating on the Root Causes

Based on previous results, this section elaborates an explanation of what may have generated the FDs we observed. While root causes behind forwarding detours may be multiple, we argue that the patterns observed seem clear cut. Indeed, even if the core contribution of this work is our methodology to detect FDs, the binary effect we found makes us believe that we are also able to pinpoint the most likely reason behind the FDs we collected. In short, the FDs we detect seem to result from scenarios involving partial-FIB routers, i.e., where routers keep IGP prefixes but delete a large fraction (if not all) of BGP prefixes from the FIB. Note that this is emphasized by the binary effect, that is even more severe that what we previously labeled as extreme-FDs.

A partial-FIB router x with no BGP prefixes installed and relying on a default route, systematically sends traffic towards a default gateway y. A priori, if y considers itself the best exit point of the AS for

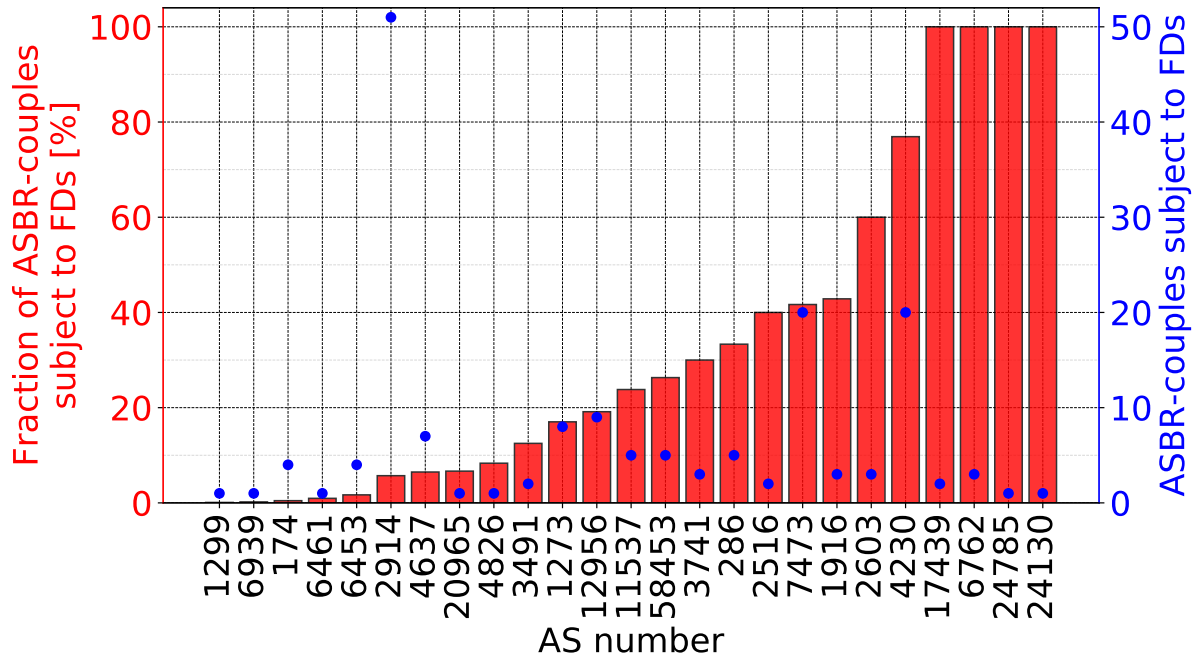


Figure III.10: Quantification of ASBR-couples subject to FDs per AS. While most ASes have less than 10 couples subject to FDs (blue dots), the fraction they represent out of the total in their AS (red bars) largely varies. This indicates that the problem of FDs is AS-dependent.

all BGP prefixes then, no FDs occur. However, depending on the best covering prefix of the destination IP address of the packets being forwarded, y may likely redirect transit traffic towards another ASBR z. This is similar to what happens with prefixes P_R and P_B in the example shown in Fig. III.6 for $x = ASBR_1$, $y = ASBR_2$ and $z = ASBR_3$, where traffic for P_B detours, but that of P_R does not. More generally, in all cases where the best IGP path from x to z does not go through y, FDs occur.

We measure the fraction of routers x that choose a y, as default gateway, which never chooses itself as exit point of the AS, and all traffic detours. This could be the case, for example, if y was not an ASBR, but rather a core router. The resulting proportion is impressive: almost 30% of x falls in this category. On the other hand, cases where only a limited fraction of the traffic detours represent an interesting case of study that may result from multiple causes. A trivial explanation could be that the default gateway was well chosen. However, other causes, more complex, are possible. For example, it could happen that traffic exited the AS before reaching the gateway, hence avoiding FDs for these egress-ASBRs. Another plausible explanation could be that the ingress-ASBR i was actually not the partial-FIB router, but rather a core router x on which i relies. In such a scenario, only those prefixes for which traffic ingresses via i, and then x is traversed, will lead to few ASBR-couples subject to FDs.

Title of the publication	Name of the venue	Year	Reference
Filtering the Noise to Reveal Inter-Domain Lies	Traffic & Measurement Analysis (**TMA**)	2019	[DFMP+19]
The Art of Detecting Forwarding Detours	Transactions on Network & Service Management (**TNSM**)	2021	[DFPM+21]

Table III.7: Summary of my publications related to routing anomalies like detours and BGP lies

Table III.7 summarizes our two main achievements looking at forwarding anomalies in transit routes.

Detours: Conclusions and Perspectives

In this section, we were interested in forwarding detours, i.e., when traffic inside ASes eventually flows through forwarding routes that divert or diverge from the expected best IGP paths. We made the following contributions:

- We investigate the root causes that produce forwarding detours, showing that they result from routing inconsistencies that generate forwarding alterations. However, not all routing inconsistencies and forwarding alterations generate forwarding detours.
- We explain why detecting forwarding detours is challenging: they generate multi-path routing patterns similar to those introduced by load balancing and traffic engineering techniques. Moreover, we take into account per-prefix LB, an LB flavor never previously studied in the literature, and propose a new taxonomy differentiating between fine-grained and coarse-grained LB types, which vary the granularity at which flows are defined with respect to the destination IP address of packets.
- We design a methodology that, without requiring privileged knowledge from the networks being analyzed, e.g., knowing the IGP metric, is able to detect whether forwarding detours occur inside them. Our methodology consists in studying the forwarding patterns inside ASes, i.e., the specific sets of forwarding routes that are revealed for different prefixes when tracing multiple IP addresses contained in each of them.
- We propose an FD-detector, to the best of our knowledge the first of its kind, tuned to detect extreme-FDs, i.e., FDs that affect numerous external prefixes. We validate the behavior of the FD-detector with emulations and on a network where we have ground truth.
- We analyze the FD-phenomenon in the wild running our FD-detector from 100 nodes of the NLNOG RING monitoring infrastructure, and find FDs in 25 out of 54 ASes. We find forwarding detours in multiple ASes, with a remarkable binary pattern in which transit traffic traversing between two border routers of an AS either never detours, or always does.

Despite these contributions, several research questions are left opens. In particular, the studies of [DFMP+19] (quantification of BGP lies) and [DFPM+21] (Forwarding Detours) are closely related: besides the fact that both allow to detect similar issues in the Internet (deflections within an AS and across them), they may sometimes share a similar root cause, i.e., forwarding scalability issues.

BGP lies occur when, for a given prefix, an AS does not forward the packet towards the neighbor it is supposed to. That is the one it advertises within its BGP offers. ASes having scalability limitations may use partial-FIB routers, using workarounds like default routes that may lead to unintentional BGP lies. Indeed, if the control plane is still working but not the forwarding ones, a mismatch between the two can occur. This brings us to this question: can we extend our methodology to discriminate malicious ASes trying to hide their lies from ASes that only suffer from technical limitations? The goal is to improve the filters designed in [DFMP+19] to look for a refined classification of lies. I aim to continue in such a direction and better analyze intersections between BGP lies and FD.

III.4 Past and Future Works: Topology Discovery, Network Analysis & Anomalies Detection

Topology Discovery, Modeling and Analysis

This long term adventure has started at the end of my Ph.D with the guidance of Jean-Jacques Pansiot, my former director, and pioneer in this topic. At the time, he was looking at a multicast tool under the radar, `mrinfo`. As we will briefly see in this section, this probing tool comes with many advantages with respect to `traceroute`. During my post-doctoral year at the Université catholique de Louvain-la-Neuve, we extended our preliminary efforts with the help of Virginie van der Schrieck, Olivier Bonaventure and Benoit Donnet to publish a first common paper in the field [MVdSD+09]. Then, we continued the project to not only map router level (blind) sub-graphs but precisely extract ISP maps with their borders [PMDB10]. It was one of the two first existing attempts to do so, and the first of this kind with `mrinfo` data. Indeed, with IGMP probing, `mrinfo` is able to capture, in a single query, all local and neighbor interfaces of a given router. Moreover, it allows for looking at layer-2 networks. Such features are very complementary to the ones offered with `traceroute` and ICMP probing in general as, with both tools, one can provide the accurate structural topology with the former and the forwarding paths in use with the later (and so its logical properties). Finally, we start to enhance our initial set of probing tools with MERLIN ([MDP+11] has been designed to ensure efficiency by decoupling the requests from the queries and thus speeding up significantly `mrinfo` probing campaigns) and then conduct new large scale campaigns during the master thesis of Pietro Marchetta [Marchetta, 2010] thanks to the collaboration of Antonio Pescape (University of Napoli). Our collaboration led to two notable papers: [MMD+12] and [MMD+10]. Wile the first focuses on combining ICMP and IGMP probing to mitigate their respective limits, the second provides an overview of our overall measurement system: while MERLIN clients are scattered on the Internet, we rely on a central controller to orchestrate and coordinate the probing.

In the meanwhile, we also have studied various characteristics of layer-2 networks thanks to our various datasets [MDBP10, TQM+13]. Either analyzing the node degree distribution or proposing a bipartite graph generator based on the patterns observed with `mrinfo`. Rather than focusing on MPLS clouds as we have done later on, with `mrinfo` data, one can access to point-to-multipoint links relying on Ethernet Switches. These two papers investigate how much such layer-2 links deform our understanding of the Internet topology. Generally speaking, this project is now terminated as `mrinfo` is not anymore under the radar as most ISPs have now turn it off or filter such queries (to avoid revealing what they consider as confidential information), at least for requests coming from outside their networks. Our set of tools are thus now deprecated but the insights we gain at the time still valuable.

As a recap, Table III.8 lists my publications in the field.

Title of the publication	Name of the venue	Year	Reference
Quantifying ASes Multi-connectivity using Multicast Information	Internet Measurement Conference (**IMC**)	2009	[MVdSD+09]
Extracting Intra-Domain Topology from mrinfo Probing	Passive & Active Measurements Conference (**PAM**)	2010	[PMDB10]
Topology Discovery at the Router Level A New Hybrid Tool Targeting ISP Networks	Journal on Selected Areas in Communications (**JSAC**)	2010	[MMD+10]
On the Impact of Layer-2 on Node Degree Distribution	Internet Measurement Conference (**IMC**)	2010	[MDBP10]
MERLIN: MEasure the router level of the INternet	Conference on Next Generation Internet (**EURO-NGI**)	2011	[MDP+11]
Quantifying and Mitigating IGMP Filtering in Topology Discovery	Global Communications Conference (**GLOBECOM**)	2012	[MMD+12]
Towards a Bipartite Graph Modeling of the Internet Topology	Computer Network (**COMNET**)	2013	[TQM+13]

Table III.8: Summary of my publications related to topology discovery and analysis

Capturing Anomalies

Finally, during the Master thesis of Andreas Guillot, [Guillot, 2018], that has also assisted us with *DCART*, we develop a framework to detect anomalies [GFW+19]. The complexity of the Internet often impedes ISPs to finely pinpoint the causes of service degradation or disruption when the problem lies outside their networks. To improve the state of the art on this topic, we have designed a tool detecting remote connectivity loss using Internet Background Radiation (IBR, [48]) through a simple and efficient method. IBR is unidirectional unsolicited Internet traffic, which is easily observable by monitoring unused address space. IBR comes with two remarkable properties: it is originated worldwide, across diverse ASs, and it is incessant. We have first shown that the number of IP addresses observed from an AS or a geographical area follows a periodic pattern. Hence, using Seasonal ARIMA ([160]) to statistically model IBR data, we are able to predict the number of IPs for the next time window. Significant deviations from these predictions indicate an outage. We evaluated our tool using data from the UCSD Network Telescope[24], operated by CAIDA, with a set of documented outages. Our experiments have emphasized the good performance of our overall method: the trade-off between true-positive rate (90%) and false-positive rate (2%) is reasonable and largely outperforms CAIDA's own IBR-based detection method. Furthermore, performing a wider comparison against other existing methods relying on distinct datasets, i.e., with BGP monitoring and active probing, we have observed that our method not only shares a large common set of outages with them, but provides in addition many specific outages that would otherwise remain undetected.

Conclusion and Perspectives

In this chapter, I have presented my main achievements in the area of IP measurements and analysis. We have first proposed a tool to reveal hidden MPLS tunnels, and their underlying invisible IP links and nodes. Second, we looked at monitoring platforms using passive and active measurements and design one able to correlate losses, loops and routing changes. We have finally studied detours and lies on the Internet, possibly leading to routing loops or at least deflections from best paths, and showed they do not seem that negligible.

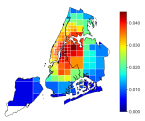
Most of the difficulties and challenges, that we have try to address, lie in the technical limits and biases of each available tool such as `ping` and `traceroute`. In particular, we have experimented and cross-validated our claims on real deployed devices, or at least relying on emulations (e.g. with on GNS3), to ensure their consistencies. We were able to discover many effects leading to specific patterns thanks to such experiments. The data and the code resulting from our efforts are freely available and we expect researchers to re-use our tools and their outcomes for other needs.

In addition to extensions proposed in the conclusion of each previous section, I have several ongoing and long term projects in the field. Most are described in Chapter V while I discuss and introduce minor other ones here. While load balancing has been extensively studied in the past, current tools still do not consider all necessary features. I aim to design a tool examining all protocols (ICMP, UDP and TCP) with distinct manipulation (port numbers but not only) on the field to be hashed. Analyzing the return paths in details, its length and other patterns of this kind can also be useful to carefully look at RTT of each subflow ([268]). Besides, I think that it is time to consider previous measurement campaigns and their outcomes to better control the probing strategy of next ones. Periodically looking at routes and their load distribution opens new model to track their evolutions in time. I aim to design several novel adaptive probing strategies to efficiently capture the path diversity and improve the state of the art methods [340, 20]. The main goal is to reduce the probing cost on the fly, e.g., specifically to an AS or a router brand, by leveraging all available information instead on relying on blind static points with no a priori knowledge, as current multi-path campaigns continue to do at each run.

Finally, I envision to restart my collaborations with RENATER and GEANT folks in a near future. Having such opportunities during the last years makes me realize how important it is to have access to real network practices and usages to design both valuable routing and measurement systems. In order to ease the innovation, the transfer from academic research to real deployed systems maintained by engineers should be achieved in a incremental and graceful manner to have a chance of being actually

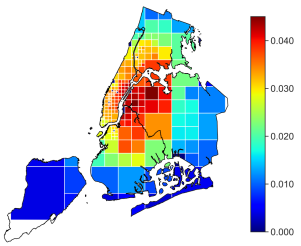
[24] https://www.caida.org/projects/network_telescope/

useful in a near future. It is beneficial for both actors since engineers can report the main difficulties they encounter in their networks to networking researchers that can in turn focus on concrete and interesting problems and take the time to solve it in depth. Too often, only short term patches are deployed while there exists more long term solutions that can be designed by the research community. Moreover, with new architectures and recent flexible hardware, many opportunities exist regarding such collaborations: progressively deploying new features directly within the production network to evaluate their benefices in real conditions becomes now possible skipping the step involving router vendors.



Chapter IV

IoT, Smart City Services and Privacy



Routing and Forwarding are not only devoted to core wired IP networks. Wireless Edges Infrastructures like Wireless Sensors Networks (WSN) or the Internet of Things also require these functions. While the deployment scale is not the same (local networks rather than inter-connected WAN), new challenges and constraints arise on the other hand: first, the medium is unstable and difficult to share due to radio interferences and collisions, second, while many deployments are constrained with low capacity (in terms of memory, battery and computation resource in general), one also possibly needs to consider mobile devices (both sensors and actuators) involved in the deployment. The former challenge is mainly tackled at the link layer with numerous Medium Access Control (MAC) proposals [198, 164, 138], both with deterministic and probabilistic schemes. The later can impact all layers, in particular the IP one: routing schemes need to be tuned to take into account limited memory, computation and battery constraints. This introduces the first topic we will look at: how to design efficient algorithms and routing protocols to extend the network life time in WSN? Rather than looking at centralized or deterministic schemes [164], we will explore the area of distributed algorithms with localized decisions.

In addition, within the same context, data transiting in IoT applications is often confidential as related to personal usage. Indeed, when measuring and collecting indicators necessary for smart cities applications, sharing them often comes at the cost of exposing confidential information that can be used for malicious purposes. We develop and propose routing architectures able to mitigate such personal exposures and leaks. We aim to rely on data aggregation and filter applications within the network (in-network processing) to avoid raw data being transmitted and forwarded as it is. Finally, we have also looked at the opportunity to model policy access with metagraphs and to deploy them using micro-services.

Contents

IV.1 Energy is Limited

My first experience outside the area of wired networks started with Antoine Gallais and the master thesis of François Clad [Clad, 2011]. Originally, we have published an exploration paper [MG09] exhibiting the potential of turning off the forwarding capabilities of a given subset of sensors. Then, we have proposed an efficient routing scheme whose goal was to achieve this objective, i.e., minimizing the number of relay nodes [CGM12]. This first research question, that we have tried to tackle, can be stated as follows:

Research Question

How to Preserve the Energy in WSN by better Decoupling their Sensing and Forwarding Abilities?

The two next sections briefly summarize the general problem, related challenges and our first contributions in the field.

IV.1.a Problem Statement in Wireless Sensor Networks

Saving energy in WSN [17] can be achieved thanks to activity scheduling [260]. In this work, we have decided to consider activity scheduling in WSN through the criterion of the area coverage. Area coverage must be ensured by connected active users in order to ensure the reliable data collection by the sink stations. Existing protocols aim at reducing the proportion of active nodes while preserving full area coverage by connected active nodes sets; they may rely on purely local decisions, and therefore take their global consistent decisions solely based on available local information. The global goal is to achieve full area coverage by connected sets while keeping a small set of active nodes or low control traffic induced by the activity scheduling protocol itself. Depending on the target optimization (proportion of active nodes, induced control traffic), proposed contributions either aim at drastically diminishing the number of involved active sensors [327] or propose to allow relatively larger proportions with lower communication overhead [130]. All solutions are studied for networks of variable density, thus showing the scalability of these protocols. For a given density, some works have also proposed to vary the coverage degree [14] or to relax the coverage constraint once a local detection threshold is considered [66]. In each case, the connectivity of the active nodes set is provided along with full area coverage.

To ensure connectivity, one of the most obvious solution is to rely on the SR/CR ratio (with SR and CR respectively denoting the Sensing and Communication Ranges – typically considering the unit disk model). In [364], authors show that ensuring full area coverage is enough to guarantee connectivity as long as $CR > 2 \times SR$. Then, most of existing contributions make the same assumption and solely focus on area coverage preservation. Other solutions are based on connected dominating sets [321] that are built with a local decision making process. A node u can indeed get passive provided that its set of active neighbors is connected, thus meaning that u is not essential to the local connectivity.

In our work [MG09], we rely on the activity scheduling protocol proposed in [130]. This area coverage solution rely on a local coverage evaluation mechanism. This phase is essential to ensure that a node would not decide to be passive without being fully covered locally. In order to tune the quality of the coverage ensured by the set of active sensors, our evaluation uses both a probabilistic detection model and an adapted coverage evaluation scheme that is detailed in [129]. A node evaluates its coverage depending on a local sensing threshold. If every physical point of its theoretical sensing area is covered with a probability greater than the given threshold, then the area is said to be covered. The variation of this threshold allows to have a more or less relaxed global coverage. Unlike most of existing solutions, this option preserves the network connectivity without relying on any assumption regarding the SR/CR ratio. Indeed, it relies on a local connectivity criterion that prevents any node to be passive if its set of active neighbors is not connected. Such a protocol ([129]) is therefore symptomatic of the previously described solutions as it allows to tune the coverage constraint but does not propose any similar option for the overall network connectivity.

In order to evaluate the communication possibilities let by this local connectivity criterion, we evaluated the path diversity of the network. Our results show that the local connectivity criterion used so far

in most of activity scheduling solutions leaves networks with a very large path diversity. In order to take advantage of this diversity, several options are possible. On the one hand, it may help the network to be more robust as the number of alternate paths from each sensor towards any sink stations is large. Such a feature is of high interest, especially in WSN where nodes could be prone to failures. It could also be used to adapt and balance the load according to the battery levels. Once some neighbors start running out of energy, the data traffic could be oriented toward nodes with higher capabilities, without having to wonder whether this node will be able to reach the sink or not. On the other hand, we can consider that this large diversity allows for more energy savings. We can indeed reduce the set of active nodes in order to have this diversity reduced in the set of active nodes. Yet, such a reduction would also impact the coverage quality in the same way as the modification of the local sensing threshold did. We therefore aim at modifying existing protocols in order to add a new activity state, that could be referred to as a *sensing-only* state. Sensing-only consists in solely sending sensed data but does not participate in multi-hop communications. In other words, sensing-only nodes would never relay communications from other sensors. Nodes in this state would solely participate in area coverage without taking part to multi-hop communications. This would reduce the communication density without altering the sensing coverage. Connectivity from each sensor to one of the sink stations should be preserved while maximizing the number of sensing-only nodes. This raises the problem of defining local decisions that would build such sets. A centralized solution could be to construct a reverse SPT (or DAG) rooted at the sink maximizing the number of leaf nodes, each sink having its own converge-cast structure (a Destination oriented DAG or DoDAG, i.e. the reverse shortest paths rooted at the sink, forming at least a tree). Sensing-only nodes would then be the leaves of this reverse SPT or DAG. We looked at localized decisions that would globally lead to the construction of a set of trees minimizing the number of forwarders node (sensing+forwarding state). Such local decisions are hard to define and the next section provides the means we rely on to tackle this challenging problem.

IV.1.b Towards an Efficient Dynamic Sink-oriented Backbone

Our goal is here to define a "virtual backbone" to enable efficient multi-hop routing protocols in WSN. The structure of this backbone should verify several properties to ensure the network efficiency and survivability. In particular, a small-sized backbone allows for saving the global energy consumption. Nodes within the virtual backbone are called *relays* and are expected to form a Connected Dominating Set (CDS) while others are called *leaves*. Thus, minimizing the size of the backbone to form a Minimal CDS (MCDS) turns equivalently the whole graph into a Maximum Leaf Spanning Tree (MLST): the more leaves, the less nodes involved in the routing plan. Subsequently, energy-efficiency can be achieved by finely adapting the communication stack (e.g., at the medium access control layer [198]).

In our model, we consider the case of a nomad sink, and two communications modes may co-exist. First, when no sink station is around, static sensors communicate for data redundancy or signalization purposes. The emphasis is then only put on energy efficiency. Second, once a nomad sink enters the network, all monitoring data should quickly reach the sink station, thus leading to a sink-oriented backbone. In such a case, the priority is to optimize convergecast routing. Since the two objectives (energy saving and convergecast routing efficiency) may co-exist in WSN deployment, our purpose is to allow static sensors to switch from a pure energy-efficient communication mode to a reliable and efficient data collection routing scheme.

Looking in particular at the energy-efficient communication mode to design the ideal backbone (minimizing the number of relays), only distributed approximations for constructing MLST are practically deployable in WSNs. Indeed, even using a centralized algorithm, the MLST problem is NP-hard [134] and, for the kind of WSN deployment we target, centralized approximations are not suitable. Furthermore, a convergecast efficient backbone rather consists in minimizing all hop distances between the sink and other sensing nodes. This computation is achieved by constructing a SPT rooted at the sink with a protocol such as a gradient [188]. Neither optimal MLST resolution is tractable ([126]), nor centralized approximations ([316], [218]) are not practically deployable in WSNs. We only rely on such techniques (on small instances) to position our heuristics regarding (near) optimal results. We adapt the formulation of Fujie [126] to implement it in CPLEX [175] as well as the approximation of Solis-Oba [316].

The solution whose design is the closest to our approach is a contribution of Misra and Mandal [243].

They describe a heuristic for approximating MCDS using collaborative cover. However, its message complexity is too high for low-power environments such as WSNs. Wu and Li [355] introduce a localized solution for constructing a non bounded approximation of a MCDS. Using solely a local 2-hop knowledge, each node determines its state, *dominating* or *dominated*, and informs its direct neighborhood. A set of pruning rules are used to limit the CDS size while maintaining its connectivity. This algorithm achieves a small-sized CDS with a low time complexity. The cost in terms of message exchange is about 1 to 2 messages per node. More recently, several derived solutions have been proposed such as energy-aware [354] and generalized [85] heuristics. In our work [CGM12], we use the *node degree based rules* presented in [354] as a building block of our proposal. The tradeoff between both communication schemes is defined by a single parameter allowing to explore the underlying routing backbone diversity: from the most convergecast efficient model to the most energy efficient one. For that purpose, we introduce a variable, denoted λ and defined in $[1, \infty[$, to make the result tend to one of the spectrum sides. A λ value close to 1 results in an energy efficient SPT generation, while largest values would tend to preserve (and even improve) the preliminary MCDS energy saving advantages.

In summary, our proposal works as follow:

1. A Wu-Li like algorithm is applied on the initial graph to obtain a preliminary small sized CDS;
2. Our gradient variant using $\lambda \to \infty$ (and an elected sink) is triggered to prune this preliminary CDS and thus reduce the backbone size;
3. As soon as a nomad sink decides to collect data, it informs the network by triggering our gradient with a λ value reflecting the chosen tradeoff between energy saving and convergecast routing efficiency;
4. Upon leaving the vicinity of the network, the sink notifies its direct neighbors in order to trigger a new gradient with $\lambda \to \infty$. The network falls back to its previous energy saving configuration.

Our version of the gradient protocol is able to prune the Wu-Li CDS to reduce the fraction of relays while, at the same time, reducing path length for convergecast communications. For every simulated topology, and when calibrated for energy efficiency, our solution systematically outperforms the 2-approximation algorithm [316] we have used as reference. We have also shown that our hybrid approach achieves an interesting trade-off between path optimality and the fraction of active relays node when considering the data collection mode. However, even active nodes can save more of their energy with either deterministic schemes or duty cycle approaches (to both reduce collisions and the energy consumption).

I envision to continue working on energy efficient routing schemes for IoT networks as developed in Sec. V.3. Indeed, energy-efficiency can be further improved with duty cycles approaches relying on multiple radios. Our activity scheduling model was only the coarse basis of the more refined proposition I will formulate.

IV.2 Privacy is Essential

This work takes place in the context of a JCJC ANR that I lead, *Nanonet*. The goal is to provide in-network data aggregation to preserve data privacy in IoT domains. Thanks to the funding of Renato Neto (under the direction of Fabrice Theoleyre since 2019, his Ph. D. defense is planned for July 2022 [Neto, 2022]), we have looked at privacy in the IoT context, and we published several papers considering a NDN architecture to handle the resulting data streams: [NMT20b], [NMT20a], [NMGT21] and [NMT21]. While the two first focus on the in-network aggregation NDN architecture we propose (Sec. IV.2.b), and the third one provides a performance analysis based on real deployment datasets (not presented here), the last one is briefly introduced in section IV.2.b. More generally speaking, the overall question that we have started to address in this project is the following:

Research Question

How to Guarantee the Privacy of Data-Streams with In-network Aggregation for Multi-Domains IoT Networks?

The next two sections elaborate a bit more on the context of application and introduce the outlines of our contributions in this new field of research to me.

IV.2.a Towards a Novel Secured Architecture with In-Network Aggregation

Privacy has always been a major requirement in IoT , particularly for sensitive information (*e.g.* healthcare, personal sensors, smart buildings). However solutions must beware of the limited resources of IoT devices since costly operations jeopardize battery life due to longer processing and extra network exchanges [215].

By using attribute based access control, cipher keys are distributed by a mediator that verifies if the attributes of the consumer match the requirements of the producer [205]. Such approach requires several exchanges with the mediator in order to acquire the access keys before each data acquisition which can be quite costly to some IoT devices.

Anonymization, aggregation and filtering are alternatives to ciphering. They provide privacy by decreasing the precision of data. Aggregating data from several sensors hides the specific values of each sensor while providing information on the global population [350]. Filtering and anonymization remove or mask sensitive information from data until some quantifiable privacy requirements are reached, such as k-anonymity [323] and ϵ-differential [98]. Data masking is a common practice when disclosing datasets with sensitive data. Removal of attributes, generalization (*e.g.* rounding or averaging numerical values) and noise addition are common operations. These increase privacy by removing exact values and descriptions from datasets while maintaining some utility. Those operations are less costly than encryption but do not provide full opacity [94].

Towards a Multi-Domain Architecture Interoperability in the network stack is a requirement to allow several autonomous IoT domains to exchange data. Typically, proxies may help to translate information between different applications [70], or even aggregate data among different domains. They can also unify the semantics of data from different networks [89]. However, these proxies solely enable inter-domain data exchange, they do not handle privacy natively. Instead, middle boxes may help to provide privacy at the borders of domains. Firewall-like devices are placed between domains to block messages that breach security policies [241]. However, the rules to apply on such devices are challenging to deploy and are only tailored for specific protocols.

We consider scenarios and applications possibly relying on large scale multi-domain IoT topologies. As such, we rely on a broad enough definition for describing such domains. They may either represent a limited collection of devices having the same owner, or having a specific application usage in common. In any cases, the key aspect is that the raw data may be exchanged within a domain, but privacy constraints hold when data exit each domain.

In the context we consider, applications generate IoT *streams*, *e.g.* chronological sequences of measurements, sent periodically. The NDN paradigm fits well with such applications because it can efficiently deal with IoT queries treated as interests. We adopt the following terminology to put the data at the core of our forwarding model:

a **chunk of data** is defined as a piece of data (*e.g.* sensor's measurement);

a **data-stream** is a temporal sequence of chunks;

a **dataset** represents the data that a domain accepts to export, *i.e.* the values and its semantic characteristics like the nature and cardinality of the dataset (*e.g.* temperature measurements from 1,000 sensors).

NDN Support of IoT streams NDN matches the design and needs of most IoT applications. The NDN protocol [365] has been used to interconnect IoT domains due to its various advantages and to the matching design of data collection [305, 24, 25]. This protocol is based on the concept of acquiring data, instead of connecting devices. By forwarding interests and datasets, routers directly manipulate chunks of data, and not anymore opaque packets. Hierarchical names replace numerical addresses, and this naming hierarchy enables route aggregation thanks to prefix based routing. Each NDN router has a cache (a.k.a. *content store*) to maximize data re-usability, in particular for popular interests. The cache policy behaves in the following way: i) a router may insert in the content store any dataset that is locally generated or forwarded; ii) routers forward an interest if the answer is not in the content store, else a reply is directly sent on the path to the inquirer.

However, some NDN features need to be adapted specifically for IoT needs [282]. In particular, to support IoT data-streams: sensors generate a sequence of measurements, that are exploited by consumers (*e.g.* HVAC uses the last temperatures measured in a room). Besides, making the reverse path entries persistent allows the consumers to implement subscriptions [64]. Producers can then keep on pushing their measurements to their *subscribers*.

The general challenge we aim to solve can be stated as follows: *How to* ***efficiently*** *collect and aggregate/combine numerous* ***data-streams*** *from multiple* ***independent domains*** *while preserving the* ***privacy constraints*** *of each data-producer?*

This question can be split into four specific objectives that are the underlying guidelines of our solution:

1. Support heterogeneous applications with multiple independent producers;
2. Maintain producer-specific offers with privacy constraints;
3. Enable large scale data-stream exchanges;
4. Answer IoT queries efficiently.

References [NMT20a] and [NMT20b] detail our architecture components, in particular the role of border routers and their transformation policies. In our architecture, the role of a such NDN routers is to apply statistical transformations on the exported datasets. They are in charge of constructing novel datasets, from the locally produced streams and imported ones. These datasets should comply to specific privacy requirements that are configured with both local and peering privacy policies. These policies typically take into account the cardinality on which the aggregation function should apply for a given type of measurements. A limit of our contribution is that we assume an underlying tree structure to easily aggregate the data without introducing duplicates. In the next section, we propose a more advanced subscription model where the routing structure is not given, but needs to be advertised according to anonymity requirements of data producers.

IV.2.b Providing a Scalable Routing Framework for Privacy

Aggregation relies on a forwarding tree topology: a device aggregates the data from different children before forwarding it. Networks use tree structures to collect and aggregate data from producers [219]. By strategically aggregating data as it flows through the tree, devices save bandwidth. The tree also enforces that the same data is only collected once in anonymous aggregation.

Existing hierarchies such as cloud, fog and edge can be used to enforce a collection tree [294]. The cloud collects data from the edge and fog which collects data from end devices, aggregated their data for local queries and pushes the resulting aggregation to the cloud for global searches. IoT devices can also create ad hoc trees with protocols such as LEACH and it successors [312]. Devices organize themselves in trees by electing neighbors as cluster heads which receive and aggregate data in order to retransmit it to sink devices. Improved-Leach [46] for instance, periodically elects cluster heads by taking into account the node's remaining battery life and distance to sink. These trees will extend battery life as most devices transmit over short distances while the cluster head is responsible for the long range ones and devices alternate being cluster heads [10].

In this second work, we do not enforce trees since they limit the aggregation opportunities of our privacy model. Our new scheme is looking at all feasible *offers* based on aggregating enough data in

order to reach the minimum amount for it to be safe to disseminate. As soon as the privacy requirement is reached for a given set of producers, their data is combined into an offer that it is advertised and can be re-combined with datasets requiring more privacy. Brokers and producers may use offers that cycle in the network: the valid offers travel through them until enough data is aggregated to unlock more sensitive data. Each offer is indeed independent and translates to a given destination in the usual IP connectivity paradigm.

Our objective is to create valid aggregations in a distributed environment system composed of multiple domains. We rely on multi-sets on which border routers can apply commutative and associative aggregation functions (e.g., min, max, average, rounded distributions). We aim to design a NDN publishing/subscription scheme that satisfies the following three properties to aggregate the data:

1. **Only data of interest must be involved:** Subscribers describe in their query their data of interest. Our subscription scheme must identify the relevant set of matching producers, *i.e.* the metadata of these producers match the query.

2. **Enforce non-intersecting anonymous datasets:** the aggregation is applied recursively to form a stream. Obviously, the same producer must not appear several times in this aggregation: we need an aggregation tree at the grain of the offer. Else, considering the same sample value multiple times leads to bias.

3. **Aggregation requirements must be preserved:** the aggregation must respect the minimum privacy requirement of each producer. This way, we enforce k-anonymous datasets [323]. We assume that a domain trusts its direct peering domains, *i.e.* they will respect the privacy requirements defined in the contracts. Further works may lift this assumption at a more expensive security cost.

Our method maintains correct aggregation through the use of *producer IDs* that identify the data used in aggregations. These IDs are locally created by each producer via *e.g.* the use of the hash of the network address and the metadata. Thus, other domains, apart from peers, cannot associate producers with their data but are still able to identify overlapping datasets. Through their use we provide both properties 1 and 2. If the hash function is sufficiently well chosen, we can neglect the probability of collisions. If unfortunately a collision occurs, it means that the two corresponding producers cannot be merged in the same stream, which does not turn to be a stringent drawback.

A producer associates to the data it generates its producer ID, and descriptive attributes called metadata. In particular, the subscriber exploits the metadata to identify the data of interest, forming the *matching producers*. The descriptive metadata of producers forms a descriptive space, where each dimension of this space consists of all possible values of the attributes of producers. Similarly, the metadata criteria of queries identify areas of this descriptive space. Thus, matching metadata to criteria is simply checking whether a given producer is inside the area of the criteria. However, highly descriptive metadata may turn possible the association of producers and IDs. Thus, we assume that some level of anonymization is applied, such that real entities cannot be associated with IDs.

High privacy requirements may lead to incomplete offers while low privacy requirements favor the emergence of *bootstrappers*, i.e., producers whose offers allow from the creation of novel ones. **Creating novel offers from incomplete offers consists in finding maximal cliques in the conflict graph: all the offers in a clique are pairwise disjoint.** Thus, creating valid offers is a NP-Hard problem as an exponential number combinations of offers is possible in such a graph [189].

In [NMT21], we then develop heuristics for both the offer and subscription phases centered around similar data. We explore and discuss distinct overall strategies to limit the exploration space: greedy, prioritization of similar data or more secured ones. Our performance evaluations highlights the efficiency of our similar data strategy to identify correctly the offers to publish and to combine. We show that our method quickly reaches acceptable levels of aggregation within the possibilities allowed by the network. Our method can be executed by resource constrained IoT gateways in order to disseminate and collect data in a multi-domain infrastructure.

We now expect to provide mechanisms to verify that aggregations cannot be deanonymized throughout the network. Indeed, a broker should be able to detect colliding offers. This filtering may be applied both during publication (not disseminating partially overlapping offers), and during subscription (active streams must concern non overlapping producers). We also expect to explore how more sophisticated

aggregation functions may scale. In particular, we aim to investigate how unaggregated data can be cached *en-route* to be re-used in streams that use different aggregation functions.

IV.3 On the Use of Micro-services and Metagraphs to Prevent Data Exposures

This last work is about securing workflows in the context of multi-party architectures. With Cristel Pelsser and Antoine Gallais as Ph. D. directors, we have supervised the works of Loic Miller on security for secured and private workflows. In [MMGP21b] we proposed a micro-services architecture while in [MMGP21c], we provide an access control verification framework relying on Metagraphs. Finally, in [MMGP21a], we have extended these two works to propose an unified solution. Loic, who has defended his thesis with success in 2022 [Miller, 2022], is now working on several proofs to exhibit the complexity of revealing redundancy and conflicts in Metagraphs. Overall, the questions that we tackle in this thesis were the following:

Research Question

How to Ensure Data Security at Rest and in Transport? How to Check and Analyze the Deployed Policy?

The two next sections present a brief summary of what we explore in this area of research, namely micro-services architectures and the use of Metagraphs to model, verify and analyze access policy.

IV.3.a A Micro-Service Architecture to Secure Workflows Architecture

Data leaks and breaches are increasingly happening. With more and more businesses using public clouds to process data [254], and this data being frequently moved around, exposures are more likely to happen than ever. Those exposures are perceived as huge losses of money for businesses like the movie industry [63], and as a loss of user privacy for applications dealing with user data [81].

The recent rise of microservices as a paradigm, and their increased use in building large, cloud-based enterprise applications [71] has increased the attack surface, meaning protecting a given network border is no longer sufficient. To prevent data leaks, one needs to consider attacks coming from inside the system (e.g. leaks stemming from the way data is processed or caused by a malicious employee). The zero-trust security model [140], where all traffic flows are required to be authenticated and authorized via fine-grained policies, provides such protection.

In accordance with such principles, we aim to achieve a secure system enabling the exchange of data between non-trusted agents in the context of workflows. The data should be secured at rest and in transport and cannot be exposed by any agent in both cases. To meet our requirements for zero-trust and prevent data leaks during the execution of workflows, we rely on a secured microservice architecture.

The microservice architecture allows one to design a system preventing data exposures that is simple, modular and scalable, thanks to its loosely coupled services. This is important when considering security mechanisms quickly become challenging to configure, manage, scale and monitor when combined, with a large number of actors using different IT environments.

We have opt for adversarial model that considers three types of attackers.

- *External attacker*: External to the workflow and the location of the deployed infrastructure. Such attackers try to gain access to the data or the business intelligence from the outside.
- *Co-located attacker*: External to the workflow, but co-located at the deployment (e.g. an attacker located in one of the third-party clouds). This co-located position opens more exploit options.
- *Malicious agent*: Internal to the workflow, this attacker tries to leak the data outside.

We propose an infrastructure to protect a workflow execution from these threats. As we need a way to prevent data leaks, we aim to control the communications an agent can engage in. To achieve this, we should control the environments the agents will be using, to make sure that all the actions of an agent follow a policy enforced by the owner. In this infrastructure, agents of our workflow are mapped to containers, which are then used in conjunction with an orchestrator, a service mesh and policy engines to enforce the policy of the owner.

Each agent is contained into a pod, including multiple containers: the service (i.e., the environment the agent will be using), a proxy and a policy sidecar. The access policies of a service are pushed in its associated policy sidecar. The proxy sidecar intercepts all traffic coming from and going to the associated service and the policy sidecar checks whether it shall be authorized or not.

The service mesh controller and the policy store are under the control of the owner. It specifies the policy to be enforced, preventing in particular the data from leaking outside. The data processed by the pods are stored encrypted on mounted Persistent Volumes (PVs), providing us with **data security at rest**. Pods also communicate according to the specified workflow and policy via `mTLS`, providing us with **data security in transport**. Communications inside a pod are not encrypted, but the isolation layers protect the data against eavesdroppers.

Our model of workflow is defined by the owner and enforced using policy sidecars, which controls the agents participating in the workflow. In their related work, Hussain et al. [170] propose and implement a security framework for the creation of a secure API service mesh using Istio and Kubernetes. They then use a machine learning based model to automatically associate new APIs to already existing categories of service mesh. Contrary to our work, they use a central enterprise authorization server, in opposition to our policy sidecars. Zaheer et al. [362] propose eZTrust, a policy-driven perimeterization access control system for containerized microservices environments. They leverage eBPF to apply per-packet tagging depending on the security context, and then use those tags to enforce policy, in opposition to our enforcement of policy which relies on policy sidecars local to the services. Weever et al. [88] investigate operational control requirements for zero-trust network security, and then implement zero-trust security in a microservice environment to protect and regulate traffic between microservices. They focus on implementing deep visibility in the service mesh, and do not propose a security or a performance evaluation.

We have indeed both realized a proof of concept of our architecture to discuss its benefits and limitations [MMGP21b], and monitored key parts of the workflow to show how the data is secured. Our experiments have shown that our approach scales well with increasing workflow complexity.

IV.3.b Modeling Access Control applied on Workflows with Metagraphs

Some of the largest cloud consumers use the cloud to deploy their workflows and enforce their processes using access control policies. Authorization is a key aspect of security in such environment, regulating the interactions taking place in a given system. For example, Netflix may often interact with their partners for some tasks, e.g. content ingestion [53, 54].

Research on policy-based management of authorization mainly focuses on three areas: policy analysis, policy refinement and policy verification. On the one hand, **policy analysis** deals with the fulfillment of specific properties by a set of policies [334], e.g. detecting when two or more policies are conflicting. On the other hand, **policy refinement** handles the translation from high-level policies into low-level configurations [244].

In our work [MMGP21c], we deal with **policy verification**, i.e., we check whether the deployment of policies actually meets their high-level specification. Policy verification plays an important role since assisting tools are not free of errors, and deployment specificities can lead the policy to become erroneous. An erroneous policy can lead attackers to view files they were not authorized to see [348], access paid content free of charge [77] and even changing access rights [249] or deleting content [13]. There exists only few works on policy verification [169, 49], when compared to the large body of work dealing with policy analysis, and none of them uses *metagraphs*.

We rely on this structure since, by design, it provides means to *locate conflicts and avoid redundancy* [279]. Metagraphs provide more fine-grained verification process than with other structures like usual graphs (even regarding hypergraphs that are similar set-to-set mapping structures). To the best of our knowledge, metagraphs belong to the rare appropriate structures able to naturally model access

control policies and projections [45] can be used to help with the visualization of very large policies.

Metagraphs are suitable modeling objects; while they enable policy analysis, we are interested in their use for a practical verification of the deployed access control policy regarding its specification. Our proposal compares the initial metagraph specification to its deployed counterpart and reveals inconsistencies. By modeling the high-level policy specification as well as the translated policy implementation as two metagraphs, we can compare both in order to track (distributed) deployment errors. When specification and implementation metagraphs match, the policy implementation has been correctly translated from the policy specification. If they do not match, the metagraphs are not equals: errors occurred during the refinement and/or deployment. We evaluate the performance of our simple approach to show its scalability.

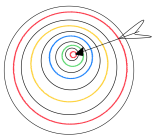
Loic Miller is now looking at the computational complexity of the problem of checking for redundancy in Metagraphs. With the two notions of *metapath dominances*, that slightly differ from what can be expressed with usual hypergraphs, in theory one can finely pinpoint edges that are not in any dominant metapath (the redundant ones). However the problem of looking if an edge is not in any such paths leads to look to forced edge in hyperpath (natively enabling the notion of edge-dominance but not the one of input-dominance). If an edge cannot be forced in any dominant hyperpath it is redundant, and if it is redundant it cannot be forced in any hyperpath (thanks to the native minimality of hypergraph regarding edges). Loic has already shown the difficulty of the problem as looking for a forced edge in an elementary path is already NP-hard (reduction to the 2 Vertex Disjoint Path Problem), and he now relies on the hypergraph literature [34, 347] to show that even in the restricted case of F-acyclic hypergraphs, the redundancy problem remains NP-hard (on the contrary to B-acyclic ones).

This chapter has briefly reported my transversal activities in other networking contexts than wired ISP ones, that is within IoT and multi-tenant domains for energy and security purposes. It concludes the part of the report dedicated to my past and ongoing activities. Table IV.1 summarizes my publications activities in these transversal fields:

Title of the publication	Name of the venue	Year	Reference
Path Diversity in Energy-efficient Wireless Sensor Networks	International Symposium on Personal, Indoor and Mobile Radio Communications (**PIMRC**)	2009	[MG09]
Energy-efficient Data Collection in WSN A Sink-oriented Dynamic Backbone	International Conference on Communications (**ICC**)	2012	[CGM12]
Enabling Privacy by Aggregation with Muti-domain IoT streams	Conference on Local Computer Networks (**LCN**)	2020	[NMT20a]
Transformation Based Routing Overlay for Privacy and Reusability in Multi-Domain IoT	International Symposium on Network Computing and Applications (**NCA**)	2020	[NMT20b]
Scalability of LPWAN for Smart City Applications	International Wireless Communications and Mobile Computing (**IWCMC**)	2021	[NMGT21]
Data Aggregation for Privacy Protection of Data Streams Between Autonomous IoT Networks	Symposium on Computers and Communications (**ISCC**)	2021	[NMT21]
Towards Secure and Leak-Free Workflows Using Microservice Isolation	International Conference on High Performance Switching and Routing (**HPSR**)	2021	[MMGP21b]
Verification of Cloud Security Policies	International Conference on High Performance Switching and Routing (**HPSR**)	2021	[MMGP21c]
Securing Workflows Using Microservices and Metagraphs	Multidisciplinary Digital Publishing Institute (**MPDI**)	2022	[MMGP21a]

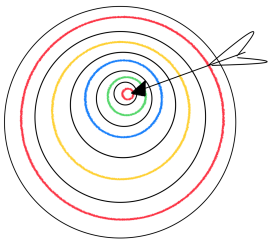
Table IV.1: Summary of my publications not related to routing and measurements in wired ISP networks.

In the next chapter, I will develop my plans for the future years, not only extending some of my activities in the radio or private networks, but also developing new projects based on recently emerging hardware opportunities, e.g., multi-radio technologies and flexible data-planes.



Chapter V

Ongoing Research Projects, Future Works and Long Term Objectives



This last chapter is dedicated to my future works and perspectives including long term research projects and directions. I will first develop my short term objectives with two routing & forwarding extensions to reach an ideal convergence. Indeed, the first thesis I aim to supervise is about providing a **comprehensive solution to ensure a fast but safe convergence in SR domains**. Only relying on an assumption as simple as considering a symmetrically weighted network seems enough to overcome the main challenges on the road to provide very efficient consistent re-routing optimal schemes. Moreover, I am interested in novel programmable data-plane paradigms allowing to implement main building blocks of such techniques at line-rate for both the local and transit traffic and prevent complex failures in massive scale networks.
Then, I will expose some of my interest in the field of **multi-radio IoT networks**. In particular, the opportunity to rely on two radios for enabling longer battery life-time.
As a third objective, I will present the future works I envision to tackle in the field of IP measurements. Three main directions are exposed: **mapping weighted ISP topologies with comprehensive probing methods, analyzing SR domains and detecting BGP anomalies**.
Finally, I will develop my long term perspectives in the area of distributed systems in general. While I am primarily interested in **self stabilizing algorithms** in particular for enabling robust routing schemes, other fields like grid exploration with weak robots also attract my attention.

Contents

V.1 Fast and Safe IP Convergence with Symmetric IGP Weights

This first section aims to tackle the two routing convergence problems addressed in chapter II as a whole. Moreover, the proposed technique respectively provides an optimal convergence as in II.3 and relies on SR as in section II.4. The enhanced re-routing solution I envision is designed for SR domains and can be used for the intra-domain traffic as well as for transit BGP traffic with a hierarchical forwarding model like the one in use with OPTIC (II.3) and developed more in details in Sec. III.3. My goal is to ensure that re-routing paths are not only transient backup but post-convergence paths deployed quickly in a safe distributed manner (exempt of anomalies like transient loops); generally speaking, the first question I aim to address is the following:

Research Question

How to Ease and Speed-up the Intra-domain Routing Convergence in SR domains?

That is, assuming a reasonable assumption in IP networks, i.e. the use of symmetric weights (with a weighted graph instead of a digraph one with possibly asymmetric valuations), enables the design of an efficient solution for deploying a **complete protection scheme being free of any forwarding loops, even transient ones lasting less than half a second in general** [MDP+18]. I argue that such a constraint (the symmetrical condition) does not hamper the deployment of TE and is often granted in current IP networks; since the link valuation in use for the IGP metric basically models bidirectional links which are symmetric regarding their physical characteristics (e.g., delays and capacities), only an excessive volume of traffic (possibly asymmetrical) in a given direction may require the use of asymmetric valuations because of saturated directed links to offload (in the case of adaptative weights optimizing the traffic distribution [119, 264]). Such an optimization objective can be fulfilled using asymmetrical LB (e.g., with more complex rules and traffic distribution than ECMP) and/or specific TE paths carrying the extra traffic away from the congested link. Indeed, most ISP decouple their best-effort deployment, that can be symmetrically designed, from their TE objectives as it only concerns few possibly large premium flows and specific congestion situations (that may be handled similarly as re-routed traffic).

Symmetric Weights Ease the Synchronisation between Adjacent routers to Detect Forwarding Loops Assuming symmetric weights comes with many advantages:

- simplifying extra SPC as best reverse distances are the same as forward ones (as well as underlying path considering a perfectly symmetric model);
- provably providing complete protection with only one SR adjacency segment at worst;
- last but not least, enabling to easily deploy a fast and correct convergence, i.e. with no transient loops, or simply *safe*.

While the two first points have been already covered with the models and the extensions proposed in section II.1.b, the last advantage has only been discussed, but a last piece is missing for a practical deployment not relying on any timers. I aim to address the same safety problem as AGBA does in the general case, i.e. preventing transient forwarding loops (Sec. II.2), but requiring much less computation and deployment complexity by opportunistically detecting forwarding loops. Indeed, **one can prove that transient loops in symmetric networks (with ECMP or an unique path) can be prevented by looking only at (elementary) circuits made of two edges**, involving only subsequent neighbors in forwarding paths. Instead of only dealing with link or node maintenance, the new proposed method is general and supports partial asynchronous deployment, that is the proposed method deals with any single failure and does not require that all devices update their forwarding states at line-rate when they receive the (first) SR encapsulated data packet(s). Such TI-LFA detoured packets contain both explicit and implicit information that can be extracted, respectively, but indirectly, the next-hop to use, and the occurrence of a network change.

As already stated at the end of Sec. II.1.c.3, the symmetry property allows to ease and speed up the convergence time with either an explicit or implicit synchronization scheme instead of relying on timers to prevent loops[1]. With the solutions developed in the following, adjacent routers can adapt their forwarding states at each packet or event according to the last routing update by just looking at the incoming/outgoing traffic for each destination. Indeed, for example considering an explicit synchronisation model, once a node has computed its new forwarding states for a given destination d, it has the ability to force its neighbor (its new outgoing interface for d) to do the same: when receiving such a request, the neighbor just has to check whether its current outgoing interface is the same as the neighbor sending the request (being an incoming interface for d). The implicit model can achieve the same but performing equivalent operations only when necessary: with SR and a fully data-plane embedded updates, we have seen that it is possible to prevent any loops. However, this operationally naive method requires that SR implicit updates are not lost or their application delayed (e.g., because operated at the control-plane instead of the data-plane). This assumption thus requires that all routers in the network have this processing ability (performing forwarding updates at line-rate); in the following, I rather opt for an incremental deployment where this feature is not (strictly) required and only used to improve the performance when possible.

The goal of the following section is to show that actually detecting loops instead of preventing them blindly is sufficient. It allows for a safe and fast opportunistic convergence not requiring timers or explicit synchronization messages. Moreover, packets are encapsulated with SR only if a loop involving them is detected and the proposed approach remains valid even if nodes do not support line-rate data-plane updates. This per packet model enables a fine-grained decision (only fixing actual issues) and I will now show that symmetric weights ease such a detection in the data-plane (with a simple incoming/outgoing traffic condition).

V.1.a Symmetric Weights, Elementary Circuits and Number of Alternances

Formally, we will first prove that **if best paths are unique and symmetric** (not only regarding their distances but also with respect to nodes and links composing them), **they cannot lead to any elementary circuit longer than two**. For any destination d, the digraph resulting from the union of the pre- and post-trees towards d, $MP(d)$, does not admit any elementary circuit longer than two in such a case. Second, still considering symmetric distances but this time with all multiple best paths used in both directions thanks to ECMP (and so relying on Reverse shortest paths DAGs), we will see that the result is less strong but still valuable for our purpose in this more general case. Relying on the notations introduced for protecting a link $l = (r, t)$ rather than on the ones extracting the constraints resulting from potential transient micro-loops, let me redefine $MP(d)$ as follows:

$$MP(d) = \bigcup_{\forall n \in V} \mathcal{D}_1(n, d) \mid l \in \mathcal{D}_1(n, d) \cup \bigcup_{\forall n \in V} \mathcal{D}_2(n, d, l)$$

$MP(d)$ is defined specifically for a given link failure and merging both these pre-impacted and post-convergence DAG may result in a digraph possibly containing elementary circuits longer than two. Nevertheless, **with ECMP, and considering that symmetric paths are all used in both directions** (i.e. the *complete* ECMP model), I will demonstrate that we are still able to rely on the following property: **for each edge** $e \in MP(d)$ **involved in a circuit we have the guarantee that** e **exists in both directions** (one direction for the pre- and the other for post-convergence forming an elementary cycle of size 2).

V.1.a.1 Two Symmetric Forwarding Models Having Distinct Implications on $MP(d)$

About Pruning $MP(d)$ Note that $MP(d)$, resulting from the merging of two reverse shortest paths DAGs initially rooted on d (or simply trees if ECMP does not offer any diversity), is now refined to be specific to the link modified in the network, $l = (r, t) \in E$: for the sake of simplicity and to avoid cumbersome notations, let us consider an implicit failed link (the cases of a node or SRLG failures are

[1] Calibrating such timers is challenging as they directly impact the trade-off between overhead (i.e. encapsulating and detouring packets when it is not anymore necessary) and reactivity (i.e. the ability to quickly switch to the new optimal state without overhead but at the risk of introducing transient loops), while our proposal allows for correct and almost instantaneous updates. Note that it is somehow similar to the Rapid Spanning Tree extension regarding its standard counterpart which rely on *paranoiac timers*.

more complex and require subtle generalization not presented here[2]). Indeed, for the analysis to come on transient loops, one can remove from $MP(d)$ all elements which cannot be involved in circuits considering a given l as the failed link. By construction, we know that circuits in $MP(d)$ cannot involve l itself and, most important, any of the nodes not using l before (and so after) the failure: they are loop-free paths regarding d and l. Considering the induced subgraph in $MP(d)$ resulting only from the subset of nodes in V using l in their pre-DAG towards d is then enough. Indeed, removing all nodes s (and their edges) not using the link l in their paths $\mathcal{D}_1(s,d)$ will also remove nodes downstream to l in their remaining paths. In the same vein, we will see latter that it is also possible to look only at links which are not both pre- and post-edge in $MP(d)$ (in this case $MP(\cdot)$ is not anymore a multi-digraph but only a directed simple one). Indeed such links cannot be involved in circuits as we will conclude after the following sequence of proofs. For the sake of the demonstrations and remain as general as possible, in this section, I will simply consider all nodes and paths in $MP(d)$ using l in their pre-DAG, in particular upstream to l. The last observation indeed results from the demonstrations to come (it is given by Cor. 5) while the former is introduced to better understand and simplify the part of the object we are interested in, only this restricted subpart of the $MP(d)$ graph may indeed contain circuits (that represent potential loops I aim to prevent). Moreover, on the contrary to AGBA or its variants, there is no practical interest to efficiently prune $MP(d)$: we here intend to exhibit a given property rather than extract numerical constraints. We will indeed show that this simple property allows us to design a very efficient algorithm not analyzing loops in details but still able to prevent all of them because of their underlying trivial patterns.

About Perfect Symmetry With ECMP, if paths are symmetric and all used in both directions (as with standard IP networks), we will then prove that despite the digraph $MP(d)$ may contain elementary circuits of size 3 or more, each involved edge in such circuits also belong to an elementary circuit of size 2 (edges exists in both direction). That is for all edges $e = (x_i, x_{i+1})$ in such a circuit $c = x_0, x_1, ..., x_i, x_{i+1}, ..., x_k$ we have (x_{i+1}, x_i) exists in the subgraph of $MP(d)$ induced by nodes $x_0, ..., x_i, x_{i+1}, ..., x_k$. Note that this is a specific kind of *chordal digraph*, fully doubly connected, where a *chord* exists for all pairs of subsequent nodes: the resulting induced subgraph (with nodes belonging to the circuit) is actually a cycle (i.e. a bidirectional ring, so not anymore directed by definition) that may or not have a chord (in the sense of non directed graph this time).
With asymmetric paths (or more generally distances), there exists cases where elementary circuits of size strictly greater than two do occur, without being composed by circuit of size two. That is there can exist elementary circuit longer than two without chord or edges existing in both directions. Among these three forwarding models (asymmetric and perfectly symmetric with or without ECMP), we will then focus on the latters which assume perfect symmetry (either unique or complete), looking specifically at ECMP as it deployed in current IP networks in particular (i.e. with complete and full symmetry existence of paths).

Specific Notations Finally, let us consider and use the notation $((x_1, x_2, x_3, \cdots))$ for denoting *an alternance path* made of subpaths from nodes x_i to x_{i+1} (subpaths $((x_1, x_2))$, x_2 to x_3, etc) such that underlying edges forming the subpath between the two subsequent x_i are of the same type (either post- or pre- edges); this way, we avoid detailing consecutive hops $(n_1, n_2, n_3, \cdots, n_i, n_{i+1} \cdots)$ of the same type between two x_i and also restrict our analysis on paths having at least one *strict* change (to possibly form a circuit). The case of edges or subpaths being both pre- and post-convergence ones will not be explicitly treated as they can be considered as being one xor the other without modifying the following statements and their proofs (that is we can ignore them in practice). To illustrate more precisely this case, and without loss of generality, let us consider that a given alternance path $p = ((x_1, x_2, x_3, \cdots))$ includes at least one convergence change such that x_1 is the first node of a pre-convergence subpath (including only pre-convergence edges), x_2 the first node of a pre- and post-convergence subpath (including only pre- and post-convergence edges), while x_3 is the first node of a post-convergence subpath (including only pre-convergence edges). Path like p are then rewritten as $((x_1, x_3, \cdots))$ xor $((x_1, x_2, \cdots))$ because we can ignore either x_2 or x_3 to count the number of strict alternances between the pre- and post-convergence safe subpaths. We are interested in paths including such strict changes (or *alternances* whose number is

[2]In section **??**, I start to investigate the case of node protection while SRLG scenarios are let for further works. Results presented here are only valid for a link removal.

equal to the number of changes plus one) and at their number as a loop cannot occur if no such change(s) trigger them: a (sub)path containing exclusively pre- or post-convergence edges (or both simultaneously) is free of any circuit by design of the routing protocol. That is at least one strict change is necessary for a circuit to occur, and so two alternances as we define them. For convenience we will consider the x_i generating the first strict changes (at the end of a path, e.g. reaching d, if there is no strict change on the last edge or subpath, one can consider the first non strict node of this subpath as the one changing the path kind – with the new kind introduced).

Alternance in a Circuit Thanks to this convenient notation summarizing subpath alternances, let us now introduce some basic definitions and results regarding the nature of circuits in $MP(d)$ depending on the forwarding model in particular.

Definition 5. *We say that a circuit (or more generally a path) in $MP(d)$ – resulting from the merging of pre- and post-convergence paths towards d, has k alternances if it alternates exactly k times between pre- and post-convergence sub-paths. It can be written as $((x_0, x_2, \cdots, x_k))$ in its alternance path notation.*

Note that such a path or circuit thus possesses at least k edges, each portion of subpath of the same convergence type including at least one edge. Let us start with the simplest general properties applying for both models.

Lemma 2. *For a circuit $c \in MP(d)$ to exist, its number of alternances is even. That is c has an even number of alternance between pre- and post-convergence sub-paths.*

Proof. Note that the case with only 1 alternance (0 change) cannot imply circuits as we assume the pre- xor post-convergence paths not containing any persistent loops (i.e. the routing is safe, except during convergence periods combining pre- and post- sub-paths). More generally, considering an even number $k-1$ of changes in a circuit c would imply that the last (non repeating) subpath $((x_{k-1}, x_k))$ of a circuit $c = ((x_0, ..., x_{k-1}, x_k = x_0))$ having k alternances belong to the same convergence type (pre- or post-) as $((x_0, x_1))$ (because changing a binary state an even number of times results in the initial state, that is no change). The $k - 1^{th}$ supposed change, $(x_{k-1}, x_k = x_0)$, should then be merged with $((x_0, x_1))$ regarding our binary subpath notations while it is in contradiction with the initial statement (c has now an odd number of changes since the last supposed change, the $k - 1^{th}$, has been removed). Overall, this result is trivial since the number of changes is odd by construction of the binary alternances in a circuit. □

Lemma 3. *For a circuit $c \in MP(d)$ having 2 alternances (but possibly more edges), each of its edges exists in both directions in $MP(d)$ considering perfect symmetry (with or without ECMP).*

Proof. Considering circuits having two alternances, we can denote x the first node of one type and y the first one of the other type on the two sub-paths composing the circuit formed with pre- and post-convergence paths towards d, denoted $c = ((x, y, x))$. Without loss of generality, let us say that $((x, y))$ is the pre-convergence subpath and $((y, x))$ the post-convergence subpath to illustrate the reasoning (Fig. V.1). Although best paths are originally computed towards d, note that isotonicity ensures that subpaths between x and y are also best paths between their extremities. Since at least two best subpaths exist in $MP(d)$ to form a circuit between x and y, for any of these pairs, there also exists two underlying sequences of edges being exactly the same in both directions otherwise it contradicts the *perfect* symmetry of the best routes: both directions between x and y share the same set of shortest subpaths. Pre- and post-subpaths among nodes in $c \in MP(d)$, as well as their one-to-one distances, are indeed symmetric because not affected by the removal of l in both directions. Nodes x and y are by construction upstream routers regarding l for d, the subpaths $((x, y))$ and $((y, x))$ are perfectly symmetric: pre-convergence subpaths between x and y do not include l, so do their respective post-convergence paths in the opposite direction. Recall that $MP(d)$ does not account for nodes and paths downstream to the failed link with respect to d, as well as non affected sources in general (they cannot be involved in any circuit for d), the distances and subpaths between any remaining pairs of nodes in $MP(d)$ are thus unaffected by the removal of l in general as they do not contain it (while they do to reach d). □

Thus when pre- and post-convergence subpaths both exists in $MP(d)$ to form a circuit $c \in MP(d)$, they are best subpaths that necessarily rely on the same set of nodes and so edges in the opposite directions

(*perfect* symmetry). Note that this is true both with ECMP (thanks to symmetrical completeness - all equal cost paths are used in both directions), and when the symmetrical unicity applies with the former model (the same unique path is used in both directions). Another observation directly implied by this lemma, and that we will generalize later when tackling circuits having more than 2 alternances, is that no edge being both pre- and post-edge can be present in c (as otherwise it implies that edge of the same kind may exist in both directions).

Without ECMP, there is Only Elementary Circuits of Size 2 (or longer simple paths but having only 2 alternances) Let us now move to a result specific to the unique forwarding path model. Our claims are here the strongest of the section as this model prevents the combination of more than two subpaths of different kinds. It then narrows the difficulty to understand the kind of loops it may trigger.

Theorem 15. *Considering the symmetric routing model with unique forwarding paths for each destination $d \in V$ (no ECMP), the number of alternances of any circuit in $MP(d)$ is equal (and so limited) to two. These 2-alternance circuits are either elementary ones containing only two edges (both directions of the same link) or simple circuits (the same node(s) are repeated) if they contain more than 2 edges.*

Proof. Let us first consider that the number of alternances exceeds two in the circuit c. It implies that there exists at least two nodes $x \neq y \in c$ having exactly one incoming edge of one type and one outgoing one of the other type. Without ECMP, regarding the couple $((x, y))$ in each direction, there exists only one subpath made of the same nodes in the reverse order. Connecting x and y in both directions thus requires at least one of these common nodes to have incoming and outgoing edges of opposite nature than the ones of x and y (to ensure the *switch* between the incoming and outgoing edges of the two types). To do so with a single intermediary node z, it is required for z to be an ECMP node (and so contradicts our initial unicity assumption): it should have two outgoing edges of the same type to switch the same incoming type in both directions of the (unique) path between x and y. Let us now consider the case of multiple switching nodes, and focus on two of them, z and w, ensuring respectively the last switch on the forward path from x to y, and the last switch on the return one from y to x; one can observe that the unicity of paths imposes an order among z and w (being the opposite between the forward and return paths). Let us handle the case where z comes before w on the forward path from x to y. This order implies that z is an ECMP node (as well as w thanks to symmetry): its two outgoing edges are of same type, as z must preserve the switch applied at w in the return path (because w is the last switching node in this subpath, z cannot switch it again and then there is no difference with z applying the switch on its own as before). The case where z comes after w cannot occur if z is not an ECMP node, switching types in both directions. Indeed, otherwise (if z preserves the edge type on the return subpath while not in the forward one), it would imply that z has both incoming and outgoing edges towards and from the same node (nodes in both subpaths are the same) of the same type. Eventually, the situation remains the same as considering one single ECMP node applying the switches for both directions.
Second, considering such circuits having exactly two alternances and the previous proof for lemma 3, we can again denote x the first node of one type and y the first one of the other type on the two sub-paths composing $c = ((x, y, x))$. We know that the two directed subpaths between x and y include the same nodes in-between: if they include more than a single edge between the two, their merging does not result anymore in an elementary but a simple circuit as only the same node(s) in-between the forward path towards y, and their reverse edges, can be used to go back to x. □

With symmetric unique paths, circuits are made of two alternances and possess only elementary circuits of length two (longer circuits are simple ones that cannot translate to loops). Only ECMP enables both more alternances and longer elementary circuits as a node with multiple outgoings edges of the same type can switch from one type to another without coming back to the same upstream node.

V.1.a.2 ECMP Makes the Problem More Challenging to Analyze but Not to be Solved

With ECMP, Elementary Circuits May Be Longer and More Alternances Can Occur Fig. V.1 illustrates the latter property on a small gadget: while there is only two alternances in the two circuits involving x, y and z, we can observe that they are both elementary circuits made of three edges. The existence of multiple paths among them allows this relaxation. On the other hand, while simple circuits in $MP(d)$

cannot contain more than 2 alternances with symmetric unique paths, they may include more than two edges (and so repeated nodes); but in practice, no such simple circuit can turn into a long forwarding loop as only subsequent elementary can occur. A node cannot use both a pre- and post-convergence edge by definition and thus, with the symmetric unique forwarding model: only one outgoing edge is used at a time – making the combination of two subsequent elementary loops impossible as well as their simultaneous existence. With ECMP, it is theoretically possible since a node may have multiple outgoing edges of the same type (e.g. in Fig. V.1): on the example, since z has two pre-convergence outgoing edges, the simple circuit x, z, y, z may occur because x and y can be up to date whereas z is not. Nevertheless, since the load balancing applied with ECMP is generally flow-based and rely on a surjective deterministic function, a packet will only loop either between x and z or y and z. Note that the two subsequent loops can exist at the same time (on the contrary to the unique path model).
In the following, we will focus on circuits having more than 2 alternances.

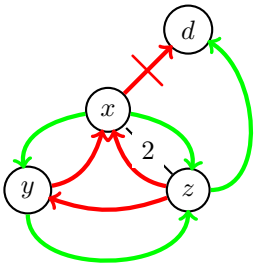


Figure V.1: Only two alternances but ECMP enables the occurrence of an elementary circuit of length 3. Here two subsequent loops can co-exist concurrently forming a simple forwarding loop.

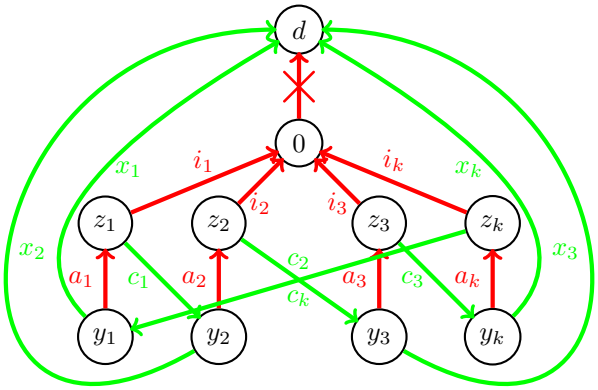


Figure V.2: An elementary circuit having 2k alternances: $((y_1, z_1, y_2, ..., y_k, z_k, y_1))$, with $k = 4$ and so 8 alternances here. Our goal is to prove that such a circuit cannot exist with the unique symmetric forwarding model while it can exist with ECMP but with conditions stated in 4: each of its edge exists in both directions in $MP(d)$.

ECMP Triggers More Complex Loops Than It Looks With ECMP, both elementary and simple circuits can contain more than 2 alternances ($2k$, $k \in \mathbb{N}$), and this is also the only case where two subsequent forwarding loops can occur in concurrence on the same intermediary node as illustrated previously. However, while a simple path of four edges can exist and form two simultaneous and adjacent forwarding loops, if each its elementary element is prevented (its sub-elementary circuits), no such loop(s) can occur. **We can then just look at elementary circuits of arbitrary length and show that their size does not matter because each of their edge exist in both direction.**

Lemma 4. *An elementary circuit $c = ((x_0, x_1, ..., x_i, x_{i+1}, ..., x_k = x_0)) \in MP(d)$ having $2k$ alternances with $k > 1$ can only exist with ECMP and if weights are symmetric we have: $((x_k = x_0, ..., x_{i+1}, x_i, ..., x_1, x_0)) \in MP(d)$.*

Proof. Let us restart from scratch the proof of Theorem 15 and naively assume that there exists an elementary circuit having $2k$ alternances without the use of ECMP (with $k > 1$). We will show that this assumption leads to a contradiction implying the presence of ECMP at each hop. More precisely, let us consider Fig. V.2 to illustrate such a circuit denoted $((y_1, z_1, y_2, \ldots, z_k, y_{k+1} = y_1))$. Underlying subpaths belonging to the circuit, or edges to simplify the illustration, are respectively denoted a_j with $1 \leq j \leq k$ and $a_{k+1} = a_1$ (indexes are circular to ease the notations) for pre-convergence subpaths, and c_j for post-convergence subpaths (with the same conditions applying on indexes). Finally, we have i_j and x_j respectively denoting pre- and post-convergence subpaths not included in the circuits (while the former use the failed link $(0, d)$, the latter do not). Note that the illustration assumes $k = 4$ for the sake of simplicity.

We consider first that the following inequalities apply for all $j \in \{1, \ldots, k\}$:

$$w(a_{j+1}) + w(i_{j+1}) < w(c_j) + w(i_j) \quad (B_j)$$

Indeed, we consider that there is no ECMP for all y_j, in particular before removing the failed link $(0, d)$. Besides, note that we extend the notation $w()$ from edges to subpaths, where it denotes the sum of weights of the underlying edges in use (instead of only consider each edge independently). With ECMP, such inequalities are not strict; one can notice it directly implies that subpaths c_j exist in both directions.

In the same vein, we have for all $j \in \{1, \ldots, k\}$:

$$w(c_j) + w(x_{j+1}) < w(a_j) + w(x_j) \quad (A_j)$$

Indeed, without ECMP for all z_j, only one path exists towards d and this is the one entirely in green in the illustration (the post-convergence one). After removing the failed link $(0, d)$, we consider that nodes have only one unique path, otherwise it implies the existence of edges a_j in both directions (or subpaths to remain general enough).

Thus, $\forall j \in \{1, \ldots, k\}$, we can conclude:

$$w(a_{j+1}) + w(i_{j+1}) + w(x_{j+1}) < w(c_j) + w(i_j) + w(x_{j+1})$$

$$w(c_j) + w(x_{j+1}) + w(i_j) < w(a_j) + w(x_j) + w(i_j)$$

The former inequalities come from (B_j) while the latter ones result from (A_j), we just add respectively the terms $w(x_{j+1})$ and $w(i_j)$. Since such inequalities are circulars modulo k, we have as many contradictions (e.g. $w(a_1) + w(i_1) + w(x_1) < w(a_1) + w(i_1) + w(x_1)$) as k (here, in the mentioned example, with $j = k$). This yield the initial statement: without ECMP, no such $2k$ alternances circuit exist.

Moreover, as discussed before for (B_j) and (A_j) statements, the only manner to avoid all these inequality contradictions is to enable non strict inequalities with the systematic use of ECMP for y_j and z_j but it implies that both subpaths c_j and a_j exist in both directions. Relying on the same arguments as in the previous proofs, it is now obvious that the circuit should exist in both directions too, forming so (a non directed) cycle.

□

With symmetric unique paths, only the case $k = 1$ exists while Fig. V.2 illustrates the case for $k = 4$ and ECMP (the counter example for the proof where all reverse edges should exist).
At the granularity of each edge (and not the alternance scale), for each circuit $c = (..., n_i, n_{i+1}, ...) \in MP(d)$, the subgraph $c' \in MP(d)$ induced by nodes $n_i \in c$ ($c' \supseteq c$ by construction), verifies $(n_{i+1}, n_i) \in c', \forall n_i \in c$. While Fig. V.2 does not exhibit such a pattern (to assume their absence and lead to a contradiction in the related proof), the proof it illustrates demonstrates that such edges should exist because of ECMP and their implication in a 2k-alternance circuit, i.e. each directed edge a_i and c_i in Fig. V.2 actually implies the existence of its respective opposite edge of the other kind. The proof indeed shows that the edges a_i actually all have an opposite post-converge edges, and respectively edges c_i implies the existence of pre-convergence edges in their opposite direction.

Theorem 16. *Considering perfect symmetry for each destination $d \in V$ (e.g. ECMP), each (elementary) circuit $c = (n_0, ..., n_i, n_{i+1}, ..., n_k = n_0) \in MP(d)$ of length $k \geq 2$ is actually a cycle. That is the edge (n_{i+1}, n_i) also exist in $MP(d)$ for any $i \in 0, ..., k-1$.*

Proof. It is just about combining the previous properties and their proofs. Indeed, from Lemma 2, we know that if a circuit exists, it has an even number of alternances. Thus, we can use Lemmas 3 and 4 (for cases $k = 1$ and $k > 1$ respectively) to conclude that such circuits exist in both directions. □

Actually, Lemma 4 is enough as it generalizes lemma 3: the underlying edges of each subpath necessarily exist in both directions to enable the subpaths to exist in $MP(d)$ for $k \geq 1$. The interest of Lemma 3 and theorem 15 is to highlight subtle differences introduced with ECMP (more alternances and longer elementary circuits) and the necessary conditions. If the perfect symmetry is not granted, Theorem 16 does not hold anymore. In practice, a degraded ECMP state may limit our ability to prevent loops depending on the implementation of the model taking benefit from Theorem 16. We discuss several model implementations in the next section. But before, let me now conclude this section with the corollary discussed at the beginning.

Corollary 5. *Considering ECMP (or the unique symmetric path forwarding model), for any failed link and edge $e \in MP(d)$, if e is both a pre- and post-convergence edge, it cannot be involved in any circuit.*

Proof. This is a direct consequence of Theorem 16 as an edge cannot be used in both directions with the same convergence type (neither for pre- nor post-ones) by definition. □

V.1.b An Efficient Data-plane Algorithm for a Safe and Fast Convergence

There exists a current trend in data-plane based routing features [211, 75, 166, 167], in particular for fast re-routing purpose. It indeed allows for faster reactions in case of unplanned events such as failure, typically by provisioning possibly sub-optimal backup routes transiently used to wait for the convergence of the control-plane. In this work, I aim to go one step further and look at optimal routing updates directly performed within the data-plane (and not only at the control-plane). That is transparently converge to the new optimal forwarding states as soon as the change is detected either with a local failure detection or looking at data-packets to detect and fix any anomalies and also mitigate suboptimal transient states.

The following theorem is the baseline we rely on to design our synchronization algorithm efficiently preventing forwarding loops.

Theorem 17. *Considering perfect symmetry, synchronizing each couple of adjacent nodes (u, v), whose one, e.g. u, aims to perform a FIB update such that $v \in \mathcal{F}_2^*(u, d, l)$) while the other verifies $u \in \mathcal{F}_1(v, d)$, is sufficient to prevent any forwarding loops during routing transitions towards a given destination d. This condition becomes necessary considering that all circuits in $MP(d)$ actually occur as forwarding loops (i.e. with the most unfavorable order of updates).*

In order to prove this statement that is the main requirement to safely enable the opportunistic data-plane update model discussed previously, let us here first consider a simple explicit synchronization algorithm to carefully update the states of nodes in a correct order (i.e. without introducing forwarding loops). This algorithm follows the guideline of the Theorem: it explicitly synchronizes adjacent nodes if (at least) one has to perform a FIB update.

V.1.b.1 The Explicit Model as a Pedagogical Baseline

First, note that this model is provided only to ease the understanding of the correctness of the proposed approach; it is safe and easier to prove but *may be slow in practice* regarding the more efficient data-plane approach I aim to design[3]. It would only degrade the convergence time, but not its correctness regarding packet losses and loops avoidance: indeed, with a re-routing solution as the one developed before in II.1, ensuring reachability is always achievable thanks to possibly sub-optimal re-routing paths offered with TI-LFA locally at the failure (globally such paths may be not elementary but are simple ones at

[3]Using opportunistic implicit updates to speed up the convergence and so reach faster and correctly the route optimality is developed at the end of the section.

worst). Said differently, keeping the pre-convergence states as long as required by a given synchronization algorithm does not provoke losses as we consider the node local to the failure to be able to handle them; however, it has an impact on its performance regarding its optimality and efficiency as sub-optimal transient paths can be used more or less for a long time. Here is the sketch of such a safe and explicit (but slow) synchronization algorithm (provided for the sake of simplicity of the proof) for a given failure l, considering an arbitrary node u (not local to l):

1. Initially, for a given couple (update numbered r, destination d), all nodes $u \in V$ are in passive state;
2. When a node u has finished to compute $\mathcal{F}_2(u,d,l)$ for a given (r,d), it applies the following procedure and then looks at requests lying in its pending list for (r,d) (and deals with them as described in item 3):
   - If $l \in \mathcal{D}_1(u,d)$, remove from $\mathcal{F}_1(u,d)$ next-hops $n_1^j(u,d)$ such that $\nexists p_1^k(u,d) = (n_1^j(u,d), \cdots, d) \in \mathcal{D}_1(u,d) \mid l \notin p_1^k(u,d)$, i.e., no path in $\mathcal{D}_1(u,d)$ starts with next-hop $n_1^j(u,d)$ and does not contain l;
   - If $c_2(u,d,l) > c_1(u,d) \wedge \mathcal{D}_2^*(l) \neq \emptyset$, that is node u aims to actually add new forwarding states because existing paths in $\mathcal{D}_1(u,d)$ are not enough for protection, then for each of its new next-hop $n_2^j \in \mathcal{F}_2^*(l) \subseteq \mathcal{F}_2(u,d,l)$, it sends an `activation request` to node $v = n_2^j$ if $v \neq d$ before actually using/installing it. By definition, this new next-hop is not included[4] in the previous next-hop set $\mathcal{F}_1(u,d)$. Otherwise ($\mathcal{F}_2^*(l) = \emptyset$), as stressed with the term *actual forwarding change*, u has nothing to do except potentially turning some next-hops off if necessary (previous step condition): u can then turn active without sending any message (u just updates its forwarding states and continue to rely on next-hops n_1^j not verifying the condition of the previous step). On the contrary ($\mathcal{F}_2^*(l) \neq \emptyset$), when an actual forwarding change do occur, pre- and post-edges do not coincide and node u should thus notify its neighbor $v = n_2^j$ that it aims to use it for destination d (this message contains the identity of u and d plus the sequence number r of the update);
3. When receiving/dealing with such an `activation request`, a node $v = n_2^j$ operates this way:
   - If it is already active regarding the couple (r,d) (it has respectively updated and installed its forwarding states with the procedure described above and after), v answers with an `activation confirmation` (this message contains r, d and its own identity);
   - Else if u belongs to the set of current active next-hops (the pre-convergence ones, $u \in \mathcal{F}_1(v,d)$ – the **incoming-outgoing loop condition**) it pushes/lets this request (once) in a pending list, or sends an `activation confirmation` otherwise.
4. When receiving such an `activation confirmation` from v, node u can turn active for the couple (r,d) and so actually uses and installs its new next-hop $v = n_2^j$ to forward packets to d; In addition, it can now answer with an `activation confirmation` for all pending requests it records.

Let me call this algorithm SEU, standing for Safe Explicit Updates, and continue to assume symmetric weights in order to conduct the following sketch of proof for theorem 17. This algorithm is indeed used as a baseline example for explicitly synchronizing adjacent nodes (like stated in Th. 17).

Proof. Let us consider an edge $e = (u,v)$ involved in a circuit towards a given destination d after the failure of a link l. We know from Theorem 16 (and its corollary 5) that (v,u) also exists in $MP(d)$. The SEU algorithm is able to detect this property thanks to its incoming-outgoing condition.
If, (i), SEU does not detect such a link symmetry for a given e (locally on u because there is no actual forwarding change), we thus know that e is not involved in any circuit and can be used safely. Moreover u satisfies the LFA condition if e belongs both to a backup path from an upstream source $s \neq u$ towards d ($e \in p_2(s,d,l)$) such that no circuit has to be prevented again up to d (Property 2 can be generalized to any source not necessarily local to the failure thanks to the general Property 3). If $s = u$ or if edge e is exclusively a pre- or post-convergence one in $MP(d)$ (the incoming-outgoing condition is not met

[4] If $|\mathcal{F}_1(u,d)| = 1 \wedge l \in \mathcal{D}_1(u,d) \wedge c_2(u,d,l) > c_1(u,d)$, n_2^j can be directly compared with n_1^1 ($n_2^j \neq n_1^1$).

neither), we need to prove that there is no operation to perform in order to avoid loops (as SEU does not handle those cases neither and, as such, considers them as safes). Since e is not involved in any circuit by itself (Theorem 16 again), in order for a loop to occur (later on the path) an actual forwarding change should occur as well, that is a post-convergence edge not used towards d before the failure l needs to be activated. Thus, only the exclusive post-convergence case requires to be considered, i.e., $\mathcal{F}_2^*(l) \neq \emptyset$. Links which are exclusively represented with a post-convergence edge (u, v) in $MP(d)$ have an advantageous property. If $(v, u) \notin MP(d)$ it implies it was not used before the failure from any source and so does not belong to $p_1(u, d)$; as such, it requires one segment (either a local push or an adjacency/node segment corresponding, by construction and in both cases, at worst to a transverse edge in $p_2^j(u, d, l) \in \mathcal{F}_2^*(l)$). We can conclude with Theorem 3: with symmetric weights, this link, and at worst the ones until the tranverse edge, are the only one towards d to be forced from u, and then, on the remainder of the path after the detour forced with the segment, pre- and post-convergence edges necessarily coincide up to d (without loops).

On the contrary, (ii), when the loop is detected with an activation request from u (thanks to the incoming-outgoing condition checked in item 3), SEU puts the activation request in a pending list and just waits to turn active. For this to happen, the node v can react immediately or not (two options in item 3 of SEU). Here we have to prove that the elementary loop of size 2 detected by v thanks to the message of u cannot occur when v finally sends the confirmation to u (item 4). It is obvious as either v has already turned active itself for this next-hop (and cannot send back the traffic to u) or waits it can. We then just need to demonstrate the progression because v also sends its request(s) to its own next-hop(s) otherwise (and may wait in-definitively). Node v will turn active if and only if all its descendants in $\mathcal{D}_2^*(l)$ are also safe. SEU indeed terminates because there exist safe post-convergence paths by definition (at worst the empty path when reaching d itself); the recursively distributed requests will eventually reach safe nodes (LFA release exit point) and be all satisfied. Hence, u will eventually activate its post-convergence next-hop after v performs the same (the non answer of v is blocking for u): the circuit (u, v, u) cannot occur (as well as (v, u, v)) even if it has only been checked from u to v (no need to verify both directions, SEU considers post-convergence edges only); and both v and then u will convergence to their new post-convergence forwarding states without letting the loop between them or any involving them occurring (Theorem 16). □

With this last property (Theorem 17) each operation required for a practical implementation are given. On this basis, one can prevent any loops (whatever the size and nature of the underlying circuit with or without ECMP as long as we have perfect symmetry) just focusing on the smallest ones (of size two) inducing all others. **Preventing such two nodes loops is then enough in theory and easy to achieve in practice (provided that weights are symmetric).**

V.1.b.2 Towards an Implicit and Opportunistic Update Model: Speeding up the Convergence Preventing Any and Only Actual Loops

If a router cannot use both, and simultaneously, an incoming post-edge and an outgoing pre-edge for a given destination (or respectively an incoming pre-edge and an outgoing post-edge – but focusing only on the first is enough as we have seen), no circuit can occur because either all edges of a path are then post-ones, or respectively pre-ones (no alternance). This property is easy to maintain between adjacent routers: if a router receives a packet that enters via the interface it aims to use as the outgoing one in the new graph, it should consider its new outgoing interface with care. Instead of just updating its forwarding entry and continue as usual, it can either wait that this neighbor notify it that it is ready to forward packets in the new graph too, and/or rely in the meanwhile on TI-LFA encapsulation to force packet on the new post-convergence path (with one adjacency segment at worst). In practice, I aim to use such a condition to detect actual forwarding loops and force TI-LFA encapsulation only when necessary. Moreover, such an encapsulation provides enough information for the outdated router to perform its update and stop the loop. The TI-LFA encapsulation is used a a safety net during the convergence, and this event based mechanism (the loop detection condition) avoids the use of timers or explicit messages.

Using such a condition, I will now progressively introduce this efficient and fast implicit notification scheme relying on pre-computed re-routing paths but not only (more options are required for also solving distant loops). Up to now, while the SEU algorithm would transiently rely on the re-routing paths

(locally optimal but not globally, especially for distant nodes) as long as necessary to update nodes in the right order to safely avoid all loops, I envision here a much more efficient approach where re-routing SR paths replace explicit triggers for updating forwarding states. The goal is to switch progressively and quickly to the (optimal) post-convergence paths rather than waiting explicit messages. All the necessary information indeed lies in the TI-LFA re-routed packets.

Using a simple condition such as **the incoming interface is equal to the outgoing one is the baseline to detect and prevent all loops in symmetric networks; with SR and new TI-LFA updates, we can go one step further and also avoid technical issues like timers even if updates are not treated instantaneously**. In theory, with a new kind of TI-LFA packet and their related data-plane based forwarding updates, one can prevent even loops occurring for a single tour by anticipating any anomalies due to the transient the loss of synchronisation. Instead of looking at the former condition, routers aiming at activating new next-hops can simply send an implicit update message in any case: this update can be implicit thanks to the new TI-LFA packet. However, this requires the update to be immediate otherwise packets need to be encapsulated until an arbitrary timer expires.

This technical difficulty, i.e. dealing with delayed or lost updates, turns the problem into a more funny game. While new next-hops can be enforced on the fly as before with TI-LFA detoured packets, applying such corrections can be performed only if necessary according to data-plane inputs (packets detected as loopy ones). That is new next-hops are activated without loop prevention (but possibly with a single encapsulated packet), and only those which are detected as loopy are encapsulated (only one tour in the two edges loop before being safely encapsulated). More precisely, with the same principle of loop detection of SEU, a router having a new post-convergence edge can choose between two options using a simple event-based algorithm (i.e. without relying on timers) based on such packets: either using its new next-hop without encapsulation if no loop is detected (e.g. for its own locally generated packets) or relying on encapsulation otherwise (note that packets sent without encapsulation first may come back as long as the neighbor has not also performed the update). While updates are usually performed in the control-plane, I argue that some, if not all, can be implemented in parallel within the data-plane to speed up the convergence (in particular regarding the explicit model). When the routers are adjacent to the failure (directly or not - as illustrated in Fig. V.3), they can proceed as follows. The one detecting locally the failure encapsulates its first detoured packet within a TI-LFA packet (having only one segment). The others will in turn update their states thanks to this single message. Either such updates are performed immediately within the data-plane, or delayed. In the latter case, a loop occur and is detected by the router local to the failure and such loopy packets are encapsulated for safety. This encapsulation is necessary for safety (when the neighbor does not succeed to update its forwarding states at line-rate) but also allow to indicate the exact update to perform. At a glance, link reversal techniques [211] can also be applied but with our approach we go one step further: links are not blindly reversed, but the content of the TI-LFA detoured packet (i.e. the destination contained in its segment) allows to precisely know what should be the new outgoing interface (no longer backup paths can be used transiently). We fasten the convergence and mitigate intermediary transient steps to reach it. A similar principle can apply for remote routers involved in loops as we will generalize a bit later.

With current LFA like solutions, timers are necessary to eventually switch the states of the routers encapsulating detoured packets; indeed, at some point, they need to stop the encapsulation and fully re-converge (packets being encapsulated in the meanwhile). Calibrating such timers is not a trivial task: either it is set too small (inducing overhead) or too large (not reactive enough). For safety reasons (to avoid all loops), one may set it up too large in practice, slowing down the total convergence time. If it is too small, transient loops are indeed likely to occur: while the source s, local to the failure, immediately does possess all its new states (thanks to its pre-computations), its neighbors do not. In such a case the transient desynchronisation between neighbors is maximized. On the contrary, with our SEU algorithm, a node knows when it can stop relying on the re-routing TI-LFA as it relies on an explicit notification. With such a recursive synchronisation between neighbors, SEU avoids the use of timers but remains too slow regarding new opportunities offered with flexible data-planes. My goal is to let routers immediately switch to their new forwarding states without waiting any explicit notification. With data-plane updates, it is possible to impose the changes from neighbor to neighbor on the fly thanks to TI-LFA messages (that contain enough information for updating next-hops along the path). This way, forwarding paths can be quickly turned into post-convergence ones anywhere in the network. A single packet being enough in

theory as the information contained in a TI-LFA packet is sufficient (the top segment and its underlying destination).

While we may generalize such results for asymmetric weights when routers (and loops) are local to the failure, it is not the case with routers involved in distant loops. Indeed, considering only pre-computed re-routing paths is not enough to deploy the new model I have in mind. Not only re-routed paths local to the failure can take benefit from this approach but the overall network and forwarding paths. In particular, routers involved in circuits remote to the failed composant which cannot rely on pre-computed backup paths for the remote failure. The same approach can work with a new kind of TI-LFA detoured packets: the transient loop cleaner ones. The only difference with the basic TI-LFA model is that the encoded detour is not anymore pre-computed but is retrieve on the fly thanks to the convergence of the control-plane (the goal is only safety as a local backup path already exists even if they are globally suboptimal – there is no black-holes).

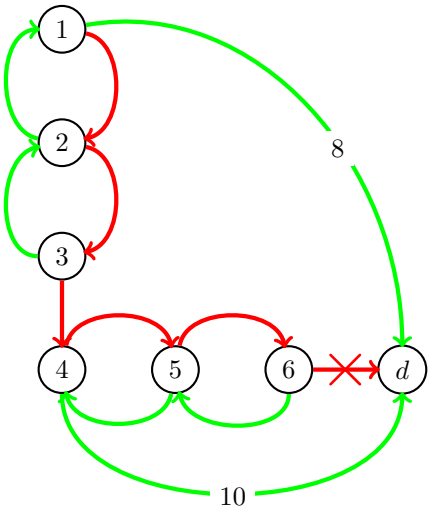


Figure V.3: Local vs. Distant Transient Forwarding Loops Resolution: while local ones (here with nodes $4, 5, 6$) can be directly prevented with the pre-computed knowledge, it is not the case for distant ones (nodes $1, 2, 3$). However a similar method can apply to solve them all.

So far, we have seen that symmetric weights ease the detection of the loops, in particular local to the failure, but we have not discussed how a router can react when it is distant from the failure as illustrated in Fig. V.3. Indeed, while routers involved in loops directly (i.e 6) or transitively adjacent (i.e 4 and 5) to the failure, can benefit from the updates of 6 that has pre-computed its backup path, it is not the case for nodes $1, 2$ and 3 also involved in potential transient forwarding loops. When a router observes that the incoming interface is equal to its outgoing interface and that it has not converged to its new state yet, it implies that no TI-LFA packet received so far has notified a more recent update than its own. Our approach consists in forcing such new TI-LFA for preventing remote loops with quick updates. However, since the failure is not local, it cannot neither rely on its pre-computed paths nor on the ones of the router to which the failed link belongs to. Instead of relying on pre-computed paths, (remote) routers aiming at applying a change can simply consider their novel post-convergence paths (that they compute at the control-plane after receiving the given LSA) and compare it with their pre-convergence SPT: they can encapsulate the packets towards the first link in the post-convergence path not in common between the two (forcing the latest exit). That is looking at the first distinct link in the path having the most nodes in common if such a path exists (otherwise it is enough to push the traffic towards the direct neighbor as it cannot result in a loop[5]). As well as with classic TI-LFA, one can rely on the latest or earliest exit approach (the available information is enough for the latter model but one may prefer to opt

[5]That is the neighbor v of u cannot send back the traffic to u triggering such a computation. Indeed, if there is a loop between the two, there is at least one node in common in the pre-SPT and the post-convergence path of u as weights are symmetric as well as paths, here just one link, between the two nodes.

for computing efficiency rather than mitigating the SR overhead).

One can thus safely opt for an opportunistic approach using TI-LFA detoured packets to only perform useful operations (without waiting for non existing or occurring loops). I call this strategy OIU for Opportunistic Implicit Updates[6]. It relies on the same principles as SEU but implements this new rule:

Definition 6.
We say that a device r applies the OIU rule for d if it updates its forwarding states when receiving a TI-LFA packet via node u (or respectively adjacency (u, v)) destined to d following this simple principle: $\mathcal{F}_1(r, d) \leftarrow \mathcal{F}_1(r, u)$ (resp. $\mathcal{F}_1(r, d) \leftarrow v$ if $r = u$ in the case of the adjacency segment).

Using this OIU rule is enough with symmetric weights as the Q-space follows the one of P if they do not intersect for the given failure, local or not (property 3). This is true for any post-convergence paths and easy to encode in one SR segment with either the latest or earliest exit model: using a *blind* test on the latest exit first and then possibly refine to check whether an earlier exit is available. In theory, each router only has to send one TI-LFA packet encoding in 1 segment the new post-convergence path (if locally necessary for the destination) to ensure that the whole network converges immediately. However, if not all routers can update their forwarding states at line-rate with embedded data-plane updates (e.g., considering incremental heterogeneous deployment or simply packet losses), this may lead to loops as the encapsulation is not anymore available. The following theorem formalizes this last opportunity along with its related limitation.

Theorem 18.
Considering a perfectly symmetric path forwarding model, an arbitrary failure l (remote or not) and a given destination d, it is sufficient that each routing device sends (only) one TI-LFA packet (made of exactly one segment) towards each its new post-convergence next-hops in $\mathcal{F}_2^*(l)$ before activating them safely (without SR) in order to globally update all forwarding states for d without creating any forwarding loops.
These conditions, and the single TI-LFA for each new next-hop in particular is indeed sufficient as long as each device receiving the TI-LFA packet updates their forwarding states immediately according to the OIU rule (this implies that all routers support embedded data-plane updates). Otherwise, if this update may be delayed (e.g., treated at the control-plane or even lost or not supported by a subset routers), adding the incoming-outgoing condition becomes necessary to prevent loops and some packets may still loop once.

Relying systematically on a single SR forced path when activating new next-hops is then risky. We prefer to opt for a lazy but more safe option. The opportunistic model I envision to develop relies on loop detection (packet looping just once at worst). This acts as the SEU signal that triggers in turn the TI-LFA encapsulation of the loopy packet to ensure its exit of the loop and possibly update recursively the neighbors in the backup path. However, such a loop may be observed in both directions:

- if a non updated device observes a new incoming SR one (while using it as its current outgoing interface – or not), it should update its states as soon as possible (and forwards immediately the packet relying on the SR encapsulation as long as the update is not performed). Here the sequence number is required in the SR packet but no error are possible thanks to this information;
- if a node (presumably having installed its new states – at least, it has not receive a non treated SR implicit update) observes an incoming normal packet (presumably sent by a non updated device) while using it as its outgoing interface, it (re-)encapsulates with SR the packet for the last change it knows. This is the correct behavior if it this node is indeed the new one (using green edges), but not otherwise (using red edges).

[6] My aim is to finish developing formally this scheme in a near future. This ongoing work is not mature enough to be fully technically detailed in this report.

When a node detects such an error (states inversion), i.e. a non updated node trying to send an out-of-date SR packet (sequence number), it re-encapsulates the packet with its more recent information. One tour and half can occur at worst for the packets leading to such errors. The new kind of TI-LFA detoured packets I have introduced acts as separators between old and new forwarding states and are used only if necessary (to avoid loops really occurring by stopping them at the packet granularity).

What we have done so far in Chapter II was not enough to have an ideal routing convergence: neither our re-routing solutions ensure the absence of transient loops, nor our mechanisms for loop-free transitions allow to handle well failures on its own (unexpected events by definition). This first project conciliates both aspects in a single option. It is not just about combining both techniques, but rather to design a new technique taking these two questions in one, natively. The detailed data-plane algorithm(s) and the proof(s) are not provided here as they are ongoing and future works, as well as results and statements in this project chapter whose proofs have to be refined before targeting a publication in a scientific venue. Technically speaking and so far, we have relied implicitly on a very simple model where only one change occur (at a time and even more globally): we have to take into account sequence numbers in a careful way to avoid troubles when multiple changes occur in a limited time scale.

In summary, with SR and symmetric weights, such a data-plane model seems very attractive. It can anticipate one event directly within the data-plane, converging on its own only looking at strictly necessary operations. My current goal in this project is to answer this question:

Research Question

How to Enable Efficient Data-Plane Routing Updates?

I have shown that we can implement it implicitly with data-plane updates based on TI-LFA detoured packets (local or not to the failures). This approach looks very promising as in theory, the network can convergence very fast and safely to its new states. The fastest routers can help slower one using source routing.

With Stefano Vissicchio, Quentin Bramas, Anissa Lamani, and presumably other folks, I will continue to explore this research path in order to target an overall technical implementation [289]. Many efforts have still to be done as the current status is not satisfying enough. In particular, one mid term direction attracts my attention: designing efficient failure detection mechanisms with triggers based on measurements. One challenge is to avoid, as much as possible, to stress the control-plane when change occur. For example, when the symmetric model is not perfect, how to ensure the same property as the ones we obtain in the other case: while the information available at the control plane is enough, the data-plane may not possess the necessary knowledge. Even worst, what to do with non symmetrical paths? The next section will elaborate on this general topic regarding at two other options.

V.2 Smart Data-Planes & Optimal Route Updates for Complex Events

In this section, I will continue to expose my main mid-term projects: pushing part of the intelligence at the data-plane to better prevent re-routing anomalies and downtime periods. In particular, I am interested in more complex scenarios than simple link updates within the IGP. I will develop here two main examples, the first one to handle router-wide re-configurations and the other to push *OPTIC* within the data-plane for BGP traffic.

V.2.a Efficient Node Protection with SR: the Advantages of Symmetric Weights

Dealing with node failure is more difficult than handling a simple link removal. It leads to several failed links to consider. Even detecting their existence is challenging as no specific advertisement can exist, and by definition, such an event should be inferred. For example, one may consider the multiple messages from neighbor nodes (e.g., with a threshold correlating their numbers in a short period of time) or using a management address with an independent protocol. Since their failures have more consequences on the

topology, both the SPC algorithm and the deployment cost are not the same as for a single link. My objective is then to answer this question:

Research Question

How to Efficiently Protect SR Networks from Router-wide Failures within an AS?

More precisely, my goal is to study this question within SR domains using TI-LFA detours. However, even with symmetric weights and with the help of SR to efficiently deploy backup routes, the challenge is more difficult than with link failures: in particular, we will see that multiple segments may be required and that *TBFH*, as it is designed in section II.1, is not enough to compute and encode the necessary segment lists to protect the network from a node failure.

More than One Segment is Required even with Symmetric Weights In the following, I will illustrate how at most $deg(f_k)-1$ segments are necessary to detour the traffic from a given neighbor f_k of the protecting source s (with $1 \leq k \leq deg(s)$). Here $f_k \in succ(s)$ simply denotes the neighbor of s which is considered to be failed. Once s detects such a neighbor-wide failure (such a detection may take the form of an inference or a prediction), it can rely on its node protecting segment lists instead of link ones (regarding the edge (s, f_k) in particular). Such lists are computed in advance and can be used safely for both node or link protection [7]; however, if used for the latter case, although safes (that is free of loops and actually able to ensure link protection), they are not necessary optimal neither in term of number of segments nor with respect to the actual post-convergence routes (the ones resulting from the link failure may differ from the ones imposed by the node failure). In this section, I focus on the problem of their computation and their cost (SR overhead) rather than on the detection mechanism allowing to determine which list to use (i.e. how to chose between node or link protection at first?).

Looking at this specific problem leads to this theorem (the proof is an ongoing work in progress illustrated in Fig. V.4 and some arguments are already developed in the following):

Theorem 19. *With $f_k \in succ(s)$, the neighbor node detected as failed from s, and $\#seg$ the maximal number of segments required to optimally detour the traffic from f_k, we have $\#seg < deg(f_k)$ for any destination (and any predecessor of f_k, including s), if the link valuation is symmetric.*

At the first glance, instead of only 2 SPC for dealing with all single link failures adjacents to a source as with *TBFH*, handling node protection looks thus to require more processing time, e.g., by computing $deg(s)$ ECMP DAG rooted at s, one without each node f_k. Nevertheless, I will now explain why only 2 SPC remains enough to compute and encode the necessary backup routes as segment lists for node protection, even when they require more than 2 segments. As long as weights are symmetric, segment lists are indeed easily retrievable for both the latest or earliest exit models. To achieve such an objective, *TBFH* requires to be extended as *TBFN*, an algorithm computing backup routes which are (optimal) post-convergence paths not including the failed node. Similarly to *TBFH*, its main ability lies in its capacity to do so for all neighbors of s, as failed neighbors, in a single extra run regarding a standard SPC algorithm.

In the following, I will explain how this theorem can hold by providing the sketch of the algorithm and its SR encoding scheme:

Theorem 20. *Computing all the segment lists for protecting all neighbor nodes of a given source can be achieved in only 2 SPC with TBFN.*

TBFN, or re-Designing TBFH for Node Protection The first obvious difference between this variant and its original counterpart is that *TBFN* searches for secondary best paths entirely ignoring each neighbor of s, i.e. by removing all f_k in its second stage (not only the incoming link (s, f_k) but also all outgoing links of f_k – other incoming ones are then silently ignored de facto). Instead of providing first-link-disjoint

[7] While the reciprocal is not true, link protecting segments list are not sufficient to handle all node failure scenarios.

secondary best paths, *TBFN* seeks for the first-node-disjoint ones. For the same reasons as with *TBFH*, node protection optimal paths are also 1-alternate paths[8] although the SPT decomposition becomes more scattered. Indeed, some internal edges become now external regarding sub-branches resulting from the removal of a given f_k (because with *TBFN* the branch originally rooted at f_k is now decomposed into as many sub-branches as its degree minus 1). Such edges are called *internal/external* and are natively embedded in the second stage of this node oriented version of *TBFH*. That is *TBFN* keeps track on the *TBFH* decomposition by grouping its subbranches for each f_k. Among these edges internal for each group of subbranches, some are equivalent to transverse edges to be forced with one adjacency segment (worst case for each neighbor of f_k), other may require a node segment satisfying the LFA condition specific to node protection (to reach the next neighbor or the destination) and finally, remaining ones are simply transparents as original internal edges (they do not cost any segment/detour to avoid f_k). The encoding of backup paths is based on the analysis of the underlying cost of using each of these internal/external edges.

At this point, let us define $k - i/e$ paths as optimal 1-alternate paths using one transverse edge and $k-1$ internal/external paths in the second stage of *TBFN*. For each of these paths, we can start by testing whether the outvertex of the transverse edge verifies the node LFA condition. If it does, we look for the earliest possible exit and conclude with only 1 segment. Otherwise, the path requires more than one edge to be forced, and as such, more than one segment. In this case, we know that this backup path will already led to at least two segments: one adjacency segment for the first transverse edge at worst, more likely a node one or a local push with the earliest exit mode, and at least one more segment will be required to deal with remaining internal/external edges among subbranches.

An Illustration of the Worst Case The example given in Fig. V.4 illustrates how *TBFN* solves this problem and efficiently returns sequences whose size is limited by the degree of the failed node. In this example, s requires two segments to reach d and avoid the failed node f_k: one detour via s_1 and then with s_2.

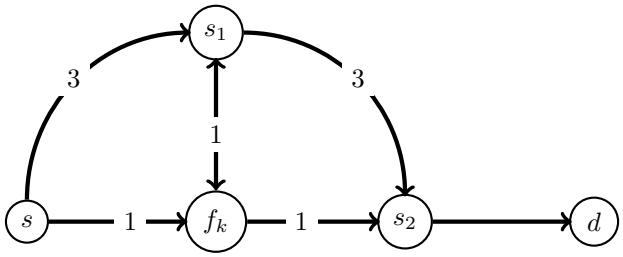


Figure V.4: With node protection, several segments can be required even with symmetric valuation: as many as the degree (minus 1) of the failed node in the worst case. Here both the transverse edge ((s, s_1)) and the internal/external edge ((s_1, s_2)) have to be forced.

Considering the SPT decomposition applied with *TBFN*, one can observe that the 1-alternate path s, s_1, s_2, d avoids f_k to reach d by using first a transverse edge, and second an internal/external edge. While the transverse edge (s, s_1) can be forced with a local push at s (instead of an adjacency segment if it would be forced remotely), (s_1, s_2) is not anymore a transparent internal edge as with *TBFH*. Indeed, by removing f_k and its links (here (s_1, f_k)), several sub-branches originally related to f_k appear: one containing only s_1 and the other with both s_2 and d (and the link between them). Then, (s_1, s_2) is an internal/external edge regarding f_k and since the path via f_k offers a better distance to connect s_1 and s_2 than the direct link between them, it cannot be used without inducing operational overhead. This example indeed shows a case where f_k always offers the best paths towards d for both s and s_1: in between subsequent sub-branches, the weights of the links to be forced are higher than best paths through f_k.Each of them thus costs one segment: with f_k having two sub-branches, and in addition to the initial transverse edge to reach the original branch pseudo-rooted at f_k, forcing the internal/external edge (s_1, s_2) between them is required. This figure can be easily extended to deal with more general cases:

[8] In order to simplify the following explanations, I prefer to ignore the case of k-alternate paths of equal distances as it has no impact on the main outcomes of this section.

while the degree of f_k determines the maximal number of segments $\#seg$, adding more links between s_i (or between f_k and s_i) has no impact on the maximal number of internal/external edges to be forced. Since only a single segment is required between subsequent (sub)branches with symmetric weights (for the same reasons as in the proof of Theorem 16) and because there is at most one internal-external in use among subbranches, only $deg(f_k) - 1$ internal-external edges at most are required to be forced on the underlying alternate path between subbranches.

Encoding List of Segments Without the SR Graph Encoding segment paths is more challenging when several segments need to be stacked. As already discussed in section II.1, two approaches are viable relying on a SR graph: either the *latest exit* (pushing the segment to the last node in P, at worst pushing at the transverse edge with an adjacency segment), or the *earliest exit*, the same greedy approach but computed in the reverse direction. Here, in the case of node protection (and symmetric weights), I will show that we can avoid using the SR graph to encode such paths both with the latest and earliest exit mode. There is enough distance information in the path computed with *TBFN* to conclude without relying on a APSP or several SPC.

The symmetric valuation, seems to offer a notable advantage: for a given requirement, e.g., avoiding a given link, node or verifying TE constraints, properties like sub-distances on a forward path are the same as the ones of the associated return path. However, avoiding the use of the SR is required for both the latest and earliest exit models. Namely, looking backward in the path built with *TBFN* to retrive the earliest exit solution is not an opportunity offered with symmetry.

Overall, one can design a round trip procedure or just compute the segment list for one of the two exit modes. For example, first computing the latest exit solution and its sequence of intermediary latest exit nodes, then looking back at earliest options before such nodes. It provides the sequence of interval(s) where to find the earliest exit node or an adjacency segment (the interval is reduced to one link in such a worst case for SR with MPLS data-plane).

More precisely, the iterative procedure consists in checking the node LFA condition on the internal-external edges and their vertex contained in the alternate path returned with *TBFN* to return the latest exits; or doing the same but looking back. Initially, from the penultimate latest internal-external edge towards the destination and, in a backward fashion, between intermediary nodes (x_{j-1}, x_{j+1}).

Obviously, the same minimal number of segments is required for both the latest and earliest exit approaches, as well as the same cardinal of detour points is mandatory with symmetric paths. These sequence of disjoint intervals can be then translate into a sequence of subsequent nodes in the path whose at least one should be forced in order to build a proper encoding.

To compute the latest exit encoding, at each step in the node protecting path where there exists an internal/external edge (between two sub branches) to process, we check whether it is necessary to force it (outside P) to reach the other branch. This greedy model return latest exits segment lists.

Let us denote such edges (i, e) for the first one, (i_2, e_2) for the second one, and (i_j, e_j) for the next j^{th} ones. Initially it leads to the usual LFA node-protection condition (with $f_k \in succ(s)$ the node to protect):

$$c(e, d) < c(e, f_k) + c(f_k, d)$$

If not satisfied, one should now check whether (i, e) must be forced with an adjacency segment or if the node segment i is enough to reach the next sub-branch (as the first latest exit). It yields to the specific adjacency equation: $w(i, e) > c(i, f_k)+c(f_k, e)$ to determine if adjacency (i, e) is mandatory[9]. If this is the case, then e is considered at the new starting point for the next step (and the first segment is the associated adjacency), otherwise, i can be chosen as the first node segment for the latest exit. Then, the procedure can restart from i (or e according to the previous step) and now considers the next internal/external edge to come. That is setting d as e_2 to look iteratively for the next internal/external edge (i_2, e_2) among other sub-branches if any, i.e. checking if $c(i, e_2) < c(i, f_k) + c(f_k, e_2)$ (or $c(e, e_2) < c(e, f_k) + c(f_k, e_2)$ if the adjacency has been forced at the previous step). As long as it is true, the procedure can continue without adding any segment, otherwise it adds either a node or an adjacency segment (with the same

[9]We do not need to verify if there is other shortest paths from i to e because if the link (i, e) is included in the best path between s and d not using f_k, it implies that the only path connecting these two extremities that may be shorter than the direct link is via f_k.

test as before adapted to the current tested internal/external edge (i_j, e_j)), and repeat the same sequence of tests until the end of path (as long as there is at least one internal/external edge to consider).

This procedure returns a latest exit solution. If one aims at determining the earliest exit without using best ECMP DAG of each required source (a kind of SR graph or APSP), she needs to start from the end, the test is about finding the earliest node in the path that satisfy the node LFA condition towards the destination. That is from d as destination and then looking backward at the path containing $(i, e), \cdots, (i_j, e_j)$, the intermediary internal-external links which require to be forced successively looking for the earliest node x before i_{j-1} where $c(x, e_j) < c(x, f_k) + c(f_k, e_j)$. This node becomes the new current destination and the same process is repeated backward until reaching the source itself as the earliest node before the initial transverse edge. Note that with MPLS, adjacency segments are special cases where no improvement in any direction can be performed, the interval between earliest and latest exits is reduced to one option (the link itself).

To summarize this project, my goal is to provide an efficient node protection feature based on SR and advanced TI-LFA mechanisms (possibly embedded within the data-plane to enable implicit updates as discussed in the previous section). At least, the control-plane can prepare the data-plane for both the best current routes and to handle any single failure, including router-wide outages, by anticipating them. Indeed, I am convinced that this proposal is light enough to fit this need in SR networks having flexible hardware support. Like OPTIC in the next section, my aim is to smartly populate the data-plane such that it has at least a step ahead in case of unexpected event.

V.2.b Implementing OPTIC in the Data-Plane

OPTIC also perfectly fits this opportunity and we aim to implement it within a flexible data-plane. Indeed, this notion of group of protection allows for both efficiency and scalability. Instead of updating the new minimal BGP next hop at the control- plane, it can be achieved at the data-plane. With Cristel Pelsser, Quentin Bramas and Jean-Romain Luttringer, we recently gain access to Tofino switches and their development environment, and start to develop our re-routing scheme in the data-plane.

The question we aim to address is the following:

Research Question

How to Leverage all Advantages of Flexible Forwarding Architectures with Complex Re-routing Models like OPTIC?

The problem is challenging as the language and the architecture are still inherently limited, e.g. not allowing simple control loops by design.

It indeed requires to deal both with such language limitations and the few number of stages offered by the pipeline and the restricted memory size and usage. Nevertheless, the challenge is attractive as the failure detection and reaction can be almost instantaneous. In practice, with OPTIC, the main advantage is that the IGP change can be directly interpreted by BGP. Updating the IGP table within the data-plane to correct the best BGP next hop (the gateway) is possible if implementing finely the set of gateways for groups of prefixes.

While we have for the moment tackled the absence of control loops in P4 with a static number of gateways to consider[10], we now explore the opportunity to both update registers and tables both at the data-plane and at the control-plane. Figure V.5 illustrates the pipeline that is currently implemented. First, for a prefix belonging to a given group g, we walk through its set of gateways ni only if its entry in the registers is not updated (checked with a sequence number: the per group number s should be equal to the global one $seqN$), otherwise the best next-hop can be directly picked from there. When the register value for a given set of next-hops is considered outdated (or null), the data-plane updates it itself. It considers a static number of steps (that can be extended if necessary) and extract the minimal value from the distance table asking for each element of the set of next-hops. Thanks to the predefined number of steps, we can avoid the use of loops by exploring chunk of next-hops with static nested conditions. In

[10] We currently explore other ways to perform such an operation using the advantages of the TCAM memory.

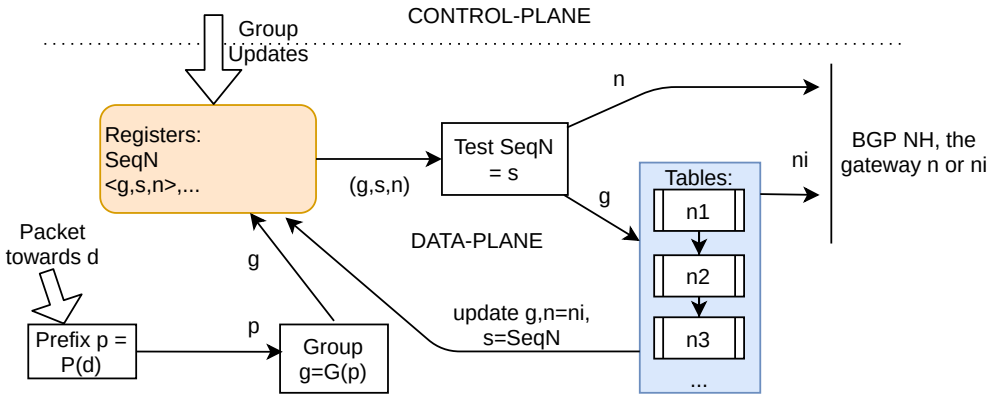


Figure V.5: Updating the *OPTIC* hierarchical FIB with both control and data-plane updates: the registers are shared between the two planes, and each of them concurrently update the entries towards each group. A group $< g, s, n >$ gives the identifier g, its sequence number s and current best next-hop n. Its identity is retrieved from p, the prefix being the most covering for d, the destination of the packet. If there exists a difference between s and $SeqN$ (with $s \leq SeqN$ by construction), then the new best gateway is not known (the control-plane has just notify that there is an entry to be updated and currently perform the necessary updates in background). In this case, a new best next-hop, ni, is computed within the data-plane at line-rate, thanks to the IGP table of distances. Finally, the $< g, s, n >$ entry is also updated in the register such that the one of the control-plane is superfluous.

any cases, although it can consume in theory one or several resubmits in the pipeline (when more steps are necessary), a single packet belonging to a given group of prefixes is enough to update the register and its sequence number (such that next packet does not need to be processed this way again, up to the next event). In parallel, the control-plane can both update IGP table distances and registers for each group, and in worst cases, setup new sets and groups if necessary (not illustrated in detail in figure V.5).

Overall the use of such a scheme enables to take the best of both planes: popular destinations are more likely to be updated with data packets (but one can mitigate this effect if they are prioritized in the control-plane) but trigger resubmit only in very rare cases while updates from the control planes are more useful for less used destinations (and avoid resubmits in any case and more processing in general). Such a method can be used in many re-routing schemes and I plan to combine them all. We also expect to combine it with solutions like PURR [75] and also look at how implementing the `min` operation using the TCAM memory ability (not per group but to encode the IGP table itself).

V.3 Multiple Topologies for Multiple Radios in Wireless Networks

This field of research is at the intersection of many of my research interests regarding routing and graphs; more specifically, it can be formulated as a multi-metric problem closely related to the one introduced in II.4. It also relies on a hierarchical routing model similar to the one we have seen in sections II.3 and III.3, with a two scale view of the network (internal and external areas are translated here to two radios having different characteristics and ranges). Before entering in the problem in more details, let me formulate the question I aim to tackle and then develop the context:

Research Question

How Efficiently Exploiting Multi-radios Routing Technologies to Preserve the Energy and Life-time of an IoT Network?

Wireless networks face several issues related to their medium and constraints. One of them is their life-time. With technology like wake up radios [138], it is now possible to better manage the battery

capacity and thus enhance wireless communications. With Julien Montavont, a member of my group already involved in this topic, we currently explore the opportunity to handle two kinds of radio links at the same time ([240, 51, 293]). He has already started to investigate the field with the thesis of Sebastian Sampayo [295] and we aim to extend their contributions with a real multi-hop architecture (for both radios).

V.3.a From Basic Assumptions Towards a Multi-Metric Model

With two radios and their distinct constraints and capabilities (in term of radios range and energy consumption in particular), several challenging routing problems occur according to the given assumptions and objectives. In particular, the challenge lies in the limits imposed with radio links: the ones available with the Wake-Up Radio (denoted Wake-Up Radio (WUR)) are not able to carry much information and exhibit a shorter range than the other one (the usual long range radio, denoted Long Range (LR)). However, they consume less energy and can be used to signal an actual message arrival hop by hop, that is relay after relay.

With such a two scale capacity, the wireless network can become a real multi-hop infrastructure where each device can act as a relay if necessary, both for wake up signals and usual LR communications. More specifically, the problem is about designing an energy efficient IoT routing protocol where the consuming radio, the one dealing with data packets, is used with care. For example, considering one technology (typically the WUR one) and its broadcast links defining the underlying subpaths to wake up the data oriented radio (the LR one), allows to ease the design of a efficient meta multi-hops system relying on simple connectivity assumptions. Since hop by hop LR communications will rely on their turn on WUR intermediary devices in this model, it is easier to reason on the WUR technology and assume that such radios form a single connected component whose links are a subset of LR ones.

Long Ranges Links Implies Short Ones Our first hypothesis to simplify the problem can be summarized as: the existence of a short range link (WUR) between two nodes $\Rightarrow$ the one of a long range link (LR) between the same two nodes. However, relying only on the short range topology is not enough to deploy (i.e. wake up) efficiently shortest LR paths: signalling messages should be advertised with both radios to deploy the two topologies and cross them.

Let us first assume a naive solution with a converge cast traffic towards a given sink that initializes the gradient process based on the WUR topological knowledge. At each hop, on the top and among the best paths offered with the wake-up radio, each forwarding device then choses the best one(s) regarding the longer range radio (depending on the available information that can be crossed). Logical information regarding the wake-up radios, e.g., the WUR best routes, can be transmitted using the LR radio as well as the LR distances. Only the initial WUR distances have to be transmitted using the short range radio. This approach has some advantages but also inherent drawbacks like Routing Protocol for Low Power and Lossy Networks (LLN) (RPL) [127] and more novel ones [42, 309]. Since we rely on wake-up radio best paths and consider at worst the same LR ones, we have a provably correct manner to wake up forwarding devices on the fly. However, since the LR radio and its distances are treated as a second metric, it is not optimal regarding this criteria (even if one can look for the best LR path made from best wake up radios ones). That is it can lead to a suboptimal situation where data forwarding LR paths are longer than necessary (shorter ones can exist). This effect can be mitigated and the path stretch analyzed to understand the negative impact of such an heuristic. While I envision to work on this kind of simple models and their related performance, let me develop here a more general framework with the good assumption and objective.

A First Model to Formulate Many Objectives With distinct and more general assumptions on the two radios (e.g. no link existence implications in any direction), the existence of paths from each device towards a given is anyway granted to result in a connected structure. More formally, consider two radios at each device and their respective graphs denoted $G_1 = (V_1, E_1, w_1)$ and $G_2 = (V_2, E_2, w_2)$ where $V_1 = V_2$ but $E_1 \neq E_2$ and the same for the respective edge valuation functions w_1 and w_2. In particular, we may have $E_1 \subset E_2$ with G_1 being the WUR graph and G_2 the LR one. We may also state that

$\forall e = (x, y) \in E_2$, $\exists$ a WUR path (with edges in E_1) between x and y, as a consequence of stating that G_1 and G_2 are connected graphs.

This leads to two possible assumptions that simplify the problem formulation at first glance:

- the existence of the WUR link $(x, y) \in E_1 \Rightarrow$ the one of the LR link ($(x, y) \in E_2$). The routing structure can/should be built (safely) around the WUR technology. At worst, the two paths may be considered the same;
- there exists at least one path between each pair of nodes in both graphs. In particular, from any node, there always exists a way to wake up any other targeted node: even if the underlying WUR subpath is long, the only necessary condition is for G_1 to remain connected to ensure this property.

These assumptions are not actually required to design a correct routing protocol as we will detail later, they are necessary and sufficient conditions to ensure the existence of feasible paths for all nodes. If G_1 is a connected graph, then the problem can be solved as added with the first assumption, it ensures that G_2 is connected too.

Let me now formulate the technical challenge to be solved. The WUR medium is very limited, it should not be stressed. Two or three identifiers in a message is a maximum. Once both LR and WUR routes advertised in the control-plane, at the data-plane, the first technical question to use them is: how to efficiently wake up intermediate (i.e. relay) nodes to enable LR communications? To avoid blind WUR flooding to initialize each LR transmission, only one efficient option is available: the WUR signal should not only contains the node to wake up, the LR relay, but also the intermediary WUR next-hop required on the underlying sub-path. Each device needs to know the WUR (sub)path towards the sink and each relay node to wake up on it, at least its first hop. The forward LR path can be then enabled at the cost of at least two information (up to three if the sink is not implicit). In the same vein, it is not only necessary to carry the best LR distance information via the LR relay in the signaling packets but also the WUR routes or at least next-hop towards it (to correlate the two information). We will see that we can deal with that issue relying on the LR radio to share iteratively the WUR neighborhood.

Several objectives are possible for computing the LR paths made of WUR segments, e.g.:

- First minimizing the WUR distance, then the LR one, or rather;
- the opposite: first the LR distance, then the WUR one.

Looking more generally at the overall multi-metric problem, these two objectives give the two extremities of the Pareto Front (being here only two dimensional). The second extremity of such a front, the most interesting problem with best LR paths first, is denoted *WUR-LR()* while the former is denoted *LR-WUR()*. Looking at the size of such a Pareto front (its range in particular) is interesting to understand the diversity of paths in typical deployment.

We can rely on our former notations to look at such problems and formulate them on the same and unique graph: for this purpose, let me define a multi-metric graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ such that $\mathcal{V} = V_2 (= V_1)$, $\mathcal{E} = E_2 \setminus E_2'$ and $w : \mathcal{E} \longrightarrow \mathbb{N}^2$. An edge does not exist in $\mathcal{E}$ while it is in E_2 if there is no path in G_1 connecting the two extremity of the edge in E_2 (it is then in E_2' by definition[11]), and w is such that $\forall e = (u, v) \in \mathcal{E}$, we have $w(e) = (x, y)$ with $x = w_2(e)$ and y being the best WUR distance of the shortest path (if any) between u and v in the graph G_1. Note that $E_2' = \emptyset$ if G_1 is a connected graph but we may add a constraint on the paths in G_1 to restrict $\mathcal{E}$ to most interesting cases, e.g., in order to avoid too long wake-up subpaths, one may enforce that they should exhibit a distance smaller than a given threshold to be considered in $\mathcal{E}$. This constraint does not hamper the existence of a feasible path in any cases (the WUR graph remains connected and, at worst, related LR edges exist) but can tend to increase the LR path length on the other hand. The first assumption is required otherwise an existing LR path can be lost because of the constraint (but feasible paths are still granted with our solution). In such an unified graph, our aim is not to solve the general multi-metric problem: we target the WUR-LR() problem, that is shortest paths minimizing first the LR distance. If there exists multiple best paths having the same LR distance in $\mathcal{G}$, among them WUR-LR() opts for the one minimizing the WUR distance as a secondary objective (other constraints or an additional arbitrary tie break may also apply).

[11] This situation cannot occur with our assumptions but our proposed protocol remains correct without as it does not return any path in such a case.

On Fig. V.6, we can illustrate our objectives and the challenges one needs to address. We assume a sink towards which each node computes and deploys its own path(s). LR paths towards the sink are given in orange (initially via two relays, 5 and 4 proposing both a LR distance of 1 in term of # hops towards the sink) while WUR paths are given in dotted black. One can observe that the range of the WUR path being smaller, several hops are necessary to reach and finally wake up the targeted LR node. The goal is then to find the best LR path that can be actually supported by an existing underlying WUR path, i.e. the shortest possible over all the best equal LR paths but also possibly subject to some constraints (e.g. two or three WUR hops among relays typically). Here we have for example a best distance of $(2, 5)$ from node 1 towards the sink looking at the problem in number of hops respectively for the LR and WUR technologies. Node 1 uses the LR path via the relay 4 to reach the sink, and it relies on subpaths $2 - 3$ and then 0 to reach the sink. The routing information for 1 to reach the sink can be summarized as a quadruplet $< (4, 2), (2, 5) >$ where the first couple provides the next-hop and the second the attached distances towards the sink of LR and WUR technology respectively. The WUR distance of 5 is known from 1 if the LR relay 4 advertises it: it allows to tie-break on the second objective even if only the best LR relay is useful for the forwarding.

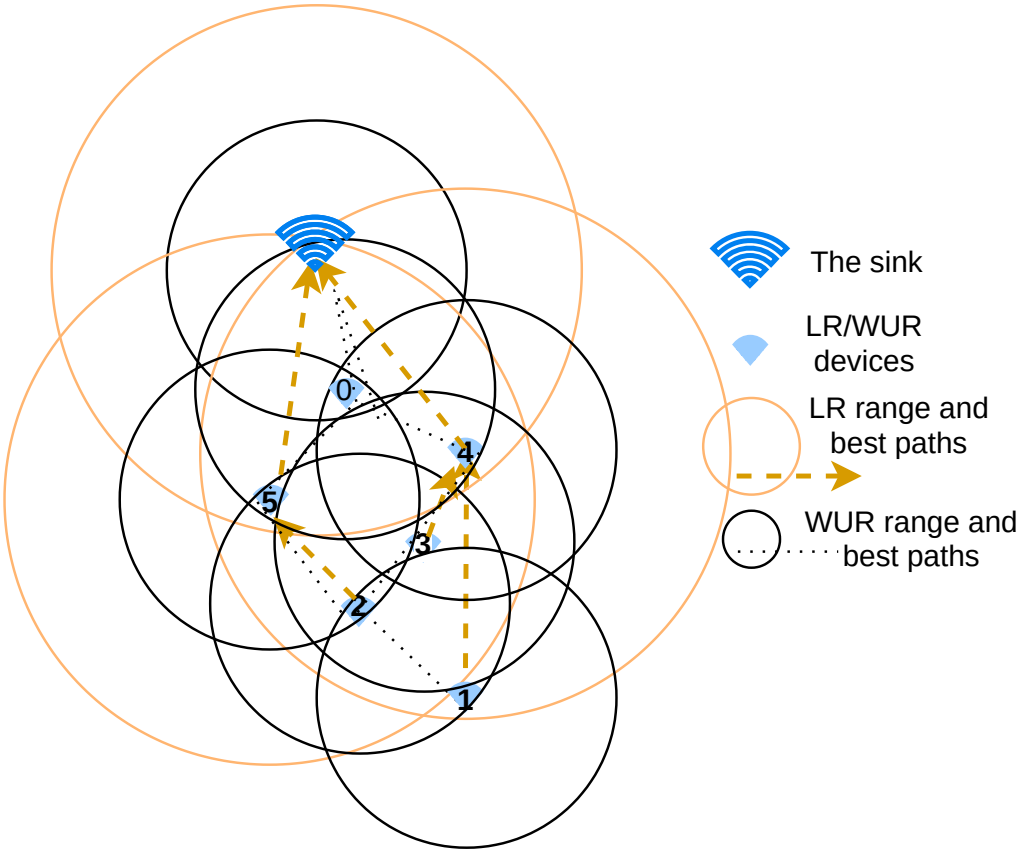


Figure V.6: Best paths are not necessarily aligned on the two metrics. Here the best LR path used by 2 (with relay 5) is not the same as the best one that 1 aims to wake up via 2. At the data-plane, node 1 should notify to 2 that it aims to reach 4 for its LR path, otherwise its path would be deflected by 2. At the control-plane, node 2 should relay WUR distances towards all LR relays in the k-hop neighborhood independently from its best LR path(s) towards the sink.

The unfied graph $\mathcal{G}$ is defined this way to enforce the deployment of LR paths according to the best available underlying WUR paths, the latter must exist under certain conditions (to ensure connectivity) and are stored with their best distances to be extended in the SPC algorithm. Paths towards the sink are computed using best LR relays based on the top of best WUR paths among them. More generally, the unified graph $\mathcal{G}$ also enables the formulation of multi-metric problems considering WUR paths in G_1

as the basis to deploy LR ones in G_2. With no (good enough) WUR path between two adjacent nodes in the LR graph G_2, there is no need to consider a link between them in $\mathcal{E}$: the LR edge is considered as not feasible, i.e. it is not deployed and the two nodes ignore the one hop LR link between them to communicate as no feasible WUR path satisfies the constraints (while they may indirectly using a longer LR path having good enough sub-WUR paths).

To design an efficient distributed routing algorithm solving WUR-LR(), we will rely on the notion of candidate paths: for each neighbor and for each radio, a node records the best advertised paths (towards the sink for LR and towards relays with the WUR metric) and cross the list of LR candidates with WUR distances towards them to validate best available LR path verifying the constraints on the WUR metric, and eventually advertised new best paths if necessary. While the LR propagation can simply follows the same principles as RPL, it is not the case for the WU radio. As already discussed, we face technical challenges due to the poor capabilities of this secondary WU radio: to deploy the forward path, we aim to rely on only two addresses in a WUR message at maximum, typically the address of the intermediary WUR receiver and the targeted relay node to wake up. The receiver looks at this target and searches for its (best) WUR path towards it and forges in turn a WUR message with the same relay target to its first WUR hop in that path (that becomes the next receiver applying the same routine). When the message reaches the targeted LR relay (to wake up its LR capacity radio), it then looks for its next best LR relay (a valid LR neighbor – except if it is the sink itself) and the same process is repeated up to the sink.

In order to enable such a process, it requires each node the ability to correctly compute the best LR paths as sub-sequences of WUR paths. Looking lexicographically at both first hops (LR best relay, and then WUR best next-hop) is then enough to wake up intermediary LR nodes assuming the required knowledge to be available. While each node is already required to know its best LR path towards the sink, in order to optimally solve WUR-LR(), a node also needs to know its best WUR paths towards *all* LR relay nodes.

Finally, note that the routing algebra in use for deploying such paths is more complex than what it may look at first glance: while the use of two entangled metrics does not hamper its overall monotonic property (strictly increasing) and so guarantees the feasibility of the solution (i.e., the routing scheme converges), the fact that the LR paths towards the sink are made of intermediary WUR nodes can induce the same effect as applying filters in BGP: the distributivity of the path comparison on the path concatenation is not granted as intermediary WUR nodes are also LR ones that can take their own selfish decisions regarding the sink, possibly diverging from the upstream nodes using them in their LR paths. Although monotonic, the routing algebra in use for deploying the global routes towards the sink is thus not necessarily isotonic when considering underlying WUR next-hops used between LR relays. Looking at the example in Fig. V.6 shows why nodes may disagree in terms of best paths: they can use a next-hop in their WUR path that prefers a distinct LR path. Here while node 1 uses 2 to reach 4 as its preferred LR relay with the WUR path $(1, 2, 3, 4)$, its next-hop, node 2 prefers node 5 as its LR relay and the traffic may be deflected from its globally optimal route if the destination 4 is not provided and forced as the LR relay within the wake-up messages from 1 (to wake up 4 via 3). To this end, the sub-path $(1, 2, 3, 4)$, or at least its distributed next-hops, need to be known from intermediary nodes, that is not only the best local LR paths, and their related sub WUR paths, towards the sink should be signaled, but also the best WUR paths towards all LR relays, otherwise node 1 cannot reach its global optimum (as 2 may not advertised the sub WUR path towards 4 because it does not need it for its own local LR traffic to reach the sink). Since links in $\mathcal{G}$ are not required to be simple directed physical edges regarding the WUR technology (some may rely on more than two hops), intermediary WUR nodes may disagree about the LR paths as their local optimal decisions can differ towards the sink. In such cases, the strictly increasing nature of the algebra allows to avoid diverging control loops but for reaching the global path optimality, it is necessary to advertize best WUR paths towards all destinations independently from the sink such that nodes can safely enforce their optimal LR decisions within WUR subpath. Finally, either only one unique globally consistent path may be deployed (e.g. using lexicographical arbitrary tie-breaks) or all equally ranked can be exploited as with ECMP. Since it increases the number of forwarding states to store, we may rather opt for the first option to mitigate the forwarding complexity and simplify the discussions.

V.3.b Designing Multiple Two-Radios Routing Protocols

To study a large range of options, I envision several modes of operations having similar objectives:

- The ideal one: assuming the whole control knowledge (LR + WUR at the cost of a consuming signalisation), compute prioritized multi-metric routes, i.e., WUR-LR() ones;
- The constrained one: limiting the flooding of LR relay distances with WUR messages to k hops, e.g., $k = 3$. The LR radio can be used to share this information;
- The deflected one: only constructing local best paths without propagating all WUR control messages (for example, node 2 can implicitly and only forwards its best LR relay via 5 as only its best LR path triggers the advertisement);
- The basic one: rely only the best WUR paths towards the sink; it induces longer LR paths but with the most efficient deployment.

In all cases, the objective remains the same (that is WUR-LR()): deploying the shortest possible LR routes towards the sink (with a more or less consuming control plane), and such routes verify first one constraint and satisfy one secondary objective to both limit the churn and the number of forwarding states:

- between each pair of subsequent LR nodes in such a route, there exists an underlying WUR path made of less thank k hops. This path should be available to efficiently wake up nodes in the shortest LR route;
- when there exists several shortest LR routes from a node to the sink, select the one minimizing the number of WUR hops as a secondary objective (another consistent tie-break may be use to ensure the unicity and to reduce the number of forwarding states to be stored).

The difficulty does not lie in the path computation problem in itself (which is polynomial) but in the technical limitations as efficiently signalling/collecting the necessary information is not an easy task with a weak radio such as the WUR one. In theory, each node needs to know its best WUR paths towards all others. To mitigate the induced WUR churn, we can limit the flooding resulting from recursive broadcast to a given vicinity (e.g. with the constraint of k hops), and design a distributed protocol that mostly rely on the LR radio to carry the topological knowledge, both for the LR and the WUR metrics (i.e. respectively the best LR distances towards the sink and the best WUR one-to-one distances among k-hops WUR neighborhood).

To illustrate the kind of solution we will evaluate, let me build a protocol sketch as follows (for the near optimal constrained solution). Initially, only the sink advertises its own destination with a distance of 0. The originality regarding RPL or similar constructions, is that this protocol relies on both technology and sends two kinds of messages: the ones about the LR radio, the others about the WUR one. While both topological information can be transmitted on the LR medium, at least one initial WUR flooding is required to learn initial WUR connections. The two kinds of messages contain similar information, the identity of the sender and a distance. Since the WU radio is very limited (and may no support the limited flooding only envisioned here to ease the understanding), the following naive version is extended afterwards.

When receiving a LR message $(v, dist)$ **(the sink can be implicit in LR messages) from** v**, a node** u **reacts as follow:** if u knows a WUR path towards v, it then looks whether the proposed path via v towards the sink is better than the best currently known LR one (thanks to $dist$ and the LR weight $w_2(u, v)$ that it compares to the best known path – this LR weight is the first element of the couple of weights in the unified graph $\mathcal{G}$). If it is the case, u replaces its best LR path in its forwarding table considering the best known WUR path towards v, and advertises (flood with the LR radio) a new LR message $(u, dist + w_2(u, v))$. Otherwise, it ignores it. If there is no WUR path, u keeps the information (the route distance and the relay id) in the candidate LR table for possible usage latter. If this is the first time a LR path is validated (or there is a change in the WUR topology), u advertises (flood with the WU radio) a new WUR message $(u, 0)$.

When receiving a WUR message $(id, dist)$ **(there is no sink here the destination is** id **itself) from** v**, a node** u **reacts as follow:** if the proposed WUR path is better than the existing one, u installs it in place of the previous one. It then looks whether there exists LR routes in the candidate LR table that can be enabled and now possibly better than previous one. If so, u continues with the LR procedure and finishes then the WUR one with $v = id$. Note that a WUR path is installed towards id (via the WUR next-hop v), not towards the sink. Moreover, if $dist$ is lower than a given threshold (e.g. 2 with a hop count metric), u advertises (flood with the WU radio) a new WUR message $(u, dist + w_2(u, id))$. Otherwise it does nothing (u stops the flooding to limit the storm area).

In order to limit the WU radio consumption and since each WUR message has to carry three information in this naive model (v, id and $dist$), our proposal is to modify the WUR signalling using the LR radio to carry this information. However, a single initial operation is required: **each device floods (only) one raw signal with its WU-radio containing only its identity**. From this information, each node knows its set of one hop WUR (incoming) neighbors. This information can be then shared on the LR radio: in this message each LR node id advertises it WUR neighorhood, i.e. a set of couple $(dist, v)$ towards id where v is a WUR neighbor at distance $dist$ (of 1 hop initially). Then, in turn (when receiving such a message), each LR node id can do the same with WUR best distances from 2 up to k-hops neighborhood. Such messages $(dist, v)$ sent by id can be handled as the $(id, dist)$ from v in the previous model (if v is a WUR neighbor).

I envision to propose this topic to a master student. The idea is to start implementing and evaluate several routing options to compare them on two criterion: the path stretch and # messages (with both technologies). In summary, I aim to investigate this wireless field with multi-radio technologies, possibly considering novel traffic models and original assumptions/correlations between metrics. The wireless medium makes the connectivity problems more challenging than with wired connexions not only because of their capacity constraints. More technical issues (e.g. broadcast directed links) and difficulties (e.g. collisions) occur to correctly control the medium and the potential mobility, and connectivity quality in general. This area is exciting as it also offers many technical challenges in terms of measurements and monitoring (as the link weights are very dynamics).

V.4 Topology Discovery and Analysis, Embedded Measurements & Anomalies Detection

Topology discovery and performance measurements are the first keys to provide reactive networks able to adapt themselves to traffic and topological changes. Current networking services like industry 4.0, 4K gaming, and large scale video-conferences have high requirements regarding both their availability (the network should remain permanently reachable even in case of sudden failures) and the constraints imposed by their operational needs (e.g. low latency and delay variation or enough bandwidth). Efficiently detecting and advertising such characteristics, their changes and anomalies is critical to enable fast and proper decisions. As a baseline to finely adapt to variations, discovering and continuously mapping network devices and their links are necessary inputs to maximize the network efficiency and reactivity.

V.4.a Comprehensive ISP Mapping and Measuring Segment Routing

In the following years, I will continue to explore the field of Internet topology discovery. Its architecture evolves slowly but surely: with the more flexible hardware deployed nowadays and new routing technologies like SR, its logical structure offers more diversity. The general question remains the same, that is *How to map the Internet?*, but adapted to recent routing features and changes to reveal more properties:

Research Question

How to Probe the Internet to Efficiently Reveal its Properties, and its Routing Features like Load Balancing and SR Paths?

In the field of IP measurements, I aim to continue to improve the state of the art with qualitative ISP intra-domain mapping (at the router level). This is a need I often meet when performing experiments to evaluate the routing algorithms I design with my co-authors. Not only the physical structure of the router level graph matters, but also its logical valuation, and all related information (e.g., the geographical location of nodes, the kind of OS and features in use). With Benoit Donnet and Emeline Maréchal, we already explore new approaches for such probing with a new tool called Anaximander [MMD22].

So far, researches on Internet topology discovery have focused on efficient data collection (e.g., Doubletree [96]), alias resolution [192], ISP mapping (e.g., Rocketfuel [319] and `mrinfo` [PMDB10]), or on Internet modeling [265]. Despite being man-made, much of the Internet is still hidden and unknown because it does not consist in a single authoritative entity. Each AS has its own commercial practices, physical infrastructure, and logical design (in particular its routing and TE strategies). In order to deploy a specific routing strategy (from best-effort traffic to more complex TE strategies, including fast re-routing), ISPs generally elaborate more or less complex strategies to control packet forwarding according to a given set of network metrics, related constraints, and technology [274].

Blindly sampling (a subset of) the Internet is not enough to reveal and discriminate such specific topological and routing patterns, or generic ones if they are any. Instead, in order to conduct relevant TE and IGP performance evaluations [137] and showcase the performance of a given routing proposal (with simulations or analytical models), it is more suitable to rely on distinct ISP maps offering various realistic situations, rather than using an arbitrary chunk of the Internet. To offer a framework for reproducible realistic experimentation, one needs to collect intra-domain networks of distinct natures and we thus aim to develop modern, accurate, and advanced topology discovery tools able to skillfully capture the reality of the Internet, in particular considering its atomistic technical nature. In particular, we look for efficient probing designs able to reveal the specifics of any given intra-domain router level map.

While Rocketfuel [319] topologies have been the de facto dataset in use for nearly two decades, we argue that both the resulting topologies and the underlying probing methods are now outdated. Indeed, the Internet structure and practices have evolved over time, and new refined measurements tools have become available as well [342] or [LVM+19]. In this work, we pursue the same objective as Rocketfuel formerly, i.e., to map specific ISPs at the router level, but we revisit the following challenge in the current Internet context: *Can we infer ISP router level maps with a reduced probing budget, but without hampering the resulting topological coverage?* Designing efficient probing campaigns is indeed essential to speed up the measurement period and so mitigate forwarding anomalies (e.g., routing changes [351, 366]), and the effects of adaptative filtering (e.g., rate limiters [281, 154]). Otherwise, the data collected may not be consistent or suffer from poor coverage. After having pointing out Rocketfuel limits for capturing nowadays Internet maps, we start by revisiting its successful components when their efficiency is still valuable. Then, we develop our own strategies and evaluate them relying on a large and recent dataset to conduct realistic simulations and support our assumptions.

Overall, this ongoing work is an exploration of best up-to-date strategies to control the probing budget. We now aim to deploy a real tool that fully handle LB and infer IGP weights of the targeted AS thanks to the collected forwarding information. To the best of my knowledge, there does not exist a model, and its companion tool, able to properly exploit load balancing information for inferring IGP weights (the seminal paper on this topic only considering standard traces without taking into account ECMP [224]). Existing tools [340, 6, 341] mostly focus on the discovery, not on the analysis. Moreover, the traces used to reveal the load balanced paths are launched without pre-calibration based on results of previous LB aware campaigns. That is, they are blind of previously collected information while they rely on usual conditional statistical methods. I envision to improve existing LB computation and probing techniques to efficiently deduce the IGP valuation in use in a given AS. Indeed, the equality among route distances translate into constraints more accurate than their inequality. I aim to design an efficient constraint pruning mechanism to determine the minimal set of constraints to be considered with respect to this advantage. Moreover, the technical knowledge acquired in section III.3 will guide my research in this area as well as the statistical analysis of traces [180]. One of remaining challenge is also to discriminate TE paths from best-effort ones to not bias the system of constraints.

Although various techniques are available for wired networks (such as active probes or routes analytics), topology discovery is even more challenging for wireless networks for which the topology is generally smaller but can be complex (in the case of multi-hop networks) and versatile. It is a challenging area

where the number of nodes is not known (the exact topology may be too variable to be accurately tracked), exhibiting a high density and highly dynamic links not always reliable [184]. I would like to investigate this area with the members of my group specialized in wireless technologies. I think there exists many research opportunities in this field, in particular considering the recent hardware and software evolutions [39].

Another direction I intend to investigate is the study of SR domains: how frequent they are and what usages are being made with their tunnels? Over the last years, many AS opt for this technology with both IPv4 (with MPLS or IPv6 encapsulation) or IPv6 networks. While the later looks to be the most promising option [109], extracting this information from IPv6 `traceroute` like data has to investigated. I intend to do so to understand if one can observe an encapsulation pattern by exploiting the available error messages considering IPv6 traces. When the MPLS data-plane is in use for IPv4 traces (and not the IPv6 one), peculiar patterns are expected to arise. On the contrary to the labels expected with LDP as control-plane, the MPLS labels used with SR should be persistent and in a given range along the SR/MPLS tunnel (with MPLS tunnels built with LDP, labels are proposed in an unordered manner). More multi-labeled packets may also be seen in the wild. I aim to explore such an analysis by deploying specific measurement campaigns. Tracking the SR evolution and deployment both in the IPv4 and IPv6 spaces is a difficult challenge as no specific tools look available so far. More generally speaking, few works specifically focus on the benefits of SR deployment [251]. This is an area I envision to investigate in the next years.

V.4.b Embedded Measurement Primitives

In this section, I will develop a project we aim to investigate with my group. As current hardware becomes more flexible than ever, in the space of network monitoring, we want to study this question:

Research Question

How, and for which Usages, Embedding Measurements and their Outcomes Directly within the Data-Plane?

Most current monitoring approaches face several limitations as they are often designed and deployed as a third-party management plan. Interactions with control and data-planes thus lack of reactivity and may fail to adapt to the real local and ground up-to-date situation. Our aim is to enable most reactions to take place directly at the data-plane to avoid such limitations. In more details, the research problem we want to tackle is the following: *what are the opportunities offered by programmable hardware in terms of monitoring and control, and how to efficiently exploit such novel features in a global architecture?*

With my group, we want to propose a set of primitives for topology discovery and performance measurements whose some can be directly handled at the data-plane both in radio and wired networks [255]. While long-term decisions can be globally piloted with algorithms and mechanisms respectively implement at the control and management plans, we argue that short-term ones can be directly piloted or provisioned within the smart data-plane. Failures and micro-congestions are typically the kind of events that are natural candidates to be evaluated and fixed at the data-plane. In wireless networks, many link quality metrics exist in the literature [361], *e.g.* Link Quality Indicator (LQI), Received Signal Strength Indicator (RSSI) and Packet Reception Ratio (PRR), and are required to be estimated in order to handle the medium versatility. The goal is to design analysis and detection methods supported by embedded measurements. Simple counters look to be enough enough to track for these kind of events (e.g. tracking the queue sizes and transmission success rates).

Many decisions can occur solely in the data-plane without burdening other planes if it is sufficiently well provisioned. In this project, our goal is to study and take benefits from the opportunities offered thanks to smart data planes:

- what are their inherent limitations?
- how to disseminate measurements to split decisions at different planes and among many distributed devices?

- how to implement efficient, correct, and near complete algorithms in such a challenging environment and various usages?

Traditional monitoring approaches do not usually consider in-band measurements and require to inject probes within the network [328] while I envision here to design and deploy new measurement primitives and their associated mechanisms. Our objective is to use passive measurements on the real traffic in transit, enabling ground and fast decisions with marginal overhead. This novel approach is adapted to all kind of networks including wireless networks, in which active measurements may have a significant negative impact on ongoing communications and energy consumption. Such fast decisions from the data-plane can be temporary while the control- and management-planes would compute more advanced long term decisions whenever necessary.

In [133], the authors discuss several modern approaches and models relying on data-plane monitoring and argue that many operations do not require to be delegated to the control-plane nor the management one. They show it is possible to implement basic statistical analysis with simple counters even when the language is inherently limited: with the use of approximations, the performance evaluation can take place as close as possible to the forwarding engine. Continuing in this direction in a good option as it is then possible to provide efficient means to monitor the network at the location where the traffic actually flows. In particular, sampling slice of traffic can be helpful for various usages like fast re-routing, anomalies detection and LB. We plan to use scalable algorithms that dynamically adapt the sampling rate and use machine-learning techniques and statistical analysis to predict the network behavior and also identify operational patterns to limit the amount of data collected.

In the case of anomaly detection [18], in particular for BGP, prediction can be offered as a service using graph based features and possibly machine learning techniques [165]. In addition, and for example, with prediction and reconstruction of time series (e.g., ARIMA, machine learning) identifying seasonal patterns (like in [GFW+19]), or global trends (per link, per area, per domain), the controller(s) should offer a consistent view of the network infrastructure despite the unreliability of the control plane, the measurement inaccuracies, or the impossibility to retrieve a high-frequency sampling data-set. By reconstructing a globally consistent view, one can predict the short, mid, and long term evolutions.

Sharing the performed measurements among device and distinct planes is one of the most challenging tasks of this project. Since they may lead to a significant amount of data and diverge, we argue that precise analysis should be first locally exploited while only general trends can be advertised at a larger scope within the network. For this, we should identify which features belong to which plane to avoid burdening the network with churn and too complex interactions. Distinct time-frames exist and adapting too frequently may lead to oscillations and a globally diverging system. We argue that the data-plane should enforce fast countermeasures or adaptations (e.g. react to a link failure) while the control-plane allows for adapting to a more stable shift of the network (new flows, new links, and operations possibly controlled at their admission in general). Embedding measurements within the data-plane is a great opportunity but efficiently splitting the distinct features and decisions inherent to each plane is challenging as each control loop should be correctly aligned with others in order to avoid inconsistent behavior possibly leading to oscillations. In summary, I aim to study such a problem and how it fits overall with the other ones developed so far using a distributed system perspective. The next and last section provides my naive vision of how relying on these more abstract models can help to design correct and performant forwarding systems adapting to challenging conditions.

V.5 Towards Self-Stabilizing Algorithms and Distributed Systems in General

This last section does neither introduce nor address a specific research question but is rather a discussion on the way I want to improve my scientific method. While the main objectives of my studies will remain to design a better Internet both in core and at edges, I would like to adopt a more formal and general analysis than what I have done so far. I actually think that distributed systems and networking practices should embrace their connections to improve the design and flexibility of current protocols. On the one hand (the one of distributed algorithms), either assumptions are often too strong for real deployments (e.g. with synchronized models, and only local views) and ignore practical implementation details or, on

the other (the one of network protocols), the solution is not properly designed to ensure consistency in corner cases (e.g. BGP).

Routing systems are distributed objects by essence. Their convergence and performance can been studied within a distributed theoretical framework like the one developed in the work of Griffin et al [84]. Proving the routing comes with many advantages for ISP as they can mitigate their troubleshooting efforts. I think this is more than a trend but the only valuable long term future for algorithms and protocols.

Auto-stabilisation under various assumptions (and adversarial model) can also offer a nice framework to study routing problems [93, 23]. In particular considering a wireless oriented adversarial model. Maintaining an efficient connected structure is a challenging area I envision to investigate. Notably in the context of my project about multi-radios for energy efficiency I have presented so far. Naturally, my data-plane oriented convergence project also falls in this area, as it can be translated to a stabilisation problem that I aim to enrich with more challenging conditions (malicious agents, several link of node failures and erroneous configurations) looking at most similar proposals in the field.

More generally speaking, I aim to collaborate with the members of the team implied in the last, and recently introduced, research field of my group (and its related topics), Quentin Bramas and Anissa Lamani. One of the topic they work on is about exploration and gathering problems with weak robots [113]. Such problems lead to interesting impossibility proofs and offer a very exciting algorithmical playground (looking at the simplest and most efficient conditions to reach a given objective according to a given set of assumptions). However, each sub problem (given by the numerous underlying parameters like the kind of space considered) looks very specific to each sub-cases and I aim to investigate the opportunity to look for more general frameworks. A last example of algorithmic problems I aim to look at in the area of graphs and distributed systems, is the stable matching problem [156] as it seems challenging under various assumptions [228]. While these two last areas of problems are not directly related to networking applications, I think they share similarities with my current activities. Not only related to the background in use (graphs and distributed asynchronous models), but also on the inherent properties (e.g. adaptability and optimality) on which rely the related algorithms. I expect that these similarities will inspire my research in networking algorithms and give me a larger perspective to solve practical problems.

Chapter VI

Conclusion & Summary of the Perspectives

Computer Networks play a critical role in nowadays societies. Sharing information has become more simpler than ever thanks to the development of the Internet and the numerous services it offers. However, its structure, the pile of protocols in use and their specificities are often too complex regarding what they should be, especially when looking at its architectural components, their interference, and underlying routing protocols co-existing to enable a broad variety of applications requiring distinct needs. Recent requirements in terms of communication reliability, latency, throughput, and quality of service in general, are especially high, typically asking for no visible service disruptions damaging the experience of the end-users. **The two main challenges I aim to tackle lies,** (i)**, in the ability of the network to quickly reconfigure itself in case of changes, whatever the service requirements and the nature of the reconvergence, and,** (ii)**, to offer efficient measurements primitives and distributed cooperation to finely monitor, understand and possibly control the network**. They translate into several research questions and related problems that I have investigated, revisited and at least partially solved in the previous chapters.

In this manuscript focusing on my activities in the field, we have discussed several means to achieve the stated objectives. In particular, we have studied algorithmical contributions allowing for faster and safer reactions in case of failures or network operations (e.g. maintenances, reconfigurations or TE optimizations in general). Preventing related disruptions and their side effects leading to network anomalies (e.g. forwarding loops) is indeed possible and enables a graceful routing convergence, either free of anomalies or short and efficient enough to mitigate most issues (by simply minimizing the downtime period if any). This can be achieved both for the internal and transit traffic when relying on BGP next-hops (the gateways at the border of ASes). We develop several frameworks taking into account the interactions between the couple of routing protocols and features that are entangled by design. We also look at orthogonal criteria when it comes to premium flows having specific TE requirements. In particular, we propose an algorithm who is able to compute DCLC paths in the operational context of SR domains.

In all these algorithmical works about improving the existing routing and forwarding protocols, not only we try to design provably correct solutions, but we also pay a special attention to their worst case theoretical complexity and practical efficiency. I particularly appreciate to look for the best achievable networking trade-offs between optimality and efficiency, seeking for the simplest appropriate data-structures and algorithms to solve graph problems. In all the solutions described in Chapter II, either we have revisited them as with the combination of *TBFH* and SR or proposed extensions considering data-plane based updates.

On the other hand, I am also interested in measuring and monitoring the Internet in order to assess the performance of current and future routing protocols. The first objective I have looked at is more oriented towards structural properties than the second: my initial activities in the field are mainly related to topology discovery and analysis, e.g., about understanding hidden corners of the Internet (like invisible MPLS clouds) at different scales; my activities in term of network monitoring are more specific to fine-grained performance measurements requiring some privileged access within the network of interest. My

experience in both these sub-areas is useful to design efficient routing protocols and reciprocally (that is, knowing current routing systems and their advanced features is useful to measure them well).

Designing valuable measurement platforms and measurement campaigns is technically challenging as it indeed requires a detailed knowledge of common routing practices and current guidelines. We have proposed several original techniques to detect, analyze and understand issues relevant to the first chapter like forwarding loops, typical patterns of downtime periods or routing detours. One of the aspects that I appreciate in this field is the technical challenges to overcome, and the difficulty in general to efficiently calibrate the probing tools in use to collect the data in the wild. In Chapter III, most of our contributions can be improved, for example looking for a comprehensive tool extracting router-level AS weighted and annotated maps. Data-plane based measurements is also an exciting area of research to enable embedded control and deploy new models to monitor and adapt the network. I will study current available architecture abilities to improve the state of the art in such measurement infrastructures.

Most of my works took place in the application context of wired core networks like ISP or Enterprise networks. However, I am also interested in other kinds of computer architectures like wireless networks in general, e.g. collection of IoT devices (possibly sensing and interacting with the physical world and forwarding data and control packets). In this context many other technical challenges emerge as such wireless networks are generally more constrained than wired ones. In particular, with the Internet of Things, not only the energy is limited but also computing resources. Within radio networks, while layer-1 and layer-2 are generally totally different from what it is used in wired networks, even at layer-3 where more similarity exist, routing features are not deployed as they are in wired networks for many reasons whose some are not only new constraints.

For example, since services and usages differ between these two classes of networks (wireless edge smart cities do not operate the same as large scale wired transit networks), it also results in distinct traffic patterns: in IoT networks, the application model often leads to converge-cast architectures rather than standard one-to-one communication existing at the macro Internet scale (e.g. with the basic unicast model). All these new requirements and constraints translate to other challenges that are exciting to tackle. In particular, I am interested in energy saving and, more recently, in the privacy of the sensitive data exchanged in such networks (cyber-physical systems of this kind can be potentially intrusive if not well designed). Indeed, we have proposed new architectures and original secure data model dissemination based on aggregation to mitigate leaks in the confidential data transmitted by sensors (typically collecting personal information like habits of inhabitants of smart cities). We have also looked at policies verification and analysis with metagraphs. Data privacy is an important topic to secure personal and sensitive information. Looking at such challenges is very interesting for societal reasons as they directly concern the end-users we all are.

The proposals made in Chapter IV consist in my preliminary works in the field. I envision to extend and revisit the already investigated privacy projects with refined models and better algorithms. For example, with the use of metagraphs, one can propose new definitions for checking metapath dominancy and better pinpoint conflicts and superfluous rules in case of distributed deployment. Also, I envision to better analyze our multiset merging strategies for in-network aggregation: my will is to design a routing algorithm auto-defining a stable tree able to merge all the data with respect to policy requirements. I aim to continue in this direction and propose better tools to support this novel area of my research.

Last but not least, this manuscript not only summarizes my main achievements in the field, and their respective refinements or perspectives, but also describes my plans for the future. While the short term agenda provided in Chapter V looks crystal clear with already concrete proposals in my favorite routing topics, my mid and long term projects are less mature and depend on recent progresses and new trends regarding the closest projects in the literature. To start as a Ph.D. supervisor, I expect to guide my first Ph.D. students on the topics I prefer, that is designing robust routing algorithms. More precisely, I have shown how one can benefit from smart data-plane routines to enable maximum flexibility in terms of network reconfigurations and convergence. Sec. V.1 already provides the main outlines of this work, also for node and transit traffic protection in the following sections, and my research group will soon have access to a convenient testbed for evaluating such modern models.

I would also aim to continue to explore other playgrounds like networking measurements and IoT networks to respectively address the problems of monitoring new technologies (like SR or control programmable/software networks) and natively consider multiples co-existing radios (having their own char-

acteristics) to preserve the wireless network life time. I expect these latter topics to be explored with Master students and eventually Ph.D. thesis projects in the next years. Sec. V.4 and Sec. V.3 respectively provide my first attempts to study and address these questions and problems.

More generally speaking, my long term objective is to progressively switch to the area of distributed systems but continuing to tackle practical networking problems. With more complex adversarial models, correct and quick routing reconfigurations become more difficult to solve as well. Most usual networking and routing proposals may be revisited under the light of more formal foundations in order to mitigate bugs and anomalies, for example by considering the more general byzantine fault model. As an even more long term perspective, with my new colleagues in the field, I aim to study the class of self-stabilizing algorithms and look at classical algorithmical problem like exploration with weak robots and stable matching in several contexts.

Chapter VII

Version française abrégée

Ce dernier chapitre présente une version très abrégée du présent manuscrit dans sa version française. Il s'agit donc d'un résumé informel de mes activités et projets de recherche.

Introduction

Internet est un réseau de domaines autonomes (chacun a ses préférences et ses propres intérêts) mais aussi inter-dépendants avec pour seul objectif véritablement commun d'assurer une connectivité globale dans la mesure du possible. Chaque sous système, c'est à dire domaine, est véritablement autonome : il est libre de ses choix politiques et financiers et dispose de plusieurs outils ou techniques pour privilégier ses propres critères locaux et pratiques (pour son trafic interne, comme externe). Par défaut, pour le trafic dit *best-effort*, les paquets IP sont acheminés au saut par saut grâce à des algorithmes et protocoles de routage distribués comme Open Shortest Path First (OSPF) ou Intermediate System to Intermediate System (IS-IS) pour la connectivité intra-domaine, et BGP pour les échanges inter-domaine. En intra-domaine, il existe également des technologies assez facilement déployables mais plus avancées comme MPLS ou SR qui permettent de contrôler plus finement les routes empruntées; respectivement en balisant préalablement la route et/ou en indiquant dans le paquet le ou les détours à utiliser plus ou moins explicitement (*loose source routing*). Alors que MPLS pré-déploie ces routes en amont via des protocoles de signalisation annexes (comme LDP pour l'extensibilité ou RSVP-TE pour l'ingénierie de trafic), SR repose sur un paradigme de routage par la source exploitant la signalisation inhérente aux protocoles de routage intra-domaine. La source peut alors indiquer directement dans le paquet les sauts intermédiaires par lesquels le paquet doit passer : cette approche est qualifiée de relâchée (*loose*) mais il existe également une approche dite explicite (*explicit*) où chacun des liens à utiliser ou partie est précisément indiqué dans chaque paquet. Dans le cadre de l'ingénierie de trafic en particulier, ce type d'approche est plus facilement extensible que l'utilisation traditionnelle de RSVP-TE ou MPLS-Fast Re-Routing (FRR).
Dans mon travail de recherche, je m'intéresse tout particulièrement à la période dite de **convergence** des protocoles de routage. C'est à dire au temps nécessaire – et aux anomalies y survenant (comme aux chemins de secours permettant de les contourner) – à la stabilisation de l'ensemble des routeurs d'un réseau IP suite à une modification (résultant de pannes, maintenances ou reconfigurations en général). Un état stable signifie que la totalité des équipements partagent un ensemble de choix cohérents, notamment en terme de prochains sauts de commutation : ceux-ci sont censés se combiner de manière cohérente pour former des routes optimales et stables de bout en bout. Avant d'atteindre cet état, les routeurs peuvent en effet disposer d'informations incohérentes résultant en un système distribué dégradé où surviennent des anomalies : boucles de commutation, chemins sous-optimaux voire absence de routes.
J'étudie ces phénomènes, et plus généralement les protocoles s'y rapportant, sous deux angles : leur mesure et supervision ainsi que la conception d'algorithmes de routage améliorant les performances d'Internet. Dans ce manuscrit, je présente dans un premier temps les solutions que j'ai proposées pour pallier aux limites des protocoles actuels (chapitre II), puis, dans un second, les plateformes de mesure que j'ai eu l'occasion de déployer pour superviser ou découvrir/quantifier les principales fonctionnalités de routage traditionnelles (chapitre III). Enfin, et respectivement dans un troisième et dernier temps, je présenterai mes activités liées à l'Internet des Objets et à la sécurité (chapitre IV), ainsi que les pistes de recherche que j'ai à cœur d'étendre et de développer dans les années à venir (chapitre V) pour continuer d'enrichir, étoffer et diversifier mes productions scientifiques.

Routage : chemins de secours, sans boucles et multi-critères

Le routage est la clé de voûte pour acheminer les paquets IP d'un domaine à l'autre. Localement et en interne d'abord, et pour le trafic de transit ensuite, les protocoles et algorithmes de routage sont les briques essentielles permettant aux routeurs de déterminer leurs tables de prochains sauts (plan de données) vers l'ensemble des préfixes destinations. Le contenu du chapitre II est majoritairement dédié à cette question : comment prévenir efficacement et correctement les changements de routage ?

Les algorithmes de routage calculent des ensembles de *meilleurs chemins* selon un ou plusieurs critères (par exemple la capacité des liens, leurs délais ou leurs bande-passantes résiduelles) alors que les protocoles s'y rapportant régissent la *signalisation* permettant aux routeurs de disposer de suffisamment d'information (correcte et cohérente) pour mener à bien ces calculs.

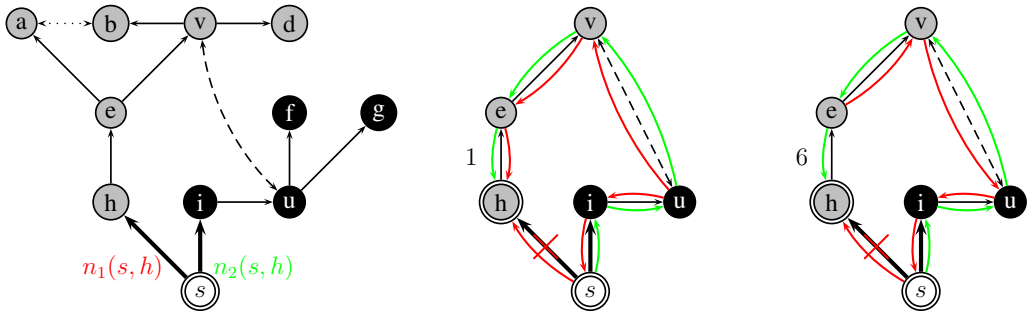
Les protocoles actuels dépendent de l'échelle à laquelle est opérée le routage : en inter-domaine, il s'agit de BGP, un protocole à vecteurs de chemin, alors qu'en intra-domaine, il s'agit d'OSPF ou IS-IS,

des protocoles à états des liens. Leurs temps de convergence divergent de plusieurs ordres de grandeur, et alors que le premier n'a pas l'assurance de converger – en fonction de sa configuration, la seconde famille converge à coup sûr grâce à ses propriétés d'*isotonie* et de *monotonie*.

Afin de pallier leurs limites respectives dans le cas de changements planifiés ou non, j'ai proposé avec mes doctorants et co-auteurs plusieurs solutions, en particulier des algorithmes de reconfiguration pour plusieurs types de modifications topologiques (pannes ou évènements planifiés, de lien ou de routeur, etc). Les trois sous-parties suivantes s'intéressent à trois cas d'étude distincts : des pannes de lien aux changements BGP pour traiter les événements internes en passant par la reconfiguration planifiée d'un routeur dans son intégralité. La dernière sous-partie traite quant à elle d'un algorithme de routage multi-critères dans le cadre d'un déploiement SR. L'ensemble des solutions décrites dans ce chapitre se basent sur la cadre formel qu'offre la théorie des graphes. Aussi, j'évalue nos propositions algorithmiques, en particulier leur efficacité en terme de temps de calcul, en analysant les données réelles ou supposées caractérisant les structures sous-jacentes aux réseaux IP.

Re-routage en cas de Panne de liens

Dans cette première partie (un résumé de la partie II.1), je revisite entièrement la publication COMNET 2011, [MFB+11], i.e. avec des algorithmes de calcul de chemins de secours directement dédiés au support de la technologie SR. L'originalité de notre approche réside dans sa capacité à calculer efficacement et exactement les chemins post-convergence sous forme de segments rapide et sans circuits, en particulier lorsque les poids du réseau sont symétriques. La figure VII.1, issue de l'illustration II.4, souligne les bases du problème traité ici[1] : comment calculer les meilleurs chemins de secours TI-LFA pour SR ? notre algorithme, TBFH, peut facilement retourner et encoder les nouveaux meilleurs chemins, les chemins post-convergence, en un seul détour au maximum si les poids sont symétriques. Soit s la source, i.e. le point de re-routage local, et soit h la destination considérées dans la figure VII.1 : comment protéger localement le lien (s, h) à destination de h (et de l'ensemble des nœuds gris appartenant à la branche grise de l'arbre des plus courts chemins) ?



(a) Arbitrary SPT decomposition (lexicographical) considering an uniform symmetric valuation ($\forall e \in E, w(e) = 1$)

(b) Pre-/post-convergence paths towards h with symmetric weights ($p_1^j(\cdot, h)$ vs. $p_2^1(\cdot, h)$)

(c) Pre-/post-convergence paths towards h with asymmetric weights

Figure VII.1: Soit le réseau donné en (a) : sa valuation est uniforme et sa décomposition en deux sous-branches de meilleurs chemins est représentée par les couleurs grise et noire. Avec une valuation symétrique (b), la protection du lien (s, h) est facile à garantir avec TBFH (un seul segment est suffisant, l'arc inter-branche, i.e. dit transverse, au pire) alors qu'en (c) un seul poids asymétrique (ici de e vers h avec un poids de 6 au lieu de 1) suffit à invalider cette propriété. Les chemins pre-convergence vers h sont donnés en rouge et ceux post-convergence le sont en vert. Si le lien (s, h) tombe en panne, l'arc transverse (u, v) n'a pas besoin d'être forcé en (b) : le segment vers le routeur u suffit en tant que simple segment de nœud. En effet, il est libre de toutes boucles de commutation (circuits avec alternances pre-rouge post-vert). En (c) avec l'asymétrie, les liens (u, v), (v, e) et (e, h) doivent tous être forcés à tour de rôle.

[1] Pour les explications formelles des notations, merci de se référer aux figures et chapitres d'origine.

Avec TBFH, les chemins de secours TI-LFA pour SR sont calculés de manière très efficace : au pire, l'arc transverse, ici (u, v), représente le premier segment à forcer si besoin (i.e. un segment d'adjacence au pire). Ce segment d'adjacence SR n'est nécessaire que si, itérativement, aucun des sauts x sur la sous branche de secours considérée (ici la noire) menant de s à u ne vérifie la condition LFA : $c(x, h) < c(x, s) + c(s, h)$. En cas de poids symétrique, cette condition est garantie au pire à partir du nœud v (à la fin de l'arc transverse menant à une autre branche par définition), sinon la séquence de re-routage peut nécessiter davantage d'éléments intermédiaires, comme dans la sous-figure VII.1(c).

Mes principales contributions sur le sujet sont les suivantes :

- un algorithme de calcul de chemins post-convergence : TBFH;
- des nouvelles preuves de sa faisabilité et de son efficacité avec les chemins de re-routage optimaux TI-LFA (FRR avec SR);
- de nouvelles primitives et propriétés pour l'encodage des meilleurs chemins TI-LFA.

Les nouveautés explorées dans le cadre de ce travail devraient prochainement donner lieu à de nouvelles publications : TBFH est l'algorithme idéal pour le calcul et le déploiement des chemins TI-LFA. En seulement deux itérations d'un algorithme des plus court chemins (e.g. Dijkstra avec une file de priorité efficace comme un tas de Fibonnacci), il est possible de connaître toutes les spécificités SR des meilleurs chemins post-convergence. Ceux-ci, les meilleurs, sont encodables avec au maximum un segment si les poids sont symétriques : ce re-routage TI-LFA coincide localement avec la reconvergence optimale et potentiellement complète du réseau. Dans mes perspectives de recherche, je décris comment étendre cette méthode à la convergence complète du réseau (plus seulement locale), en particulier avec cette même hypothèse de valuation symétrique et si le réseau accepte les mises à jour au niveau du plan de données.

Reconfigurations pour les Événements Planifiés à l'échelle du routeur

Cette seconde partie (II.2) présente les travaux réalisés pendant la thèse de François Clad [Clad, 2014] avec Stefano Vissicchio, Pierre François et Jean-Jacques Pansiot (directeur). Ces travaux ont donné lieu à plusieurs publications prestigieuses Q1 et A (INCP 2013, ToN 2014, ToN 2015) : [CMV+13, CMP+14, CVM+15]. Ces trois publications apportent des solutions algorithmiques à plusieurs type de problème de re-routage à l'échelle d'un lien ou, plus généralement, d'un nœud entier. La figure VII.2 tirée de la figure II.5 illustre le problème abordé ici : comment calculer des séquences d'incréments évitant les boucles de commutation produites lors des états transitoires de convergence IGP ? notre algorithme, (A)GBA ou ses variantes, peut résoudre ce problème efficacement. Soit le nœud 4 la destination et 0 le nœud à modifier représentés dans la figure VII.2 : lors d'une période de convergence, les états de commutation avant (pre-) et après (post-convergence) peuvent se superposer et générer des circuits dans la fusion des deux graphes acycliques à l'origine, i.e. générer des boucles de commutation dans un cas défavorable (à cause de l'ordre des mises à jour non contrôlé).

Avec nos algorithmes, nous proposons de contrôler l'ordre des mises à jour avec des séquences d'incréments croissants appliquées depuis le routeur à modifier, 0 sur l'exemple : notre solution ne requiert rien de plus que les messages d'états des liens pré-existants (ici un seul message suffit pour s'assurer que d convergence avant d). Mais les difficultés pour résoudre correctement et optimalement le problème dans son ensemble ne manquent pas : en particulier avec les contraintes dynamiques résultants de nouvelles boucles potentielles.

Nos contributions majeures sont les suivantes :

- un cadre formel pour la résolution des circuits en cas de panne (de lien ou) de routeur;
- des algorithmes de calcul et de résolution efficace des contraintes statiques, i.e., les circuits transitoires pre- & post-convergence, en particulier GBA;
- des preuves de sa correction et optimalité avec des séquences (strictement) croissantes;

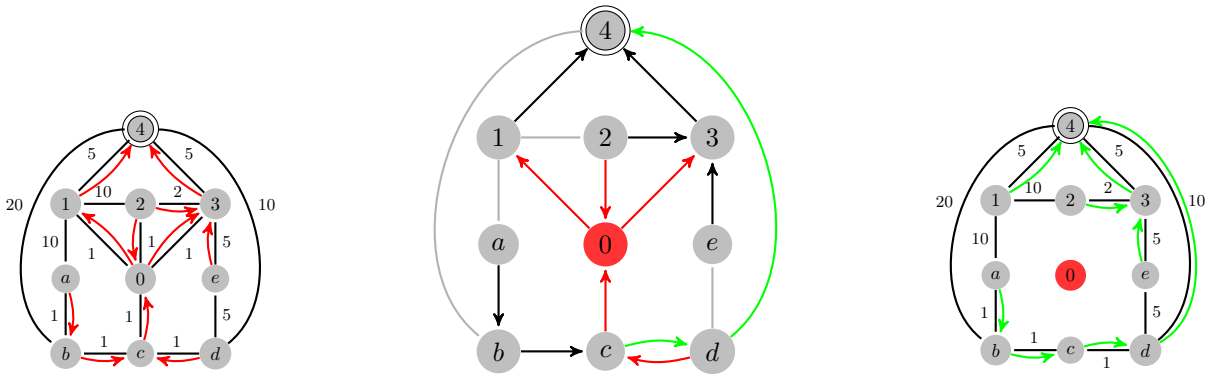


Figure VII.2: Une boucle de commutation transitoire peut survenir entre les nœuds c et d pour la destination 4 durant la convergence du routage IGP. Nos algorithmes préviennent ces problèmes en injectant des séquences d'états des liens depuis le nœud 0.

- des variantes algorithmiques pour la gestion des contraintes dynamiques et de leurs effets, telles que AGBA ou DGBH.

Ces contributions s'inscrivent dans un cadre assez large de solutions pour la reconfiguration des réseaux IP disposant ou non d'une architecture logicielle [117]. Elles sont néanmoins originales et directement déployables car elles ne pré-supposent pas l'utilisation de nouveaux protocoles ou primitives, seuls les messages de signalisation existants sont sollicités pour mettre à jour l'état du réseau. Nous avons montré comment les boucles statiques pouvaient être modélisées et leurs contraintes satisfaites pour définir des séquences minimales d'incréments croissants (sur chaque lien), que ce soit pour un seul lien dirigé mais aussi pour un routeur entier (problème multi-dimensionnel). Dans ce dernier cas, de nouveaux problèmes peuvent se produire : les boucles dynamiques, en particulier autour du nœud à éteindre. Afin de répondre à ces défis, nous avons proposé plusieurs systèmes de résolution de contraintes, à la fois sur un plan physique et aussi logiciel. Nos méthodes produisent des résultats convaincants : les séquences calculées sont relativement courtes sur les topologies typiques des opérateurs et pour la plupart des nœuds et destinations, de même les temps de calcul restent assez faibles dans la majorité des cas.

Plusieurs pistes d'amélioration sont pourtant envisageables : pour commencer, il s'agit de mieux appréhender d'un point de vue théorique le problème dans sa version la plus générale avec des séquences d'incréments non strictement croissantes. De même qu'avec les boucles incidentes au routeur à modifier induites par des changements de commutation, de nouvelles contraintes apparaissent alors dynamiquement : est-il possible de tirer partie efficacement de cette généralisation du problème ? La difficulté, i.e. la complexité réelle, de ces variantes du problème n'est pas connue, probablement NP-difficile. Avec Stefano Vissicchio et Quentin Bramas nous avons l'intention d'analyser de s'y confronter pour mieux cerner le problème dans son ensemble et y répondre avec des méthodes encore plus efficace.

Protection du Trafic de Transit

Dans cette troisième partie, je décris les travaux réalisés pendant la thèse de Master de Jean-Romain Luttringer [Luttringer, 2019] prolongée durant sa thèse de doctorat avec Quentin Bramas et Cristel Pelsser. Ce travail a pour but d'améliorer la gestion de la *patate chaude* par BGP lors de changements de routage internes au domaine considéré. Nous avons publié nos propositions à ce sujet dans une conférence de rang A* (INFOCOM 2021): [LBPM21]. La figure VII.3 (extraite de la partie II.3) illustre les principaux enjeux du défi à relever : **comment définir efficacement l'ensemble des routes nécessaires à la re-convergence BGP, de manière rapide et optimale, en cas de changement intra-domaine** ? notre solution, OPTIC, résout efficacement ce problème. Soit un AS donné et un préfixe destination p, et soit s le nœud à considérer comme source représentée en haut à gauche dans la figure VII.3 : lors d'une période de convergence BGP liée à une panne interne, s doit choisir sa nouvelle meilleure passerelle BGP pour atteindre p. Sur

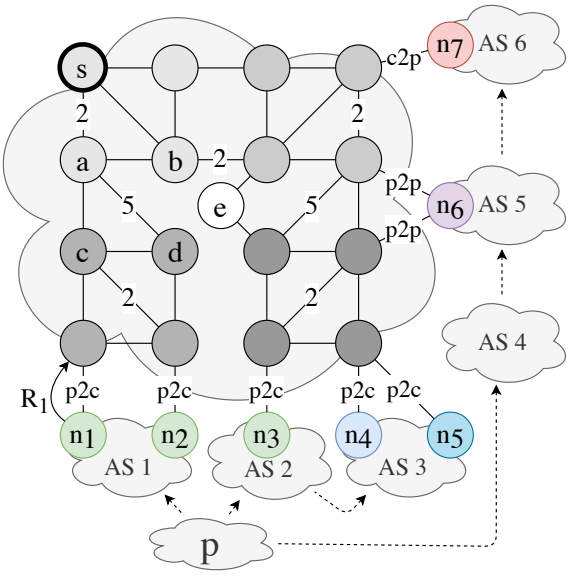


Figure VII.3: Cet exemple (provenant de la figure II.8) décrit les informations pour atteindre le préfixe BGP p depuis s. Les liens BGP sont annotés avec le type de relation (p2c signifie *provider-to-customer*, p2p et c2p, *peer-to-peer* et *customer-to-provider* respectivement). En pratique, le premier type de relation est préféré aux deux suivants pour des raisons de coût. Chaque prochain saut BGP n_x est attaché à la route R_x annoncée. Les liens sans indication de valuation explicite IGP ont un poids de 1.

l'exemple, avant changement, la passerelle n_1 dispose de la meilleure distance IGP vers p (parmi les routes BGP ayant les meilleurs attributs inter-domaines). En revanche, après changement, il pourrait s'agir d'un autre prochain saut BGP : n_x avec $x \neq 1$[2].

Par exemple, la panne du nœud a partitionne le réseau interne et implique l'utilisation de n_3 (de même que la panne du lien (a, c) nécessite cette même passerelle pour rester optimal vis à vis de la patate chaude) alors que les autres pannes peuvent se contenter du prochain saut n_2 (le meilleur si n_1 tombe en panne notamment). La question que nous résolvons ici est la structuration efficace des prochains sauts BGP pour prévenir optimalement toutes les pannes (simples) internes sans nécessiter la re-convergence complète de notre nouveau modèle dans la plupart des cas (ou au moins avoir un coup d'avance sur les changements IGP pour avoir le temps de le faire sans coupure).

Nos contributions phares, toujours en cours d'extension, sont les suivantes :

- un cadre technique pour la résolution du problème de routage de la patate chaude (iBGP) dynamique;
- des structures de données et des algorithmes appropriés, OPTIC, pour le maintien des meilleures routes iBGP en cas de changement IGP;
- une étude analytique du nombre de groupes de préfixes (et de leurs ensembles de passerelles).

Notre approche est originale car elle se positionne à l'interface de BGP et de l'IGP en place au sein du domaine considéré : comment appréhender optimalement, ou au moins efficacement et correctement, le problème BGP de la patate chaude en cas de changements de routage IGP ? Pour cela il faut au préalable de la visibilité sur les routes disponibles au sein du domaine (via iBGP), assez pour assurer la convergence optimale, typiquement avec AddPath [335], et également un mécanisme de table de routage pour une commutation hiérarchique composant les fonctions externes et internes (typiquement comme avec PIC [110]). Nous avons (re)combiné ces deux principes pour produire OPTIC et ses groupes de préfixes par

[2]Les prochains sauts n_4 et n_5 sont discriminées par l'attribut MED alors que le nombre de sauts d'AS intervient pour différencier les routes R_3 et R_4 par exemple.

ensemble de passerelles BGP. Notre solution garantit les meilleurs résultats pour la protection optimale : les routes sont calculées et organisées plus finement que celles atteignables avec les solutions existantes. Il s'agit véritablement de l'ensemble des meilleures routes BGP après convergence (pour tous les événements internes) et non pas d'états sous optimaux dégradés à cause d'un manque de complétude. En revanche, le nombre de groupes, et leurs tailles, deviennent les facteurs limitants de notre solution. En particulier si l'implémentation n'est que réalisée dans le plan de contrôle. Nous avons analysé le nombre de groupes inhérents à notre modèle renforcé en fonction de différents paramètres (taille et niveau du réseau de transit notamment). Nos résultats montrent que notre solution est particulièrement adaptée aux réseaux d'Entreprise et aux réseaux de transit dont la taille est modérée. Notre modèle d'analyse théorique est perfectible quant aux fonctions de distribution en entrée (uniforme par défaut) : des données de terrain pourraient améliorer la finesse de notre analyse.

Nous étudions actuellement l'opportunité d'explorer notre solution dans le cadre des plans de donnés programmables : en effet cela peut grandement limiter l'impact du nombre de groupes sur les mises à jour immédiates. La bascule vers la meilleure passerelle peut se faire à la granularité des paquets en transit sans nécessiter une mise à jour linéaire en le nombre de groupes. L'application d'une fonction aussi basique qu'un minimum pose néanmoins déjà des défis en P4[3] : nous en discuterons plus en détail dans l'avant dernier chapitre résumant mes perspectives de recherche.

Déploiement de Chemins Multi-Critères

Dans cette dernière partie du premier chapitre de contributions, je présente les travaux effectués durant le doctorat de Jean-Romain Luttringer [Luttringer, 2022] co-encadrée avec Quentin Bramas et Cristel Pelsser (directrice). Cette thèse a pour objectif de définir des algorithmes efficaces pour le calcul de chemins contraints. Ces travaux ont donné lieu à deux publications (NCA 2020, COMNET 2022) : [LAM+20, LAM+22].

La figure VII.4 (voir partie II.4 pour la figure d'origine) illustre les concepts nécessaires à l'étude du problème abordé dans cette partie : comment calculer et maintenir efficacement l'ensemble des routes nécessaires au *front de Pareto* induit par DCLC, c'est à dire l'ensemble des routes non dominées requises pour obtenir le meilleur chemin IGP (least cost) contraint par un délai donné (delay constrained) ? **notre solution, BEST2COP, résout efficacement le défi DCLC dans le contexte d'un déploiement SR** (avec une contrainte opérationnelle en plus sur le nombre de segments maximum, DCLC-SR considère ainsi 3 métriques dont l'une est optimisée sous les contraintes des deux autres). Soit un AS donné, ici une partie de GEANT, et un couple source-destination, ici Francfort−Vienne dans la figure VII.4 : il existe trois chemins dans le front de Pareto pour ce couple, chacun un meilleur candidat DCLC avec sa contrainte de délai associée.

Dans la partie II.4, j'explique comment prendre en compte la contrainte du nombre de segments avec le graphe SR multi-métrique. L'ensemble de ces chemins (et leur liste de segments) peut-être exponentiel en taille. Or il faut pouvoir maintenir ce front pour manipuler des résultats exacts : avec Best2Cop, nous avons décidé de contrôler les dimensions de ce front avec une discrétisation des délais. De plus, nous avons mis en oeuvre plusieurs techniques originales pour prendre en charge efficacement la contrainte opérationnelle SR, le nombre de segments maximum.

Nos principales contributions sont les suivantes :

- un large état de l'art du problème DCLC et de l'utilisation de SR dans les réseaux IP;
- un algorithme de calcul efficace et correct (à la granularité de la mesure des latences) pour le résoudre avec la contrainte SR additionnelle : BEST2COP;
- une implémentation et évaluation de BEST2COP par comparaison (avec T/SAMCRA [336, 197]) sur de nombreux réseaux à très large échelle.

L'originalité de nos contributions réside dans la prise en compte native de SR dans notre calcul de chemins contraints (ou et optimaux). Cette contrainte représente un défi particulier car il s'agit d'un

[3] https://opennetworking.org/p4/

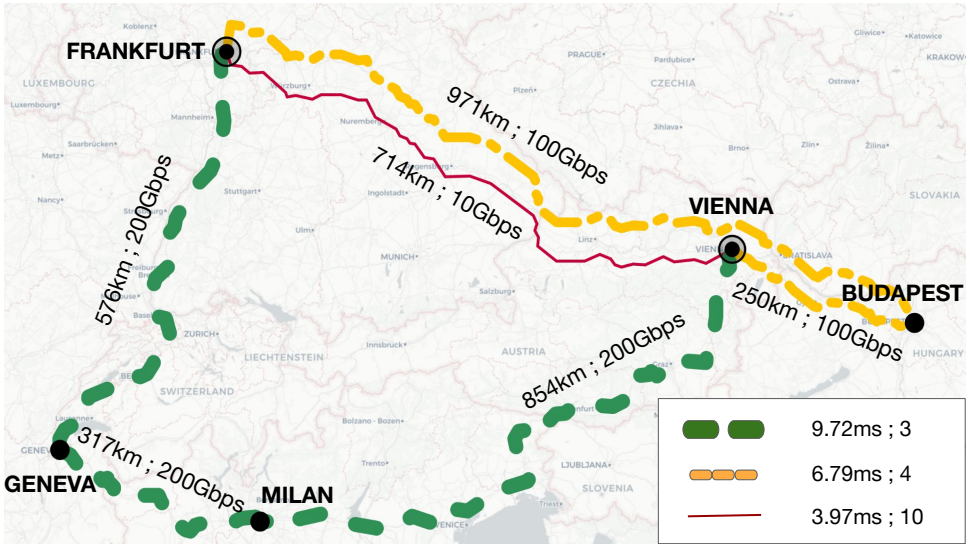


Figure VII.4: Cette illustration met en évidence la pertinence des chemins DCLC dans le réseau GEANT (la figure II.10 est celle d'origine). En fonction de la contrainte de délai requise par le service applicatif à fournir, chacun de ces trois chemins entre Francfort et Vienne peut se révéler le plus performant. Il ne sont pas directement comparables (i.e. non dominés) : il s'agit du front de Pareto nécessaire au calcul de DCLC.

compteur de segments, une métrique non directement valuée dans le graphe multi-métrique de départ : nous proposons plusieurs modèles de calcul, n premier lieu un graphe SR multi-valué sur lequel on peut soit opérer une conversion *online* des chemins durant l'exécution de notre algorithme principal, soit appliquer une transformation de graphes en amont. Avec les hypothèses adéquates sur les délais de propagation (leur mesure est biaisée au moins en terme de fidélité bien que leur résolution, voire leur justesse, puisse sembler meilleure a priori), il est également possible de discrétiser leur mesure sans perte critique d'information afin de réduire et donc contrôler la taille maximale du front de Pareto. Nous proposons un algorithme exploitant ces propriétés, facilement parallélisable et capable d'amortir la complexité de gestion du front de Pareto alors que la plupart des variantes existantes ne cumulent pas tous ces bénéfices. Nous avons expérimentalement étudié et comparé les temps de calcul de nos algorithmes par rapport à l'une des meilleures options concurrentes : notre approche offre des temps de calcul très raisonnables et meilleurs qu'avec l'existant, même dans le cas de graphes d'opérateurs de grande taille.

Ces travaux ont démontré l'intérêt de notre nouvelle solution que nous espérons améliorer de plusieurs manières : la rendre robuste aux changements de délais et de poids IGP est le principal défi : est-il possible de continuer à garantir ce service premium dans ces conditions ? Nous voulons également proposer un cadre de comparaison multi-métriques pour évaluer les nombreuses variantes algorithmiques existantes. Pour finir, l'encodage minimal SR des chemins selon des propriétés plutôt que sur base d'un chemin strict est un problème encore relativement ouvert.

Mesurer et superviser les domaines Internet

Mesurer, sonder voire découvrir Internet est un sujet particulièrement excitant : comment sont configurés les routeurs ? quels configurations et comportements sont les plus populaires et répandus ? quels effets provoquent les changements de routes ? pourquoi certaines routes ne sont pas optimales même à l'intérieur d'un domaine ? Toutes ces questions sont développées dans le chapitre III.

Beaucoup de problèmes se posent pour mesurer ces phénomènes et les défis techniques pour y parvenir sont nombreux. En particulier, il existe beaucoup d'exceptions (sauts absents, adresses tierces, et tunnels invisibles) et le coût de sondage peut se révéler intrusif si celui-ci n'est pas contrôlé efficacement.

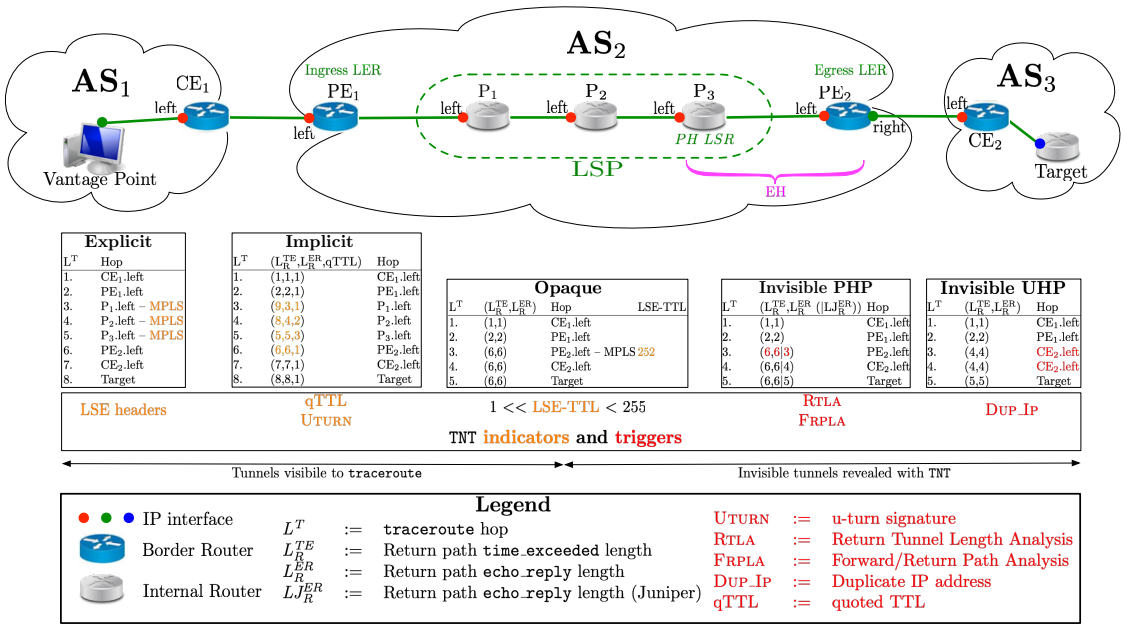


Figure VII.5: Illustration de notre classification MPLS (voir la description d'origine de la figure III.1 pour davantage de détails) : elle caractérise les différents comportements observés avec l'outil `traceroute`. La partie supérieure de la figure représente physiquement le tunnel : P_1 est le premier LSR et P_3 l'avant dernier. En cas de Penultimate Hop Popping (PHP), P_3 est responsable de la suppression du dernier label alors qu'en Ultimate Hop Popping (UHP) c'est PE_2. La partie centrale donne notre classification MPLS en fonction du contenu de la trace et des différents Time to Live (TTL) (donnant des indications sur le nombre de sauts aller et retour). Enfin, la dernière partie décrit les déclencheurs et indicateurs utilisés par `TNT` pour respectivement révéler et indiquer les tunnels résultant d'une trace. Des informations supplémentaires sont également données en légende.

Mesurer Internet nécessite une compréhension fine des protocoles existants, notamment les plus déployés comme MPLS. Chaque réseau a également besoin d'un service de supervision précis afin de contrôler ses performances et détecter d'éventuelles erreurs et dysfonctionnements.

Mes travaux sur le sujet s'inscrivent dans ces deux contextes : comment mieux comprendre, analyser et contrôler les routes empruntées sur Internet ? Découvrir, superviser et analyser les réseaux IP et leurs protocoles de routage sont les trois principaux aspects de mes recherches traitées dans ce chapitre.

Découverte des Tunnels MPLS

Je décris ici les travaux réalisés pendant la thèse d'Yves Vanaubel [Vanaubel, 2018] co-encadrée avec Benoit Donnet (directeur) et avec la participation de Jean-Jacques Pansiot. Ces travaux ont donné lieu à de nombreuses publications prestigieuses dans le domaine (plusieurs IMC, PAM, TMA et TNSM) : [DLMP12, VPMD13, VMPD16, VMPD15, VMPD17, VLM+19, LVM+19]. Ces publications se focalisent essentiellement sur l'étude de MPLS et de son déploiement effectif et nous ont permis de partager plusieurs outils et jeux de données avec la communauté des mesures IP.

La figure VII.5 (détaillée dans la partie III.1) illustre les principaux éléments techniques du problème : comment classifier les déploiements MPLS en fonction de leur visibilité dans les traces actives ? notre outil de traçage actif, TNT, est capable de répondre à cette question. Soit un AS donné et un tunnel MPLS entre PE_1 et PE_2, comme représenté dans la figure VII.5 : il existe quatre principaux types de tunnels MPLS, fonction de leur comportement face à `traceroute`, **les explicites, les implicites, les opaques et les invisibles**.

Cette figure illustre les difficultés à résoudre pour obtenir la plus fine des classifications : chaque type de tunnels a son indicateur ou son déclencheur et l'estimation du TTL initial et du nombre de sauts en résultant est critique pour caractériser les traces. C'est le comportement de PE_1 à l'aller et au retour du paquet qui conditionne la réaction de `TNT`. Si celui-ci est configuré pour cacher le tunnel à l'aller, il peut être lui même responsable d'un indice indiquant un tunnel sur le chemin retour. En particulier grâce à

une astuce des constructeurs, l'application d'une opération minimum entre les TTL IP et MPLS pour éviter une communication explicite entre les deux PE, mais qui sélectionne alors le mauvais TTL dans les messages d'erreur retour.

Nos contributions les plus marquantes sont les suivantes :

- un cadre technique pour décrire et classifier le déploiement pratique de MPLS;
- des algorithmes et leurs implémentations pour l'extraction des tunnels invisibles : l'outil TNT;
- des expérimentations par émulation et plusieurs campagnes de mesure et de performance.

Nous avons été parmi les premiers à proposer des méthodes de révélation des tunnels MPLS (avec [317]), les pionniers de manière concrète et structurelle avec l'outil TNT. L'originalité de notre approche consiste en la détection de motifs particuliers sur les TTL collectés des messages d'erreur renvoyés par les routeurs. Sur base de ces signatures, notre technique de sondage, TNT, envoie de nouveaux messages afin de révéler le tunnel caché détecté (s'il existe et est extractible). Nous avons pu mettre en lumière la large proportion de tunnels MPLS cachés dans Internet et l'efficacité de TNT : nos outils ont révélé de nombreux cas de figure avec des configurations provoquant la présence de tunnels a priori invisibles.

Ces contributions permettent de mieux analyser et appréhender Internet en général en contribuant à la correction des traces de routage afin de corriger les cartes et les propriétés semblant en résulter. Nous voulons continuer à partager nos résultats en étendant leurs portées à d'autres phénomènes liés aux réseaux de niveau 2 ou au déploiement de SR. Je développerai ce projet dans plusieurs directions dans le dernier chapitre indiquant mes perspectives de recherche.

Supervision des Réseaux IP

Cette partie est dédiée aux travaux réalisés dans le cadre de mes collaborations avec RENATER et GEANT, respectivement dans un projet national et européen. Il s'agit de superviser des réseaux IP large échelle en intra- et inter-domaine. Mes principaux travaux, avec de nombreux co-auteurs locaux ou internationaux, ont été publié dans des journaux réputés (COMNET 2019 et COMCOM 2018) : [MDP+18, VBD+20].

L'originalité de ces deux contributions réside dans les caractéristiques suivantes :

- une corrélation fine entre plusieurs sources de données : actives (sondes et messages d'erreur) et passives (protocolaires et manuelles) avec DCART;
- une reconstruction efficace des segments de routes inter-domaines pour déterminer le domaine provoquant des performances dégradées voire des pannes.

La figure VII.6 (provenant originellement de la partie III.2) illustre l'une des principales sources des résultats obtenus avec DCART : le collecteur d'événements de routage IS-IS déployé sur RENATER. Notre plateforme, DCART, se base sur celui-ci et répertorie tous les changements pour par exemple distinguer les événements isolés de ceux des liens dont l'état oscille rapidement entre actif et fautif. Sur cette figure VII.6 où les routeurs sont triés en fonction de leur degré, nous observons plusieurs phénomènes : des lignes (typiquement le *flap*, des agrégats d'événements liés) et des colonnes (des événements à l'échelle d'un routeur). RENATER subit de nombreux changements, de l'ordre de plusieurs par jour (magnitude en la dizaine) et parfois beaucoup plus. Ces changements entraînent à leur tour plusieurs effets : des coupures et des boucles de commutation principalement.

Cette figure offre une vision précise de la fréquence et nature des changements de configuration (intentionnels ou non) dans un opérateur IP. Ils sont nombreux, parfois avec des épisodes intenses, et pour la plupart groupés dans des agrégats. Grâce aux autres sources de données, nous avons pu mesurer dans quelles proportions ils sont la cause des coupures et des boucles de commutation observées. Les coupures longues ($>$ 1 seconde) sont majoritairement dues à des événements de routage et les boucles de commutation surviennent principalement à l'ajout de lien ou de routeur (typiquement faisant suite à une panne).

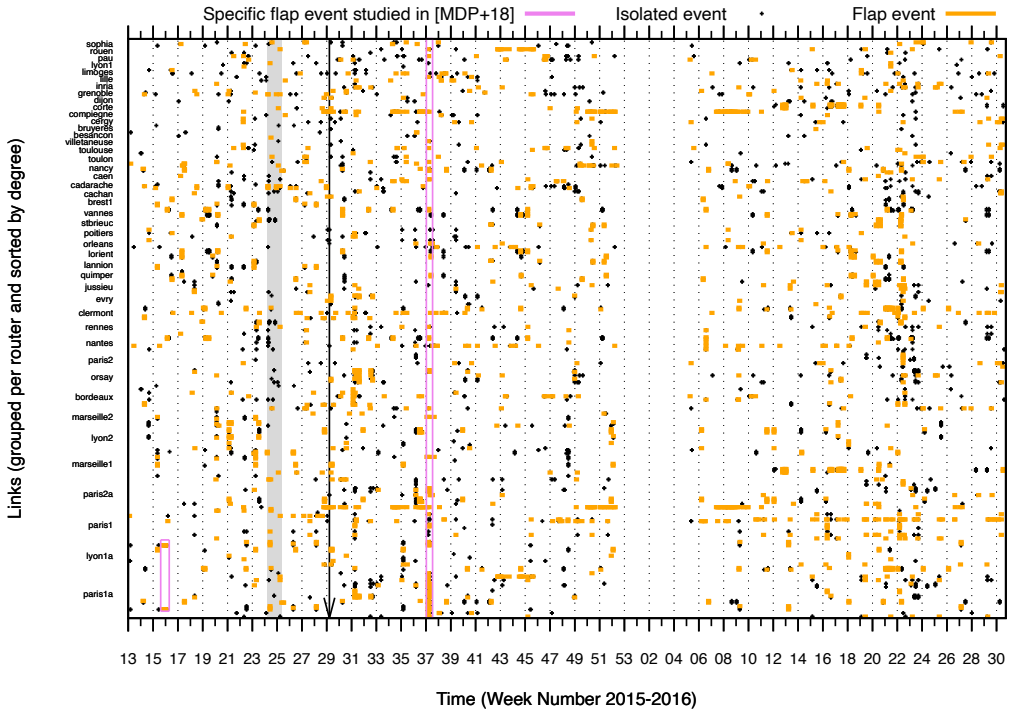


Figure VII.6: Répartition spatio-temporelle des événements de routage triés par degré des routeurs (en y). Les événements isolés sont donnés en noir alors que les agrégats sont oranges (la longueur de l'intervalle donne la durée). Plus d'explications techniques sont données dans la partie de la figure d'origine III.4.

Nos travaux sont originaux pour plusieurs raisons : bien qu'il existe de nombreuses options pour superviser les réseaux IP, celles-ci ne sont généralement pas conçues pour être flexibles et elles ne s'intègrent que partiellement au système de routage dans son ensemble : **nous avons ici finement corrélé quatre sources de données et démontré l'importance d'une mesure coordonnée pour en extraire les phénomènes complexes comme les boucles de commutation**. Grâce à nos mesures, nous avons notamment pu observer un phénomène massif de lien erratiques, leurs états instables générant de nombreuses pertes de paquets. Nous avons également remarqué que la plupart des boucles de commutation semblent se produire au (re)démarrage des liens, ou plus marqué encore, des routeurs dans leur intégralité.

Détours et Deflections

Cette partie décrit les travaux réalisés pendant la thèse de Julian Del-Fiore [Del Fiore, 2021] co-encadrée avec Cristel Pelsser (directrice). Ces travaux sur la mesure des chemins et routes IP ont donné lieu à deux publications [DFMP+19, DFPM+21] (TMA 2019 et TNSM 2020) en collaboration avec l'équipe COMICS de Naples, notamment avec les chercheurs Antonio Pescape et Valerio Persico. L'originalité de notre approche est basée sur le recoupement par partitionnement des routes multiples (en jouant sur les multiples destinations de chaque préfixe /24 associé à une route) afin de les classifier correctement vis à vis de l'équilibrage de charge (distribution uniforme du trafic à une certaine granularité) et de l'ingénierie de trafic (spécifique à quelques préfixes).

Nos contributions les plus significatives sont les suivantes :

- nous avons développé des algorithmes détectant les différences entre les chemins de données et de contrôle inter-domaine afin d'étudier les mensonges BGP;
- nous avons élaboré une méthode capable de discriminer les détours de commutation de l'ingénierie de trafic ou de l'équilibrage de charge;

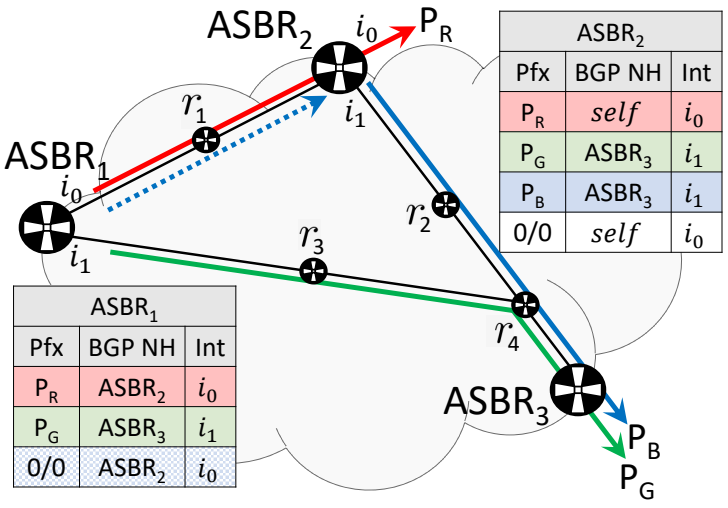


Figure VII.7: Incohérences et détours de routage (la description complète est fournie avec la figure d'origine III.6). La route par défaut de $ASBR_1$, ne disposant que d'une FIB partielle, provoque des incohérences de routage avec $ASBR_2$ pour le préfixe bleu P_B absent. Comme $ASBR_2$ redirige son trafic vers P_B via $ASBR_3$, la meilleure route ne correspond plus avec la meilleure route IGP de $ASBR_1$ vers $ASBR_3$. De plus, le préfixe P_G n'étant pas sujet à de telles incohérences, un motif multi-route apparaît entre $ASBR_1$ et $ASBR_3$: ici la route IGP vers $ASBR_3$ est la même que pour P_G.

- enfin, plusieurs campagnes de mesures ont été effectué avec ces nouveaux outils pour mesurer ces phénomènes à l'échelle de l'Internet actuel.

La figure VII.7 (originellement décrite dans la partie III.3) illustre l'origine du problème des détours sur un exemple de FIB partiel. La difficulté qui se pose dans ce cas est : comment distinguer les détours, de l'équilibrage de charge et de l'ingénierie de trafic ? En effet, ils résultent tous les trois en des motifs de commutation multi-chemins. Notre solution analyse la dispersion des destinations et préfixes sur les routes de l'ensemble de départ (entrée-sortie identiques) et associe la route IGP de base (la directe) à celle obtenue en traçant l'interface IP de sortie du prochain saut BGP, la sortie associée à cet ensemble de routes (pour une entrée donnée). Pour cette analyse, il s'agit de fusionner les routes collectées entre elles si possible (ECMP) ainsi que les préfixes associés. Ceux pour lesquels l'une de leurs destinations utilise une route incluse dans un autre ensemble sont fusionnés : ces routes et préfixes appartiennent à un ensemble ECMP. En pratique, ces ensembles et leurs fusions sont explorables et faisables dans la majorité des cas car le nombre de routes IP visibles est limité (alors qu'en théorie et dans le pire cas le nombre de routes ECMP peut se révéler factoriel en la taille du réseau). Sur cette figure (VII.7) et l'exemple simplifié qu'il représente, on peut observer deux routes vers $ASBR_3$ depuis $ASBR_1$. Elles sont provoquées par un état de routage incomplet sur $ASBR_1$: le trafic est détourné via $ASBR_2$, ce qui provoque un détour visible si celui-ci n'est pas sur la meilleure route IGP. En fonction de la répartition du trafic sur ces deux routes, notre méthode en déduit l'origine (cause du multi-route).

Sur cette figure, si le trafic est équitablement réparti, nous en déduirons un motif ECMP alors que si la route interne est isolée (la seule à ne pas être fusionnée avec les autres), nous conclurons à un détour extrême (toutes les routes BGP sont détournées de la route directe pour un couple). Notre outil est aussi capable de distinguer le type d'implémentation ECMP en jeu : répartition par destination, par port ou par préfixe. Les résultats obtenus constituent une sous estimation des origines du phénomène : outre la confusion possible avec d'autres techniques multi-chemins, il existe plusieurs configurations où les incohérences restent invisibles car elles ne provoquent pas toujours des détours (visibles).

Ces travaux nous ont permis de constater que de nombreux opérateurs semble avoir recours à des solutions à court terme pour prévenir les manques de performances de certains de leurs routeurs (ceux ne pouvant supporter l'échelle de BGP) : malgré les limites d'une telle mesure, la proportion de détours

est significative et ceux-ci sont extrêmes quand ils se produisent : la majorité du trafic, tout le trafic de transit BGP, est dévié sur un ensemble de routes annexes à l'exception de la route directe (vers l'IP entrante du routeur de sortie) qui n'est pas utilisée pour le trafic inter-domaine. Ces déviations internes peuvent provoquer à leur tour des différences BGP entre les routes réellement empruntés en opposition aux chemins théoriquement annoncés.

Nous envisageons de continuer à explorer ce domaine en croisant ces deux types d'information pour mieux cerner la nature des mensonges sur Internet et alors proposer des méthodes protocolaires pour les prévenir ou au moins les limiter. Par ailleurs, nous sommes aussi intéressés par des modèles probabilistes plus efficaces pour la découverte de routes multiples et de l'équilibrage de charge, à la fois sur le plan technique et théorique.

Objets Connectés

L'Internet des Objets (IoT) et les réseaux de capteurs/actionneurs sont un terrain d'application dont la particularité est un medium instable et complexe à partager équitablement entre les stations. De plus, les protocoles de routage pour les réseaux sans fils n'ont ni les mêmes besoins, ni les mêmes contraintes que les réseaux filaires. Leurs ressources sont plus limités en énergie, en mémoire et en calcul. Par ailleurs, le trafic collecté dans ces réseaux feuilles est souvent critique en terme de confidentialité : les données personnelles échangées doivent être anonymisées dès leur émission en les agrégeant avec des fonctions commutatives et associatives. Dans ce chapitre (plus développé dans sa version longue en anglais IV), nous allons étudier les différentes pistes que je poursuis afin de répondre à ces problématiques : économie d'énergie, confidentialité des flux IoT multi-opérateurs et déploiement sécurisé de politiques d'accès dans le *cloud*.

Économie d'énergie dans les Réseaux de Capteurs

Dans cette partie, je décris brièvement les travaux réalisés pendant la thèse de master de François Clad [Clad, 2011] co-encadrée avec Antoine Gallais. Ces travaux, portant sur les réseaux de capteurs économes en énergie, ont donné lieu à deux publications (PIMRC 2019, ICCCN 2011) : [MG09, CGM12].

Nos principales contributions sont les suivantes :

- une analyse des opportunités et bénéfices de la diversité des chemins dans les réseaux d'objets contraints;
- une heuristique distribuée de construction d'arbres maximisant le nombre de feuilles;
- une analyse et évaluation par comparaison (méthode optimale ou approximation).

Ces travaux ont été parmi les premiers à généraliser l'idée d'ordonnancer les réseaux de capteurs de manière hétérogène dans le temps et l'espace pour économiser la consommation liée à la commutation de paquets. Les performances de notre solution sont encourageantes dans le cadre des déploiements évalués. Il s'agit d'une base intéressante pour économiser de l'énergie au niveau des communications radios et ainsi prolonger la durée de vie de ce type de réseaux.

Confidentialité et Vie Privée dans l'Internet des Objets

Cette partie synthétise les travaux effectués dans le cadre d'un projet ANR JCJC que je supervise, NanoNet : il s'agit essentiellement des travaux issus de la thèse de Renato Neto [Neto, 2022] co-encadrée avec Fabrice Theoleyre (directeur). L'objet du projet est de proposer une architecture de routage avec agrégation des données pour les réseaux de bordure pour l'IoT. Ces travaux ont donné lieu à quatre publications (LCN 2020, NCA 2020, IWCMC 2021, ISCC 2021) : [NMT20b, NMT20a, NMGT21, NMT21].

Nos principales contributions sont les suivantes :

- une architecture réseau pour l'agrégation et la transformation de flux IoT inter-opérateurs;

- un algorithme de routage NDN pour l'agrégation et la confidentialité des données;
- une analyse d'un déploiement réaliste multi-opérateurs avec LORA.

L'originalité et la pertinence de notre approche résident dans la transformation des données au sein du réseau durant leur collecte. Plutôt que de déléguer ce travail au niveau applicatif, nous proposons une architecture réseau orientée données pour les flux IoT confidentiels. La difficulté du problème est lié aux possibles duplication de données si elles ne sont pas identifiables (au moins de manière anonyme) et lorsque la structure sous-jacente peut contenir des circuits. Le problème devient alors de construire des annonces ne contenant que des offres dont l'intersection des identifiants est nulle. Le nombre de combinaisons étant exponentiel, il s'agit de réduire l'espace de recherche de manière efficace en agrégeant en priorité les données aux similarités élevées. Nos simulations indiquent que malgré l'apparente difficulté du problème, nos heuristiques produisent des résultats satisfaisants dès lors que certains producteurs de données ont des exigences d'agrégation raisonnable.

Plusieurs limites de nos solutions sont perfectibles : un attaquant peut tenter d'isoler des données en combinant des annonces provenant de routeurs différents (sans pour autant totalement dé-anonymiser les données). Nous aimerions maintenant démontrer que cet effet est limité en pratique et que le risque est suffisamment faible pour garantir la viabilité de notre solution.

Meta-graphes et Micro-services pour Prévenir les Fuites de Données

Dans cette partie, je présente les contributions proposées pendant la thèse de Loic Miller [Miller, 2022] co-encadrée avec Cristel Pelsser et Antoine Gallais, co-directeurs de la thèse. C'est un financement Cisco qui nous a permis de financer l'étude de la sécurité des données dans le cadre de déploiement de workflows distribués. Ces travaux ont donné lieu à trois publications (deux HPSR 2020 et MDPI 2021) : [MMGP21b, MMGP21c, MMGP21a].

Nos deux principaux travaux et réalisations sont les suivants :

- une architecture micro-services pour les workflows distribués;
- une représentation et vérification des politiques d'accès représentés sous forme de metagraphes.

Ces travaux nous ont permis d'investiguer plusieurs pistes pour mettre en oeuvre un workflow sécurisé dans le *cloud*. D'abord en ce qui concerne les aspects techniques et pratiques, nous avons évalué une architecture micro-services adaptée à un tel déploiement où les données sont chiffrées et leurs traitements isolés autant que possible, et où les accès sont vérifiés et les communications interceptées. Loic Miller a proposé une preuve de concept pour un tel déploiement avec tous les éléments nécessaires à son fonctionnement correct et efficace. Ensuite, sur un plan plus théorique et en s'appuyant sur les metagraphes à l'expressivité appropriée à la spécification des politiques d'accès, il a élaboré plusieurs méthodes d'analyse et de conversion des règles : de leur spécification en objets aux relations n-aires à leur déploiement dans un langage concret. Pour finir, il s'est intéressé à la complexité inhérente aux primitives permettant d'analyser la redondance et les conflits présents dans les politiques spécifiées. En effet, la propriété de *dominance* propre aux metagraphes [45] se révèle très utile pour la détection de redondances (les hypergraphes fournissent néanmoins un cadre similaire avec davantage de résultats connus). Malheureusement, ce problème est NP-difficile comme Loic l'a démontré (ainsi que ses variantes plus spécifiques) mais nous avons pu élaborer plusieurs heuristiques dont les performances sont prometteuses.

Perspectives de Recherche

Ce dernier chapitre décrit les pistes de recherche que j'envisage d'explorer à moyen et long terme. Alors que mes perspectives à court terme ont déjà été introduites dans les chapitres précédents essentiellement sous forme d'extensions de mes travaux en cours ou passés, je m'intéresse ici à de nouveaux projets dans plusieurs domaines. Grâce à l'essor de nouvelles technologies plus ouvertes et flexibles, le développement de nouveaux services répondant à des exigences toujours plus élevées en matière de qualité est aujourd'hui

plus simple qu'hier : avec les réseaux orientés logiciel et équipements programmables, plusieurs pistes sont envisageables pour faciliter le transfert technologique et l'innovation.

En particulier, l'émergence des plans de données programmable représente une opportunité remarquable pour y déployer des techniques de re-routage rapide là où elles peuvent se révéler les plus efficaces. À moyen terme, la première thèse que je souhaite encadrer concerne en effet ma spécialité préférée : la convergence rapide et sûre des protocoles de routage IP en profitant des nouveaux moyens techniques à disposition. En particulier, en faisant également l'hypothèse d'une valuation symétrique des liens, de nouveaux protocoles sont possibles et moins tributaires des temporisateurs généralement utilisés pour ce type de service. Je décris également dans ce chapitre mes projets concernant les réseaux sans fil et la mesure sur Internet. De manière générale, et à long terme, l'algorithmique distribuée et les algorithmes auto-stabilisants sont les deux thématiques où j'aimerais davantage m'investir dans les décennies à venir.

Convergence Rapide et Correcte avec Valuation Symétrique

Dans cette partie, je développe les contributions revisitées au début du manuscrit, TBFH et AGBA en particulier, dans un nouveau contexte technologique (les réseaux programmables, en particulier au niveau du plan de données) et avec une hypothèse forte en théorie mais raisonnable en pratique. Avec une valuation symétrique, il est en effet possible d'aisément détecter et ainsi facilement éviter les boucles de commutation durant la convergence en cas de panne. **Dans un réseau avec des poids symétriques, tous les liens de tous les circuits associés à ces boucles existent dans les deux directions (avec ou sans ECMP)**. De nouvelles approches pour améliorer l'état de l'art sont possibles avec cette propriété : il suffit de superviser les nouveaux liens, en particulier ceux impliqués dans les deux directions pour un même paquet (facilement et rapidement détectable), en utilisant SR pour éviter que ces paquets ne bouclent. Cette approche peut être déployée à la fois pro-activement et ré-activement. D'abord, pour chaque nouveau lien est envoyée une information de signalisation au nouveau voisin, potentiellement implicite via le plan de données et des détours SR; ensuite tant que certains paquets entrent par leur nouvelle interface de sortie, ceux-ci sont re-expédiés avec un détour SR. Si la première action ne suffit pas (tant que la mise à jour n'a pas été opérée à cause d'une perte ou ralentissement), la seconde devient nécessaire, il s'agit d'un filet de sécurité.

Une telle approche dispose de deux avantages majeurs : elle ne nécessite pas de temporisateur, ni de signalisation explicite. D'une part, le filet de sécurité permet de ne plus se soucier de la synchronisation entre équipements voisins : elle se fait de manière implicite et évènementiel sur chaque paquet sans besoin de calibrer un temporisateur (dont le but serait de prédire la durée maximale de la perte de synchronisation). D'autre part, les mises à jour au niveau du plan de données permettent d'accélérer la synchronisation entre équipements : le premier routeur à jour indique indirectement à ses voisins la nouvelle route à suivre avec sa déviation SR. Cela permet aux routeurs de réaliser leurs mises à jour instantanément dans le plan de données et dans l'ordre de circulation des paquets sans avoir besoin d'attendre la fin des opérations de mise à jour provenant du plan de contrôle. Mon objectif est de vérifier la faisabilité et l'efficacité d'une telle approche avec les équipements de commutation programmables déjà disponibles sur le marché. Dans la partie suivante, ma démarche est similaire (appliquer les mises à jour en concurrence entre le plan de données et de contrôle) mais pour des évènements plus complexes comme la panne d'un routeur (en entier) ou la convergence BGP suite à un changement IGP dans le registre d'OPTIC.

Mises à Jour dans le Plan de Données pour les Événements Complexes

Nous avons déjà initié certains de ces travaux et ainsi commencé à explorer ces pistes, notamment dans le cadre de la thèse de Master de Thomas Alfroy [Alfroy, 2020] et son implémentation d'OPTIC en Free Range Routing[4]. Il s'agit d'une part d'implémenter OPTIC au niveau des plans de contrôle et de données pour évaluer sa faisabilité et ses bénéfices pratiques dans le contexte des réseaux programmables (sur les deux plans et leurs interactions), et d'autre part, de traiter le cas d'événements de routage posant davantage de défis que la simple panne de liens en interne, typiquement la panne de nœud en entier. En effet, **dans le cas d'un réseau aux poids symétriques, je conjecture que le nombre de segments nécessaires**

[4] https://frrouting.org/

pour traiter le cas d'une panne de nœud est au pire égal au degré de celui-ci (au lieu d'un segment au pire pour un lien). Je travaille actuellement sur un algorithme et sa preuve pour traiter ce sous problème.

En ce qui concerne l'extension P4 d'OPTIC, nous réfléchissons actuellement aux moyens les plus efficaces pour contourner la limite du nombre de groupes. Alors que cette limite s'applique au plan de contrôle et en terme de mémoire, nous pensons qu'il est possible et bénéfique de réaliser les mises à jour directement depuis le plan de données : le nombre de groupes n'auraient alors plus d'effet sur la bascule vers la nouvelle meilleure passerelle. Là encore, nous souhaitons réaliser des expériences sur des équipements programmables modernes afin de quantifier les performances, avantages et limites de nos propositions.

Multi-topologies pour les Réseaux Multi-Radios Efficaces en Énergie

Il s'agit d'un travail entamé avec Julien Montavont et Léon Niess dans le cadre d'un stage de Master portant sur la collecte de données dans l'Internet des Objets. Notre objectif est de poursuivre et surtout étendre les propositions réalisées durant la thèse de Sebastien Sampayo [295]. Les réseaux bi-radios sont une alternative asynchrone aux réseaux déterministes : une des deux radios, peu consommatrice d'énergie, permet de réveiller l'autre, la principale, à la demande, plus consommatrice mais disposant d'une plus grande portée et d'un meilleur débit.

Notre principal objectif est de proposer une architecture asynchrone de routage multi-sauts sur les deux radios : les meilleurs chemins empruntés seront ceux minimisant la métrique de la radio principale pour lesquels il existe des sous chemins (contraints et optimaux dans ce cadre) sur l'autre technologie. Ainsi, le réveil des nœuds intermédiaires permettra de progressivement joindre le puits du réseau (trafic convergecast). Nous sommes actuellement en train de finaliser la spécification et l'implémentation dans un simulateur de nos propositions et variantes algorithmiques et protocolaires. La radio de courte portée est très contrainte en pratique car c'est sa modulation assez grossière qui lui permet de réaliser des économies d'énergie : par exemple, elle ne peut pas être utilisée pour partager des tables de routage (trop d'information), seulement pour initier la découverte de ses liaisons et réveiller spécifiquement un nœud identifié (un ou deux identifiants par paquet au maximum). Nos propositions visent à solliciter la radio longue portée pour embarquer ces informations lors de la construction de l'architecture de routage. De plus, nous appliquons une limite de profondeur sur ce (sous) graphe pour éviter les inondations et assurer la fiabilité des sous chemins courte portée (distance limitée). Nos objectifs à moyen terme sont les suivants : évaluer les bénéfices de notre solution et de ses variantes par rapport à l'existant en fonction de plusieurs critères (par exemple, l'écart à l'optimal en terme de longueur de chemins, temps de convergence, fiabilité et coût en messages, consommation énergétique pour le maintien de la structure et son utilisation) sur des scénarios réalistes.

Découverte de Topologie, Mesures Embarquées et Détection d'Anomalies

Fort de mon expérience dans ce domaine, je souhaite poursuivre et étendre les travaux entamés avec Benoit Donnet et Emeline Maréchal [MMD22]. Depuis le projet Rocketfuel [319], peu de chercheurs se sont intéressés à la question de la collecte fine des domaines de routage régissant la composition d'Internet. S'il existe de nombreux outils pour collecter et résoudre des sous ensembles topologiques d'Internet au niveau routeur, les cartes en résultant manquent souvent d'information en terme de valuation (poids des liens et/ou délais) et de découpage par domaine. Notre premier objectif est donc de pallier ce manque en prenant en compte les nouvelles spécificités logicielles et physiques de l'Internet actuel : émergence de l'équilibrage de charge (ECMP) et de technologies comme SR, et son évolution structurelle (CDN et IXP) remettant en cause certaines des propriétés exploitées jusqu'ici par les outils existants. Il s'agit d'unifier et de mettre à jour les principaux outils de la communauté mesure IP.

Mon deuxième objectif est de développer des mesures embarquées dans le plan de données pour mieux réagir, i.e. au plus près, des informations de terrain : la gestion des micro-congestions (files pleines sur de très court laps de temps) avec de l'équilibrage de charge ECMP ou étendu est un exemple de ce qu'il peut se faire à cette échelle et dans ce cadre. De manière générale se posent les questions de contrôle et de réaction pour le routage, voire de détection d'anomalies et d'attaques. Quelles sont les décisions à déléguer et à quelle entité ? Il est nécessaire de redéfinir le rôle de chaque plan (données, contrôle et management) afin de partager ces mesures et disséminer les réactions intelligemment. J'aimerais étudier

ces opportunités pour savoir ce qu'il est judicieux de mesurer et d'opérer dans le plan de données : sur base des équipements à notre disposition, je souhaite définir un cadre pratique et formel pour analyser et résoudre cette question et l'appliquer éventuellement à la détection d'anomalies.

Conclusion et Travaux Futurs

A travers ce manuscrit, j'espère avoir démontré mon expérience dans mon domaine de recherche, les réseaux IP, du routage à sa mesure en particulier. Depuis ma prise de fonction en tant que maître de conférences permanent en 2011, j'ai maintenu une activité de publications soutenue (près de trois par an en moyenne) et de qualité avec plusieurs journaux Q1 et conférences de rang A ou A*. La plupart de ces travaux ont été réalisés dans le cadre de thèses de doctorat. J'ai eu la chance d'en co-encadrer six dans des domaines divers : routage (2) et mesures IP (2) mais aussi sur la confidentialité dans l'IoT et la sécurité dans les réseaux virtualisés aux opérations distribués. L'ensemble de ces thèses ont été menées à bien et ont donné lieu à au moins trois publications internationales par doctorant.
Mes deux principales thématiques de recherche sont (i) le calcul de meilleurs chemins (potentiellement contraints) dans le cadre de modifications topologiques dans les réseaux dynamiques et, (ii), la mesure et supervision des réseaux IP, en particulier en matière de routage. Ces deux thèmes sont complémentaires et se nourrissent l'un de l'autre : mes activités en routage m'ont permis de développer une expertise avancée pour la mesure des phénomènes s'y rapportant (par exemple avec la découverte de tunnels MPLS ou la révélation des détours), et réciproquement, les analyses effectuées sur nos collectes de données sont précieuses pour l'évaluation des solutions de routage proposées (vis à vis des architectures et des topologies réseaux typiques) ainsi que pour mieux appréhender les règles de bonnes pratiques les plus répandues. Cette complémentarité est bénéfique à mon travail dans chacun de ces deux sous thèmes et m'a offert la chance de publier dans des revues et conférences parmi les plus prestigieuses des deux sous-domaines. J'ai aussi étudié des problèmes différents dans d'autres domaines d'application : l'économie d'énergie dans les réseaux IoT de capteurs/actionneurs, la confidentialité des données pour les flux multi-opérateurs sans-fil, et la vérification des politiques d'accès dans les workflows distribués. Ces travaux ont élargi mes compétences vers de nouveaux champs disciplinaires où nous avons réussi à contribuer au travers de plusieurs publications scientifiques.
Enfin, je propose dans ce mémoire plusieurs pistes de recherche à court et moyen termes : au delà des simples extensions des projets déjà explorés dans les premiers chapitres, je souhaite m'investir dans quatre directions en particulier : la convergence IP en cas de valuation symétrique (et sa détection/mesure par le plan de données), les mises à jour du routage dans le plan de données par le plan de données, les réseaux sans fil bi-radios, asynchrones et multi-sauts pour l'économie d'énergie, ainsi que la mesure et la cartographie des technologies émergentes.
A plus long terme, j'aimerais investir mon temps dans de nouveaux domaines de recherche connexes à mes activités actuelles comme les systèmes et algorithmes distribués. En particulier les algorithmes auto-stabilisants avec des hypothèses de travail apportant de nouveaux défis en réseaux, comme la prise en compte des pannes byzantines plutôt que seulement des pannes franches. Je pense en effet que les algorithmes de routage pourraient grandement bénéficier de ce genre de cadre formel plus large et plus théorique que ceux considérés en pratique. Je n'exclus pas non plus l'opportunité de travailler sur des problèmes encore plus éloignés des recherches conduites jusqu'ici, par exemple avec l'exploration de structures discrètes (i.e. des graphes plus ou moins régulier et dynamiques) par des robots contraints et aux capacités faibles. Je serai soutenu dans ces activités par les arrivées relativement récentes de Quentin Bramas et Anissa Lamani dans notre équipe de recherche. Leur intégration a en effet permis l'essor dans notre groupe d'une nouvelle thématique traitant de ce type de sujets.

Bibliography

[1] Archipelago (ark). https://www.caida.org/projects/ark/.

[2] As rank. https://asrank.caida.org.

[3] Cef polarization. https://www.cisco.com/c/en/us/support/docs/ip/express-forwarding-cef/116376-technote-cef-00.html.

[4] Configuring per-prefix load balancing. https://www.juniper.net/documentation/en_US/junos/topics/usage-guidelines/policy-configuring-per-prefix-load-balancing.html.

[5] How does load balancing work? https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/5212-46.html.

[6] Paris Traceroute. https://paris-traceroute.net/.

[7] Per-flow and per-packet load balancing. https://support.huawei.com/enterprise/en/doc/EDOC1100055041/ebc8ad42/per-flow-and-per-packet-load-balancing.

[8] Ripe atlas. https://atlas.ripe.net/.

[9] Understanding the algorithm used to load balance traffic on mx series routers. https://www.juniper.net/documentation/en_US/junos/topics/concept/hash-computation-mpcs-understanding.html.

[10] Soroush Abbasian Dehkordi, Kamran Farajzadeh, Javad Rezazadeh, Reza Farahbakhsh, Kumbesan Sandrasegaran, and Masih Abbasian Dehkordi. A survey on data aggregation techniques in IoT sensor networks. Wireless Networks, 26(2):1243–1263, feb 2020.

[11] Emile Aben. 768k Day. Will it Happen? Did it Happen? https://labs.ripe.net/Members/emileaben/768k-day-will-it-happen-did-it-happen.

[12] Mehran Abolhasan, Tadeusz Wysocki, and Eryk Dutkiewicz. A review of routing protocols for mobile ad hoc networks. Ad Hoc Networks, 2(1):1–22, January 2004.

[13] Ahmed Aboul-Ela. Delete credit cards from any twitter account, 2014. https://hackerone.com/reports/27404.

[14] Z. Abrams, A. Goel, and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. pages 424–432, Berkeley, CA, USA, 2004.

[15] Heidi Adams. Segment routing in the 5g era. https://www.juniper.net/assets/de/de/local/pdf/articles/3200083-en.pdf. (Accessed on 06/21/2021).

[16] P. Agarwal and B. Akyol. Time-to-live (TTL) processing in multiprotocol label switching (MPLS) networks. RFC 3443, Internet Engineering Task Force, January 2003.

[17] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. Computer Networks (Elsevier), 38:393–422, 2002.

[18] Bahaa Al-Musawi, Philip Branch, and Grenville Armitage. Bgp anomaly detection techniques: A survey. IEEE Communications Surveys Tutorials, 19(1):377–396, 2017.

[19] David L. Alderson, John C. Doyle, and Walter Willinger. Lessons from "a first-principles approach to understanding the internet's router-level topology". Comput. Commun. Rev., 49(5):96–103, 2019.

[20] Rafael Almeida, Renata Teixeira, Darryl Veitch, Christophe Diot, et al. Classification of load balancing in the internet. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pages 1987–1996. IEEE, 2020.

[21] G. Almes, S. Kalidindi, and M. Zekauskas. A round-trip delay metric for ippm. RFC 2681, RFC Editor, September 1999.

[22] G. Almes, S. Kalidindi, M. Zekauskas, and A. Morton. A one-way delay metric for ip performance metrics (ippm). STD 81, RFC Editor, January 2016.

[23] Karine Altisen, Stéphane Devismes, Swan Dubois, and Franck Petit. Introduction to Distributed Self-Stabilizing Algorithms, volume 8 of Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool, April 2019.

[24] Marica Amadeo, Claudia Campolo, Antonio Iera, and Antonella Molinaro. Named data networking for IoT: An architectural perspective. In European Conference on Networks and Communications, pages 1–5. IEEE, jun 2014.

[25] Marica Amadeo, Giuseppe Ruggeri, Claudia Campolo, and Antonella Molinaro. IoT Services Allocation at the Edge via Named Data Networking: From Optimal Bounds to Practical Design. IEEE Transactions on Network and Service Management, 16(2):661–674, jun 2019.

[26] Michele Amoretti, Riccardo Pecori, Yanina Protskaya, Luca Veltri, and Francesco Zanichelli. A Scalable and Secure Publish/Subscribe-Based Framework for Industrial IoT. IEEE Transactions on Industrial Informatics, 17(6):3815–3825, 2021.

[27] Tore Anderson. Slimming down the Internet routing table. https://www.redpill-linpro.com/sysadvent/2016/12/09/slimming-routing-table.html.

[28] L. Andersson and R. Asati. Multiprotocol label switching (MPLS) label stack entry: EXP field renamed to traffic class field. RFC 5462, Internet Engineering Task Force, February 2009.

[29] L. Andersson, I. Minei, and T. Thomas. LDP specification. RFC 5036, Internet Engineering Task Force, October 2007.

[30] Y. P. Aneja and K. P. K. Nair. The constrained shortest path problem. Naval Research Logistics Quarterly, 25(3):549–555, 1978.

[31] Francois Aubry. Models and Algorithms for Network Optimization with Segment Routing. PhD thesis, 2019.

[32] François Aubry, David Lebrun, Stefano Vissicchio, Minh Thanh Khong, Yves Deville, and Olivier Bonaventure. Scmon: Leveraging segment routing to improve network monitoring. In IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pages 1–9, 2016.

[33] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding traceroute anomalies with paris traceroute. In Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, pages 153–158, 2006.

[34] Giorgio Ausiello and Luigi Laura. Directed hypergraphs: Introduction and fundamental algorithms——a survey. Theoretical Computer Science, 658:293–306, 2017.

[35] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP tunnels. RFC 3209, Internet Engineering Task Force, December 2001.

[36] D. Aydin. CISCO vs. Juniper MPLS, June 2014. See http://monsterdark.com/cisco-vs-juniper-mpls/.

[37] Vaibhav Bajpai and Jürgen Schönwälder. A survey on internet performance measurement platforms and related standardization efforts. IEEE Communications Surveys Tutorials, 17(3):1313–1341, 2015.

[38] Hitesh Ballani, Paul Francis, Tuan Cao, and Jia Wang. Making routers last longer with viaggre. In NSDI, volume 9, pages 453–466, 2009.

[39] Manu Bansal, Jeffrey Mehlman, Sachin Katti, and Philip Levis. Openradio: A programmable wireless dataplane. In Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12, page 109–114, New York, NY, USA, 2012. Association for Computing Machinery.

[40] Paul Barford and Joel Sommers. Comparing Probe- and Router-Based Packet-Loss Measurement. IEEE Internet Computing, 2004.

[41] Colby Barth. LINX103: Scale Out Route-Server, 2018.

[42] Stefano Basagni, Chiara Petrioli, and Dora Spenza. CTP-WUR: The collection tree protocol in wake-up radio WSNs for critical applications. In 2016 International Conference on Computing, Networking and Communications (ICNC), pages 1–6, Kauai, HI, USA, February 2016. IEEE.

[43] Ahmed Bashandy, Clarence Filsfils, Bruno Decraene, Stephane Litkowski, Pierre Francois, Daniel Voyer, Francois Clad, and Pablo Camarillo. Topology Independent Fast Reroute using Segment Routing. Internet-Draft draft-bashandy-rtgwg-segment-routing-ti-lfa-05, IETF Secretariat, October 2018.

[44] Ahmed Bashandy, Clarence Filsfils, Stefano Previdi, Bruno Decraene, Stephane Litkowski, and Rob Shakir. Segment Routing with the MPLS Data Plane. RFC 8660, December 2019.

[45] Amit Basu and Robert W Blanning. Metagraphs and their applications, volume 15. Springer Science & Business Media, 2007.

[46] Zahra Beiranvand, Ahmad Patooghy, and Mahdi Fazeli. I-LEACH: An efficient routing algorithm to improve performance & to reduce energy consumption in Wireless Sensor Networks. In The 5th Conference on Information and Knowledge Technology, pages 13–18. IEEE, may 2013.

[47] John Bellardo and Stefan Savage. Measuring packet reordering. In Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, pages 97–105, 2002.

[48] K. Benson, A. Dainotti, k. claffy, A. Snoeren, and M. Kallitsis. Leveraging internet background radiation for opportunistic network analysis. In ACM Internet Measurement Conference (IMC), 2015-10.

[49] Padmalochan Bera, Soumya Kanti Ghosh, and Pallab Dasgupta. Policy based security analysis in enterprise networks: A formal approach. IEEE Transactions on Network and Service Management, 7(4):231–243, 2010.

[50] R. Beverly. Yarrp'ing the Internet: Randomized high-speed active topology discovery. In Proc. ACM Internet Measurement Conference (IMC), November 2016.

[51] Khadak Singh Bhandari, In-Ho Ra, and Gihwan Cho. Multi-Topology Based QoS-Differentiation in RPL for Internet of Things Applications. IEEE Access, 8, 2020.

[52] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman. Optimized network traffic engineering using segment routing. In 2015 IEEE Conference on Computer Communications (INFOCOM), pages 657–665, 2015.

[53] Netflix Technology Blog. Netflix conductor: A microservices orchestrator, 2016.

[54] Netflix Technology Blog. Evolution of netflix conductor: v2.0 and beyond, 2019. https://netflixtechblog.com/evolution-of-netflix-conductor-16600be36bca.

[55] L. Blunk, M. Karir, and C. Labovitz. MRT Routing Information Export Format. RFC 6396, 2011.

[56] R. Bonica, D. Gan, D. Tappan, and C. Pignataro. ICMP extensions for multiprotocol label switching. RFC 4950, Internet Engineering Task Force, August 2007.

[57] Quentin Bramas, Pascal Mérindol, Cristel Pelsser, and Jean-Romain Luttringer. A Fast-Convergence Routing of the Hot-Potato: The Tool to Perform your own Evaluation, February 2020. https://doi.org/10.5281/zenodo.3972128.

[58] Anat Bremler-Barr, Yehuda Afek, and Shemer Schwarz. Improved BGP Convergence via Ghost Flushing. In IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428), volume 2, pages 927–937. IEEE, 2003.

[59] Gerth Stølting Brodal, George Lagogiannis, and Robert E. Tarjan. Strict fibonacci heaps. In Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, STOC '12, page 1177–1184, New York, NY, USA, 2012. Association for Computing Machinery.

[60] J. Brumbaugh-Smith and D. Shier. An empirical investigation of some bicriterion shortest path algorithms. European Journal of Operational Research, 43(2):216–224, Nov 1989.

[61] Alexander Brundiers, Timmy Schüller, and Nils Aschenbruck. On the benefits of loops for segment routing traffic engineering. In 2021 IEEE 46th Conference on Local Computer Networks (LCN), pages 32–40, 2021.

[62] Randy Bush, Olaf Maennel, Matthew Roughan, and Steve Uhlig. Internet optometry: Assessing the broken glasses in Internet reachability. In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, IMC '09. ACM, 2009.

[63] Simon Byers, Lorrie Cranor, Dave Korman, Patrick McDaniel, and Eric Cronin. Analysis of security vulnerabilities in the movie production and distribution process. In Proceedings of the 3rd ACM workshop on Digital rights management, pages 1–12. ACM, 2003.

[64] Rute C. Sofia and Paulo M. Mendes. An Overview on Push-Based Communication Models for Information-Centric Networking. Future Internet, 11(3):74, mar 2019.

[65] Matthew Caesar, Donald Caldwell, Nick Feamster, Jennifer Rexford, Aman Shaikh, and Jacobus van der Merwe. Design and Implementation of a Routing Control Platform. In Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2, NSDI'05, page 15–28, USA, 2005. USENIX Association.

[66] Yanli Cai, Wei Lou, Minglu Li, and X.-Y. Li. Target-oriented scheduling in directional sensor networks. pages 1550–1558, Anchorage, AK, USA, 2007.

[67] Berat Can Şenel, Maxime Mouchet, Justin Cappos, Olivier Fourmaux, Timur Friedman, and Rick McGeer. Shared internet-scale measurement platforms. In NSF Workshop on Overcoming Measurement Barriers to Internet Research (WOMBIR 2021), extended abstract, Online, United States, January 2021.

[68] J. Camilo Cardona, Pierre Francois, Bruno Decraene, John Scudder, Adam Simpson, and Keyur Patel. Bringing High Availability to BGP. Comput. Netw., 91(C):788–803, November 2015.

[69] Radu Carpa, Olivier Glück, and Laurent Lefevre. Segment routing based traffic engineering for energy efficient backbone networks. In 2014 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), pages 1–6, 2014.

[70] Angelo P Castellani, Thomas Fossati, and Salvatore Loreto. Http-coap cross protocol proxy: an implementation viewpoint. In MASS. IEEE, 2012.

[71] Ramaswamy Chandramouli and Zack Butcher. Building secure microservices-based applications using service-mesh architecture. Technical report, National Institute of Standards and Technology, 2020.

[72] Jaideep Chandrashekar, Zhenhai Duan, Z-L Zhang, and Jeffrey Krasky. Limiting path exploration in bgp. In Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., volume 4, pages 2337–2348. IEEE, 2005.

[73] Chi-kin Chau, Richard Gibbens, and Timothy G. Griffin. Towards a Unified Theory of Policy-Based Routing. INFOCOM, 2006.

[74] Marco Chiesa, Andrzej Kamisiński, Jacek Rak, Gábor Rétvári, and Stefan Schmid. A Survey of Fast-Recovery Mechanisms in Packet-Switched Networks. IEEE Communications Surveys Tutorials, 23(2):1253–1301, 2021.

[75] Marco Chiesa, Roshan Sedar, Gianni Antichi, Michael Borokhovich, Andrzej Kamisiński, Georgios Nikolaidis, and Stefan Schmid. Fast ReRoute on Programmable Switches. IEEE/ACM Transactions on Networking, 29(2):637–650, 2021.

[76] Chen-Nee Chuah, Supratik Bhattacharyya, and Christophe Diot. Measuring I-BGP Updates and Their Impact on Traffic. Technical report, SPRINT ATL res. rep. TR02-ATL-051099, 1999.

[77] Catalin Cimpanu. Steam bug could have given you access to all the cd keys of any game, 2018.

[78] Cisco Networks. Cisco content hub - performance measurement for traffic engineering. https://content.cisco.com/chapter.sjs?uri=/searchable/chapter/content/en/us/td/docs/ios-xml/ios/seg_routing/configuration/xe-17/segrt-xe-17-book/performance_measure_TE.html.xml, September 2020.

[79] kc claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov. Internet mapping: from art to science. In Proc. IEEE Cybersecurity Application and Technologies Conference for Homeland Security (CATCH), March 2009.

[80] Thomas Clausen, Jiazi Yi, and Ulrich Herberg. Lightweight On-demand Ad hoc Distance-vector Routing - Next Generation (LOADng): Protocol, extension, and applicability. Computer Networks, 126:125–140, October 2017.

[81] Privacy Rights Clearinghouse. Data breaches, 2020. https://privacyrights.org/data-breaches.

[82] Florin Coras, Damien Saucez, Loránd Jakab, Albert Cabellos-Aparicio, and Jordi Domingo-Pascual. Implementing a BGP-free ISP core with LISP. In 2012 IEEE Global Communications Conference (GLOBECOM), 2012.

[83] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. Introduction to Algorithms. McGraw-Hill Higher Education, 2001.

[84] Matthew L. Daggitt, Alexander J. T. Gurney, and Timothy G. Griffin. Asynchronous convergence of policy-rich distributed bellman-ford routing protocols. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '18, page 103–116, New York, NY, USA, 2018. Association for Computing Machinery.

[85] F. Dai and J. Wu. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. IEEE Transactions on Parallel and Distributed Systems, 15(10):908–920, 2004.

[86] L. De Ghein. MPLS Fundamental: A Comprehensive Introduction to MPLS (Theory and Practice). CISCO Press, November 2006.

[87] H. De Neve and P. Van Mieghem. Tamcra: A tunable accuracy multiple constraints routing algorithm. Comput. Commun., 23(7):667–679, March 2000.

[88] Catherine de Weever and Marios Andreou. Zero trust network security model in containerized environments. 2020.

[89] Pratikkumar Desai, Amit Sheth, and Pramod Anantharam. Semantic Gateway as a Service Architecture for IoT Interoperability. In International Conference on Mobile Services, volume 32, pages 313–319. IEEE, jun 2015.

[90] Shivani Deshpande, Marina Thottan, and Biplab Sikdar. An online scheme for the isolation of bgp misconfiguration errors. IEEE Transactions on Network and Service Management, 5(2):78–90, 2008.

[91] G. Detal, b. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet. Revealing middlebox interference with tracebox. In Proc. ACM Internet Measurement Conference (IMC), October 2013.

[92] Marco Di Bartolomeo, Valentino Di Donato, Maurizio Pizzonia, Claudio Squarcella, and Massimo Rimondini. Extracting routing events from traceroutes: A matter of <italic>empathy</italic>. IEEE/ACM Transactions on Networking, 27(3):1000–1012, 2019.

[93] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. Commun. ACM, 17(11):643–644, nov 1974.

[94] Josep Domingo-Ferrer and Jordi Soria-Comas. Data Anonymization. In Risks and Security of Internet and Systems, pages 267–271. Springer, 2015.

[95] B. Donnet and T. Friedman. Internet topology discovery: a survey. IEEE Communications Surveys and Tutorials, 9(4):2–15, December 2007.

[96] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Efficient algorithms for large-scale topology discovery. In Proc. ACM SIGMETRICS, June 2005.

[97] Ali Dorri, Salil S. Kanhere, Raja Jurdak, and Praveen Gauravaram. Blockchain for IoT security and privacy: The case study of a smart home. In 2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017, number March, pages 618–623, 2017.

[98] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, Automata, Languages and Programming, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[99] K. Edeline and B. Donnet. A first look at the prevalence and persistence of middleboxes in the wild. In Proc. International Teletraffic Congress (ITC), September 2017.

[100] Theodore Elhourani, Abishek Gopalan, and Srinivasan Ramasubramanian. Ip fast rerouting for multi-link failures. IEEE/ACM Transactions on Networking, 24(5):3014–3025, 2016.

[101] Ahmed Elmokashfi and Amogh Dhamdhere. Revisiting bgp churn growth. ACM SIGCOMM Computer Communication Review, 44(1):5–12, 2013.

[102] Funda Ergun, Rakesh Sinha, and Lisa Zhang. An improved fptas for restricted shortest path. Information Processing Letters, 83(5):287–291, Sep 2002.

[103] Nick Feamster, David G Andersen, Hari Balakrishnan, and M Frans Kaashoek. Measuring the effects of internet path faults on reactive routing. ACM SIGMETRICS Performance Evaluation Review, 31(1):126, June 2003.

[104] G. Feng, K. Makki, N. Pissinou, and C. Douligeris. Heuristic and exact algorithms for qos routing with multiple constraints. IEICE Transactions on Communications, 85:2838–2850, 2002.

[105] C. Filsfils. Segment routing - cisco live barcelona 2019. https://www.segment-routing.net/conferences/2019-01-30-CLEUR-3122/, January 2019.

[106] C. Filsfils, K. Michielsen, and K. Talaulikar. Segment Routing Part I. Number partie. 1 in Segment Routing. CreateSpace Independent Publishing Platform, 2017. https://books.google.fr/books?id=1zMyMQAACAAJ.

[107] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir. Segment routing architecture. RFC 8402, RFC Editor, July 2018.

[108] Clarence Filsfils. Network programming with srv6. https://orbi.uliege.be/bitstream/2268/245292/4/network-programming-phd-final.pdf, April 2020.

[109] Clarence Filsfils, Pablo Camarillo, John Leddy, Daniel Voyer, Satoru Matsushima, and Zhenbin Li. Segment Routing over IPv6 (SRv6) Network Programming. RFC 8986, February 2021.

[110] Clarence Filsfils, Pradosh Mohapatra, John Bettink, Pranav Dharwadkar, Peter De Vriendt, Yuri Tsier, Virginie Van Den Schrieck, Olivier Bonaventure, and Pierre Francois. Bgp prefix independent convergence (pic) technical report. Cisco, Tech. Rep, Tech. Rep, 2011.

[111] Clarence Filsfils, Pradosh Mohapatra, John Bettink, Pranav Dharwadkar, Peter De Vriendt, Yuri Tsier, Virginie Van Den Schrieck, Olivier Bonaventure, and Pierre Francois. Bgp prefix independent convergence (pic) technical report. Cisco, Tech. Rep, Tech. Rep, 2011.

[112] T. Fiola and J. Panagos. This Week: Deploying MPLS. Junos Networking Technologies Series. Juniper Networks Books, April 2011.

[113] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed computing by oblivious mobile robots. Synthesis lectures on distributed computing theory, 3(2):1–185, 2012.

[114] K. Foerster, M. Parham, M. Chiesa, and S. Schmid. Ti-mfa: Keep calm and reroute segments fast. In IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pages 415–420, 2018.

[115] Klaus-Tycho Foerster, Andrzej Kamisiński, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan. Grafting arborescences for extra resilience of fast rerouting schemes. In IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, pages 1–10, 2021.

[116] Klaus-Tycho Foerster, Mahmoud Parham, Marco Chiesa, and Stefan Schmid. Ti-mfa: Keep calm and reroute segments fast. In IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pages 415–420, 2018.

[117] Klaus-Tycho Foerster, Stefan Schmid, and Stefano Vissicchio. Survey of consistent software-defined network updates. IEEE Communications Surveys & Tutorials, 21(2):1435–1461, 2019.

[118] Romain Fontugne, Esteban Bautista, Colin Petrie, Yutaro Nomura, Patrice Abry, Paulo Gonçalves, Kensuke Fukuda, and Emile Aben. Bgp zombies: An analysis of beacons stuck routes. In International Conference on Passive and Active Network Measurement, pages 197–209. Springer, 2019.

[119] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing ospf weights. In Proceedings IEEE INFOCOM 2000, volume 2, pages 519–528. IEEE, 2000.

[120] P. Francois, M. Shand, and O. Bonaventure. Disruption free topology reconfiguration in ospf networks. In IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications, pages 89–97, 2007.

[121] Pierre Francois. Improving the Convergence of IP Routing Protocols. Ph.d. thesis, October 2007.

[122] Pierre Francois and Olivier Bonaventure. Avoiding transient loops during the convergence of link-state routing protocols. IEEE/ACM Trans. Netw., 15(6):1280–1292, December 2007.

[123] Pierre Francois, Clarence Filsfils, John Evans, and Olivier Bonaventure. Achieving Sub-second IGP Convergence in Large IP Networks. SIGCOMM Comput. Commun. Rev., 35(3), July 2005.

[124] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. volume 34, page 596–615, New York, NY, USA, jul 1987. Association for Computing Machinery.

[125] Jing Fu, Peter Sjodin, and Gunnar Karlsson. Loop-free updates of forwarding tables. IEEE Transactions on Network and Service Management, 5(1):22–35, 2008.

[126] Tetsuya Fujie. An exact algorithm for the maximum leaf spanning tree problem. Computers & Operations Research, 30(13):1931–1944, 2003.

[127] Olfa Gaddour and Anis Koubâa. RPL in a nutshell: A survey. Elsevier Computer Networks, 56(14), 2012.

[128] E. Gafni and D. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. IEEE Transactions on Communications, 29(1):11–18, 1981.

[129] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenović. Ensuring k-coverage in wireless sensor networks under realistic physical layer assumptions. In Proc. of 5th IEEE Conference on Sensors (Sensors), Daegu, Korea, 2006.

[130] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenović. Localized sensor area coverage with low communication overhead. IEEE Transactions on Mobile Computing, 7(5):661–672, May 2008.

[131] Adrian Gämperli, Vasileios Kotronis, and Xenofontas Dimitropoulos. Evaluating the Effect of Centralization on Routing Convergence on a Hybrid BGP-SDN Emulation Framework. ACM SIGCOMM Computer Communication Review, 44(4):369–370, 2014.

[132] Lixin Gao and J. Rexford. Stable internet routing without global coordination. IEEE/ACM Transactions on Networking, 9(6):681–692, 2001.

[133] Sam Gao, Mark Handley, and Stefano Vissicchio. Stats 101 in P4: towards in-switch anomaly detection. In HotNets '21: The 20th ACM Workshop on Hot Topics in Networks, 2021, 2021.

[134] Michael R. Garey and David S. Johnson. Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., USA, 1990.

[135] Rosario G. Garroppo, Stefano Giordano, and Luca Tavanti. A survey on multi-constrained optimal path computation: Exact and approximate algorithms. Computer Networks, 54(17):3081–3107, Dec 2010.

[136] S. Gay, Renaud Hartert, and Stefano Vissicchio. Expect the unexpected: Sub-second optimization for segment routing. IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, pages 1–9, 2017.

[137] S. Gay, P. Schaus, and S. Vissichio. REPETITA: Repeatable experiments for performance evaluation of traffic-engineering algorithms. Technical Report 1710.08665, October 2017.

[138] Mayssa Ghribi and Aref Meddeb. Survey and taxonomy of mac, routing and cross layer protocols using wake-up radio. Journal of Network and Computer Applications, 149:102465, 2020.

[139] S. Giacalone, D. Ward, J. Drake, A. Atlas, and S. Previdi. Ospf traffic engineering (te) metric extensions. RFC 7471, RFC Editor, March 2015.

[140] Evan Gilman and Doug Barth. Zero Trust Networks. O'Reilly Media, Incorporated, 2017.

[141] L. Ginsberg, S. Previdi, S. Giacalone, D. Ward, J. Drake, and Q. Wu. Is-is traffic engineering (te) metric extensions. RFC 8570, RFC Editor, March 2019.

[142] A. Giorgetti, P. Castoldi, F. Cugini, J. Nijhof, F. Lazzeri, and G. Bruno. Path encoding in segment routing. In 2015 IEEE Global Communications Conference (GLOBECOM), pages 1–6, 2015.

[143] A. Goel, K.G. Ramakrishnan, D. Kataria, and D. Logothetis. Efficient computation of delay-sensitive routes from one source to all destinations. In Proceedings IEEE INFOCOM 2001, volume 2, pages 854–858 vol.2, 2001.

[144] B. J. Goodchild, Y.-C. Chiu, R. Hansen, H. Lua, M. Calder, M. Luckie, W. Lloyd, D. Choffnes, and E. Katz-Bassett. The record route option is an option! In In Proc. ACM Internet Measurement Conference (IMC), November 2017.

[145] M. Goyal, M. Soperi, E. Baccelli, G. Choudhury, A. Shaikh, H. Hosseini, and K. Trivedi. Improving convergence speed and scalability in ospf: A survey. IEEE Communications Surveys Tutorials, 14(2):443–463, 2012.

[146] J.-F. Grailet and B. Donnet. Towards a renewed alias resolution with space search reduction and IP fingerprinting. In Proc. IFIP Network Traffic Measurementand Analysis Conference (TMA), June 2017.

[147] J.-F. Grailet and B. Donnet. Revisiting subnet inference WISE-ly. In Proc. IFIP Network Traffic Measurementand Analysis Conference (TMA), June 2019.

[148] J.-F. Grailet, F. Tarissan, and B. Donnet. TreeNET: Discovering and connecting subnets. In Proc. Traffic Monitoring and Analysis Workshop (TMA), April 2016.

[149] T.G. Griffin, F.B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. IEEE/ACM Transactions on Networking, 10(2):232–243, 2002.

[150] Timothy G. Griffin and Joäo Luís Sobrinho. Metarouting. SIGCOMM Comput. Commun. Rev., 35(4):1–12, aug 2005.

[151] Timothy G Griffin and Gordon Wilfong. On the correctness of IBGP configuration. ACM SIGCOMM Computer Communication Review, 2002.

[152] Jochen W. Guck, Amaury Van Bemten, Martin Reisslein, and Wolfgang Kellerer. Unicast qos routing algorithms for sdn: A comprehensive survey and performance evaluation. IEEE Communications Surveys & Tutorials, 20(1):388–415, 2018.

[153] Rabah Guedrez, Olivier Dugeon, Samer Lahoud, and Géraldine Texier. Label encoding algorithm for mpls segment routing. In 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA), pages 113–117. IEEE, 2016.

[154] H. Guo and J. Heidemann. Detecting ICMP rate limiting in the Internet. In Proc. Passive and Active Measurement Conference (PAM), 2018 March.

[155] Liang Guo and I. Matta. Search space reduction in qos routing. In 19th IEEE International Conference on Distributed Computing Systems, page 142–149, Jun 1999.

[156] Dan Gusfield and Robert W. Irving. The Stable marriage problem - structure and algorithms. Foundations of computing series. MIT Press, 1989.

[157] H. Haddadi, G. Iannaccone, A. Moore, R. Mortier, and M. Rio. Network topologies: Inference, modeling and generation. IEEE Communications Surveys and Tutorials, 10(2):48–69, April 2008.

[158] F. Hao, M. Kodialam, and T. V. Lakshman. Optimizing restoration with segment routing. In IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pages 1–9, 2016.

[159] Renaud Hartert, Pierre Schaus, Stefano Vissicchio, and Olivier Bonaventure. Solving segment routing problems with hybrid constraint programming techniques. In Gilles Pesant, editor, Principles and Practice of Constraint Programming, pages 592–608, Cham, 2015. Springer International Publishing.

[160] A. C. Harvey. ARIMA Models, pages 22–24. Palgrave Macmillan UK, London, 1990.

[161] Refael Hassin. Approximation schemes for the restricted shortest path problem. Mathematics of Operations Research, 17(1):36–42, 1992.

[162] David Hauweele, Bruno Quoitin, Cristel Pelsser, and Randy Bush. What do parrots and bgp routers have in common? ACM SIGCOMM Computer Communication Review, 46(3):2, 2018.

[163] Urs Hengartner, Sue Moon, Richard Mortier, and Christophe Diot. Detection and Analysis of Routing Loops in Packet Traces. In Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement, pages 107–112, Marseille, France, November 2002.

[164] Rodrigo Teles Hermeto, Antoine Gallais, and Fabrice Theoleyre. Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey. Computer Communications, 114:84 – 105, 2017.

[165] Kevin Hoarau, Pierre Ugo Tournoux, and Tahiry Razafindralambo. Suitability of graph representation for bgp anomaly detection. In 2021 IEEE 46th Conference on Local Computer Networks (LCN), pages 305–310, 2021.

[166] Thomas Holterbach, Edgar Costa Molero, Maria Apostolaki, Alberto Dainotti, Stefano Vissicchio, and Laurent Vanbever. Blink: Fast Connectivity Recovery Entirely in the Data Plane. In 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), pages 161–176, Boston, MA, February 2019. USENIX Association.

[167] Thomas Holterbach, Stefano Vissicchio, Alberto Dainotti, and Laurent Vanbever. SWIFT: Predictive Fast Reroute. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '17, page 460–473, New York, NY, USA, 2017. Association for Computing Machinery.

[168] Xiaolan Hou, Muqing Wu, and Min Zhao. An optimization routing algorithm based on segment routing in software-defined networks. Sensors, 19:49, 12 2018.

[169] Graham Hughes and Tevfik Bultan. Automated verification of access control policies using a sat solver. International journal on software tools for technology transfer, 10(6):503–520, 2008.

[170] Fatima Hussain, Weiyue Li, Brett Noye, Salah Sharieh, and Alexander Ferworn. Intelligent service mesh framework for api security and management. In 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pages 0735–0742. IEEE, 2019.

[171] Geoff Huston. BGP in 2018 – BGP Churn. https://blog.apnic.net/2019/01/22/bgp-in-2018-bgp-churn/.

[172] Geoff Huston. CIDR-REPORT Status summary. https://www.cidr-report.org.

[173] Geoff Huston. http://bgp.potaroo.net.

[174] Gianluca Iannaccone, Chen-Nee Chuah, Richard Mortier, Supratik Bhattacharyya, and Christophe Diot. Analysis of Link Failures in an IP Backbone. In Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement, pages 237–242, Marseille, France, November 2002.

[175] IBM ILOG. CPLEX Optimizer, high-performance mathematical programming solver for linear programming, mixed integer programming, and quadratic programming. http://www.ibm.com.

[176] Hiro Ito, Kazuo Iwama, Yasuo Okabe, and Takuya Yoshihiro. Avoiding Routing Loops on the Internet. Theory of Computing Systems, 36:597–609, 2003.

[177] Jeffrey M. Jaffe. Algorithms for finding paths with multiple constraints. Networks, 14(1):95–116, 1984.

[178] Pooyan Jamshidi, Claus Pahl, Nabor C Mendonça, James Lewis, and Stefan Tilkov. Microservices: The journey so far and challenges ahead. IEEE Software, 35(3):24–35, 2018.

[179] E. Jasinska, N. Hilliard, R. Raszuk, and N. Bakker. Internet Exchange BGP Route Server. RFC 7947, RFC Editor, September 2016.

[180] E. T. Jaynes. Probability theory: The logic of science. Cambridge University Press, Cambridge, 2003.

[181] Zhanfeng Jia and P. Varaiya. Heuristic methods for delay constrained least cost routing using k-shortest-paths. IEEE Transactions on Automatic Control, 51(4):707–712, Apr 2006.

[182] Cheng Jin, Abhinav Srivastava, and Zhi-Li Zhang. Understanding security group usage in a public iaas cloud. In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, pages 1–9. IEEE, 2016.

[183] A. Juttner, B. Szviatovski, I. Mecs, and Z. Rajko. Lagrange relaxation based method for the qos routing problem. In Proceedings IEEE INFOCOM 2001, volume 2, pages 859–868 vol.2, 2001.

[184] Mohamed Amine Kafi, Jalel Ben Othman, and Nadjib Badache. A survey on reliability protocols in wireless sensor networks. ACM Comput. Surv., 50(2), may 2017.

[185] Enio Kaljic, Almir Maric, Pamela Njemcevic, and Mesud Hadzialic. A survey on data plane flexibility and programmability in software-defined networking. IEEE Access, 7:47804–47840, 2019.

[186] Andrzej Kamisiński. Evolution of ip fast-reroute strategies. In 2018 10th International Workshop on Resilient Networks Design and Modeling (RNDM), pages 1–6, 2018.

[187] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the Tightrope: Responsive Yet Stable Traffic Engineering. In ACM SIGCOMM, 2005.

[188] Holger Karl and Andreas Willig. Protocols and Architectures for Wireless Sensor Networks. Wiley-Interscience edition, 2007.

[189] Richard M Karp. Reducibility among combinatorial problems. In Complexity of computer computations, pages 85–103. Springer, 1972.

[190] Elliott Karpilovsky, Matthew Caesar, Jennifer Rexford, Aman Shaikh, and Jacobus Van Der Merwe. Practical network-wide compression of ip routing tables. IEEE Transactions on Network and Service Management, 9(4):446–458, 2012.

[191] E. Katz-Bassett, H. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. van Wesep, A. Krishnamurthy, and T. Anderson. Reverse traceroute. In Proc. USENIX Symposium on Networked Systems Design and Implementations (NSDI), June 2010. See https://www.revtr.ccs.neu.edu.

[192] K. Keys. Internet-scale IP alias resolution techniques. ACM SIGCOMM Computer Communication Review, 40(1):50–55, January 2010.

[193] Elie F. Kfoury, Jorge Crichigno, and Elias Bou-Harb. An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends. IEEE Access, 9:87094–87155, 2021.

[194] T. Korkmaz and M. Krunz. Multi-constrained optimal path selection. In Proceedings IEEE INFOCOM 2001., volume 2, page 834–843 vol.2, Apr 2001.

[195] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, and Hicham Lakhlef. Internet of things security: A top-down survey. Computer Networks, 141:199–221, 2018.

[196] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz. An overview of constraint-based path selection algorithms for qos routing. IEEE Communications Magazine, 40(12):50–55, Dec 2002.

[197] F. A. Kuipers and P. F. A. Van Mieghem. Conditions that impact the complexity of qos routing. IEEE/ACM Transactions on Networking, 13(4):717–730, 2005.

[198] Romain Kuntz, Antoine Gallais, and Thomas Noel. From versatility to auto-adaptation of the medium access control in wireless sensor networks. Journal of Parallel and Distributed Computing, 71(9):1236–1248, 2011.

[199] Nate Kushman, Srikanth Kandula, and Bruce M. Maggs. R-BGP: Staying Connected in a Connected World. In 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI 07), Cambridge, MA, April 2007. USENIX Association.

[200] Amund Kvalbein, Audun Fosselie Hansen, Tarik Cicic, Stein Gjessing, and Olav Lysne. Multiple routing configurations for fast ip network recovery. IEEE/ACM Transactions on Networking, 17(2):473–486, 2009.

[201] Kin-Wah Kwong, Lixin Gao, Roch Guerin, and Zhi-Li Zhang. On the feasibility and efficacy of protection routing in ip networks. In 2010 Proceedings IEEE INFOCOM, pages 1–9, 2010.

[202] F. Lazzeri, Gianmarco Bruno, J. Nijhof, A. Giorgetti, and P. Castoldi. Efficient label encoding in segment-routing enabled optical networks. 2015 International Conference on Optical Network Design and Modeling (ONDM), pages 34–38, 2015.

[203] W. C. Lee, M. G. Hluchyi, and P. A. Humblet. Routing subject to quality of service constraints in integrated communication networks. IEEE Network, 9(4):46–55, 1995.

[204] Ka-Cheong Leung, Victor OK Li, and Daiqin Yang. An overview of packet reordering in transmission control protocol (tcp): problems, solutions, and challenges. IEEE transactions on parallel and distributed systems, 18(4):522–535, 2007.

[205] Bing Li, Dijiang Huang, Zhijie Wang, and Yan Zhu. Attribute-based Access Control for ICN Naming Scheme. IEEE Transactions on Dependable and Secure Computing, 15(2):194–206, mar 2018.

[206] S. Lindsey, C. Raghavendra, and K. Sivalingam. Data gathering in sensor networks using the energy*delay metric. In Proceedings of the 15th International Parallel and Distributed Processing Symposium, pages 2001–2008, April 2001.

[207] S. Lindsey and C. S Raghavendra. PEGASIS: power-efficient gathering in sensor information systems. In Proceedings of the IEEE Aerospace Conference, pages 1125–1130, March 2002.

[208] S Litkowski, B Decraene, and P Francois. Microloop prevention by introducting a local convergence delay. Internet-draft, IETF, 2013.

[209] Stephane Litkowski, Ahmed Bashandy, Clarence Filsfils, Pierre Francois, Bruno Decraene, and Daniel Voyer. Topology Independent Fast Reroute using Segment Routing. Internet-Draft draft-ietf-rtgwg-segment-routing-ti-lfa-07, Internet Engineering Task Force, June 2021. Work in Progress.

[210] Gang Liu and K.G. Ramakrishnan. A*prune: an algorithm for finding k shortest paths subject to multiple constraints. In Proceedings IEEE INFOCOM 2001, volume 2, pages 743–749 vol.2, 2001.

[211] Junda Liu, Aurojit Panda, Ankit Singla, Brighten Godfrey, Michael Schapira, and Scott Shenker. Ensuring connectivity via data plane mechanisms. In 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), pages 113–126, Lombard, IL, April 2013. USENIX Association.

[212] W Liu, W Lou, and Y Fang. An efficient quality of service routing algorithm for delay-sensitive applications. Computer Networks, 47(1):87–104, Jan 2005.

[213] Ritika Lohiya and Ankit Thakkar. Application Domains, Evaluation Data Sets, and Research Challenges of IoT: A Systematic Review. IEEE Internet of Things Journal, 8(11):8774–8798, jun 2021.

[214] Qasim Lone, Matthew Luckie, Maciej Korczyński, and Michel van Eeten. Using loops observed in traceroute to infer the ability to spoof. In Mohamed Ali Kaafar, Steve Uhlig, and Johanna Amann, editors, Passive and Active Measurement, pages 229–241, Cham, 2017. Springer International Publishing.

[215] Javier Lopez, Ruben Rios, Feng Bao, and Guilin Wang. Evolving privacy: From sensors to the Internet of Things. Future Generation Computer Systems, 75:46–57, 2017.

[216] Dean H. Lorenz and Danny Raz. A simple efficient approximation scheme for the restricted shortest path problem. Operations Research Letters, 28(5):213–219, Jun 2001.

[217] Pierpaolo Loreti, Andrea Mayer, Paolo Lungaroni, Francesco Lombardo, Carmine Scarpitta, Giulio Sidoretti, Lorenzo Bracciale, Marco Ferrari, Stefano Salsano, Ahmed Abdelsalam, and et al. Srv6-pm: A cloud-native architecture for performance monitoring of srv6 networks. IEEE Transactions on Network and Service Management, 18(1):611–626, Mar 2021.

[218] Hsueh-I Lu and R. Ravi. Approximating maximum leaf spanning trees in almost linear time. Journal of Algorithms, 29(1):132–141, 1998.

[219] Rongxing Lu, Kevin Heung, Arash Habibi Lashkari, and Ali A Ghorbani. A Lightweight Privacy-Preserving Data Aggregation Scheme for Fog Computing-Enhanced IoT. IEEE Access, 5:3302—-3312, 2017.

[220] Matthew Luckie. Scamper: A scalable and extensible packet prober for active measurement of the internet. In Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC '10, page 239–245, New York, NY, USA, 2010. Association for Computing Machinery.

[221] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and kc claffy. AS Relationships, Customer Cones, and Validation. In Proceedings of the 2013 conference on Internet measurement conference - IMC '13, pages 243–256, Barcelona, Spain, 2013. ACM Press.

[222] Liang Ma, Ting He, Ananthram Swami, Don Towsley, and Kin K Leung. On optimal monitor placement for localizing node failures via network tomography. Performance Evaluation, 91:16–37, September 2015.

[223] Leonardo Maccari and Renato Lo Cigno. Pop-routing: Centrality-based Tuning of Control Messages for Faster Route Convergence. In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, pages 1–9. IEEE, 2016.

[224] Ratul Mahajan, Neil Spring, David Wetherall, and Tom Anderson. Inferring link weights using end-to-end measurements. In Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment, IMW '02, page 231–236, New York, NY, USA, 2002. Association for Computing Machinery.

[225] A Mahimkar, J Yates, Y Zhang, A Shaikh, and J Wang. Network-wide Information Correlation and Exploration (NICE): Framework, Applications, and Experience, 2007.

[226] Ajay Anil Mahimkar, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Qi Zhao. Towards automated performance diagnosis in a large IPTV network. ACM SIGCOMM CCR, 39(4):231, August 2009.

[227] Ajay Anil Mahimkar, Han Hee Song, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Joanne Emmons. Detecting the performance impact of upgrades in large operational networks. ACM SIGCOMM CCR, 40(4):303, August 2010.

[228] David F. Manlove. Algorithmics of Matching Under Preferences, volume 2 of Series on Theoretical Computer Science. WorldScientific, 2013.

[229] P. Marchetta, W. de Donato, V. Persico, and A. Pescapé. Experimenting with alternative path tracing solutions. In Proc. IEEE Symposium on Computers and Communications (ISCC, July 2015.

[230] P. Marchetta and A. Pescapé. DRAGO: Detecting, quantifying and locating hidden routers in traceroute IP paths. In Proc. Global Internet Symposium (GI), April 2013.

[231] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, Yashar Ganjali, and Christophe Diot. Characterization of Failures in an Operational IP Backbone Network. IEEE/ACM Trans. Netw., 16:749–762, August 2008.

[232] Pedro Marques, Rex Fernando, Enke Chen, Pradosh Mohapatra, and Hannes Gredler. Advertisement of the Best External Route in BGP. Internet-Draft draft-ietf-idr-best-external-05, IETF Secretariat, January 2012.

[233] J. Martin and A. Nilsson. On Service Level Agreements for IP Networks. In INFOCOM, 2002.

[234] Ernesto Queirós Vieira Martins. On a multicriteria shortest path problem. European Journal of Operational Research, 16(2):236–245, May 1984.

[235] Ernesto Queirós Vieira Martins and José Luis Esteves Dos Santos. The labelling algorithm for the multiobjective shortest path problem. 1999.

[236] Satoru Matsushima, Clarence Filsfils, Zafar Ali, Zhenbin Li, and Kalyani Rajaraman. Srv6 implementation and deployment status. Internet-Draft draft-matsushima-spring-srv6-deployment-status-11, IETF Secretariat, February 2021. https://www.ietf.org/archive/id/draft-matsushima-spring-srv6-deployment-status-11.txt.

[237] Cristina Mayr, Eduardo Grampín, and Claudio Risso. Optimal route reflection topology design. In Proceedings of the 10th Latin America Networking Conference, LANC '18, page 65–72, New York, NY, USA, 2018. Association for Computing Machinery.

[238] D. McPherson. Intermediate System to Intermediate System (IS-IS) Transient Blackhole Avoidance. RFC 3277, IETF, April 2002.

[239] A. Medem, R. Teixeira, N. Feamster, and Mickael Meulle. Joint Analysis of Network Incidents and Intradomain Routing Changes. In CNSM, 2010.

[240] Michael Menth and Ruediger Martin. Network Resilience through Multi-Topology Routing. In proc. of the Design of Reliable Communication Networks (DRNC) Workshop, October 2005.

[241] Lionel Metongnon, Ramin Sadre, and Eugene C. Ezin. Distributed Middlebox Architecture for IoT Protection. In CNSM. IEEE, oct 2019.

[242] D. Minoli et al. IoT Considerations, Requirements, and Architectures for Smart Buildings—Energy Optimization and Next-Generation Building Management Systems. IEEE Internet of Things Journal, 4(1):269–283, Feb 2017.

[243] R. Misra and C. Mandal. Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks. IEEE Transactions on Parallel and Distributed Systems, 21(3):292–302, 2010.

[244] Jonathan D Moffett and Morris S Sloman. Policy hierarchies for distributed systems management. IEEE Journal on Selected Areas in Communications, 11(9):1404–1414, 1993.

[245] Richard Mortier. PyRT: Python Routeing Toolkit. https://github.com/mor1/pyrt.

[246] Reza Motamedi, Reza Rejaie, and Walter Willinger. A survey of techniques for internet topology discovery. IEEE Communications Surveys & Tutorials, 17:1044–1065, 2015.

[247] Reza Motamedi, Bahador Yeganeh, Balakrishnan Chandrasekaran, Reza Rejaie, Bruce M. Maggs, and Walter Willinger. On mapping the interconnections in today's internet. IEEE/ACM Trans. Netw., 27(5):2056–2070, 2019.

[248] Murtaza Motiwala, Megan Elmore, Nick Feamster, and Santosh Vempala. Path Splicing. In ACM SIGCOMM Computer Communication Review, 2008.

[249] Laxman Muthiyah. Hacking facebook pages, 2018. https://thezerohack.com/hacking-facebook-pages.

[250] Eugene S Myakotnykh, Otto J Wittner, Bjarne E Helvik, Atef Abdelkefi, Jon Kåre Hellan, Olav Kvittem, Trond Skjesol, and Arne Øslebø. An analysis of interdomain availability and causes of failures based on active measurements. Telecommunication Systems, September 2011.

[251] Ryo Nakamura, Kazuki Shimizu, Teppei Kamata, and Cristel Pelsser. A first measurement with bgp egress peer engineering. In Passive and Active Measurement, pages 199–215, Cham, 2022. Springer International Publishing.

[252] João Carlos Namorado Climaco and Ernesto Queirós Vieira Martins. A bicriterion shortest path algorithm. European Journal of Operational Research, 11(4):399–404, Dec 1982.

[253] Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu, Zhi-Li Zhang, and Chen-Nee Chuah. Fast local rerouting for handling transient link failures. IEEE/ACM Transactions on Networking, 15(2):359–372, 2007.

[254] Nick Galov. Cloud Adoption Statistics for 2020, January 2020. https://hostingtribunal.com/blog/cloud-adoption-statistics/.

[255] Bin Niu, Jiawei Kong, Shaofei Tang, Yingcong Li, and Zuqing Zhu. Visualize your ip-over-optical network in realtime: A p4-based flexible multilayer in-band network telemetry (ml-int) system. IEEE Access, 7:82413–82423, 2019.

[256] Nordic Gateway for Research and Education. GÉant topology map. https://www.nordu.net/content/géant, January 2014.

[257] Yevheniya Nosyk, Maciej Korczyński, and Andrzej Duda. Routing loops as mega amplifiers for dns-based ddos attacks. In Passive and Active Measurement, PAM, Berlin, Heidelberg, 2022. Springer-Verlag.

[258] Yasuhiro Ohara, Shinji Imahori, and Rodney Van Meter. MARA: Maximum Alternative Routing Algorithm. In IEEE INFOCOM, 2009.

[259] D. Oran. OSI IS-IS Intra-domain Routing Protocol. RFC 1142, IETF, February 1990.

[260] E. Ilker Oyman. Multiple Sink Location Problem and Energy Efficiency in Large Scale Wireless Sensor Networks. PhD thesis, Bogazici University, 2004.

[261] J. Paixão and J. L. Santos. A new ranking path algorithm for the multi-objective shortest path problem. 2008. https://www.semanticscholar.org/paper/A-new-ranking-path-algorithm-for-the-shortest-path-Paix%C3%A3o-Santos/d113dc9676607aa8018e972220f6cbb70838146d.

[262] Jean-Jacques Pansiot and Dominique Grad. On routes and multicast trees in the internet. SIGCOMM Comput. Commun. Rev., 28(1):41–50, jan 1998.

[263] C.H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In Proceedings 41st Annual Symposium on Foundations of Computer Science, page 86–92, Nov 2000.

[264] Mahmoud Parham, Thomas Fenz, Nikolaus Süss, Klaus-Tycho Foerster, and Stefan Schmid. Traffic engineering with joint link weight and segment optimization. In Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '21, page 313–327, New York, NY, USA, 2021. Association for Computing Machinery.

[265] R. Pastor-Satorras and A. Vespignani. Evolution and Structure of the Internet: A Statistical Physics Approach. Cambridge University Press, 2004.

[266] Dan Pei, Matt Azuma, Dan Massey, and Lixia Zhang. BGP-RCN: Improving BGP Convergence through Root Cause Notification. Computer Networks, 48(2):175–194, 2005.

[267] C. Pelsser, T. Takeda, E. Oki, and K. Shiomoto. Improving Route Diversity through the Design of iBGP Topologies. In 2008 IEEE International Conference on Communications, pages 5732–5738, Beijing, China, 2008. IEEE.

[268] Cristel Pelsser, Luca Cittadini, Stefano Vissicchio, and Randy Bush. From Paris to Tokyo: On the Suitability of ping to Measure Latency. In IMC, 2013.

[269] Igor Polyantchikov, Eduard Shevtshenko, Tatjana Karaulova, Taivo Kangilaski, and Luis M Camarinha-Matos. Virtual enterprise formation in the context of a sustainable partner network. Industrial management & data systems, 2017.

[270] P. Pongpaibool, R. Doverspike, M. Roughan, and J. Gottlieb. Handling IP Traffic Surges via Optical Layer Reconfiguration. In OFC, 2002.

[271] Sumet Prabhavat, Hiroki Nishiyama, Nirwan Ansari, and Nei Kato. On load distribution over multipath networks. IEEE Communications Surveys & Tutorials, 14(3):662–680, 2011.

[272] K Pradeep, OYK Alani, et al. Reducing BGP Convergence Time by Fine Tuning the MRAI Timer on Different Topologies. In 13th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting PGNET 2012. The School of Computing and Mathematical Sciences, Liverpool John Moores, 2012.

[273] GSMA Future Networks Programme. Network slicing : Use case requirements. https://www.gsma.com/futurenetworks/wp-content/uploads/2020/01/2.0_Network-Slicing-Use-Case-Requirements-1.pdf. (Accessed on 05/26/2021).

[274] P. Psenak, S. Hegde, C. Filsfils, and A. Gulko. ISIS segment routing flexible algorithm. Internet Draft (Work in Progress) draft-hegdeppsenak-isis-sr-flex-algo-02, Internet Engineering Task Force, February 2018.

[275] Peter Psenak, Shraddha Hegde, Clarence Filsfils, Ketan Talaulikar, and Arkadiy Gulko. Igp flexible algorithm. Internet-Draft draft-ietf-lsr-flex-algo-07, IETF Secretariat, April 2020. http://www.ietf.org/internet-drafts/draft-ietf-lsr-flex-algo-07.txt.

[276] Himabindu Pucha, Ying Zhang, Z. Morley Mao, and Y. Charlie Hu. Understanding Network Delay Changes Caused by Routing Events. SIGMETRICS Performance Evaluation Review, 35(1):73–84, June 2007.

[277] Andrea Raith and Matthias Ehrgott. A comparison of solution strategies for biobjective shortest path problems. Computers & Operations Research, 36(4):1299–1331, Apr 2009.

[278] Alex Raj and Oliver C. Ibe. A Survey of IP and Multiprotocol Label Switching Fast Reroute Schemes. Computer Networks, 51(8):1882 – 1907, 2007.

[279] Dinesha Ranathunga, Matthew Roughan, and Hung Nguyen. Verifiable policy-defined networking using metagraphs. IEEE Transactions on Dependable and Secure Computing, 2020.

[280] Robert Raszuk, Bruno Decraene, Christian Cassar, Erik Aman, and Kevin Wang. BGP Optimal Route Reflection (BGP ORR). RFC 9107, August 2021.

[281] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Characterizing ICMP Rate Limitation on Routers. In ICC, 2015.

[282] Ravi Ravindran, Yanyong Zhang, Luigi Grieco, Anders Lindgren, Jeff Burke, Bengt Ahlgren, and Aytac Azgin. Design Considerations for Applying ICN to IoT. Technical report, ICN Research Group, 2019.

[283] D.S. Reeves and H.F. Salama. A distributed algorithm for delay-constrained unicast routing. IEEE/ACM Transactions on Networking, 8(2):239–250, Apr 2000.

[284] Glenn Robertson, Nirupam Roy, Phani Krishna Penumarthi, Srihari Nelakuditi, and Jason M. O'Kane. Loop-free convergence with unordered updates. IEEE Transactions on Network and Service Management, 14(2):373–385, 2017.

[285] E. Rosen and Y. Rekhter. BGP/MPLS IP virtual private networks (VPNs). RFC 4364, Internet Engineering Task Force, February 2006.

[286] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta. MPLS label stack encoding. RFC 3032, Internet Engineering Task Force, January 2001.

[287] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, Internet Engineering Task Force, January 2001.

[288] Ćiril Rožić and Galen Sasaki. Optical protection cost of loop free alternates on completely connected ip networks over optical rings. IEEE/ACM Transactions on Networking, 29(3):1116–1127, 2021.

[289] Nicolas Rybowski and Olivier Bonaventure. Evaluating ospf convergence with ns-3 dce. In Workshop on ns-3 (WNS3), 2022.

[290] Gábor Rétvári, János Tapolcai, Gábor Enyedi, and András Császár. Ip fast reroute: Loop free alternates revisited. In 2011 Proceedings IEEE INFOCOM, pages 2948–2956, 2011.

[291] Sartaj Sahni. General techniques for combinatorial approximation. Operations Research, 25(6):920–936, 1977.

[292] Amit Sahoo, Krishna Kant, and Prasant Mohapatra. BGP Convergence Delay After Multiple Simultaneous Router Failures: Characterization and Solutions. Computer Communications, 32(7-10):1207–1218, May 2009.

[293] Sebastian Sampayo, Julien Montavont, and Thomas Noël. REFLOOD: Reactive Routing Protocol for Wake-Up Radio in IoT. Elsevier Ad Hoc Networks, 121, 2021.

[294] Sunny Sanyal and Puning Zhang. Improving quality of data: Iot data aggregation using device to device communications. IEEE Access, 6:67830–67840, 2018.

[295] Sebastian Lucas Sapamyo. Polymorphic network protocol suite in heterogeneous wake-up IoT networks. Ph.d. thesis, 2021.

[296] Pushpasis Sarkar, Stephane Litkowski, Bruno Decraene, Clarence Filsfils, Syed Raza, and Martin Horneffer. Operational Management of Loop-Free Alternates. RFC 7916, July 2016.

[297] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas Anderson. The end-to-end effects of internet path selection. SIGCOMM Comput. Commun. Rev., 29(4):289–299, August 1999.

[298] Klaus Schneider, Beichuan Zhang, and Lotfi Benmohamed. Hop-by-hop multipath routing: Choosing the right nexthop set. In IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, pages 2273–2282, 2020.

[299] Tibor Schneider, Rüdiger Birkner, and Laurent Vanbever. Snowcap: Synthesizing network-wide configuration updates. In Proceedings of the 2021 ACM SIGCOMM 2021 Conference, SIGCOMM '21, page 33–49, New York, NY, USA, 2021. Association for Computing Machinery.

[300] J. Scudder, R. Fernando, and S. Stuart. BGP Monitoring Protocol (BMP). RFC 8671, RFC Editor, June 2016.

[301] Aman Shaikh, Rohit Dube, and Anujan Varma. Avoiding Instability During Graceful Shutdown of Multiple OSPF Routers. IEEE/ACM Trans. Netw., 14(3):532–542, June 2006.

[302] Aman Shaikh and Albert G Greenberg. OSPF Monitoring - Architecture, Design, and Deployment Experience. NSDI, 2004.

[303] M. Shand and S. Bryant. A Framework for Loop-Free Convergence. RFC 5715, IETF, January 2010.

[304] Mike Shand and Stewart Bryant. IP Fast Reroute Framework. RFC 5714, January 2010.

[305] Wentao Shang et al. Named Data Networking of Things. In IoTDI, pages 117–128. IEEE, apr 2016.

[306] Prateek Shantharama, Akhilesh S. Thyagaturu, and Martin Reisslein. Hardware-accelerated platforms and infrastructures for network functions: A survey of enabling technologies and research studies. IEEE Access, 8:132021–132085, 2020.

[307] R. Sherwood, A. Bender, and N. Spring. Discarte: a disjunctive Internet cartographer. In Proc. ACM SIGCOMM, August 2008.

[308] R. Sherwood and N. Spring. Touring the internet in a TCP sidecar. In Proc. ACM Internet Measurement Conference (IMC), October 2006.

[309] Abhimanyu Venkatraman Sheshashayee and Stefano Basagni. Multi-Hop Wake-Up Radio Relaying for the Collection Tree Protocol. In 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall), pages 1–6, Honolulu, HI, USA, September 2019. IEEE.

[310] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5):637–646, Oct 2016.

[311] Xiaokui Shu, Danfeng Yao, and Elisa Bertino. Privacy-preserving detection of sensitive data exposure. IEEE transactions on Information Forensics and Security, 10(5):1092–1103, 2015.

[312] Sunil Kumar Singh, Prabhat Kumar, and Jyoti Prakash Singh. A Survey on Successors of LEACH Protocol. IEEE Access, 5:4298–4328, 2017.

[313] J. L. Sobrinho, D. Fialho, and P. Mateus. Stabilizing BGP through distributed elimination of recurrent routing loops. In 2017 IEEE 25th International Conference on Network Protocols (ICNP), volume 00, pages 1–10, Oct. 2017.

[314] João Luís Sobrinho, Laurent Vanbever, Franck Le, and Jennifer Rexford. Distributed Route Aggregation on the Global Network. In Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, CoNEXT '14, page 161–172, New York, NY, USA, 2014. Association for Computing Machinery.

[315] Joao Luis Sobrinho, Laurent Vanbever, Franck Le, Andre Sousa, and Jennifer Rexford. Scaling the Internet routing system through distributed route aggregation. IEEE/ACM Trans. Netw., 24(6):3462–3476, December 2016.

[316] Roberto Solis-Oba. 2-Approximation algorithm for finding a spanning tree with maximum number of leaves. In Algorithms - ESA' 98, volume 29, pages 132–141. Springer Berlin / Heidelberg edition, 1998.

[317] J. Sommers, B. Eriksson, and P. Barford. On the prevalence and characteristics of MPLS deployments in the open Internet. In Proc. ACM Internet Measurement Conference (IMC), November 2011.

[318] Meongchul Song and S. Sahni. Approximation algorithms for multiconstrained quality-of-service routing. IEEE Transactions on Computers, 55(5):603–617, May 2006.

[319] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In Proc. ACM SIGCOMM, August 2002.

[320] Ashwin Sridharan, Sue B. Moon, and Christophe Diot. On the correlation between route dynamics and routing loops. In Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement, IMC '03, page 285–294, New York, NY, USA, 2003. Association for Computing Machinery.

[321] I. Stojmenović, M. Seddigh, and J. Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. TDPS, 13(1):14–25, 2001.

[322] Stephen D. Strowes. Visibility of ipv4 and ipv6 prefix lengths in 2019. https://labs.ripe.net/Members/stephen_strowes/visibility-of-prefix-lengths-in-ipv4-and-ipv6.

[323] Latanya Sweeney. k-Anonymity: a Model for Protecting Privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10, oct 2002.

[324] R. Teixeira and J. Rexford. Managing Routing Disruptions in Internet Service Provider Networks. IEEE Communications Magazine, 44(3):160 – 165, March 2006.

[325] R. Teixeira, A. Shaikh, T.G. Griffin, and J. Rexford. Impact of Hot-Potato Routing Changes in IP Networks. IEEE/ACM Transactions on Networking, 16(6):1295–1307, December 2008.

[326] Mikkel Thorup and Uri Zwick. Compact Routing Schemes. In Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '01, page 1–10, New York, NY, USA, 2001. Association for Computing Machinery.

[327] D. Tian and N. D. Georganas. Connectivity maintenance and coverage preservation in wireless sensor networks. Ad Hoc Networks Journal (Elsevier), 3:744–761, November 2005.

[328] Olivier Tilmans, Tobias Bühler, Ingmar Poese, Stefano Vissicchio, and Laurent Vanbever. Stroboscope: Declarative network monitoring on a budget. In Sujata Banerjee and Srinivasan Seshan, editors, 15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018, pages 467–482. USENIX Association, 2018.

[329] Andree Toonk. What caused today's Internet hiccup. https://www.bgpmon.net/what-caused-todays-internet-hiccup/.

[330] M. E. Tozal and K. Sarac. TraceNET: an Internet topology data collector. In Proc. ACM Internet Measurement Conference (IMC), November 2010.

[331] George Tsaggouris and Christos Zaroliagis. Multiobjective optimization: Improved fptas for shortest paths and non-linear objectives with applications. Theory of Computing Systems, 45(1):162–186, Jun 2009.

[332] Steve Uhlig and Sebastien Tandel. Quantifying the BGP Routes Diversity Inside a Tier-1 Network. In NETWORKING, pages 1002–1013, 2006.

[333] Jim Uttaro, Pierre Francois, Keyur Patel, Jeffrey Haas, Adam Simpson, and Roberto Fragassi. Best Practices for Advertisement of Multiple Paths in IBGP. Internet-Draft draft-ietf-idr-add-paths-guidelines-08, IETF Secretariat, April 2016.

[334] Fulvio Valenza, Cataldo Basile, Daniele Canavese, and Antonio Lioy. Classification and analysis of communication protection policy anomalies. IEEE/ACM Transactions on Networking, 25(5):2601–2614, 2017.

[335] Virginie Van den Schrieck, Pierre Francois, and Olivier Bonaventure. BGP Add-Paths: The Scaling/Performance Tradeoffs. IEEE Journal on Selected Areas in Communications, 28(8):1299–1307, October 2010.

[336] P. Van Mieghem and F. A. Kuipers. Concepts of exact qos routing algorithms. IEEE/ACM Transactions on Networking, 12(5):851–864, 2004.

[337] Laurent Vanbever, Stefano Vissicchio, Cristel Pelsser, Pierre Francois, and Olivier Bonaventure. Lossless Migrations of Link-State IGPs. IEEE/ACM Trans. Netw., 20(6):1842–1855, December 2012.

[338] D. Veitch, B. Augustin, R. Teixeira, and T. Friedman. Failure control in multipath route tracing. In IEEE INFOCOM 2009, pages 1395–1403, 2009.

[339] Pier Luigi Ventre, Stefano Salsano, Marco Polverini, Antonio Cianfrani, Ahmed Abdelsalam, Clarence Filsfils, Pablo Camarillo, and Francois Clad. Segment Routing: a Comprehensive Survey of Research Activities, Standardization Efforts and Implementation Results. arXiv:1904.03471 [cs], July 2020. arXiv: 1904.03471.

[340] Kevin Vermeulen, Justin P Rohrer, Robert Beverly, Olivier Fourmaux, and Timur Friedman. Diamond-miner: Comprehensive discovery of the internet's topology diamonds. In 17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI}), pages 479–493, 2020.

[341] Kevin Vermeulen, Stephen D Strowes, Olivier Fourmaux, and Timur Friedman. Multilevel mda-lite paris traceroute. In Proceedings of the Internet Measurement Conference 2018, pages 29–42, 2018.

[342] Fabien Viger, Brice Augustin, Xavier Cuvellier, Clémence Magnien, Matthieu Latapy, Timur Friedman, and Renata Teixeira. Detection, understanding, and prevention of traceroute measurement artifacts. Computer Networks, 52(5):998 – 1018, 2008.

[343] Stefano Vissicchio, Luca Cittadini, and Giuseppe Di Battista. On IBGP Routing Policies. IEEE/ACM Trans. Netw., 23(1):227–240, February 2015.

[344] Stefano Vissicchio, Luca Cittadini, Maurizio Pizzonia, Luca Vergantini, Valerio Mezzapesa, and Maria Luisa Papagni. Beyond the best: Real-time non-invasive collection of BGP messages. In Internet Network Management Workshop, San Jose, CA, USA, April 2010.

[345] Stefano Vissicchio, Luca Cittadini, Laurent Vanbever, and Olivier Bonaventure. iBGP deceptions: More sessions, fewer routes. In INFOCOM, pages 2122–2130, March 2012.

[346] Stefano Vissicchio, Laurent Vanbever, Luca Cittadini, Geoffrey G. Xie, and Olivier Bonaventure. Safe update of hybrid sdn networks. IEEE/ACM Transactions on Networking, 25(3):1649–1662, 2017.

[347] Antonio P Volpentesta. Hypernetworks in a directed hypergraph. European Journal of Operational Research, 188(2):390–405, 2008.

[348] Vulners. Razer us: Database credentials lea, 2017. https://vulners.com/hackerone/H1:293470.

[349] Srinivas Naga Vutukury. Multipath Routing Mechanisms for Traffic Engineering and Quality of Service in the Internet. PhD thesis.

[350] Isabel Wagner and David Eckhoff. Technical Privacy Metrics: A Systematic Survey. ACM Computing Surveys, 51(3):57:1–57:38, 2018.

[351] Feng Wang, Zhuoqing Morley Mao, Jia Wang, Lixin Gao, and Randy Bush. A measurement study on the impact of routing events on end-to-end internet path performance. ACM SIGCOMM CCR, 36(4):375, August 2006.

[352] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. COPE: Traffic Engineering in Dynamic Networks. In ACM SIGCOMM, 2006.

[353] Ron Widyono and Tenet Group. The design and evaluation of routing algorithms for real-time channels, 1994.

[354] Jie Wu, Fei Dai, Ming Gao, and Ivan Stojmenovic. On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. Journal of communications and networks, 4(1):59–70, 2002.

[355] Jie Wu and Hailan Li. A Dominating-Set-Based routing scheme in ad hoc wireless networks. Telecommunication Systems, 18(1):13–36, 2001.

[356] Xin Yuan and Xingming Liu. Heuristic algorithms for multi-constrained quality of service routing. In Proceedings IEEE INFOCOM 2001, volume 2, pages 844–853 vol.2, 2001.

[357] X. Yang and D. Wetherall. Source Selectable Path Diversity via Routing Deflections. In ACM SIGCOMM, 2006.

[358] Keiichi Yasumoto, Hirozumi Yamaguchi, and Hiroshi Shigeno. Survey of Real-time Processing Technologies of IoT Data Streams. Journal of Information Processing, 24(2):195–202, 2016.

[359] Halil Yetgin, Kent Tsz Kan Cheung, Mohammed El-Hajjar, and Lajos Hanzo Hanzo. A survey of network lifetime maximization techniques in wireless sensor networks. IEEE Communications Surveys Tutorials, 19(2):828–854, 2017.

[360] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. Comput. Netw., 52(12):2292–2330, aug 2008.

[361] Dingwen Yuan, Salil S. Kanhere, and Matthias Hollick. Instrumenting wireless sensor networks — a survey on the metrics that matter. Pervasive and Mobile Computing, 37:45–62, 2017.

[362] Zirak Zaheer, Hyunseok Chang, Sarit Mukherjee, and Jacobus Van der Merwe. eztrust: Network-independent zero-trust perimeterization for microservices. In Proceedings of the 2019 ACM Symposium on SDN Research, pages 49–61, 2019.

[363] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. IEEE Internet of Things Journal, 1(1):22–32, Feb 2014.

[364] H. Zhang and J. C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. AHSWN, 1:89–123, 2005.

[365] L. Zhang et al. Named data networking. ACM SIGCOMM Computer Communication Review, 44(3):66–73, jul 2014.

[366] Y. Zhang, Z. M. Mao, and J. Wang. A framework for measuring and predicting the impact of routing changes. In Proc. IEEE INFOCOM, May 2007.

[367] Yuanxun Zhang, Saptarshi Debroy, and Prasad Calyam. Network-wide anomaly event detection and diagnosis with perfsonar. IEEE Trans. Netw. Serv. Manag., 13(3):666–680, 2016.

[368] Xiaoliang Zhao, Dante J Pacella, and Jason Schiller. Routing scalability: an operator's view. IEEE Journal on Selected Areas in communications, 28(8):1262–1270, 2010.

[369] Zheng Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. IEEE Journal on Selected Areas in Communications, 14(7):1228–1234, 1996.

[370] Jianzhong Zhou. A new distributed routing algorithm for supporting delay-sensitive applications. In ICCT'98. 1998 International Conference on Communication Technology. Proceedings (IEEE Cat. No.98EX243), volume 2, pages 7 pp. vol.2–, Oct 1998.

[371] Chaoshun Zuo, Zhiqiang Lin, and Yinqian Zhang. Why does your data leak? uncovering the data leakage in cloud from mobile apps. In 2019 IEEE Symposium on Security and Privacy (SP), pages 1296–1310. IEEE, 2019.

Thesis References

[Alfroy, 2020] Alfroy, T. (2020). Towards a PoC for an Optimal Protection Technique for Inter-intra-domain Convergence. Master's thesis, Université de Strasbourg. Thèse de Master 2 dirigée par Mérindol, Pascal - Informatique.

[Clad, 2011] Clad, F. (2011). Diffusion Efficace dans les Réseaux de Capteurs Économes en Énergie. Master's thesis, Université de Strasbourg. Thèse de Master 2 dirigée par Mérindol, Pascal & Gallais, Antoine - Informatique.

[Clad, 2014] Clad, F. (2014). Disruption-free Routing Convergence: Computing Minimal Link-state Update Sequences. PhD thesis, Université de Strasbourg. Thèse de Doctorat dirigée par Pansiot, Jean-Jacques - Informatique.

[Del Fiore, 2021] Del Fiore, J. M. (2021). Detecting Hidden Broken Pieces of the Internet: BGP Lies, Forwarding Detours and Failed IXPs. PhD thesis, Université de Strasbourg. Thèse de Doctorat dirigée par Pelsser, Cristel - Informatique.

[Ehlinger, 2017] Ehlinger, T. (2017). IP Monitoring for Multi-Domain Network Service. Master's thesis, Université de Strasbourg. Thèse de Master 2 dirigée par Mérindol, Pascal - Informatique.

[Gamba, 2017] Gamba, J. (2017). (i)BGP Anomalies, Towards New Guidelines. Master's thesis, Université de Strasbourg. Thèse de Master 2 dirigée par Mérindol, Pascal & Pelsser, Cristel - Informatique.

[Guillot, 2018] Guillot, A. (2018). Using Internet Background Radiation Traffic to Detect Outages. Master's thesis, Université de Strasbourg. Thèse de Master 2 dirigée par Mérindol, Pascal & Pelsser, Cristel - Informatique.

[Holterbach, 2014] Holterbach, T. (2014). Micro-boucles de Routage et Stabilité. Master's thesis, Université de Strasbourg. Thèse de Master 2 dirigée par Mérindol, Pascal - Informatique.

[Luttringer, 2019] Luttringer, J.-R. (2019). Inter- and Intra-domain Routing Interactions: Towards Efficient Overall Updates. Master's thesis, Université de Strasbourg. Thèse de Master 2 dirigée par Mérindol, Pascal - Informatique.

[Luttringer, 2022] Luttringer, J.-R. (soutenance prévue en 2022). Computing and Deploying Forwarding Paths for IP Networks: From Reliable Hot-Potato Routing to Delay Constrained Least Cost Tunnels. (Tentative title). PhD thesis, Université de Strasbourg. Thèse de Doctorat dirigée par Pelsser, Cristel - Informatique.

[Marchetta, 2010] Marchetta, P. (2010). An Internet Topology Discovery Approach based on Multicast. Master's thesis, Université de Naples (Italie) - Louvain la Neuve (Belgique). Thèse de Master 2 dirigée par Mérindol, Pascal, Donnet, Benoit & Antonio Pescape - Informatique.

[Miller, 2022] Miller, L. (2022). Securing Workflows: On the Use of Microservices and Metagraphs to Prevent Data Exposures. PhD thesis, Université de Strasbourg. Thèse de Doctorat dirigée par Pelsser, Cristel & Gallais, Antoine - Informatique.

[Neto, 2022] Neto, R. (soutenance prévue en 2022). Privacy-aware Aggregation of IoT Streams in Multi-Owner Infrastructures. PhD thesis, Université de Strasbourg. Thèse de Doctorat dirigée par Theoleyre, Fabrice - Informatique.

[Otto, 2016] Otto, J. (2016). On the Impact of Path Changes on TCP Flows. Master's thesis, Université de Strasbourg. Thèse de Master 2 dirigée par Mérindol, Pascal, Pelsser, Cristel & Cateloin, Stéphane - Informatique.

[Rodau, 2017] Rodau, A. (2017). An Algebraic Theoritical Framework for Path-vector Protocols. Master's thesis, Université de Strasbourg. Stage de Magistère dirigée par Mérindol, Pascal & Pelsser, Cristel - Mathématique.

[Vanaubel, 2018] Vanaubel, Y. (2018). Revealing and Characterizing MPLS Networks. PhD thesis, Université de Liège. Thèse de Doctorat dirigée par Donnet, Benoit - Informatique.

Personal References

[CGM12] Francois Clad, Antoine Gallais, and Pascal Mérindol. Energy-efficient Data Collection in WSN: A Sink-oriented Dynamic Backbone. In International Conference on Communications (IEEE ICC), 2012.

[CMP+14] Francois Clad, Pascal Mérindol, Jean-Jacques Pansiot, Pierre François, and Olivier Bonaventure. Graceful Convergence in Link-State IP Networks - A Lightweight Algorithm Ensuring Minimal Operational Impact. IEEE/ACM Transactions on Networking, 2014.

[CMV+13] Francois Clad, Pascal Mérindol, Stefano Vissicchio, Jean-Jacques Pansiot, and Pierre François. Graceful Router Updates in Link-state Protocols. International Conference on Network Protocols (IEEE ICNP), 2013.

[CVM+15] Francois Clad, Stefano Vissicchio, Pascal Mérindol, Pierre François, and Jean-Jacques Pansiot. Computing Minimal Update Sequences for Graceful Router-Wide Reconfigurations. IEEE/ACM Transations on Networking, 2015.

[DFMP+19] Julián Martin Del Fiore, Pascal Mérindol, Valerio Persico, Cristel Pelsser, and Antonio Pescapè. Filtering the Noise to Reveal Inter-Domain Lies. Traffic & Measurement Analysis (IFIP TMA), 2019.

[DFPM+21] Julián Martin Del Fiore, Valerio Persico, Pascal Mérindol, Cristel Pelsser, and Antonio Pescape. The Art of Detecting Forwarding Detours. IEEE Transactions on Network and Service Management, 2021.

[DLMP12] Benoit Donnet, Matthew Luckie, Pascal Mérindol, and Jean-Jacques Pansiot. Revealing MPLS Tunnels Obscured from Traceroute. ACM Sigcomm Computer Communication Review, 2012.

[DMP+19] Julian Martin Del Fiore, Pascal Mérindol, Valerio Persico, Cristel Pelsser, and Antonio Pescapé. Filtering the noise to reveal inter-domain lies. In 2019 Network Traffic Measurement and Analysis Conference (TMA), pages 17–24, June 2019.

[DPvdL+13] Gregory Detal, Christoph Paasch, Simon van der Linden, Pascal Mérindol, Gildas Avoine, and Olivier Bonaventure. Revisiting Flow-based Load Balancing: Stateless Path Selection in Data Center Networks. Elsevier Computer Networks, 2013.

[GFW+19] Andreas Guillot, Romain Fontugne, Philipp Winter, Pascal Mérindol, Alistair King, Alberto Dainotti, and Cristel Pelsser. Chocolatine - Outage Detection for Internet Background Radiation. Traffic & Measurement Analysis (IFIP TMA), 2019.

[LAM+20] Jean-Romain Luttringer, Thomas Alfroy, Pascal Mérindol, Quentin Bramas, Francois Clad, and Cristel Pelsser. Computing Delay-Constrained Least-Cost Paths for Segment Routing is Easier Than You Think. Network Computing and Applications (IEEE NCA), 2020.

[LAM+22] Jean-Romain Luttringer, Thomas Alfroy, Pascal Mérindol, Quentin Bramas, François Clad, and Cristel Pelsser. Deploying Near Optimal Delay Constrained Paths with Segment-Routing in Massive Scale Networks. Elsevier Computer Networks, 2022.

[LBPM21] Jean-Romain Luttringer, Quentin Bramas, Cristel Pelsser, and Pascal Mérindol. A Fast-Convergence Routing of the Hot-Potato. IEEE INFOCOM, 2021.

[LVM+19] Jean-Romain Luttringer, Yves Vanaubel, Pascal Mérindol, Jean-Jacques Pansiot, and Benoit Donnet. Let There Be Light: Revealing Hidden MPLS Tunnels with TNT. IEEE Transactions on Network and Service Management, pages 1–1, 2019.

[MCP09] Pascal Mérindol, Stéphane Cateloin, and Jean-Jacques Pansiot. Low Complexity Link State Multipath Routing. Global Internet Symposium (IEEE GIS), INFOCOM Workshops, 2009.

[MDBP10] Pascal Mérindol, Benoit Donnet, Olivier Bonaventure, and Jean.-Jacques. Pansiot. On the impact of layer-2 on node degree distribution. In Proc. ACM Internet Measurement Conference (IMC), November 2010.

[MDP+11] Pascal Mérindol, Benoit Donnet, Jean-Jacques Pansiot, Matthew J Luckie, and Young Hyun. MERLIN - MEasure the router level of the INternet. 7th EURO-NGI Conference on Next Generation Internet, 2011.

[MDP+18] Pascal Mérindol, Pierre David, Jean-Jacques Pansiot, Francois Clad, and Stefano Vissicchio. A Fine-grained Multi-source Measurement Platform Correlating Routing Transitions with Packet Losses. Elsevier Computer Communications, 2018.

[MFB+11] Pascal Mérindol, Pierre Francois, Olivier Bonaventure, Stéphane Cateloin, and Jean-Jacques Pansiot. An Efficient Algorithm to Enable Path Diversity in Link State Routing Networks. Elsevier Computer Networks, 2011.

[MG09] Pascal Mérindol and Antoine Gallais. Path Diversity in Energy-efficient Wireless Sensor Networks. IEEE PIMRC, 2009.

[MMD+10] Pietro Marchetta, Pascal Mérindol, Benoit Donnet, Antonio Pescapè, and Jean-Jacques Pansiot. Topology Discovery at the Router Level - A New Hybrid Tool Targeting ISP Networks. IEEE Journal on Selected Areas in Communications, 2010.

[MMD+12] Pietro Marchetta, Pascal Mérindol, Benoit Donnet, Antonio Pescapè, and Jean-Jacques Pansiot. Quantifying and Mitigating IGMP Filtering in Topology Discovery. IEEE GLOBECOM, 2012.

[MMD22] Emeline Marechal, Pascal Mérindol, and Benoit Donnet. ISP probing reduction with anaximander. In Passive and Active Measurement, Springer PAM, 2022.

[MMGP21a] Loïc Miller, Pascal Mérindol, Antoine Gallais, and Cristel Pelsser. Securing workflows using microservices and metagraphs. Electronics MDPI, 2021.

[MMGP21b] Loïc Miller, Pascal Mérindol, Antoine Gallais, and Cristel Pelsser. Towards Secure and Leak-Free Workflows Using Microservice Isolation. International Conference on High Performance Switching and Routing (IEEE HPSR), 2021.

[MMGP21c] Loïc Miller, Pascal Mérindol, Antoine Gallais, and Cristel Pelsser. Verification of Cloud Security Policies. International Conference on High Performance Switching and Routing (IEEE HPSR), 2021.

[MVdSD+09] Pascal Mérindol, Virginie Van den Schrieck, Benoit Donnet, Olivier Bonaventure, and Jean-Jacques Pansiot. Quantifying ASes Multiconnectivity using Multicast Information. Internet Measurement Conference (ACM IMC), 2009.

[NMGT21] Renato Neto, Pascal Mérindol, Antoine Gallais, and Fabrice Theoleyre. Scalability of LPWAN for Smart City Applications. International Wireless Communications and Mobile Computing (IEEE IWCMC), 2021.

[NMT20a] Renato Neto, Pascal Mérindol, and Fabrice Théoleyre. Enabling Privacy by Aggregation with Mulltidomain IoT streams. Local Computer Networks (IEEE LCN), 2020.

[NMT20b] Renato Neto, Pascal Mérindol, and Fabrice Theoleyre. Transformation Based Routing Overlay for Privacy and Reusability in Multi-Domain IoT. Network Computing and Applications (IEEE NCA), 2020.

[NMT21] Renato Neto, Pascal Mérindol, and Fabrice Theoleyre. Data Aggregation for Privacy Protection of Data Streams Between Autonomous IoT Networks. In Symposium on Computers and Communications (IEEE ISCC), 2021.

[PMDB10] Jean-Jacques Pansiot, Pascal Mérindol, Benoit Donnet, and Olivier Bonaventure. Extracting Intra-Domain Topology from mrinfo Probing. Passive and Active Measurements (Springer PAM), 2010.

[TQM+13] Fabien Tarissan, Bruno Quoitin, Pascal Mérindol, Benoit Donnet, Jean-Jacques Pansiot, and Matthieu Latapy. Towards a Bipartite Graph Modeling of the Internet Topology. Elsevier Computer Networks, 2013.

[VBD+20] Pavle Vuletić, Bartosz Bosak, Marinos Dimolianis, Pascal Mérindol, David Schmitz, and Henrik Wessing. Localization of Network Service Performance Degradation in Multi-tenant Networks. Elsevier Computer Networks, 2020.

[VLM+19] Yves Vanaubel, Jean-Romain Luttringer, Pascal Mérindol, Jean-Jacques Pansiot, and Benoit Donnet. TNT, Watch me Explode - A Light in the Dark for Revealing MPLS Tunnels. Traffic & Measurement Analysis (IFIP TMA), 2019.

[VMPD15] Yves Vanaubel, Pascal Mérindol, Jean-Jacques Pansiot, and Benoit Donnet. MPLS Under the Microscope. In Internet Measurement Conference (ACM IMC), New York, New York, USA, 2015.

[VMPD16] Yves Vanaubel, Pascal Mérindol, Jean-Jacques Pansiot, and Benoit Donnet. A Brief History of MPLS Usage in IPv6. Passive and Active Measurements (Springer PAM), 2016.

[VMPD17] Yves Vanaubel, Pascal Mérindol, Jean-Jacques Pansiot, and Benoit Donnet. Through the Wormhole - Tracking Invisible MPLS Tunnels. Internet Measurement Conference (ACM IMC), 2017.

[VPMD13] Yves Vanaubel, Jean-Jacques Pansiot, Pascal Mérindol, and Benoit Donnet. Network fingerprinting - TTL-based router signatures. Internet Measurement Conference (ACM IMC), 2013.

Acronyms

AGBA Adjusted Greedy Backward Algorithm. 3, 13, 39, 44

APSP All Pairs Shortest Path. 15

AS Autonomous System. 6, 10

BEST2COP Best Exact Segment Track for 2-Constrained Optimal Paths. 3

BGP Border Gateway Protocol. 6, 7, 10, 180

DC Downstream Criteria. 27

DCART Dynamic Changes Analysis with Routing Traces. 3, 89

DCLC Delay Constraint Least Cost. 8, 68, 176, 185

ECMP Equal Cost MultiPath. 6

FD Forwarding Detours. 3, 89

FIB Forwarding Information Base. 10, 41

FRR Fast Re-Routing. 180

IGP Internal Gateway Protocol. 6, 7, 21

IoT Internet of Things. 2, 4

IP Internet Protocol. 2, 180

IS-IS Intermediate System to Intermediate System. 180

ISP Internet Service Providers. 6, 9, 177

LDP Label Distribution Protocol. 6, 180

LFA Loop Free Alternate. 16

LQI Link Quality Indicator. 173

LR Long Range. 166

LSA Link-State Advertisements. 39–41

MAC Medium Access Control. 136

MPLS Multi Protocol Label Switching. 3, 6, 9, 89, 180, 187

NREN National Research and Educational Network. 89

OPTIC Optimal Protection Technique for Inter-intra-domain Convergence. 3, 13

OSPF Open Shortest Path First. 180

PHP Penultimate Hop Popping. 187

PQ Priority Queue. 14

PRR Packet Reception Ratio. 173

RPL Routing Protocol for Low Power and Lossy Networks (LLN). 166

RSSI Received Signal Strength Indicator. 173

RSVP-TE Resource ReSerVation Protocol for Traffic Engineering. 6, 180

SLA Service Level Agreements. 3, 7, 9

SPC Shortest Path Computation. 8, 14, 20, 21, 147

SPDAG Shortest Path Directed Acyclic Graph. 35

SPF Shortest Path First. 16

SR Segment Routing. 3, 6–8, 12, 16, 21, 68, 161, 176, 177, 180, 181, 194

SRLG Shared Risk Link Groups. 52

TBFH Two Best First Hops algorithm. 3, 13

TE Traffic Engineering. 3, 6, 7, 9, 51, 89, 147, 176

TI-LFA Topology Independent Loop Free Alternate. 21

TNT Trace the Naughty Tunnels. 3, 89

TTL Time to Live. 187

UHP Ultimate Hop Popping. 187

WSN Wireless Sensor Networks. 4

WUR Wake-Up Radio. 166