PH.D. THESIS

presented at
Louis Pasteur University, Strasbourg
Department of Computer Science
LSIIT Laboratory, UMR CNRS-ULP N:7005

For obtaining the degree:
Louis Pasteur University
Doctor of Philosophy (Ph.D) in
Computer Science

by

Emil Ivov

Optimizing Real-Time Communications over the Internet Protocol

Public defense on April 1st 2008 with the following jury :

Andrzej Duda, External evaluator
Professor at ENSIMAG, Grenoble
Eckhart Koerner, External evaluator
Professor at the University of Mannheim
Thomas Noel, Thesis advisor
Professor at Louis Pasteur University, Strasbourg
Jean-Jacques Pansiot, Internal evaluator
Professor at Louis Pasteur University, Strasbourg
David Simplot-Ryl, Examinator
Professor at the Lille University of Science and Technology

Acknowledgements

Getting to the end of a Ph.D. is a tough matter and one could hardly get through it alone. Throughout my last years in the Louis Pasteur University I have been fortunate to meet and receive help from many people. One of the pleasures of finally finishing this task is the opportunity to thank them.

First and foremost, I would like to express my gratitude to Thomas Noel. I could hardly overstate his role in my life during the last years: not only would I have never even thought of starting a Ph.D. if it hadn't been him but I would have certainly not finished it. Thank you, Thomas, for guiding me when I most needed it, for helping me stay on track, and for being a great friend during the last years.

My work on the field of peer-to-peer real-time communication has evolved in a context of tight collaboration with Enrico Marocco from Telecom Italia Labs, Turin. I would like to express my most sincere gratitude to him for sharing his vast experience in the fields of SIP and P2P overlays.

I have started my Ph.D. together with Julien Montavont and we have worked together on subjects related to seamless IPv6 mobility. We have shared many common problems and many of the ideas presented in this thesis have came up during discussions with him. In addition to helping me out with his experience in geolocation assisted mobility, he was also the first of us to finish his thesis and he has provided me with generous advice while I was preparing mine.

Through my entire life I have always been able to count on the unconditional support of my family. Mom, Dad, Tsveti, thank you very much for always being there for me and for tolerating all the whims and the weird effects that a Ph.D. thesis may have on a student :).

Experimentation has been a very important part of my Ph.D. and work on the SIP Communicator and fmipv6.org projects has taken a substantial amount of the effort that I've spent on it. I would therefore like to thank Yana Stamcheva and Martin Andre for working with me on these projects. I really enjoyed working with you! Thank you!

One of the most difficult things for a Ph.D. student is having to learn and adopt a proper research attitude. Knowing what to take for granted, and what to prove, when to lean on previous work, and when to question it are choices that are far from being easy. I esteem myself particularly lucky in this respect because I have had the possibility of observing and learning this research attitude from a true scientist. I would hereby like to express my profound admiration for Jean-Jacques Pansiot, who represents in my eyes the very image of a researcher and thank him for all the intersting discussions: no matter how minor they may have seemed, to me they have always been a source of inspiration and a display of a model to follow.

Many thanks to the members of my jury: Andrzej Duda, Eckhart Koerner, and David Simplot-Ryl for spending the effort of reading the whole lot of pages contained in the manuscript of my thesis. Thank you, I do appreciate the effort!

I have shared the time spent in the ULP Network Research Team with many students, engineers and interns, all of which I am more than glad to have met. I would like to take this chance and also say thanks to all of them for all the fun we've shared. Alexander Pelov, Antoine Gallais, Arnaud Frey, Christophe Jelger, Cristina Tabacaru, Guillaume Schreiner, Jean Lorchat, Jean-Marc Muller, Koshiro Mitsuya, Mickael Hoerdt, Nicolas Dichtel, Nicolas Montavont, Pascal Merindol, Romain Kuntz, Symphorien Wanko, Tom Remoleur, and Vincent Lucas, guys, it has been a pleasure working with you!

A considerable part of my efforts during the past years has been dedicated to development. One of the people that I have most learned from and that I consider in many aspects to be my mentor is Mudumbai Ranganathan from the Advanced Network Technologies Division of the National Institute of Standards and Technology in USA. Ranga, thanks for always being there and always taking the time to discuss all the matters that I needed help on!

Last but certainly not least, I would like to profoundly thank my beloved Véronique Dupont, for sharing my life since the very start of my Ph.D. adventures ;), for giving me all necessary support and reassuring me every time I needed it while writing this thesis (and that means many times indeed). Thank you, Véro!

To my family

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The work that we present in this thesis focuses on the optimization of two different aspects of real-time communication over the Internet Protocol.

First, in part I we will describe different ways of improving network mobility over IEEE 802.11 wiress LANs with IPv6, including: usage of link layer triggers in application layer mobility; usage of a secondary wireless interface; and an optimization for Fast Handovers with Mobile IPv6 that allows assisting discovery of neighboring access points with geographic positioning information.

In the second part of this thesis we will be presenting a solution that allows using Peer-to-Peer technologies in real-time communication. The solution includes the definition of two new protocols used in the maintenance of such overlay networks and the transport of DHT operation requests.

1.1 Background

During the last several years use of the Internet has become quite popular and white-spread. Numerous existing applications and services that we have been using for tens of years are now being migrated over the Web. Telephony and television are one such example with their web counterparts: VoIP and (live) video streaming. This evolution is boosted even further by the constant development of wireless technologies such as IEEE 802.11 networks, more commonly known as Wi-Fi (Wireless Fidelity). Wireless devices are now offering bandwidth and reliability that are sufficient for the transporta-

tion of real-time media. Their massive adoption by both homes and enterprises have clearly shown the user eagerness for mobility.

Along with the growth of the Internet we have also witnessed the increase of problems related to the limited number of publicly routable addresses. For several years now, different sources have been predicting that the unallocated pool of addresses usable with the currently deployed version of the Internet Protocol (IPv4) will run out by 2011 at the latest [161]. In order to address this problem the Internet Engineering Task Force has defined Internet Protocol version 6 which, among other things, allows for a virtually unlimited number of addresses. Deployment of IPv6 has taken a while to get into gear (given that the first Internet Standard defining it has been available for more than ten years) but it now seems to be finally taking up [11, 61, 50]. In addition to the greater space of internet addresses IPv6 is also offering the possibility of improving existing services such as IP mobility.

Mobility for real-time applications in Wi-Fi networks can occur and be handled on different layers of the OSI ISO model. Simply moving from one wireless Access Point (AP) to another, located in the same subnet, is a link layer event that is transparent for the IP protocol stack and the applications themselves. If however, the second point of attachment is located behind a different access router, a mobile node would have to update its network layer configuration in order to avoid ingress filtering. This qualifies the process as a network layer handover and in the case where the mobile node is not using any mobility management mechanisms such as Mobile IP or Mobile IPv6, networking applications would have to re-establish their ongoing sessions from within the new network.

Depending on the layer where mobility occurs, a node could experience different problems. After disconnecting from a particular access point for example, and before being able to connect to the new one, a mobile node has to discover all APs available in the surroundings. The process could vary from several hundred milliseconds to more than ten seconds - a delay which is unacceptable for most real-time applications. Selecting the Access Point to connect to might also be a problem as the link layer protocols do not provide a way for a network interface driver to determine the availability of network connectivity for a specific AP before actually connecting to it. Reconfiguring the IPv6 stack so that it would be operational on the new subnet in the case of a network layer handover could also prove a lengthly process. The address autoconfiguration and location update procedures, described in detail in the rest of the document, could often last for more than a second.

There have already been a number of studies and a large set of optimizations that try to address these issues. In this thesis we will be presenting those that we consider most interesting and efficient, we will be pointing their weaknesses and we would also de-

scribe our own work on getting closer to a solution that could provide seamless mobility with an acceptable level of complexity and deployment cost.

Another consequence of the popularity of the Internet comes from the greater exposure of network services and the fact that any particular provider could easily address vast numbers of potential users regardless of otherwise blocking criteria such as geographic location. As a result, classic centralized architectures like those adopted by conventional telephony are proving inefficient when used in offering large scale real-time services over the Internet. Back in the late 1990's Shawn Fanning from the Northeastern University in Boston created and launched the Napster application [10] which allowed users to share and download content in a distributed manner. This model proved quite successful and was later adopted by many other applications. It is now mostly referred to as Peer-to-Peer (P2P) networking. In August 2003 the Skype Limited company has launched the first telephony service and application [13] using a P2P protocol. In addition to simply distributing registration servers, usage of P2P in real-time communication has the advantage to also offer an easy way of decentralizing algorithms for traversal of Network Address Translation (NAT) gateways.

In the second part of this thesis we will describe our work on a protocol whose purpose would be to build and maintain an open and relatively generic overlay network.

1.2 Managing micro mobility

Our work on the subject of optimizing Wi-Fi mobility has started at the application layer. Application layer mobility consists in having all applications manage handovers by themselves. A software phone using the SIP protocol for example would send a reINVITE request to its correspondent as soon as it detects movement and would thus reestablish all ongoing sessions. There are numerous advantages of handling mobility by applications themselves. Deployment for example is relatively simple since it does not require configuration or installation of network layer entities (e.g. routers or home gateways). Other advantages include reduced complexity, and the possibility to optimize performance for one particular application. One of the difficulties, however, comes from the fact that applications would only become aware of a handover once all of the underlying layers have completed it and this is often too late.

In [67] we have therefore investigated the possibility of assisting application layer mobility with information from the lower layers. We have implemented a solution that uses a standard UNIX mechanism (RTNETLINK sockets) to subscribe for and receive events from the network card driver and the IP stack. Using this implementation we

have completed a number of evaluation experiments which show that the use of such mechanisms greatly optimizes handover performance and brings resulting latency and packet loss down to reasonable levels.

It is clear however, that hoping for explicit mobility support in all applications that might need it is an overly optimistic assumption and that in order to expect to one day have reliable seamless handovers we would need to work on transparent mobility supporting lower layers. Our work has therefore continued with an evaluation of one of the most popular solutions for IPv6 mobility - Mobile IPv6 and its Fast Handovers extension the FMIPv6 protocol. Back at the time the FMIPv6 protocol was relatively new and there were no implementations that we could use for our work. We have therefore created the open source fmipv6.org implementation suite [66] and used it to complete a comprehensive evaluation of the protocol [93].

Our analysis have shown that the FMIPv6 protocol helps to completely eliminate handover latency and packet loss in many cases. However, we have also seen that all advantages brought by the protocol are greatly outweighed by the connection disruption necessary for a node to scan its environment during discovery of neighbor access points. The reason why this happens is that IEEE 802.11 devices are unable of using more than one radio frequency at a time. WLAN scanning therefore requires nodes to disconnect from their current channel and only then consecutively probe the rest of the Wi-Fi channels. This has led us to the next stage of our work. In [92] we define and evaluate a way for WLAN nodes to execute soft (also referred to as "make before break") handovers through the use of a secondary wireless interface. With 0 packets lost and handover latency completely eliminated, the simulational evaluation of the solution is quite encouraging. Our next step has been to port this solution and adapt it to a standardized mechanism such as the FMIPv6 protocol. As we have already pointed out, handovers in FMIPv6 are sufficiently optimized and it is only the discovery phase that causes problems. We have therefore simplified our soft handover approach and kept the use of the secondary wireless interface to the scanning phase only. [65] presents a thorough description and an experimental evaluation of this approach. As expected, the results are quite satisfactory with handovers being completely seamless in cases when triggered by the mobile and hardly perceptible for unexpected loss of connectivity.

We are of course conscious that existing mobile devices rarely come with more than one wireless interface which has made us work on alternative schemes for the optimization of candidate access point discovery. In [98] we present the design, implementation and evaluation of a comprehensive solution that allows assisting Wi-Fi handovers with positioning information obtained by the mobile device. The decision to direct our work this way has been influenced by the growing number of network enabled devices that

come equipped with a GPS. As expected the evaluation presented in [98] shows results which are quite similar to those obtained with the multi interface solution.

It is probably worth mentioning that work on this topic has been completed in the context of a contract with Orange Labs France, which would probably help further explain our desire to keep the proposed solutions relatively applicable, implementation friendly, and compatible with existing standardization efforts.

1.3 Using Peer-to-Peer in real-time communication

The second part of this thesis concentrates on the use of Peer-to-Peer networks in real-time communication. Work on it has been completed in a joint effort with Telecom Italia Labs, Turin, represented by Enrico Marocco, and partially with Columbia University, NYC USA, represented by Dhruv Chopra and Henning Schulzrinne, and it is currently continued in the context of the P2PSIP working group of the Internet Engineering Task Force.

There are numerous reasons why one might want to use P2P in real time communication. We go through them in detail later in this thesis but to put it short it's all about the fact that applications such as telephony, streaming, video conferencing and other kinds of real-time communication are generally relatively expensive resource-wise. This is particularly important for algorithms that deal with the traversal of Network Address Translation (NAT) gateways. Most such mechanisms get down to relaying media exchanged between NAT-ed clients through a point on the public Internet. Using a single server for all relaying necessary for a particular provider is quite an expensive solution and, depending on the scale, it may even be impossible in certain cases. Other than NAT and firewall traversal P2P networks are also used for the distribution of the registration services and routing policies.

A common concern related to P2P overlay networks is that of security. We therefore begin the presentation of our work on peer-to-peer with a state of the art description of different security pitfalls associated to this networks, as well as the existing solutions. Results presented in this part of the thesis have been published in [53].

Keeping security constraints in mind we have started work on two protocols that allow to build and maintain P2P overlays the primary use of which would be real-time communication with the SIP protocol.

XPP [126] (the eXtensible Peer Protocol) is the first of these protocols. Its purpose is to serve as a transport mechanism for all overlay and DHT maintenance operations.

One of the primary constaints that we have kept in mind while designing it was the fact that it should allow maximum tolerance for firewalls and NATs for a minimum cost. XPP sessions are therefore initiated "simultaneously" by both sides using the SIP protocol and in a way that very much resembles initiation of normal SIP calls. Another important characteristic of the protocol comes from the fact that it could be used with virtually any Distributed Hash Tables algorithm.

In parallel with XPP we have also completed a second protocol that we will refer to as XPP-PCAN (XPP Extensions for Implementing a Passive P2PSIP Overlay Network based on the CAN Distributed Hash Table) [127]. It consists in a set of XPP extensions required for creating an overlay network based on the CAN distributed hash table algorithm. It specifies how peers and clients must behave in order to maintain the overlay and use it for the establishment of multimedia communication sessions.

To limit the overhead due to maintenance operations and to allow the adoption of security policies for preventing malicious nodes to damage the overlay, making the decision when a user/client of the overlay should contribute to it by joining as a peer belongs to existing peers (hence the passive in PCAN).

Both protocols have been implemented in an open source project [125] and their integration in the SIP Communicator [15] client is pending.

1.4 The rest of this paper ...

... is organized as follows: Part I contains the description of our work on optimizing seamless mobility. This part starts with a problem statement describing the different parts of mobility over IEEE 802.11 networks and a state of the art classification of existing solutions and optimizations in Chapter 2. Next, in Chapter 3 we go over our efforts on improving application layer mobility through the use of link layer information. In Chapter 4 we present our evaluation of the FMIPv6 protocol.

The following Chapter 5 contains our work on the network layer. We describe in detail our solution for achieving soft handovers through the use of a secondary Wi-Fi interface in Section 5.2. The application of this solution to the FMIPv6 protocol as well as its evaluation are presented in Section 5.3, and finally, our geolocation extensions of the FMIPv6 protocol are described in Section 5.3

The second part of this thesis (i.e. Part II) contains our work in the field of Peer-to-Peer real-time communication. Chapter 7 begins with an introduction containing the reasons why we believe P2P networks to be important for services like Internet tele-

phony as well as a state of the art presentation of security concerns and solutions often associated with P2P. The following Chapter 9 presents the mechanics of the eXtensible Peer Protocol, and Chapter 10 the description of the XPP PCAN extension.

Part III concludes this thesis and presents possible perspectives and future work.

We have decided to include in this thesis a brief annex (Part IV) which describes the history, architecture, and features of the SIP Communicator application. Though not directly related to the content of our research work, SIP Communicator has been used in a number of evaluations. It represents a substantial amount of the effort that we've spent during the last several years and would probably be a key factor our future work.

Part I

Transparent mobility for real-time communication

Chapter 2

Introduction and state of the art in the field of IP mobility

Nowadays Internet is arguably, one of the most widely deployed networks for real-time communication. The main protocol used for the transportation of data over this network is called the Internet Protocol (IP). According to the OSI and TCP/IP network models, this is a network layer (layer 3) protocol. Currently it is the version four (IPv4 [151]) of this protocol that is most widely deployed over the Internet. However, usage of this version of the Internet Protocol is now becoming problematic mainly because of the limited number of available IP addresses used for identifying and locating network enabled devices. As a result of the rapid expansion of the Internet network, the number of connected devices, that is those that are assigned an IP address, greatly increased. Today numerous sources predict that in the following several years there would be no more unallocated IPv4 addresses [161].

The reason for this is that IPv4 addresses are encoded as integer numbers over 32 bits. Theoretically, this allows for the simultaneous identification of a maximum of 2^{32} devices, or in other words 4 billion approximately. In reality, this number is reduced even further by inefficiencies like subnetting and the fact that many of the segments in the address space have been attributed to countries or organizations who do not necessarily use all available addresses.

In order to address this problem, the Internet Engineering Task Force (IETF) [89] has designed a new version of the IP protocol, often referred to as IPv6 [60]. The main difference brought about by this new version comes from the fact that it defines an encoding of IP addresses based over 128 bits. In other words this would make it possible to simultaneously address (superficially) an unthinkable number of machines:

about 340 trillion, trillion, trillion ($3.4x10^38$). In reality, as with IPv4, addresses are structured, and as a result the number effectively available is somewhat less according to the administrative policy adopted. For example, on one model, each site running IPv6 would be given a 48-bit prefix, leaving 80 bits for local use. There could be 35,184,372,088,832 such prefixes - 35 trillion IPv6 sites, which seems to be enough for 9 billion people.

Other than that, the IPv6 protocol also makes possible the adding of new options in the IP header which greatly facilitates the development of new protocols extensions. Another new and interesting feature is the definition of a stateless address autoconfiguration [192], which allows network connected nodes to automatically obtain an IPv6 address withour using external centralized servers or other mechanisms such as DHCP [64].

These, as well as other new features and possibilities, the version 6 of the IP protocol is expected to soon be adopted as the default protocol for Next Generation Networks.

In this chapter, we will present excising mechanisms and procedures that allow mobile terminals to move through multiple IPv6 Wi-Fi networks. We will particularly focus on the standard procedures, their limitations, as well as some of the optimizations that have been suggested in the literature and that we esteem most efficient.

2.1 Mobility Background

In most cases mobility in IP networks happens at two levels. A change of the physical point of attachment to the network is often referred to as *layer 2 handover* because it only uses the first two layers of TCP/IP network model (i.e. the physical and data link layers). In Wi-Fi networks for example, layer 2 handovers can occur when a mobile terminal disconnects from its current access point (AP) and associates with a new one (see the figure 2.1).

Following a layer 2 handover, a mobile node is able to communicate using its link layer connection (e.g., ethernet or Wi-Fi) with other devices located in the same link segment. Moreover, if after the link layer handover the IP subnet of a mobile has not changed (e.g. both Wi-Fi access points are on the same IP subnet) it would also be able to retain its layer 3 connectivity.

The possible second phase of a mobility event is called a *layer 3 handover* and corresponds to the change of the IP subnet that occurs as a result of the handover. When, for example, the new point of attachment of a node is not located in the same subnet as

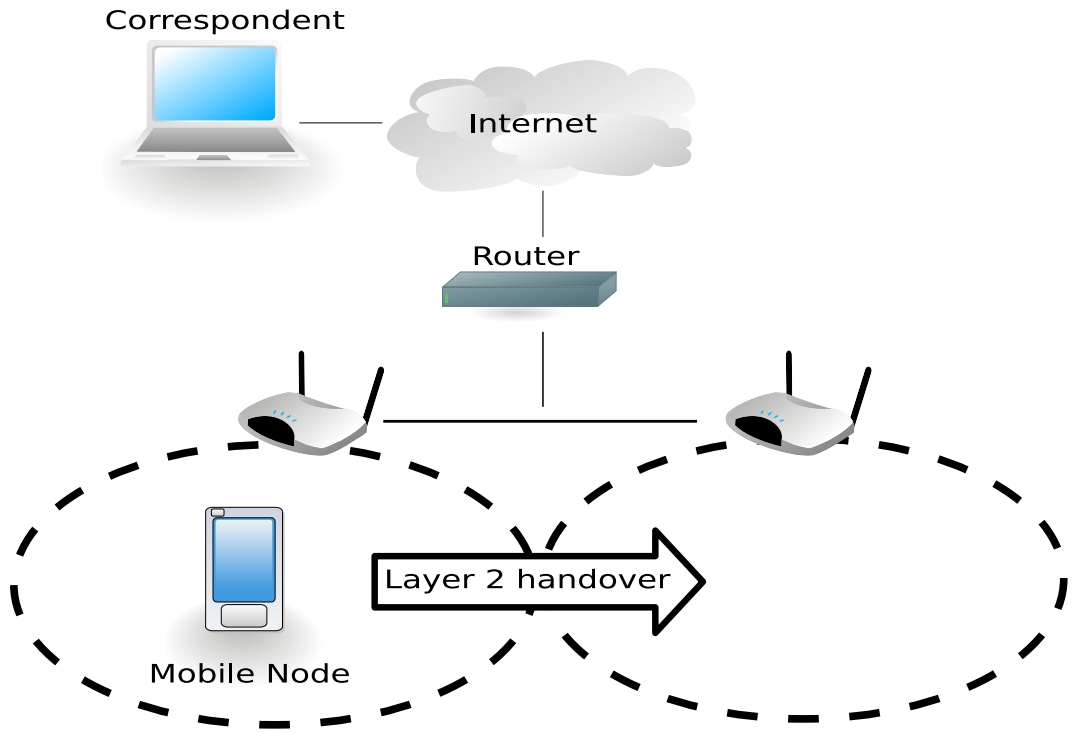


Figure 2.1: Layer 2 handover

the one it was using before, the node would also have to go through a layer 3 handover before being able to restore its network connectivity (see figure 2.2).

2.2 Layer 2 mobility in IEEE 802.11 WLAN networks

When a mobile node enters or simply wakes up in the coverage area of a new access point, it must connect or associate with it before actually being able to communicate. This procedure is defined by the 802.11 standard and it consists of the following three phases: scanning, authentication, and association.

2.2.1 Scanning

The first stage of the 802.11 association procedure consists in the discovery of an access point whose characteristics best correspond to those of the mobile terminal. The

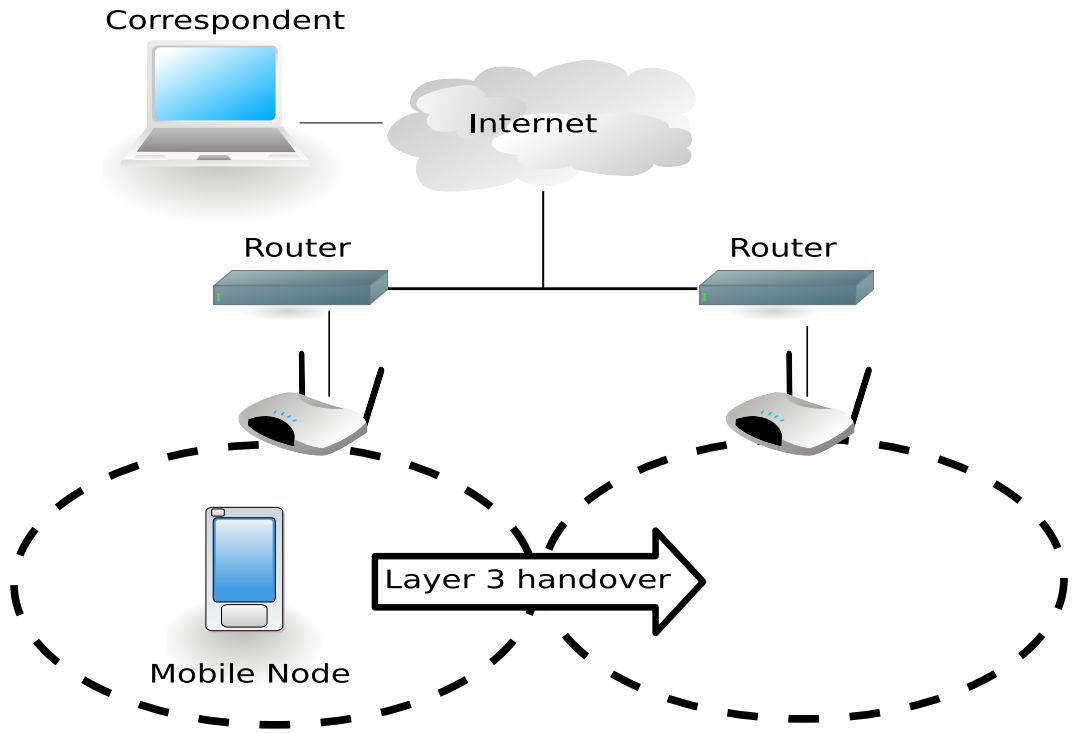


Figure 2.2: Layer 3 handover

scanning phase defines exactly how the discovery of surrounding access points is to be accomplished. The 802.11 standard defines two methods for access point discovery: active and passive.

Active scanning

When performing active scanning, most mobile nodes probe all radio channels used by Wi-Fi devices by broadcasting *Probe Request* messages. When an access point receives such a message, it answers the sending terminal by a *Probe Response* containing the various parameters which it supports.

Depending on the number of access points in the vicinity, sending a single *Probe Request* can generate the transmission of multiple *Probe Response* messages. Every time the scanning node receives a *Probe Response*, it records the various parameters of the access point that is described by the message, and it continues its search through the rest of the available frequencies. After having probed all (or a prefined list of)

radio channels, the node selects an access point among those which it discovered and it initiates the authentication phase.

It is worth mentioning that the *Probe Request*s are the only messages of the association procedure which do not require any acknowledgement. This means that the absence of a *Probe Response* following the sending of a *Probe Request* on a particular channel can be interpreted in two different ways. One possibility would be that a collision has occurred during the transmission of the *Probe Request*, another option would mean that none of the access points located in the vicinity operate on this channel. Knowing that the *Probe Response* messages require an delivery acknowledgement, if an access point sends a *Probe Response* to the terminal, it is virtually certain to receive it because of the retransmissions mechanism. To prevent mobile terminals from being delayed on channels which are not used, the 802.11 standard defines two timers that are used when scanning a channel: *MinChannelTime* and *MaxChannelTime*. Following the emission of the first *Probe Request* on a channel, the terminal starts the *MinChannelTime* timer. If when the timer fires the terminal has not received any *Probe Response* messages, it can conclude that there are no access points on this channel and moves on to the next one. In the case where one or more *Probe Response* messages have been received on a channel, the mobile node would wait for the *MaxChannelTime* timer to fire before sorting all received *Probe Response* messages and moving to the next channel. The standard only defines the relation $MinChannelTime \leq MaxChannelTime$ without specifying any exact numerical values. As we will observe in the rest of this thesis, these parameters can strongly vary according to the maker and the implementation of the wireless device drivers.

Passive Scanning

The 802.11 standard also defines a method of passive access point discovery, in which a terminal would only intercept the signalling frames coming from surrounding access points. These signalling frames are called *Beacon*s and are generally sent every 100 milliseconds. The time spent on each channel is calculated the same way as for the active scanning.

2.2.2 Authentication

Once a mobile node has completed the discovery phase and has as a result selected its new access point, it must try and authenticate with it. The 802.11 standard defines two possible methods of authentication. The open method requires only one sequence of

two messages (request/answer). This method is used when terminals are systematically authorized to connect to the network. The second option, referred by the standard as a shared key method, requires the exchange of a sequence of four messages. The node initiates the sequence by sending its identity to the access point. The AP then requires the terminal to respond to a challenge in order to prove knowledge of a shared secret (password or key). Depending on the validity of the message that the node has sent as a response to the challenge, the access point would either authenticate or reject the mobile.

2.2.3 Association

The association phase represents the last stage of a layer 2 handover and it consists of the messages *Association Request* and *Association Response*. Following a successful authentication, the association phase allows a mobile node and an access point to negotiate the various parameters that they will use during communication. A mobile node is able to both transmit and receive data once this association phase has been completed.

Figure 2.3 illustrates all the stages of a layer 2 handover. For reasons of readability, the figure does not represent any of the acknowledgement messages.

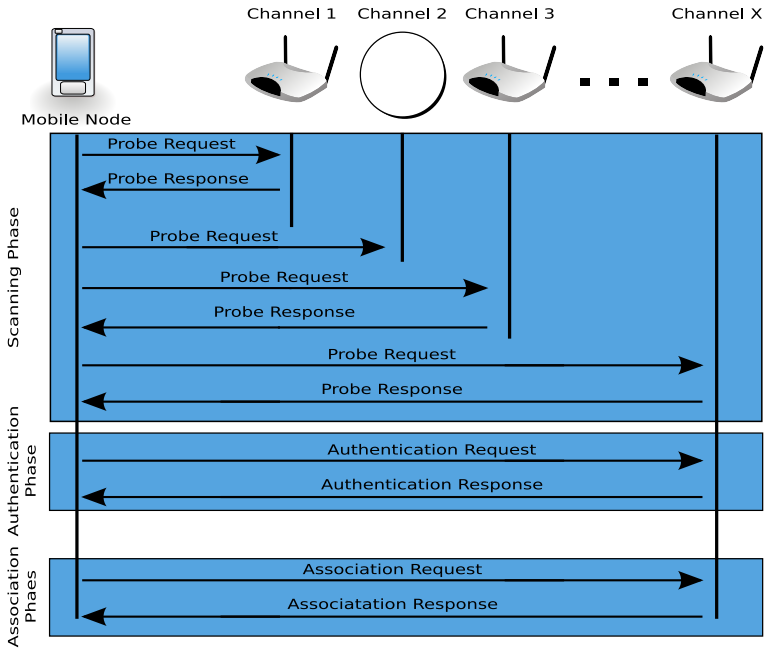


Figure 2.3: Layer 2 handover using active scanning in a 802.11 wireless network

2.3 Layer 3 mobility in IPv6 networks

In the previous sections we have described the procedures that the 802.11 standard defines so that mobile nodes could move from one access point to another. These procedures allow nodes to maintain layer 2 connectivity while roaming through different wireless networks as long as they remain in the same IP subnet. Quite naturally, the 802.11 standard does not go beyond the data link layer of the TCP/IP model and it does not define the way a node would restore connectivity on layer 3 - the network layer of the TCP/IP model. While movement between two access points located on the same IP link does not imply any routing problems, changing subnets requires an update of the IP address of the terminal. In the absence of any specific IP layer support, none of the correspondent of the moving node would be able to detect the change in its address and would therefore continue sending data to the old address of the terminal which has become obsolete after the handover. As a result, the mobile terminal would have to re-initialize all of its ongoing sessions after the completion .

In order to handle these problems, the IETF organization has recently defined and standardized two protocols addressing mobility management in IPv6 networks. On the one hand, the *Mobile IP* protocol allows movement of mobile terminals and lets them control their mobility. On the other hand, the *NEMO (Network Mobility) BASIC Support* protocol [62] concentrates all mobility management inside special routers, which allows the movement of whole networks without exposing user nodes to the complexity of managing this movement. Even though the NEMO BASIC Support protocol was especially defined for IPv6 networks, Mobile IP itself exists in two versions. One, MIPv4 [69], is specifically designed for IPv4 networks, whereas the other, MIPv6 [103], as an extension to the IPv6 protocol. In this document we are mainly interested on Next Generation Networks and we therefore focus on the IPv6 version - MIPv6.

2.3.1 The Mobile IPv6 protocol

The main idea behind the MIPv6 protocol is to use a centralized node that relays all communication bound to mobile nodes towards their current location. From the MIPv6 perspective, all mobile nodes therefore regard the Internet as two different kinds of subnets: the visited and home networks. While in its home network, a mobile node communicates using its primary, home address (HoA), just as any other internet connected node would. This case is illustrated on figure 2.4.

The relay node responsible for that terminal, also referred to as home agent (HA), is located in the same home network. When the terminal leaves its home network and

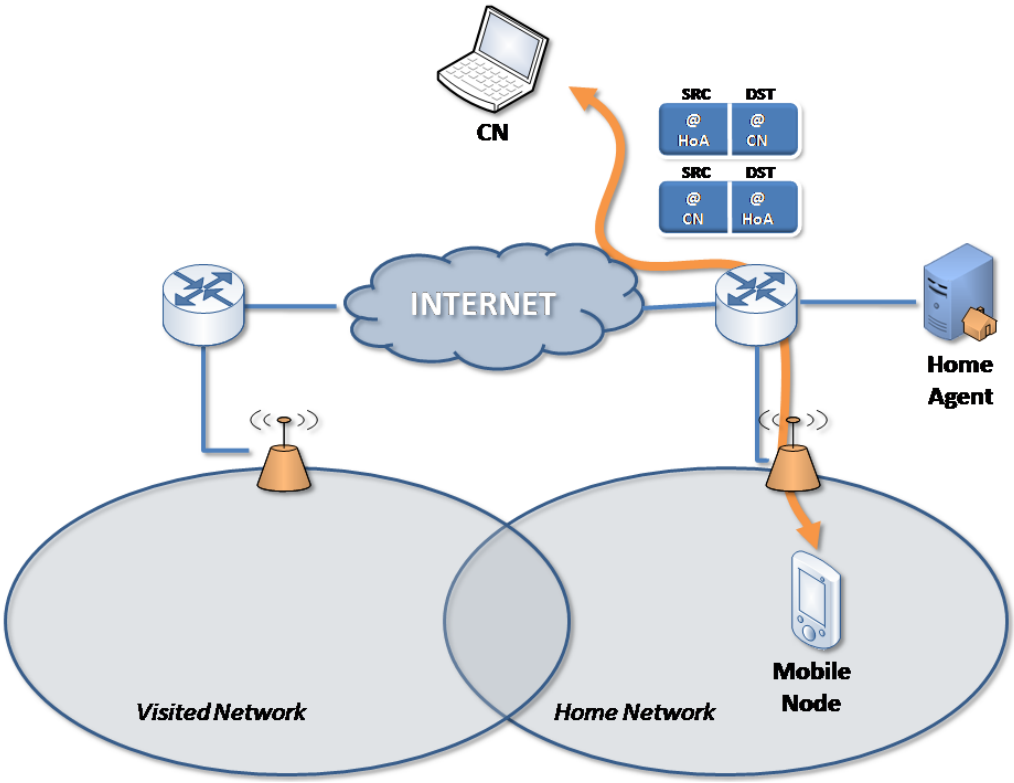


Figure 2.4: A mobile node in its home network

moves to a visited one, it obtains a second IPv6 address. This new address is only temporary and is called a Care-of Address (CoA). It identifies the terminal inside the new network that it has moved to. In order to maintain its ongoing sessions, the terminal must inform its home agent any time it moves to such new networks and keep its CoA record with it up to date. This way, once a node has left its home network, the home agent intercepts all packets that are bound to the mobile node (with MIPv6, when not using route-optimization procedures, packets are always sent to the home address of the terminal) and redirects them towards its currentlocation using the CoA. Any time when sending packets from a visited network, a mobile node would also use its home agent so that its movement would remain transparent for its correspondents. In other words, the home agent and the mobile node establish a bidirectional communication tunnel (see figure 2.5).

To carry out these operations, the home agent maintains up to date a list of mappings between the home and the temporary addresses of all nodes that it is responsible for. It updates this list every time it receives a BU (*Binding Update*) or a BACK (*Binding Ac-*

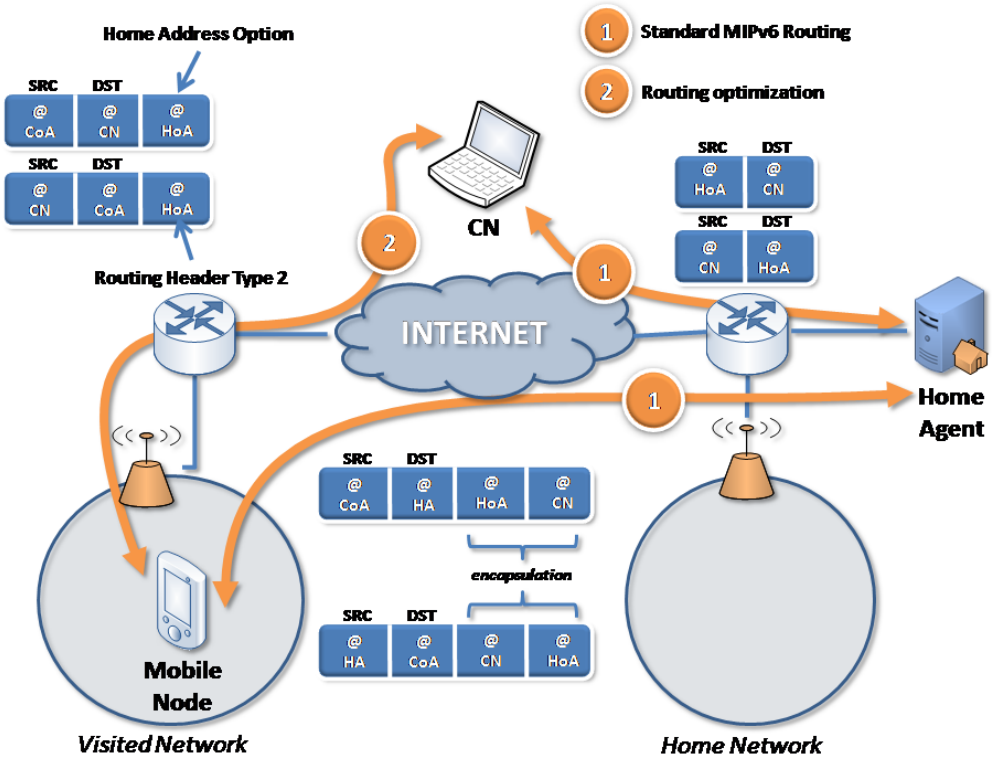


Figure 2.5: A mobile node in a visited network

knowledgement) message from a mobile node. These messages can also be exchanged securely by using the IPsec protocol [21]. The redirection of the IPv6 packets by the home agent, and the transmission of packets using the home address of a mobile as a source even when it is topologically invalid uses the extension header mechanisms permitted by the IPv6 protocol. In other words sending MIPv6 devices add to all IPv6 packets an extra header of additional routing information (see the figure 2.5).

The basic specifications of the MIPv6 protocol also define two routing optimizations (see figure 2.5). The first alternative to the bidirectional tunnel established between a mobile node and its home agent is called triangular routing. In this mode, the mobile node can directly send its IPv6 packets to the correspondents (CN) without passing through the home agent. With this intention, the terminal uses its temporary address but adds an option containing its home address. The addition of the home address makes it possible for the correspondents to identify the sending terminal. Packets coming from the correspondents on the other hand, are still being relayed through the home agent (hence the name - triangular routing).

A second optimization, which is often simply referred to as *Routing Optimization* tries to resolve the constraint of triangular routing. It makes it possible for correspondents to use the temporary address of the mobile node in order to communicate with it. Same as the home agent, the correspondents that support this mode maintains a list of mappings in order to keep track of the relations between home and temporary addresses. As a result, the mobile node must also notify all its correspondents when moving using the BU and BACK messages. Each update of a correspondent is preceded by a *Return Routability* procedure. This procedure aims at preventing malicious users from sending fictitious updates to the correspondents assuming the identity of a mobile terminal. This procedure allows correspondents to make sure that the same terminal is reachable through its home and temporary addresses. Once the cache entry of a mobile has been updated, correspondents start adding a particular routing header to all packets intended for this mobile node. This routing header contains the home address of the mobile node and it indicates the final destination of the packet. Although this routing technique helps avoiding packet relayring, it requires a certain amount of extra management by correspondents themselves and can therefore not be used with correspondents that do not support the MIPv6 standard. For this reason, through the rest of this paper we will only focus on standard MIPv6 routing using the bidirectional tunnel between a home agent and its mobile nodes.

The mechanisms introduced by the MIPv6 protocol, including all the messages exchanged between the arrival of a node in a visited network and the re-establishment of the bidirectional tunnel are often referred to as a *layer 3 handover*. Once it has completed a layer 2 handover, a mobile node has to determine if its new point of attachment is in a new subnet. This detection is generally based on the reception of a*Router Advertisement* (RA) messages. These messages are periodically sent by the access routers of the link and are used at the time of the stateless address autoconfiguration [192] by the neighbor discovery protocol [142]. Upon the reception of a router advertisement message, the terminal could detect that it has moved to a new subnet and can thus configure its new temporary address. After having checked that this address is not used by other nodes on the same link, a procedure also known as Duplicate Address Detection (DAD), the terminal can send a BU to update the binding cache of its home agent. When the update is completed, the home agent sends a BACK to the terminal in order to notify it of the end of the layer 3 handover. Figure 2.6 illustrates the complete layer 3 handover various procedure.

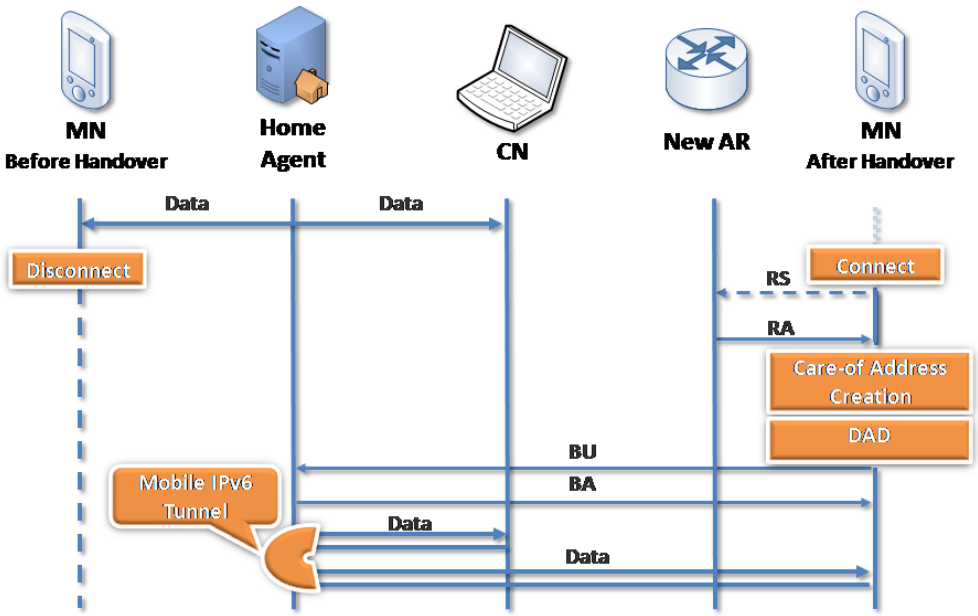


Figure 2.6: Management of layer 3 handovers with the MIPv6 protocol

2.3.2 Network mobility (NEMO)

The mechanisms introduced by NEMO BASIC Support protocol mainly reuse those defined by MIPv6. The main difference comes from the migration of all mobility management within the routers, which are now called mobile routers. This allows entire networks to support mobility, while keeping movements completely transparent to all devices located behind the mobile router. All terminals located behind a mobile router would therefore only have to support the IPv6 protocol. However, we could still have mobile nodes supporting MIPv6 behind a mobile router and the NEMO BASIC Support protocol therefore defines three possible types of devices. *Local Fixed Node*s (LFN) are nodes or routers that belong to the mobile network and that do not support any form of mobility (i.e. they have no support for MIPv6 or NEMO). In other words they are not capable of changing their point of attachment to the network without interrupting their ongoing communication sessions.

*Mobile Visiting Node*s (VMN) on the other hand, are mobile devices (terminals or router) which are capable of moving without interrupting their active communication

sessions. Such devices would therefore support either the MIPv6 protocol, in the case of a terminal, or the NEMO BASIC Support protocol in the case of a router. For such nodes, the mobile network represents a mere visited network which they use in a temporary way.

Finally, *Local Mobile Node*s (LMN) are similar to VMNs with the only difference being the fact that their home network is actually the mobile network whereas this is not the case for VMNs.

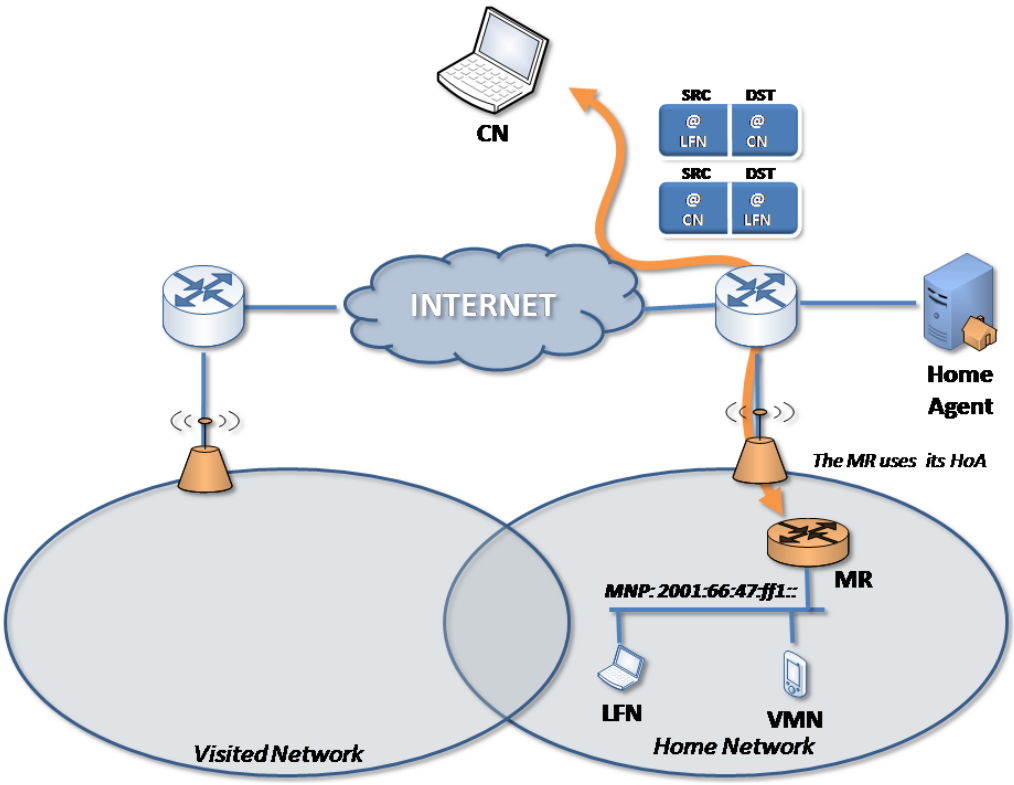


Figure 2.7: A NEMO mobile router in its home network

The NEMO BASIC Support protocol completely reuses the MIPv6 protocol architecture and it adds to it the concept of *Mobile Network Prefix*es (MNP). Just as with the MIPv6 protocol, a mobile router will have two addresses and will establish a bidirectional tunnel between its current localization and its home agent. Because of its role as a router, it would announce an IPv6 prefix to all devices located behind it using standard Router Advertisement (RA) messages. This prefix is often referred to as an MNP prefix and remains the same regardless of the location of the router. The IPv6 packets sent towards an MNP prefix are automatically delivered to its home agent using the standard

IPv6 routing mechanisms. The home agent would then also act as a router and would either route such packets directly to the mobile router when it is in the home network, or use the bidirectional mobility tunnel that it has previously established with it.

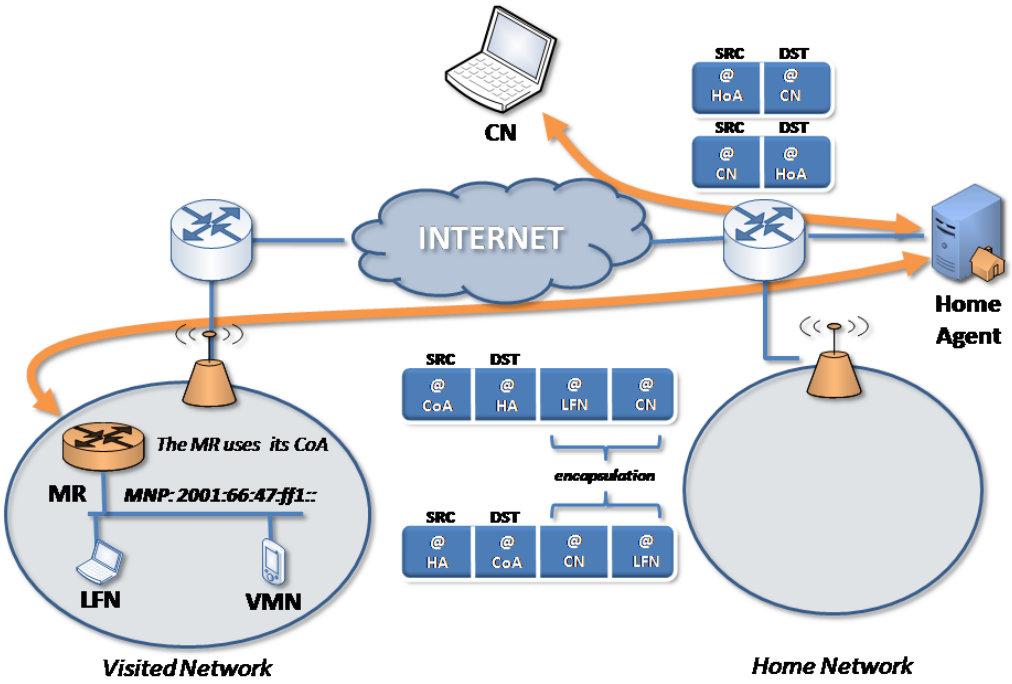


Figure 2.8: A NEMO mobile router in a foreign network

[135] provides a practical evaluation of this protocol based on an IPv6 Wi-Fi network deployed in the Louis Pasteur University in Strasbourg.

2.4 Application layer mobility

In addition to handling terminal mobility on layer 2 and 3 it is also possible to develop real-time applications in a way that makes them capable of handling it themselves. One way of doing this is by using the IETF *Session Initiation Protocol* (SIP) [167]. This approach brings numerous advantages such as the fact that SIP mobility allows to avoid the encapsulation and the triangular routing typical for Mobile IP. Another advantage of

handling mobility on the application layer comes from the fact that it is easily deployed and does not require deployment of any network layer entities. It is therefore possible to use this solution even in cases where one does not have control over the network infrastructure, which is the case of most home and small office users. The following sections present this protocol and the way it handles mobility.

2.4.1 Session Initiation Protocol Basics

The *Session Initiation Protocol* (SIP) [167] is a protocol for establishing and tearing down multimedia sessions. SIP can also support various types of mobility such as terminal mobility, session mobility, personal mobility, and service mobility [179]. Since our focus in this thesis is on terminal mobility, in the rest of this section we only describe SIP terminal mobility after briefly mentioning the basics of the *Session Initiation protocol*.

Figure 2.9, shows a typical example of a SIP message exchange between two users, Alice and Bob. In this example, Alice uses a SIP application on her PC (referred to as a softphone) to call Bob on his SIP phone over the Internet. Also shown are two SIP proxy servers that act on behalf of Alice and Bob to facilitate the session establishment. This typical arrangement is often referred to as the "SIP trapezoid" as shown by the geometric shape of the dotted lines in the figure.

Alice "calls" Bob using his SIP identity, a type of Uniform Resource Identifier (URI) called a SIP URI. It has a form similar to an email address, typically containing a username and a host name. In this case, it is sip:bob@biloxi.com, where biloxi.com is the domain of Bob's SIP service provider. Since Alice's softphone does not know the location of Bob or the SIP server in the biloxi.com domain, the softphone sends an initial SIP INVITE request to the SIP server that is responsible for Alice's domain, atlanta.com. The proxy server receives the INVITE request and sends a 100 (Trying) response back to Alice's softphone. The 100 (Trying) response indicates that the INVITE has been received and that the proxy is working on her behalf to route the INVITE to the destination. The atlanta.com proxy server locates the proxy server at biloxi.com and forwards, or proxies, the INVITE request there. The biloxi.com proxy server receives the INVITE and responds with a 100 (Trying) response back to the atlanta.com proxy server to indicate that it has received the INVITE and is processing the request. It then consults a database, generically called a location service that contains the current IP address of Bob and proxies the INVITE to Bob's SIP phone.

Bob's SIP phone receives the INVITE and alerts Bob for the incoming call from Alice so that Bob can decide whether to answer the call, or in other words, Bob's phone

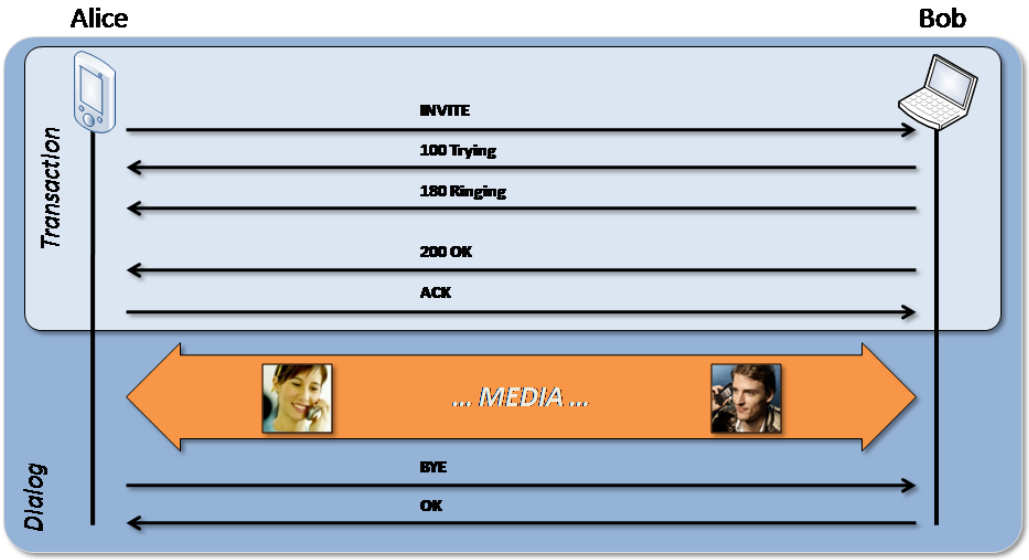


Figure 2.9: A basic Session Initiation Protocol scenario

rings. Bob's SIP phone indicates this in a 180 (Ringing) response, which is routed back through the two proxies in the reverse direction. When Alice's softphone receives the 180 (Ringing) response, it passes this information to Alice, perhaps using an audio ring back tone or by displaying a message on Alice's screen.

In this example, Bob decides to answer the call. When he picks up the handset, his SIP phone sends a 200 (OK) response to indicate that the call has been answered. Finally, Alice's softphone sends an acknowledgement message, ACK, to Bob's SIP phone to confirm the reception of the final response (200 (OK)). The ACK is sent directly from Alice's softphone to Bob's SIP phone, bypassing the two proxies. This occurs because the endpoints have learned each other's address from the Contact header fields through the INVITE/200 (OK) exchange, which was not known when the initial INVITE was sent.

Alice and Bob's media session has now begun. During the session, either Alice or Bob may change the characteristics of the media session (e.g. media formats or

endpoint location). This is accomplished by sending a re-INVITE containing a new media description. A re-INVITE scenario is discussed a bit later.

At the end of the call, Bob disconnects (hangs up) first and generates a BYE message. This BYE is routed directly to Alice's softphone, again bypassing the proxies. Alice confirms receipt of the BYE with a 200 (OK) response, which terminates the session.

Session Initiation Protocol Mobility

[179] shows how SIP can be used to support terminal mobility and its advantages over other mobility protocols. It is important to note that SIP-based terminal mobility does not add anything new to the standard SIP protocol in order to support mobility. For the completeness of the paper, we briefly illustrate mid-call mobility [179] on figure 2.10. Midcall mobility allows a node to continue an ongoing session with a peer after changing networks. Figure 2.10, shows an example of how mid-call mobility is supported by SIP. In this example, an MN sends a re-INVITE request with its new IP address to the CN (1), and the CN directly sends packets to the MN at the new point of attachment to the network (2, 3).

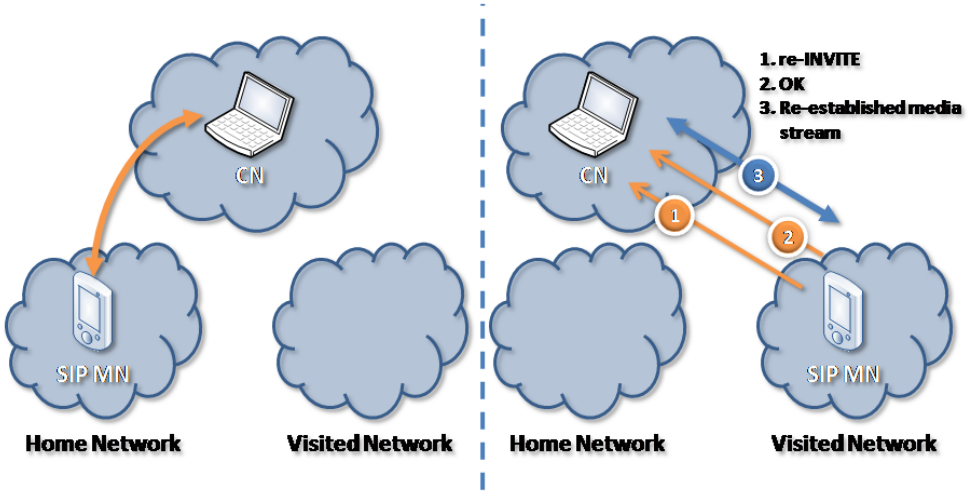


Figure 2.10: Midcall mobility with SIP

2.5 Limitations of existing standards

In the previous section we have described existing standard protocols which make it possible for mobile nodes to move through IPv6 WLAN networks without interrupting their ongoing communication sessions. However, during the time of the handover itself, it is not possible for a node to neither receive nor send any data. Depending on the applications that are being used, such connection disruptions can be more or less perceptible to the users. The increasing popularity of IEEE 802.11 networks and the ever growing bandwidth provided by this technology, more and more users count on them for the transportation of *real-time* traffic. The term real-time communication is used when referring to applications that are subject to a time constraint in the reception or the transmission of data. *Voice over IP* (VoIP) and *Video conferencing* are among the most common examples of such applications. In order for real-time communications traffic to have a quality satisfactory for the end user, the latency accumulated during a layer 2 and layer 3 handover should not exceed 150 milliseconds [106].

2.5.1 Layer 2 handover latency

Numerous studies carried out on 802.11 networks during the last few years show that layer 2 mobility, such as it is described in the standard, does not allow for handovers, rapid enough to satisfy the constraints of real-time applications [133, 195, 196]. In most cases, the latency accumulated during a handover would exceed the 150 milliseconds threshold that we have mentioned above. The candidate access point discovery phase of the layer 2 handover takes approximately 90% of the total handover time which makes it the principal cause for the accumulated latency [133]. According to the distribution of the surrounding access points over the radio frequencies, a node may have to probe several channels before discovering a new access point. As a result, all unused channels would be probed in vain during *MinChannelTime* each one, while the terminal would have to spend *MaxChannelTime* on those that are occupied. According to the numerical values of the *MinChannelTime* and *MaxChannelTime* parameters, the terminal can largely spend more than one second to probe all of the 14 available channels.

In practice, however, we could also note that in most cases, in addition to the numerical values of the parameters *MinChannelTime* and *MaxChannelTime*, the scanning algorithms can also vary from one device to the next. This comes from the fact that the IEEE 802.11 standard does not define the order in which devices should go through the wireless channels nor the number of channels to probe before completing a handover. As a result, some devices would systematically probe all radio operator channels, while others would enter the authentication phase as soon as they have discovered an access

point and after having spent *MaxChannelTime* on a particular channel. Even in this last case, layer 2 handovers could still remain perceptible to the user. The latency accumulated during a layer 2 handover would generally lie between 50 and 400 milliseconds when using recently manufactured wireless cards [133, 195, 196]. We have already mentioned that in order not to disturb user quality, this interval should not exceed 150 ms [106].

2.5.2 Layer 3 handover latency

As we have mentioned in previous sections, in some cases, mobile nodes would also have to carry out a layer 3 handover following the one on layer 2. The total connection disruption time can therefore be further increased by the mechanisms of the MIPv6 protocol [136, 137]. This is mainly caused by the following three factors: the detection of the fact that a mobile node has moved to a new IPv6 link (*movement detection*), the verification of the uniqueness of the newly configured care-of address, and the update of the binding cache entry that the home agent keeps for this address and node. Layer 3 movement detection is terminated by the first *Router Advertisement* message (RA) received on the new link. The time necessary for this phase would therefore depend on the frequency at which these messages are being broadcasted by the access routers. This frequency would generally lie between 200 and 600 seconds [142], which is insufficient for prompt mobility management. The MIPv6 protocol therefore overrides these values and encourages network administrators to increase this frequency down to between 30 and 70 milliseconds but still stipulates that these values should not be used by default. With the retransmission interval being between 30 and 70 milliseconds, a node would need an average of 50 milliseconds to detect a new IPv6 link and thus obtain its new temporary address.

In order to avoid IP address conflicts, the stateless address autoconfiguration protocol specifies that, after a terminal has acquired a new IPv6 address, and before actually using it, it must verify that it is unique on the current link and that it is not being used by another node. Therefore after configuring their new address mobile nodes should go through the *Duplicate Address Detection* procedure (DAD) defined in [192]. The procedure uses messages and semantics defined by the neighbor discovery protocol for IPv6 [142]. This procedure adds an additional delay to the total duration of the layer 3 handover since it delays the emission of the BU messages towards the home agent. After the configuration of a new IPv6 address, a mobile node must wait between 0 and 1 second before sending its first DAD verification request, which is actually a *Neighbor Solicitation* (NS) message [142, 192]. The default values in the definition of the DAD procedure specify that a node would generally broadcast a single NS message. If at the

end of one second it has not received a *Neighbor Advertisement* (NA) indicating that its new address is already allocated to someone else, it can conclude that there is no address conflict on the link and can therefore start using it to communicate. In other words, if no collision is detected DAD would generally add approximately 1500 milliseconds to the duration of a handover.

Finally, the time of routing of the packets between the home agent and the current location of the mobile node can also contribute to the amount of the total time of connection disruption. Following the move of a mobile node to a new subnet, a home agent would continue sending data packets towards the old location of the terminal until it receives a new location update. Since, after a handover, a terminal is no longer connected to its previous subnet, these packets are lost. After configuring its care-of address (assuming that the node uses oDAD), the time of reception of the data packets on the new IPv6 link would directly depend on the time required to route these packets from the home agent to the mobile node and vice versa. The greater the topological (or geographical) network distance between the two, the longer it would take to complete the update of the CoA, HoA binding that the home agent maintains for this mobile.

As a result of the poor performance (in terms of achieving a seamless handover) of the standard 802.11 handovers and MIPv6 protocol, we have seen many proposals attempting to reduce the time of connection disruption accumulated during handovers. In the next section, we will describe those that we consider most popular and/or most efficient.

2.6 Work on optimizing the Layer 2 handover procedure

A possible classification of the various optimizations suggested in the related work would be to separate them depending on the network layer that they operate on. Some optimization proposals for example focus on layer 2 handovers in IEEE 802.11 networks. Others operate on layer 3 and focus on the MIPv6 protocol. A third type of proposals proposals provide complete solutions, addressing at the same time layer 2 and layer 3 issues. In this section we focus on optimizations targetting the layer 2 handover.

2.6.1 IAPP and Context Caching using Neighbor Graphs

[88] describes a method that allows APs to build neighbor graphs within a set of IEEE 802.11 wireless subnets, also referred to as Extended Service Set (ESS). The authors propose using that graph for caching and transferring MN authentication context information across APs with the purpose of accelerating the authentication stage of the 802.11 handoff process.

The optimization proposed in this document is based on the Inter Access Point Protocol (IAPP) [88]. IAPP was conceived in order to ensure a Single Association Invariant (i.e. maintaining a single association of a station with the wireless network) and the secure transfer of state and context information between APs involved in a reassociation reducing or eliminating the latency due to context transfer.

The protocol specifies two types of interaction for completing context transfer. The first form of interaction occurs between APs during a handoff and is achieved by the IAPP protocol, and the second form of interaction is between an AP and the RADIUS server.

Association and reassociation events change a station's point of access to the network. When a station first associates to an AP, the AP broadcasts an Add-Notify message notifying all APs of the station's association. Upon receiving the Add-Notify, the APs clear all stale associations and state for the station. This enforces a unique association for the station with respect to the network. When a station reassociates to a new-AP, it (the new AP) informs the old-AP of the reassociation using IAPP messages. Figure 2.11 shows the sequence of messages involved in the reassociation.

At the beginning of a reassociation, the new-AP can optionally send a Security Block message to the old-AP, which acknowledges with an Ack-Security-Block message. This message contains security information to establish a secure communication channel between the APs. The new-AP sends a Move-Notify message to the old-AP requesting station context information and notifying the old-AP of the reassociation. The old-AP responds by sending a Move-Response message.

For confidentiality of the context information, the IAPP draft recommends the use of a RADIUS server (to obtain shared keys) to secure the communication between APs. The RADIUS server can also provide the address mapping between the MAC addresses and the IP addresses of the APs, which is necessary for IAPP communication at the network layer.

The Inter AP Protocol was originally designed to only transfer MN context reactively. In other words, the context transfer was initiated only after the mobile station

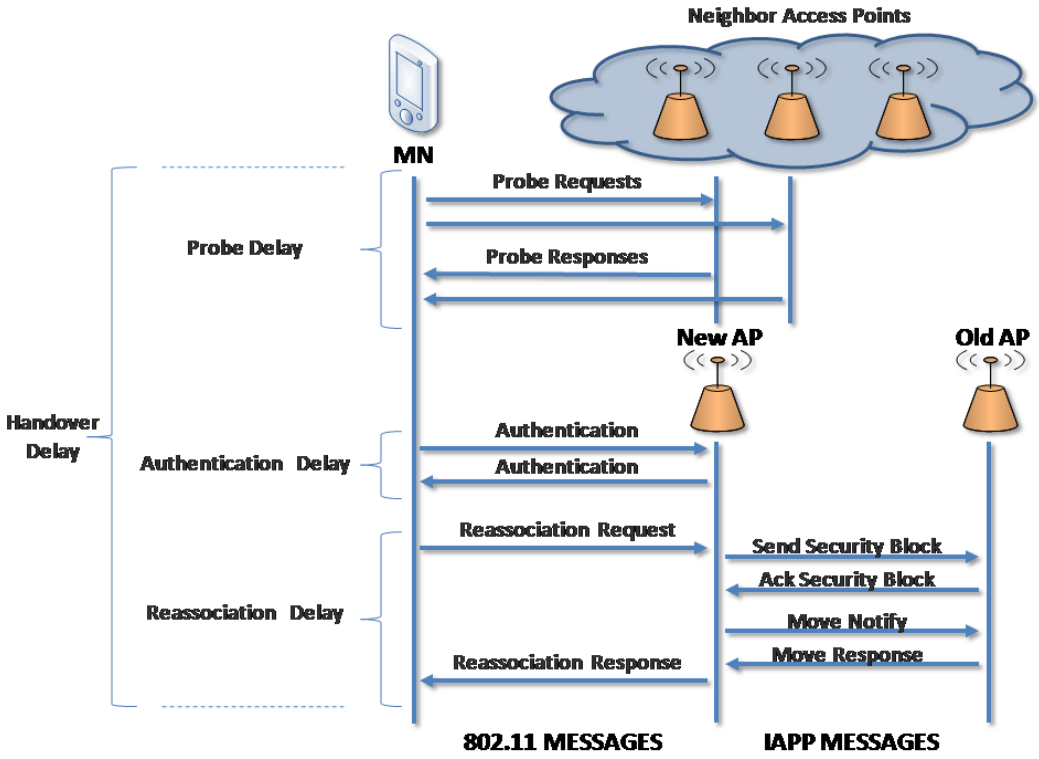


Figure 2.11: The WLAN Association Process (Active Scanning) with IAPP

had associated with the next base station or access router. This inevitably led to overall increase in the latency of the handover rather than reducing it.

With the addition of the CACHE-notify and CACHE-response messages [148], it became possible to proactively transfer the context of a roaming host. The CACHE-notify message is sent from an AP to its neighbors and carries the context information pertaining to the client. It is sent following a reassociation or an association request. A Cache-Response is sent in order to acknowledge the receipt of Cache-Notify. A timeout on this message results in the removal of the edge connecting the two APs in the neighbour graph as the new AP might not be alive (see Figure 2.12).

In order to actually build a neighbor graph, an access point could either use 802.11 reassociation requests (these requests contain the BSSID of MN's previous AP) or listen for incoming Move-Notify requests sent upon MN reassociation from neighbor APs via IAPP.

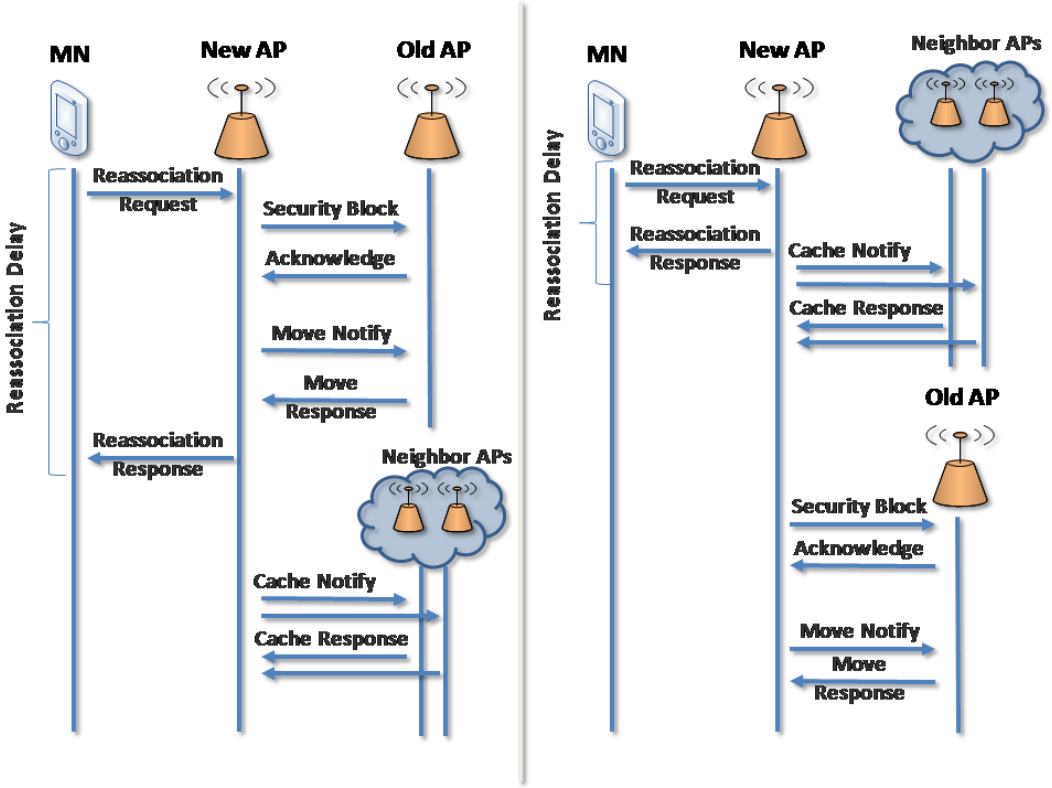


Figure 2.12: Cache miss and cache hit during reassociation with IAPP

The algorithm uses an LRU (Least Recently Used) approach for refreshing the graph which ensures removal of stale nodes. The last is also achieved by eliminating nodes that do not return a CACHE-response after receiving a CACHE-notify message.

Authors provide both simulation and experimental evaluation of their proposition. In the experiments they conduct, 114 reassociations occur with an average reassociation latency of 23.58 ms (including the one outlier) and 15.37 ms (without the outlier) for a cache-miss (traditional handoff), and 1.69 ms for a cache-hit.

2.6.2 Selective scanning and AP cache

Some mobility optimizations propose storing and using the roaming history of a mobile in order to optimize its layer 2 handovers. One of the most popular solutions of this kind is called *Selective Scanning and Caching* [173]. With it a mobile node would create a radio channels mask which would contain the Wi-Fi frequencies that have to be probed first during a handover. Over the time, together with going through successive

handovers, the terminals also build a history of the access points that they have completed association with. With every next handover, they will use the information that they have accumulated in their roaming history in order to avoid having to go through the whole phase of candidate access point discovery. As mentioned previously, this discovery phase represents the main source of the latency accumulated during the layer 2 handover. The authors of [173] also propose an approach that allows limiting the duration of the scanning phase even if the history does not contain any information from previous handovers. The details of this approach are described in the rest of this section.

After a terminal has activated its IEEE 802.11 interface, it needs to go through an active scanning procedure over all the 14 Wi-Fi channels in order to initialize its radio channel mask. For each *Probe Response* that it receives, the mobile would record the channel that the message originated on (see figure 2.13).

By default, the mobile would also add channels 1, 6 and 11 since they are generally preferred in many Wi-Fi deployments as the transmissions on any of them do not cause any interference with devices using the other two. In addition, they are also allowed in the majority of the countries in spite of the various frequency restrictions that have been imposed around the world.

Once this initialization has been completed, the mobile node would associate with an access point and would remove the channel that it uses since, chances are, that adjacent access points of the same operator would not use the same radio channel. During every next handover, the mobile node would start by only probing the channels that are present in its mask. If the mobile does not receive an answer on any of these channels, it would invert its mask and start the scanning procedure all over again. If after this second scan, there is still no response, the mobile would go back to the standard scanning procedure: it would go through all channels and create a new mask. This procedure is called *Selective Scanning* and according to the evaluation presented by the authors, it generally reduces the duration of the layer 2 handover to approximately 130 milliseconds.

Another important aspect of this solution is the fact that mobile nodes are expected to store a history of the access points that they have associated with. When a terminal connects to an access point, it creates a new entry in its roaming history and records the link layer address of its wireless interface as well as the number of the channel that it has connected with the AP on. For each entry in its history, the mobile node also records the details (link layer address and frequency) of the second closest access points (see 2.13). These details are discovered at the time a *Selective Scanning* run, or during one of the handovers. Before the terminal initiates a handover, it looks up in its history the parameters of the access points that it has recorded as being closest to its current attachment point. If there are no such records yet, the mobile executes the complete *Selective Scanning* procedure. In the opposite case, when there are records of

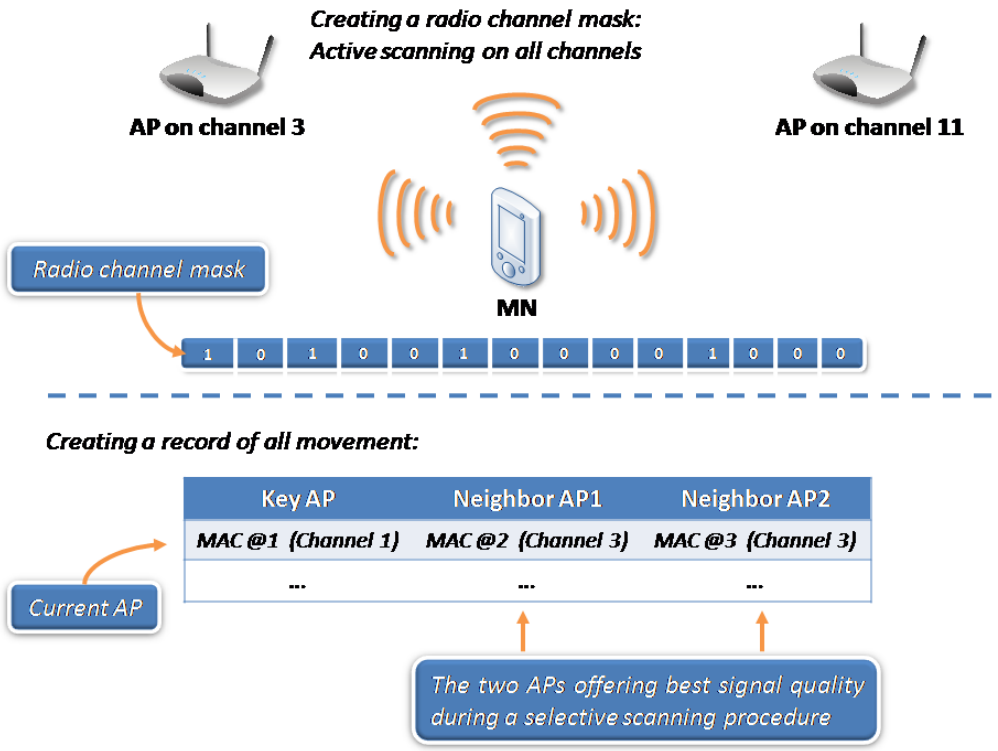


Figure 2.13: Creating a roaming history and a channel mask

two alternative access points located in the vicinity, it would try to connect with the first of them. If the AP is still available and the mobile succeeds in authenticating with it, the mobile moves to the phase of association. If, on the other hand, the authentication fails (the terminal is not authorized to use this access point or the access point is not within its range any more), it would move down its history list and try to with the second nearest alternative access point in it. If this authentication also fails, the terminal executes the complete *Selective Scanning* procedure.

Combined use of a roaming history and the *Selective Scanning* procedure makes it possible to reduce the latency accumulated during a layer 2 handover to approximately 3 milliseconds in some of the best cases [173].

2.6.3 Synchronized Beacons

The authors of an optimization technique called *SyncScan* propose a way of achieving rapid handovers based on a system that synchronizes the 802.11 control frames [157].

The idea behind it is that a mobile could periodically "listen" to the various *Beacon* frames broadcasted by the surrounding access points in order and use them to proactively build a list of the access points that are most closely located to it. The solution completely removes the use of active scanning and as a result allows to shorten the layer 2 handover procedure by bringing it down to only authentication and association. Moreover, such an approach allows for a finer management of the handover process and particularly, better choice of the "right" moment to trigger it since the mobile node is constantly monitoring the quality of the signal being emitted by the various surrounding access points. A terminal is therefore able to anticipate or delay the execution of a handover depending on the intensity of the signal being casted upon it by the most closely located access points.

In order for this solution to work the system clocks of the participating access points need to be synchronized as closely as possible. In order to achieve this the authors of the solution propose using the *Network Time Protocol* (NTP) [132]. Then, at a given instant, t, all access points implementing the solution and operating on channel 1 will emit a *Beacon*. At instant $t+d$, that is d milliseconds later, all access points set to operate on channel 2 would do the same and also broadcast a *Beacon*. The same repeats on all channels. This way, a mobile node associated with a particular access point, operating on channel c would be able to detect the access points which use channel $(c + 1)$ if it switches to that channel d milliseconds after the reception of a *Beacon* coming from its current access point.

In order to limit the risk of collisions during the synchronized broadcast of the *Beacon* frames coming from access points which use the same radio channel, the exact time of emission of the *Beacon* frames for a given channel varies slightly (e.g. $t+x, x \in [0; 3]$ milliseconds).

Periodically, over a certain period of time (e.g. every 500 milliseconds), a mobile node would execute a partial passive research with the intention of intercepting *Beacon* frames broadcasted by the nearby access points. Each one of these partial scans is carried out on a new channel.

Every time the mobile is about to switch to a new frequency in order to passively scan it, it indicates to its current access point that it would temporarily enter an energy save mode. This request is part of the IEEE 802.11 standard and it would make the access point put on standby all data packets that it needs to send to the mobile. This way the MN would be able to turn off its radio interface for a certain length of time. In the case of *SyncScan* this length would be a multiple of the frequency *Beacon* frames broadcast frequency and the MN would use it to switch to the channel that it wishes to scan. Using the energy save mode allows the mobile to complete its periodic passive scans without losing data packets. The change of the frequency that the mobile is listen-

ing to should always be synchronized with the broadcast of the *Beacon* frames. After scanning a new channel long enough to receive a *Beacon* frame, the node would record the parameters of the access point that it has just discovered, and return to its operating channel. It would then send a new request to its current access point, indicating that it is leaving its energy save mode.

The mobile repeats this procedure until it has accumulated the complete list of the surrounding access points. A node can thus directly begin the layer 2 handover procedure with the phase of authentication, which is enough to make link layer handovers imperceptible to the end user [157].

One of the problems with the *SyncScan* solution is that in reality, it is quite a challenging task to synchronize all participating access points. This is even more so for wireless devices because of the specificities of the physical medium.

Another problem comes from the fact that, depending on the wireless device and the implementation of the device drivers, switching from one Wi-Fi channel to another channel may have a considerable cost in terms of time. The switching interval may vary from 5 to 40 milliseconds [157]. That leaves a relatively small margin for adjustment to the exact time of the *Beacon* broadcast which makes fine grained control of the wireless devices especially important..

2.6.4 Reverse engineering Cisco System's Wireless Domain Service

Cisco Systems, the Sillicon Valley device manufacturer have implemented and propose in a number of their networking products a complete solution for fast and protected handover management spanning over both layer two and three [188, 189, 190]. Although this solution was designed for use in IPv4 networks, we also believe the general model to be applicable with IPv6 without substantial modifications. The name of this solution is *Wireless Domain Services* (WDS) and it is entirely based on proprietary protocols and requires specific network equipment (see figure 2.14).

The WDS solution introduces a new network entity called the WDS server. The role of this WDS server is, among other things, the authentication of access points as well as mobile terminals that are part of or use the system. It is also in charge of the management of all handovers of the using mobile nodes. In the following sections, we will describe the algorithms that this system uses in order to achieve rapid handovers. We will also present the mechanisms that it employs in order to guarantee network security.

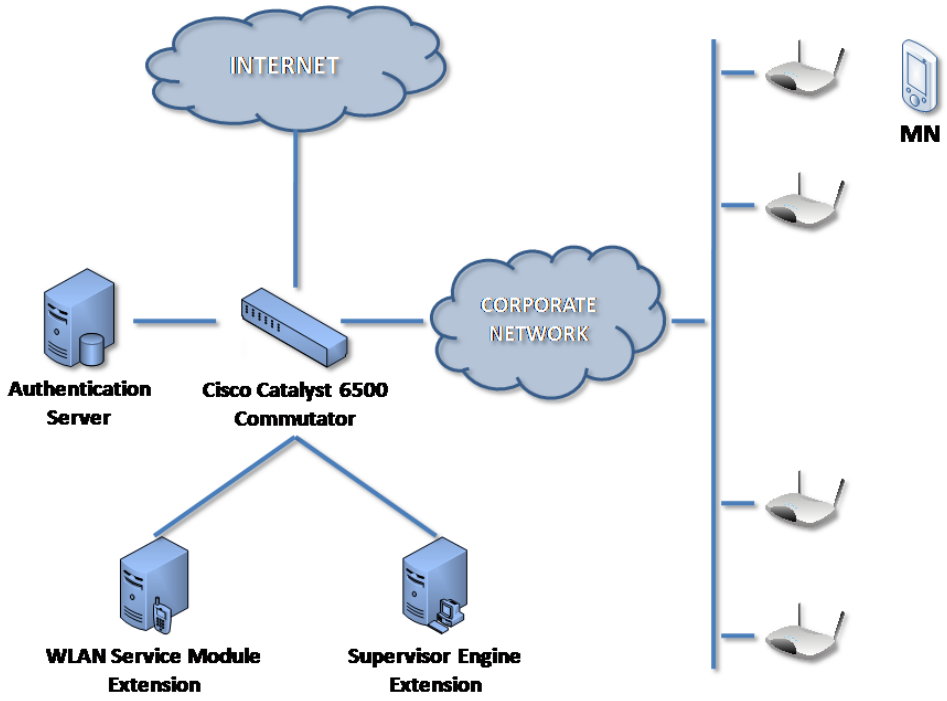


Figure 2.14: Various entities in the Wireless Domain Service

Layer 2 handovers

The major improvement that the WDS system brings to layer 2 handovers comes from the fact that the WDS server provides to all mobile nodes the list of access points located in their vicinity in order to optimize the scanning procedure defined in the IEEE 802.11 standard. Each access point in the network has to dynamically build a neighbor cache using information provided by the mobile nodes, through the WDS protocols, after they complete their associations. This information consists mainly in the parameters that mobiles have used to connect to their previous access points, including the radio channel, and their SSID.

When filling in their neighbor cache, access points also receive from mobile nodes the exact time when they have (been) disconnected from their previous access point. When this time exceeds 10 seconds, the corresponding access point is not considered as a neighbor and are as a result not considered as candidates for the WDS neighbor cache.

The time necessary for an access point to build a complete neighbor cache depends directly on the number of surrounding access points, the number of mobile nodes and the rate at which the mobiles move through the access points.

After a mobile node has associated with an access point, it would receive from it the content of the AP's WDS neighbor cache. During its next layer 2 handover, the terminal could therefore use this list and only probe the channels that are expected to be used by the neighboring access points. Right after obtaining its first response from one of the access points on the list, the terminal immediately authenticates and associates with it. This procedure thus makes it possible to strongly reduce the latency time accumulated during a layer 2 handover.

It is also worth noting that whether or not a mobile would use the WDS optimizations for a layer 2 handover depends on the traffic that was being exchanged by the mobile right before the handover. If the MN has received a data packet within 500 milliseconds directly preceding the handover, it will use the WDS optimizations the way we have just described. In the opposite case, that is, if the terminal did not have any time-critical communication sessions in progress (e.g. real time), it is assumed that a fast handover would be unnecessary. In this case, the terminal does not use the neighbor cache and carries out a standard 802.11 handover.

In order to benefit from the layer two handover optimization of the WDS system, it is necessary for both the wireless network cards and access points to be manufactured by Cisco Systems. For layer 2 only optimizations, there is no need for extra equipment since one of the access points could also play the role of a WDS server and handle the authentication of the access points and the mobiles.

Layer 3 handovers

In the WDS system, support for layer 3 mobility is not based based on the Mobile IP protocol but it uses a relatively similar approach. A logical overlay layer 3 network is set up above over the IP network. This overlay network will make it possible for mobile terminals to keep the same IP address independently of IP subnet that it is actually on. As a result, this system goes around one of the fundamental principles of the protocol IP which states that the IP address of a terminal also corresponds to its current location.

In order to activate management of layer 3 mobility, it is necessary for a network to be equipped with a *Cisco Catalyst 6500 Series Wireless LAN Services Module* (WLSM) and a *Supervisor Engine 720* (SE). In this setup, it is the Cisco Catalyst that plays the role of the WDS Server (again through the use of the WLSM module).

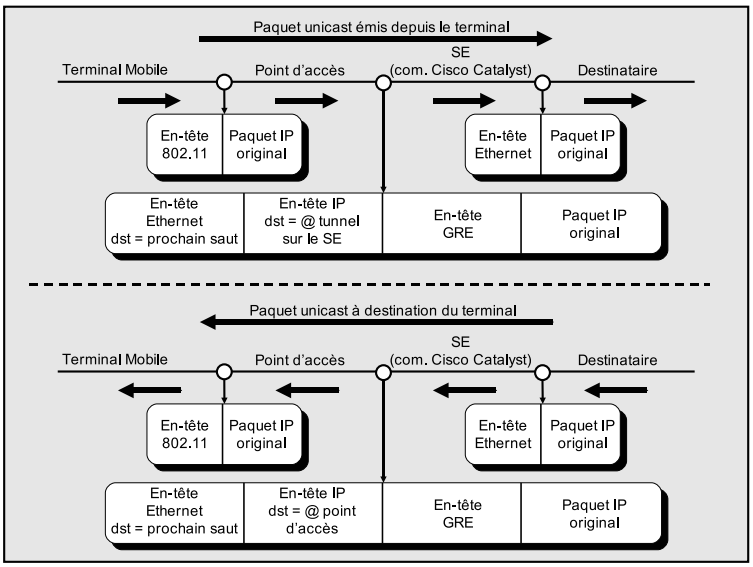


Figure 2.15: Using an overlay network for the transfer of unicast data packets

Creating the logical network used by WDS for layer 3 handover management is above all the role of the SE and the access points. The SE uses an alternative of the *Generic Routing Encapsulation* mechanism (GRE [74]) called *multipoint Generic Routing Encapsulation* (mGRE). This mechanism makes it possible to create simple communication tunnels between a single source and multiple recipients. Usage of various tunnels using the mGRE encapsulation between the SE and the access points is what actually constitutes the logical layer 3 network. All the unicast IP data packets must go through this logical network in order to reach their destinations. As a result, the SE can be regarded as the central point of the network (see figure 2.15). On the other hand, all control traffic (between the access points and the WDS server) does not use the logical network and employs another proprietary protocol called *Wireless LAN Context Protocol* (WLCP). This last is a somewhat modified version *Light-Weight Access Point Protocol* (LWAPP [38]) which is standardized by the IETF. The primary purpose of these protocols is to centralize the management and the configuration of all access points in a wireless network.

The SE also maintains an up to date local copy of mobility cache, in which it records IP address, link layer address and the current access point of each mobile currently in the system.

The SE also creates a tunnel for each IP subnet controlled by the WDS server, and assigns it to a mobility group. When a mobile arrives in the network, it must be identified by the WDS server before it is granted access to the network. At the time of the request

of the mobile, the WDS server transmits to the SE a message including the IP address and the MAC address of the terminal as well as the identifier of its current access point. After receiving such a message, the SE updates its mobility neighbor cache and adds the terminal to a mobility group according to its IP address. It also registers the access point that the terminal is currently associated to, as a new end point of the tunnel which corresponds to this mobility group. NEXT, the SE sends a message to the WDS server notifying it of this last update. The WDS server then transmits this message to the current access point of the mobile node so that it could also update its own associations table. This table makes it possible to transmit the IP packets which arrive through the tunnel towards this mobile node and vice versa.

When moving through access points that lie in different IP subnets, mobiles carry out almost the same procedure as the one that we described in section 2.6.4. The mobile node itself remains associated with the same mobility group since it does not change its IP address and regardless of the actual IP subnet corresponding to its new location.

However, what does change is the tunnel associated with this mobility group. The new access point is therefore added as one of the endpoints of this tunnel while the old access point may be removed since it may not be used any more by any of the mobile in this mobility group. In other words, mobile nodes in the Cisco Systems WDS system do not in any way intervene in the layer 3 handover management, and the entire procedure is only managed by the network itself (see figure 2.16).

Security

The WDS System uses the 802.1X protocol [18] to guarantee security in the network. When using the basic scenario of this solution (i.e. the one that only covers layer 2 mobility), mobile nodes have to authenticate against the WDS authentication server after each layer 2 handover. However, such authentications tend to be relatively time consuming since they require a long exchange of authentication messages. As a result, these procedures may account for a substantial part of the layer 2 handover latency. WDS optimizes this stage with the joint use of a centralized key management system called *Cisco Centralized Key Management* (CCKM) and the *Cisco Lightweight Extensible Authentication Protocol* (Cisco LEAP) protocol which make it possible for the WDS server to also the role of an authentication server.

The phase of authentication against the WDS server is executed during the association phase of the layer 2 handover and introduces only two extra WLCCP messages between the access point and the WDS server: a pre-registration request and response.

Finally the WDS system reduces the latency accumulated by the layer 2 and layer 3 handovers to less than 50 milliseconds [188, 189, 190]. Figure 2.16 shows examples of layer 2 and layer 3 handovers in a WDS system.

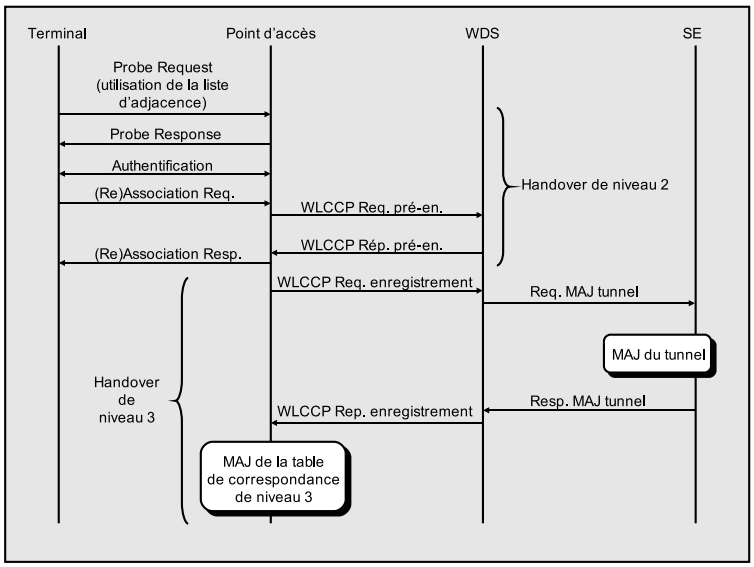


Figure 2.16: Layer 2 and layer 3 handovers with the WDS system

2.7 Layer 3 Movement Detection Optimizations

Movement detection is the task of determining whether an IPv6 node has moved to a new subnet. This detection is important since in the case that the device has moved, the addresses that it had been using so far become invalid and additional configuration must be undertaken in order to re-establish or maintain upper layer connectivity.

Movement detection is accomplished by determining that the current router is unreachable, and discovering a new router available on the particular subnet. The following sections go through some of the most popular techniques in optimizing this part of the network handover.

Subsection 2.7.1 presents an optimization of router discovery that consists in making ARs broadcast RA messages on a high frequency and thus allowing newly arrived mobile nodes to quickly detect subnet configuration settings. Section 2.7.2 goes through a description of a mechanism proposing immediate responses to Router Solicitations (as opposed to those randomly delayed by an interval between 0ms and 500ms by standard). In 2.7.3 we expose a scheme that makes APs cache the last seen RA and re-emit

it each time an MN associates with that particular AP. Section 2.7.4 describes the very common notion of aiding movement detection through link layer triggers and 2.7.5 proposes incorporating link identifiers into RAs and thus aiding unreachability detection of a previous AR.

2.7.1 Periodic Router Advertisement Beaconing (Dense RAs)

Beaconing is a term used to refer to broadcasting of network identification information at regular intervals. Mobile IPv6 reduces the lower bound of the MinRtrAdvInterval and MaxRtrAdvInterval to 30 and 70 ms respectively [103]. With these settings, beacons will be sent no more closely than 30 ms apart, and with no greater separation than 70ms. Routers are required to send the beacons at random times within this interval. This means that an MN will receive an RA within 70ms of arriving on the link, and may expect to receive an RA within 25ms, if we assume MN entry into the network to be randomly distributed in the interval.

This technique requires no action on the part of the MN other than listening to RA multicasts. The bandwidth consumption by multicast beacons is 14 kbps when RAs only include one Prefix Information option. Addition of a Source-Link-layer-Address option and a MIPv6 Advertisement Interval option typically increase this to 16.6 kbps.

On some networks, such overhead (20 multicast packets per second) causes a serious burden on network bandwidth. In these cases, RFC2461 [142] specified intervals should be used, if other movement detection mechanisms are available.

Additionally, the reduced interval between messages may have side effects for non-MIPv6 nodes on the same networks. The AdvDefaultLifetime value is used to set the lifetime of the default router in seconds, as advertised in the Router Lifetime field of the RA. The minimum value specified in [142] for this value is MaxRtrAdvInterval. This value is less than one second when using MIPv6 specified advertisement intervals. Even if default router lifetimes are rounded up to the nearest second, nodes which assume MaxRtrAdvInterval is at RFC2461 values could be confused about the lifetime of their default router. Routers should ensure that AdvDefaultLifetime is greater than or equal to 4 seconds, in order to avoid this confusion.

2.7.2 Fast Router Advertisement (FastRA)

Fast Router Advertisement [107] removes the random delay required of a router (between 0 and 500 ms as specified by Neighbor Discovery) before it responds to RS mes-

sages. It relies on the fact that only one router on a given subnet would be configured for FastRA, so that responses are not simultaneous. FastRA principally aims at delivery of unicast RA messages, since the rate limiting of multicast RA messages specified in RFC2461 mandates that RA messages may not be sent within 3 seconds of each other.

This could be easily dealt with if FastRA adopted the MinDelayBetweenRAs value advised by MIPv6. This way, responses to RS request would only be delayed if the last multicast RA occurred more recently than MinDelayBetweenRAs before the RS arrived at the router (or MaxFastRAs has been consumed). This could work well if the arrival of mobile nodes occurred much less frequently than the unsolicited multicast RA interval.

FastRA incorporates a rate limiting feature aimed at diminishing the potential effect of FastRA traffic on nodes which are already connected to the network. Routers may transmit no more than MaxFastRAs advertisements in an interval before discarding solicitations until the next unsolicited multicast RA.

Either of the solicitation mechanisms may be used to get FastRA response from a router, although Advertisement Interval timeout will only be invoked on packet loss if Link-triggers are available. Movement detections times are bounded only by the time to send the Multicast RS message and send the unicast RA response. Recent testing has indicated 95% of RAs were received within 15 ms of sending an RS on 802.11b networks, when Neighbor Discovery was being performed on the MN's Link-Local address. If RS messages include Source link-layer Address options or are multicast responses with no timer delays, movement detection time will be lower.

2.7.3 RA caching in Access Points (Fast Router Discovery)

One method which requires no solicitation from the MN is network triggering of RA. Router advertisements are sent to the MN when it attaches to an access point (AP) associated with this network. In network deployments, the router may not be the link-layer device which the MN connects to, and therefore may be unaware of MN link-connection. Only in the case where the the AP advises the router of connection or AAA state, can the router send (unscheduled) unsolicited RAs before receiving packets from the MN.

The Fast Router Discovery (FRD) [95] places the responsibility of sending triggered RA messages upon APs. The Access Points cache RA's recently sent from the router, and deliver a frame to the MN when it connects. This frame is datalink-unicast to the MN and contains the most recent unsolicited RA.

In this case, less frequent transmission of unsolicited multicast RA messages may be used. At the same time, the first frame which is queued for the MN is the RA required for movement detection. Deployment of FRD requires each of the APs for the network to be capable of both the caching and triggered sending operations. Analysis and experimental results indicate that this is potentially the fastest network-layer based movement detection optimization, dependent on AP processing capacity.

2.7.4 Link Layer triggers on the Mobile Node

For situations where RA packets have been transmitted correctly, the Advertisement Interval's best case occurs when the timeout occurs just after the MN joins a new link. In environments where the MN can receive an indication that a link has been joined (e.g. WLAN association has ended), this information can be used to either simply trigger an RS immediately (although once again a random delay before sending RSs is advised by [142]) or play a role in determining a mobility management policy by the mobile node and assist upper layers during handover [191, 67]

Additionally, even if the MN doesn't send an RS upon receiving a link-up trigger, it can use the trigger to validate received RA messages for movement detection with eager-binding. The MN may be able to enter an 'eager-binding' state until it receives its first RA on the new link. If it receives an RA from a previously unseen router, it may be useful to confirm bidirectional reachability with it, and then undertake movement signaling.

If the MN joins a subnet, then until it knows the identity of the segment (i.e. before it has received an RA), it MUST assume that it is on a new link, where its Link-Local address is tentative. This means that RS messages will either be deferred until DAD operations have been performed on the Link-Local address or the RS MUST be sent with an unspecified address. A multicast response will be scheduled no sooner than:

$$Max(LastMcastRATime + MinDelayBetweenRAs, \quad (2.1)$$
$$now + 0 - 500msRSResponseDelay) \quad (2.2)$$

Where MinDelayBetweenRAs could be as high as 3 seconds. Even if the response is not multicast, the RS response delay is still incurred.

2.7.5 RA Link Identification for Mobile IPv6 Movement Detection

In IPv6 multiple routers are allowed on a link, and these routers do not have to advertise overlapping prefixes. Therefore, reception of an RA from a new router does not always mean that movement has occurred. For reliable movement detection, nodes can check the reachability of the current router. Determination that the current router is unreachable is typically a slow process, but provides safeguards which allow nodes to be sure that movement has occurred.

Link Identifiers (LinkIDs) [28] are numeric values automatically configured on a link, which are extremely unlikely to conflict with an identifier on an adjacent link.

The Link Identifier is randomly generated by one of the routers on a link and all of the other Link Identifier supporting routers on that link agree to use that identifier.

Mobile Nodes (MNs) receiving Router Advertisements (RAs) with LinkID options can use the LinkID value to identify the link that they are attached to. This may aid movement detection by allowing MNs to determine when the link changes. A change to the LinkID implies to the MN that currently configured router is unreachable.

All routers supporting the Link Identifier Option advertise it in each of their Router Advertisements. Advertised Link Identifiers are consistent within any one link.

Mobile Nodes store the Link Identifier internally, for comparison with subsequently received Router Advertisements.

Upon receiving an RA with a LinkID that differs from the MN's currently recorded value of LinkID, the MN can assume that it has moved to a new network and that its current default router is unreachable. This information may be used to initiate further stateful or stateless autoconfiguration procedures, or appropriate mobility signalling by the MN.

When an MN receives an RA from a previously unseen router, which contains the same LinkID as its default, this means the MN is on the same link, but does not guarantee reachability for the current default router.

2.8 Optimizing DAD

In order to generate a unique address, IPv6 nodes perform the Duplicate Address Detection (DAD) procedure [142]. The neighbor discovery specification states that payload

packets can not be sent or received before this procedure is complete. In an environment where frequent movements are expected, this delay can be problematic. As the reduction or removal of the delay would be quite useful, different DAD optimizations have been proposed. Optimistic DAD (presented in section 2.8.1) consists in allowing the use of tentative addresses before DAD has actually completed. Subsection 2.8.2 describes a modification of ARs making them keep a list of all MLD listener reports sent by nodes whenever they perform DAD and then transmit this list to newly arrived MNs which may thus determine whether or not they are the first to join this group and thus the only one to be using the corresponding address.

2.8.1 Optimistic Duplicate Address Detection

Optimistic DAD [139] is a modification of the existing IPv6 Neighbor Discovery [142] and Stateless Address Autoconfiguration process. Its intention is to minimize address configuration delays in the case of successful duplicate address detection, and to reduce disruption as far as possible in the case DAD fails.

Optimistic DAD is based on the assumption that DAD is far more likely to succeed than fail for a well-distributed random address. Disruption is minimized by limiting nodes' participation in Neighbor Discovery while their addresses are still Tentative.

The main idea behind preventing disruption in the case of a failure is making sure that an Optimistic Node (ON) does not, while tentative, send any messages which will override its neighbors' Neighbor Cache (NC) entries for the address it is trying to configure: doing so would disrupt the rightful owner of the address in the case of a collision. The document offers a set of necessary modifications and additions to [142] and [192] so that such an ON implements the proper behavior.

2.8.2 Duplicate Address Detection Optimization using IPv6 Multicast Listener Discovery

[75] describes a possible optimization to Duplicate Address Detection (DAD) which can be used to successfully terminate DAD for a relatively short period, based on the presence of listeners on the link-scope solicited nodes multicast address (i.e. the address of a multicast group where all potentially conflicting addresses belong). The document makes use of the idea that Nodes that have completed DAD, and own an address listen on the link scope solicited-node multicast address [60] related to the IPv6 address which they have completed DAD on. When a new node attempts to configure an address on

the network, it sends a neighbor solicitation message to the same address, and succeeds if no device responds to this message within a timeout period. As part of this process, the new node also listens to the solicited-node multicast address

The MLD RFC requires a node to send an MLD report before listening to the solicited-node multicast address [197]. This process allows informing the router on a link about the presence of listeners for the address, so that a multicast-group can be managed.

To sum up, the optimization described in this subsection allows nodes to ask the router to tell them if they are the first node to enter this multicast group. If they are the first to enter the group then it follows that no-one else is currently performing DAD defense against the required unicast address.

2.9 Work on optimizing the Layer 3 handover procedure

2.9.1 Hierarchical Mobile IPv6

The purpose of introducing a hierarchical architecture is reducing considerably the time necessary to update the binding cache entries that a home agent uses to map care-of and home addresses. One way to do this is to separate the network in different domains which are independent of the actual IP subnets and whose size is defined by the operator or the administrative entity in charge of the network. Throughout the years there have been various models suggesting to do this in different ways. The *Hierarchical Mobile IPv6 Mobility Management* (HMIPv6) is one such solution. Work on it started at INRIA Rhone-Alpes by Claude Castelluccia around the year of 2000 [40], and it was then continued in the MIPSHOP working group at the engineering task force until it was standardised as RFC 4140 [184] in August 2005.

For every administrative domain, the HMIPv6 protocol introduces, a new network entity called a *Mobility Anchor Point* (MAP). The MAP primarily plays the role of a local home agent, making it possible to mask from home agents and correspondents, the majority of the moves that a mobile terminal would undertake, in order to minimize the traffic between the mobiles and their home agents. In addition to traffic, the HMIPv6 protocol also allows to decrease the time necessary for a mobile to update its address record after moving. Figure 2.17 illustrates the procedure defined by the HMIPv6 protocol.

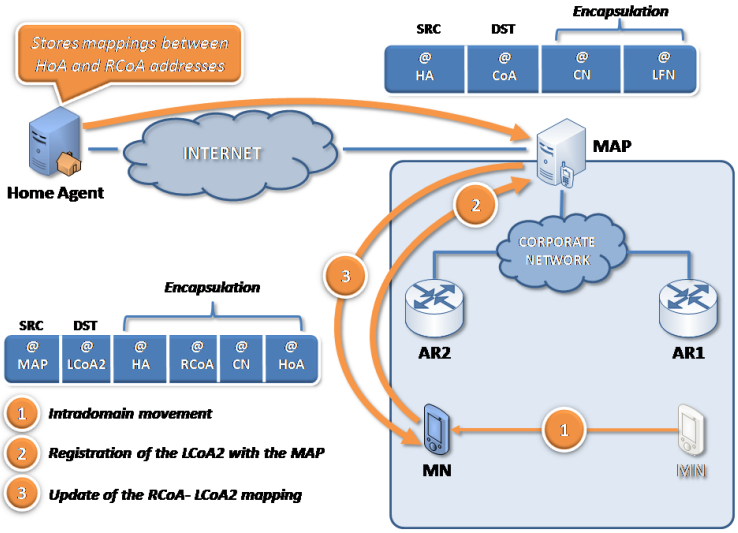


Figure 2.17: The HMIPv6 protocol

When a mobile node arrives in a new domain, it receives the address of the MAP (or MAPs) responsible for this domain through a newly defined option of the RA messages. This new option also allows the mobile node to configure a valid temporary address in the same sub-network as that of the MAP. Such addresses are called *Regional Care-of Addresses* or RCoA and will be preserved by the terminal as long as it remains in the domain. In addition to the RCoA and according to the IPv6 mechanisms that we've described in previous sections, the reception of an RA allows the mobile to configure a care-of address for use in its current sub-network. In HMIPv6 CoA addresses are called *one-Link Care-of Addresses* or LCoA. After obtaining these two addresses, the node sends a first BU message to the MAP in order to establish a new binding between its RCoA and LCoA addresses. This way the mobile creates a bidirectional tunnel, similar to the one used in the standard MIPv6 protocol, between the MAP and its current location. In the same time, the mobile node also sends a second BU, this time to its home agent, in order to create a binding between its RCoA and its home address. After receiving this BU, the home agent updates its binding cache and starts relaying towards the RCoA address that it has just learned all MN bound packets. These packets would be intercepted by the MAP which would, on its turn, relay them to the current location of the terminal (presumable in the same domain) through the bidirectional tunnel that was established with the first BU (see figure 2.17).

After moving from one subnet to a new one inside the domain, the mobile node obtains a new LCoA which it would again send to the MAP. As long as it remains in the same domain however, the MN would keep the same RCoA which makes its movement

completely transparent at the level of the home agent (see figure 2.17). As a result, the time necessary to update the temporary addresses of a mobile is considerably reduced because the MAP and the MN are both located in the same domain.

After changing a domain the mobile node would have to obtain a new RCoA address, and would therefore have to update its registration with the home agent. The change of domain is detected through a new option transported by the router advertisements.

2.9.2 The FMIPv6 protocol

The protocol Fast Handovers for Mobile IPv6 (FMIPv6) is an improvement of the MIPv6 protocol and is, without a doubt, among the most promising handover optimizations. Work on it was first described in [111] by Rajeev Koodli and Charles Perkins and it was later accepted as an Internet Standard [70]. Its purpose is reducing to a maximum the time of connection disruption on the layer 3 and limit, as much as possible, the loss of data packets. The main idea behind is for the mobile node to try and determine the IPv6 subnet that a node is going to connect to before it has actually done so.

The FMIPv6 protocol defines new mechanisms that allow mobile terminals to ask their current access routers for information on all surrounding access points as well as the layer 3 parameters of the IPv6 subnets that they are located on. Requests for such information are sent out through a new type of ICMP messages called *Router Solicitation for Proxy Advertisement* requests (RtSolPr). Responses to such requests are sent by the currently active access router through another type of specially defined ICMP messages called *Proxy Router Advertisement* (PrRtAdv). The RtSolPr messages are anycasted by mobile nodes on the local link and they are generally addressed to all current access routers (the ff02::2 anycast address). These messages can contain the identifier of one or more access points that the mobile node would like to retrieve information for. Exactly which access point identifiers would be included in the RtSolPr request is outside the scope of the FMIPv6 protocol but it is generally expected that after performing a layer 2 scanning procedure the mobile node would come up with a list of surrounding access points and their IDs. It could then use these IDs (generally the layer 2 address of the access point) in the RtSolPr message since they are the ones that the mobile node is likely to move to.

A node could also request information for the complete list of access points and routers known to its current access router by sending a request for a special access point address (the all zero address).

When it receives an RtSolPr message, an access router would reply by sending to the mobile a corresponding PrRtAdv message containing the requested information. This information is transported in the form of one or more [AP-ID, AR-Info] t-uples. Each t-uple contains the MAC address of the access point that the information is about, the MAC address of the access router behind it, the IPv6 prefix announced by this access router and the length of this prefix. After obtaining such information a mobile node is ready to initiate a fast handover although it would not always do so. One characteristic of the FMIPv6 protocol is that it allows to completely separate the candidate access point discovery phase from the rest of the handover. The FMIPv6 scanning phase is described on figure 2.18.

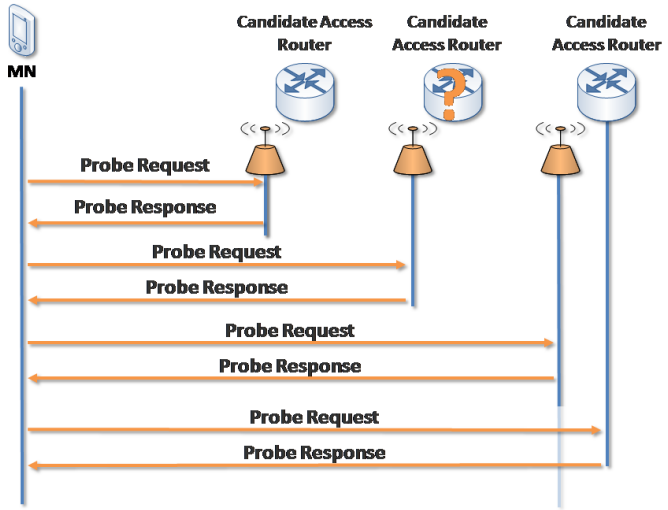


Figure 2.18: Candidate access point discovery with FMIPv6

The FMIPv6 protocol defines two ways to actually proceed with a handover. They were originally referred to as *Predictive* and *Reactive* Context transfer by [111] and were later described by RFC 4068 as two different modes of operation called *Predictive* and *Reactive* mode [70].

A mobile node would use the predictive mode of operation in cases where it is capable of anticipating the need for carrying out a handover (e.g., by using information of level 2 such as the quality of the radio signal). After detecting the need for a handover the mobile would send to its current access router a message defined by the FMIPv6 protocol, and using the mobility header packet format, called a *Fast Binding Update* (FBU). This router is referred by the FMIPv6 specification as a *Previous Access Router* (PAR) since the mobile node expected to leave the link that it is responsible for. FBU messages contain the current temporary address of the mobile terminal (called Previous

CoA or PCoA for the same reasons as above) as well as the identifier of the access router towards which the terminal is planning on moving. This new router is called *Next Access Router* or NAR as the mobile node is expected to connect to a link that it is responsible for after leaving its current subnet.

After receiving an FBU, the PAR sends a *Handover Initiate* message (HI) to the NAR. HI messages contain various information on the mobile node such as its link layer address, PCoA and (optionally) the address that the terminal would like to use in the sub-network of the NAR (NCoA). This message is used to notify the NAR of the intentions of the mobile terminal in order to as a first step in verifying whether the handover is feasible. The NAR can either accept or refuse the handover using a *Handover Acknowledge* message (HACK). After receiving the HACK, the PAR sends to the terminal a *Fast Binding Acknowledgement* message (FBACK, another mobility header message) in response to its earlier FBU. The FBACK is the last of the FMIPv6 messages exchanged during a handover and once it has received it, the mobile node is ready to move.

It is important to note that in addition to simply triggering the HI message the FBU also serves as a means for the mobile node and the PAR to establish a bi-directional data tunnel. All the data packets destined for the mobile node are intercepted by the PAR and transmitted to the node's NCoA, and hence the NAR, through this tunnel.

When it receives a message destined for the NCoA, the NAR could either transmit it directly to the MN, provided it has already arrived, or use the buffering mechanisms defined by the neighbor discover protocol in order to store them until the MN has turned up in its local link, at the end of the handover.

After arriving on the link of the NAR, the terminal would notify the NAR of its arrival using a *Fast Neighbor Advertisement* (FNA) mobility header message, which is supposed to update its neighbor cache and unlock the transmission of all data packets that it has buffered for the time of the handover. The FNA is thus the last message of signalling defined FMIPv6 protocol.

The bidirectional tunnel established between the PAR and the NAR is generally maintained over a period of time that is sufficiently long for the terminal to update its registration with the Home Agent, and notify it of its new temporary address (NCoA), all this without losing any data packets. As long as this tunnel is active, (given the fact that it is bidirectional) the terminal is also able to send data packets using its previous care-of address (PCoA). The predictive mode is illustrated on figure 2.19.

The FMIPv6 protocol also defines a reactive mode of operation, which represents the cases where the mobile terminals are either unable to anticipate their movement or are

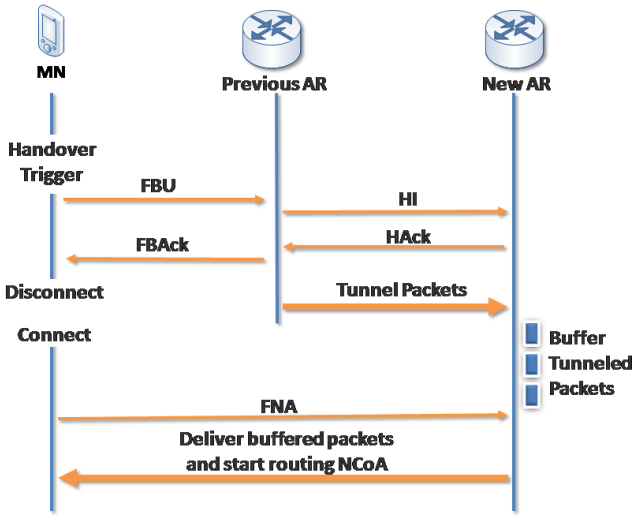


Figure 2.19: Predictive mode of operation of the FMIPv6 protocol

forced to leave their current link without waiting for the *Fast Binding Acknowledgement*. In such cases most of the FMIPv6 is carried out once the layer 2 handover has been completed. In these particular cases, the FBU is sent encapsulated in an FNA and from within the new subnet (the one of the NAR). This is done regardless of whether an FBU has also been sent from the previous link, since the mobile has no way of knowing whether this FBU has been received.

After receiving an FNA containing an FBU, an NAR would extract and transmit the FBU to the PAR. From this point on, the exchange of the HI/HACK messages proceeds the same way as in the predictive mode, which allows the PAR to transmit data packets towards the NCoA of the mobile through the same bi-directional tunnel that it uses in predictive mode. The reactive mode of operation is illustrated on figure 2.20.

2.9.3 S-MIP: A Seamless Handoff Architecture for Mobile IP

In [85] authors define the following as primary objectives of the S-MIP protocol: “Extreme Low Handoff Latency, Minimal Handoff Signaling, Indoor Large Open Space Environment, Scalability, High Availability, Fault Tolerance”.

The key concepts in the article are:

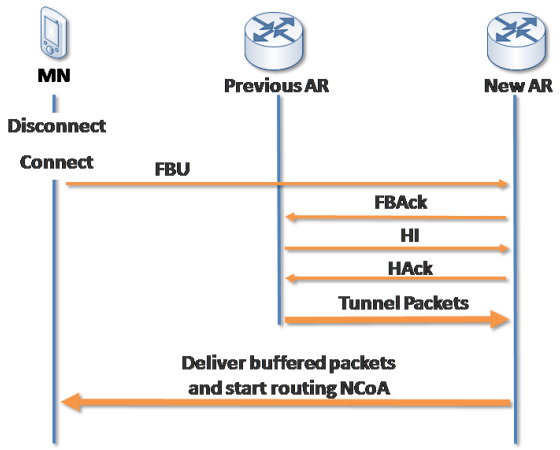


Figure 2.20: Predictive mode of operation of the FMIPv6 protocol

Combined usage of the FMIPv6 [70] and HMIPv6 [184] protocols: No changes are made to FMIPv6 and HMIPv6 protocol semantics.

Synchronized-Packet-Simulcast (SPS): By SPS authors mean simulcasting (i.e. bi-casting) packets to the current network that the mobile node is attached to, and to the potential access network that the mobile node is "asked" to switch onto.

A Decision Engine: The Decision Engine (DE) is similar to a MAP in its scope, and makes handoff decision for its network domain. Through periodic feedback information from individual ARs, the DE maintains a global view of the connection state of any mobile devices in its network domain, as well as the movement patterns of all these mobile devices. The DE is also capable of offering load-balancing services where it is able to instruct ARs, responsible for fewer mobile devices, to take on new mobile device, rather than ARs that are closer to the mobile device.

Figure 2.21 illustrates a basic S-MIP scenario. S-MIP defines six new messages:

- Current Tracking Status (CTS) message from the MN to DE. It contains location tracking information.
- Carrying Load Status (CLS) message from the ARs to DE. The CLS message contains the information regarding how many mobile devices an AR is currently managing.

- Handoff Decision (HD) message from the DE to ARs. The HD message contains the outcome of the handoff decision at the DE, namely which AR a MN should handoff to.
- Handoff Notification (HN) message from the PAR to MN. The HN contains the indication from the PAR to the MN, directing exactly which NAR the MN should handover to (this is sent in combination with the PrRtAdv). The PAR derives the content of the HN message from the received HD message.
- Simulcast (Scast) message from PAR to MAP. The Scast message triggers the start of the SPS process.
- Simulcast Off (Soff) message from NAR to MAP. This message terminates the SPS process.

Like HMIPv6 [184] the S-MIP protocol also modifies RAs in order to distribute the address of the Decision Engine.

Note: During the following authors seem to consider an AP and an AR to always coincide and call both an AR. It should also be noted that FMIPv6 protocol semantics been different at the time the article was written as they do not match those of the latest FMIPv6 version even though authors explicitly declare them to remain unchanged.

Upon receiving a beacon advertisement message from the newly discovered ARs (or rather APs), the MN initiates a handoff by sending the RtSolPr message to the PAR (currently a handover is only initiated after an FBU message and not an RtSolPr). At that point the PAR sends HI messages, to all the potential NARs identified by the MN in the RtSolPr message. This HI message contains within the requested care-of address (CoA) on the NAR and the current care-of address being used at the PAR. All NARs will respond to the HI message with a HAck message either accepting or rejecting the new CoA. As specified in [85], if the new CoA is accepted by the NAR, the PAR sets up a temporary tunnel to the new CoA. Otherwise the PAR tunnels packets destined for the MN to the NAR, which takes care of forwarding packets to the MN temporarily. In response to the RtSolPr message, the MN will receive a PrRtAdv messages from the PAR, similar to the hierarchical scheme.

ARs send CLS messages to the DE periodically (every 3 seconds approximately) as a reply to the modified Router Advertisement message (with DE reply option). The CLS message indicates how many mobile devices are associated with a particular AR as well as the IP addresses of those MNs. A CTS message is generated by the MN every time it receives a L2 beacon advertisement from the ARs. Each CTS message contains the signal strength of the detectable AR and the respective Ids of the AR. The

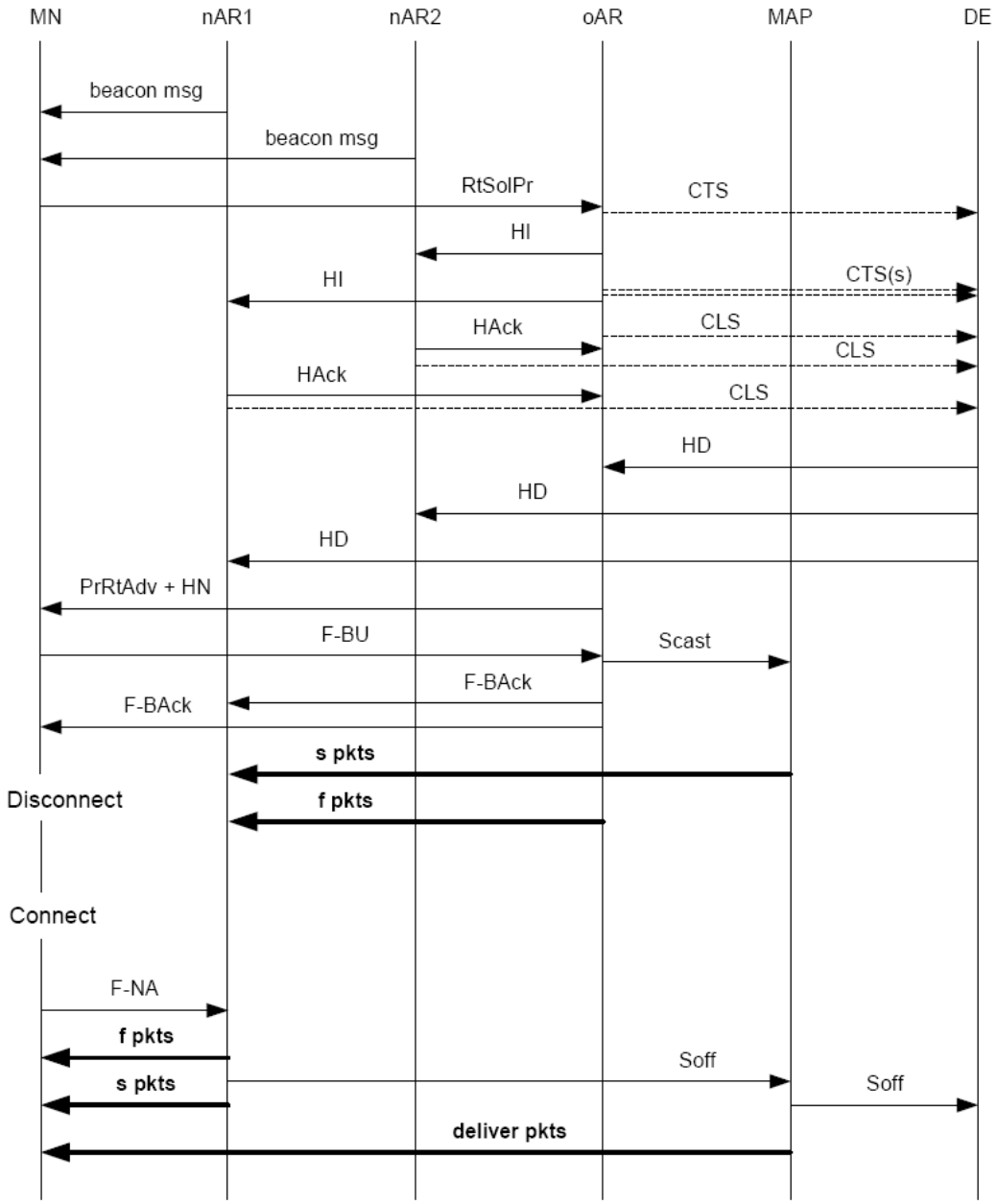


Figure 2.21: A handover scenario using S-MIP

signal strength and AR Ids serve as MN's location tracking information. The current AR will forward this tracking information (CTS) every second until the reception of the HD message from the DE. The MN may send the tracking information to other ARs, other than the current AR, if the connection to the current AR is poor. In this

case, duplicated CTS messages will arrive at the DE, which will simply discard the duplicated messages. After analyzing the CTS and CLS messages, including tracking the mobile node's movement for a short period (minimum of 3 seconds, analysis shown in Appendix C), the DE sends HD messages to all participating ARs for the specific mobile node requesting a seamless handoff. In turn, the PAR sends a HN message together with the PrRtAdv to the MN indicating exactly which AR that the MN should switch to.

Next, after receiving the HN, the MN sends a FBU message to the PAR containing the new care-of address. Upon receiving it, the PAR would send an Scast message to the MAP initiating the simulcasting of packets. All packets received by the MAP and destined for the MN are now sent to both the PAR and the NAR. After receiving the FNA the NAR will start delivering packets from the FMIPv6 buffer to the MN and once this buffer is empty it would turn off bicasting by issuing an Soff message towards the MAP.

Authors provide an analytical evaluation (simulations) of S-MIP that "prove" it to be an effective protocol. It should be noted that very little detail of these simulations is actually given to the reader. Furthermore, the exposed results show a connection loss inferior to 100ms which most probably means that L2 handover time and connection loss are most probably not considered. Nevertheless, the document introduces some worthwhile concepts such as the presence of a Decision Engine, and network aided handover based on AP load.

2.9.4 Bi-casting

One way of extending the FMIPv6 protocol involves a method often referred to as Bi-casting or n-casting [124]. The main idea behind it is to simply duplicate all traffic intended for a mobile terminal over a short period of time, with the purpose of transmitting all MN bound data packets not only towards its current location, but also towards one or more other locations where the terminal is likely to move. The result intended by this technique is reducing the number of packets lost during the time of a handover and clearing any hesitations over choosing the optimal moment of initiating the tunnelling of data packets between the PAR and the NAR in the FMIPv6 protocol.

An extra advantage comes from the fact that bi-casting data flows is one good solution of the *ping pong* effect (successive handovers among the same two IPv6 subnets). The bi-casting method can also be used in the case of the traditional MIPv6 protocol or the HMIPv6 protocol. The procedures being almost identical in these various cases (with the only difference being the entity that is actually doing the bi - casting: e.g.,

access router, home agent or MAP). For the sake of simplicity we will mainly focus on the traditional MIPv6 protocol.

In order to use bi-casting, a mobile node must be able to simultaneously manage multiple care-of addresses. On the other side, the home agent must also be able to associate more than a single care-of address per terminal, which actually amounts to being able to support multiple binding-cache entries for the same node. The bi-casting method does not require new messages, all information that needs to be exchanged between a mobile and the bicasting entity is piggybacked in existing MIPv6 messages.

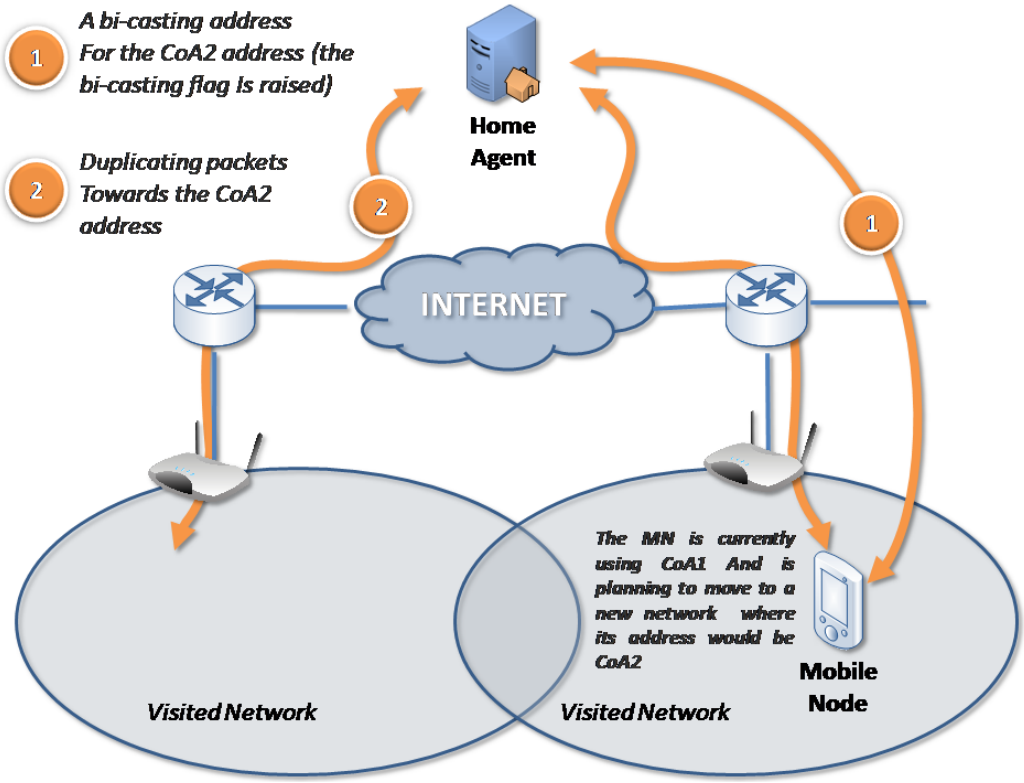


Figure 2.22: Bi-casting

When node is ready to start the bi-casting procedure (e.g., in a case) of pending handover, all it has to do is send to its home agent a Binding Update containing the new CoA address that it is going to use in addition to the one it currently has. Setting specific bit in the BU to 1, allows the mobile to indicate to its home agent, that the CoA transported in the BU is a secondary, additional address, and that the home agent should not remove its existing association.

When it receives a message with the bi-casting bit set to 1, the home agent creates a new entry in its binding cache and it initiates the transmission of data packets towards all the care-of addresses known for this particular terminal. In case it has active sessions with correspondent nodes that also support the MIPv6 protocol, the mobile node can send bi-casting BUs to them too. If this is not the case, or the correspondent nodes simply do not support multiple associations, they will simply ignore the bi-casting bit and proceed as with a normal handover.

The amount of time that the home agent, and those of the correspondents that support bi-casting, should keep the multiple associations active is indicated in the BU that initiates the whole bi-casting procedure. After the expiration of the bi-casting timer, the home agent would update its binding cache and replace all earlier CoA bindings with the new ones. Figure 2.22 illustrates the whole bi-casting procedure.

It is important to note that having the home agent do all the bi-casting for the terminal is a solution that lacks scalability. In this situation all traffic bound for the mobile terminal is duplicated on all the way from the home agent to the terminal while in most cases all data packets would generally follow the same route almost all the way to the MN except from the last several hops. This generates a significant amount of extra network traffic. It is therefore reasonable to only deploy bi-casting together with protocols such as FMIPv6 and HMIPv6 where all packets could be duplicated by either a PAR or a MAP.

2.10 Conclusion

High popularity of wireless devices and rapid deployment of Wi-Fi networks have raised user expectations for uninterrupted connectivity. Members of the research community have been working for several years on different mechanisms that would allow for transparent mobility and enable users to roam through wireless networks without interrupting their communication sessions or experiencing lower quality.

In this chapter we have presented the mechanisms that allow network protocols on the different OSI model layers to handle such mobility and we have also pointed out some of their shortcomings:

- The IEEE 802.11 standard defines basic mechanisms that enable mobile Wi-Fi terminals to reestablish layer 2 connectivity when moving from one access point to another. However, the scanning phase of the layer 2 handover, as defined in the

IEEE 802.11 standard, may be excessively long and cause interruptions of more than several seconds.

- On the network layer, the support for transparent layer 3 mobility in IPv6 networks has been addressed by the design and implementations of a new protocol. Among the various solutions proposed for this problem it now seems that the MIPv6 protocol has been widely accepted as the de-facto standard for handling network mobility. Yet, some of the mechanisms associated with this protocol are still causing lengthy connection disruption intervals during the layer 3 handovers.

- Finally, we have also described solutions for handling mobility at the application layer using the SIP protocol. Such solutions have the advantage of being very easily deployed in existing network configurations since they do not require modification of lower layer protocol entities or OS components. However, being situated at the topmost OSI stack layer, such applications are subject to movement detection delays accumulated by the lower layers

In the following chapters of this thesis we will present our work on evaluating and optimizing mobility on each of these layers. We start by describing a way of using link layer triggers with SIP mobile applications in chapter 3. In chapter 4 we go through an extensive experimental evaluation of one of the most popular mobility solutions - the Fast handovers for Mobile IPv6 protocol, and then in chapter 5 we describe a set of mobility optimizations using various techniques such as the use of multiple wireless interfaces and an FMIPv6 implementation aided by geographic positioning information.

Chapter 3

Optimizing VoIP mobility at the application layer

As we have mentioned earlier (Section 2.4), one of the ways to manage mobility is to handle it in the applications themselves. There are numerous advantages to this approach, such as the fact that application layer mobility allows avoiding the encapsulation and the triangulation typical for Mobile IP.

Handling mobility with application layer protocols such as SIP is an easy to setup and deploy solution that does not require the installation of any new network layer entities such as the Mobile IPv6 Home Agent for example. It is therefore possible to use it even in cases where one does not have control over the network infrastructure, which is the case of most home and small office users. In this chapter we present an evaluation of one such solution. We also describe the various optimizations that we have experimented with, such as using Cross Layer Triggers (CLT), and removing DAD.

Acknowledgement: Work described in this chaper has been funded by a research contract with Orange Labs, France.

3.1 Analytical evaluation of application layer handovers

The handover process over WLAN, when controlled by an application layer entity, consists of 3 major phases : Layer 2 handover, Layer 3 address configuration and Application Layer Handover which gives us a total handover time of :

$$T = T(L2) + T(L3) + T(APP) \tag{3.1}$$

1. **T(L2) - Layer 2 handover** - synchronization, authentication, and association. We will be calling the time necessary for a Layer 2 handover - T(L2) 160ms [137].

2. **T(L3) - Layer 3 address construction** - Router Solicitation, and Router Advertisement. The time needed by Layer 3 to construct a new network address is referred to as T(L3). T(L3)= T(movement detection) +T(DAD). Assuming that the movement detection process is terminated by the reception of an RA we have T(movement detection) 750ms [192], and T(DAD) 1500ms [192, 141]

3. **T(APP) - Application Layer handover** - Session reinitialization, referred to as T(APP) consists in the time that it takes the application layer to detect the address change, send the SIP re-INVITE request, and re-establish the media flow towards the new location of the MN. In our testbed this comes down to the following T(APP) T(app layer movement detection) + 50 ms

In this thesis we focus on handover optimization by bringing T(L3) and T(APP) to a minimum. T(L3) depends mainly on the delay accumulated before receiving an RA. To bring that to a minimum an MN may send a Router Solicitation (RS) or a router may send RAs at a small interval. Both have their inconveniences: [141] specifies that responses to RSs must be randomly delayed by 0-500 ms and having dense RAs incurs extra traffic of approximately 14 kbps when sending RA frames at their minimum size of 88 bytes. On some networks, consumption of such bandwidth may be undesirable. T(APP) may vary according to implementations. The only way for an application entity to get notified for a mobility event is to cache a node's network address and perform periodic verifications for a change. The exact interval at which these verifications would occur is quite delicate to determine since big values bring to a delay in detecting node movement and short intervals may take significant resources on the host machine.

In our solution a mobile node would send an RS right after moving to a new subnet. This would cause a nearby router to respond with an RA (after introducing an average delay of 250 ms) and speed up address construction by Layer 3. Once the MN receives the RA, it does not perform DAD in order to avoid the significant delay it implies, as explained in section 1.2. Right after that the application entity, controlling handover and mobility, would send its session re-initialization message to a CN.

Table 3.1 shows estimated time values needed by each layer to complete the handover process.

Table 3.1: Handover duration by layers

	SIP	SIP+DenseRA	SIP+CLT r
L2 Handover - T(L2)	160 ms	160 ms	160 ms
L3 Autoconf - T(L3)	1900 ms	50 ms	260 ms
SIP Handover	50 ms + detection	50 ms + detection	50 ms

3.1.1 Description of the optimization

SIP is an application layer protocol and as such standard SIP entities could only rely on indirect mobility notification such as constantly scanning local addresses for a change. A mobility event would reach a SIP application only after all procedures described in section 1 have completed and, assuming that there are no lower layer mobility handlers (such as MIPv6 [103]), it is obliged to acquiesce to delays imposed by lower layers (e.g. waiting for an RA).

On the link layer level, though, there is an underlying awareness of connection events, which one might convey to a SIP application. If this information is available, it is always available faster than multicast-ed RAs and hence address changes.

Link-Layer movement, however doesn't necessarily mean that subnet and network address have changed so an RA must be solicited to confirm movement.

In our solution we introduce a Cross Layer Module (CLM), figure 3.1, that interacts with link, network, and application layer entities. Many network cards support notification model that enables communication with user-space processes. The CLM listens for event notifications and would send an RS every time an AP change is reported (i.e. right after the reception of an Association Response). The corresponding RA response is received and processed by the Network Layer. The network address is thus updated. The CLM on its turn uses the RA as a trigger to send a notification to a pre-registered application layer SIP entity and it is able to send its session re-initialization message (re-INVITE).

The implementation of CLM does not require any changes in the network configuration and will behave the same way over networks with dense RAs and standard RFC2461 [141] compliant networks (i.e. with standard density of RA emissions).

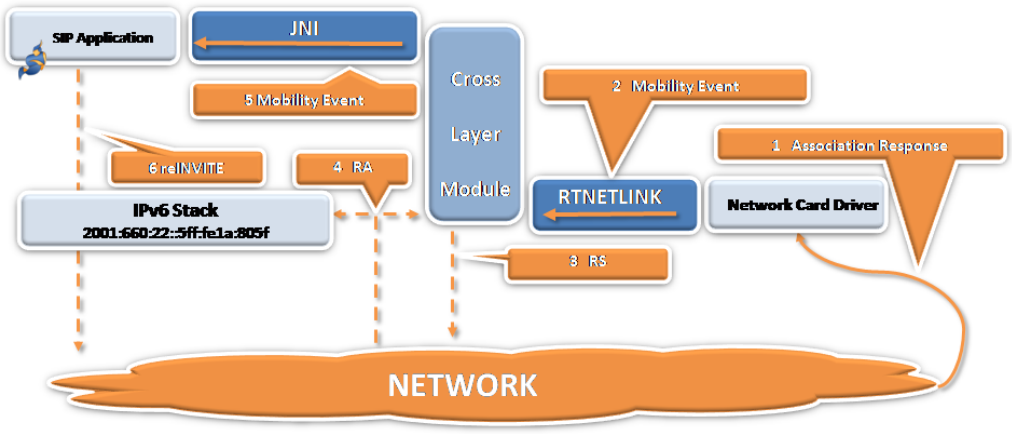


Figure 3.1: Cross Layer Module Architecture

3.2 Implementation and testing

3.2.1 Testbed

The testbed used for testing the proposed optimization consists of three IPv6 access networks. Two of these are equipped with an 802.11b Wireless LAN access point. All APs are Cisco Aironet 350 series. Handovers occur back and forth the two IPv6 networks equipped with an AP. Network topology is shown on figure 3.2.

MNs are Linux equipped terminals with modified IPv6 stacks so that no DAD is performed, as proposed by [140].

We implemented the CLM as described in the previous section. We also modified a SIP client called the SIP Communicator [15] so that it would handle incoming reINVITEs and interact with the CLM (i.e. send a re-INVITE upon CLM mobility notification).

3.2.2 Experimental evaluation results

We have measured the handoff delay with SIP terminal mobility in our IPv6 testbed. Three different scenarios have been considered: (a) SIP mobility over a network with standard RA density (400 s) and with DAD; (b) SIP mobility over a network with dense

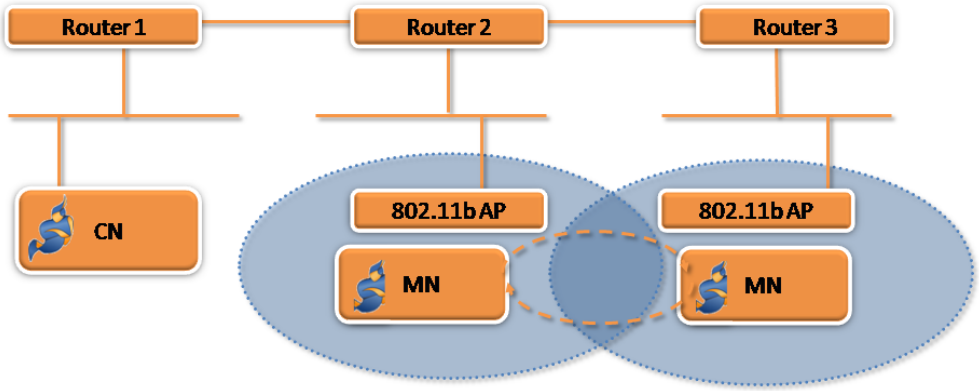


Figure 3.2: Testbed network configuration

Table 3.2: Mean values for results from experiments with cross layer triggers

Case	L2 start	L2 end	Address Autoconf	reINVITE	OK	Data
(a)	0	0,165	407,54	409,149	409,192	409,2
(b)	0	0,165	0,755	2,149	2,199	2,247
(c)	0	0,168	0,666	0,699	0,756	0,797
(d)	0	0,167	0,313	0,346	0,395	0,454

RAs (750 ms) and with DAD; (c) Standard SIP mobility over a network with dense RAs (750 ms) and without DAD; (d) SIP mobility of a node using Cross Layer Triggers (CLT). The table shows the handoff delay for each of these configurations. Values shown in the table are the average result of ten successive handovers for each configuration.

Table 3.2 shows the signalling delay measured from the first Probe Request sent by the Mobile Node after entering the new BSS until the first UDP data packet received by the MN on its new location. As we can see, Layer 2 trigger usage combined with kernel modifications (removed DAD) have reduced handoff delay by 1500 ms to over 400 s for some cases. The same tests are illustrated on figure 3.3.

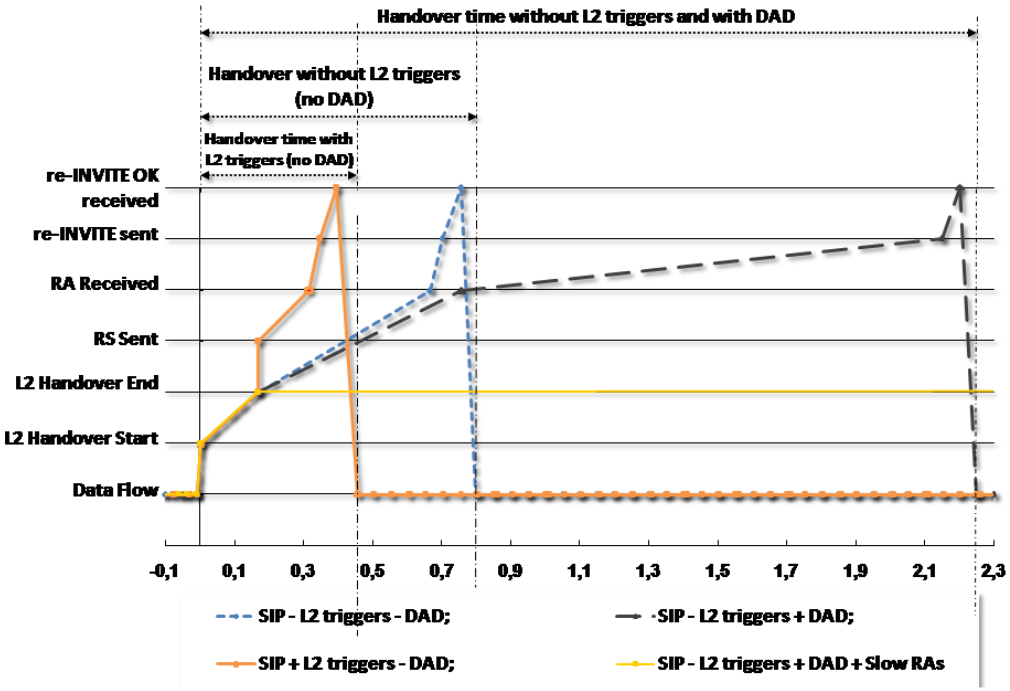


Figure 3.3: Cross layer triggers experimentation results

3.3 Conclusion

In this chapter, we have described an optimization of the handover process, its analytical evaluation, and sample results for SIP mobility in an IPv6 laboratory testbed. In our performance study we concentrate on optimizing the movement detection part of the handover process. We observe that using Cross Layer Triggers (CLT) and removing DAD greatly improve SIP mobility. Results shown here reflect handoff delays for standard network configurations, networks with dense RA emissions, and MNs with and without kernel modifications and CLT usage.

It is however still clear that mobility solutions implemented solely at the application layer could not serve the purposes of a long-term deployment. Despite the fact that even though a handover latency of 450ms is impressive from a Layer 7 perspective, it is far from enough when trying to obtain completely seamless mobility. In addition to this, SIP mobility would only give the possibility to roam uninterrupted, to applications that explicitly implement it.

For all of the above reasons we have decided to direct our next efforts to the lower layers and to the optimization of Mobile IPv6 based mobility.

Chapter 4

Evaluating a generic layer 3 solution

As we have mentioned in Section 2 we consider the FMIPv6 protocol as one of the most promising mobility solutions. As a result a considerable amount of time of this thesis has been spent on the implementation, evaluation and improvement of this protocol. The Louis Pasteur University in Strasbourg, has produced one of its first implementations, which was, by the way, the first to be distributed under an open source license as fmipv6.org FMIPv6 implementation [66].

In this chapter we go through an experimental evaluation of some of the most important scenarios and use cases of the FMIPv6 protocol. The purpose of this evaluation has mainly been to understand the strengths and weaknesses of this protocols, and see which of its features, if any, would need further work and optimization. *Acknowledgement: Work described in this chaper has been funded by a research contract with Orange Labs, France.*

4.1 Previous evaluations of the FMIPv6 protocol

To this date, numerous studies exist on the behavior of FMIPv6. [111] is one of the early papers available on the protocol. Authors provide an evaluation based upon a proprietary experimental implementation and also propose a context transfer scheme (the latter being outside the scope of our current study). The paper was published a relatively long time ago (2001 - approx. 5 years) when still little was known about FMIPv6 and there were many uncertainties concerning its performance. It was therefore of considerable use and had a significant impact on following work in the IETF. Yet we find the experimentation setup somewhat unrealistic and results to be incomplete from

a today's point of view. Handovers, for example are triggerred upon reception of a user issued command, and do not make use of L2 triggering or any other scheme. More importantly, experimentation results only include the length of the handover procedure but not the handover latency (the amount of time that connection was unavailable and packets were being dropped by the AR). Last but not least, there is no consideration of candidate AP discovery which we believe to be critical in the current state of the protocol (we talk about this in section 4.3.5).

In [80] authors provide an analytical evaluation and comparison of the FMIPv6, HMIPv6 [184], and MIPv6 protocols, based on default values provided by the respective RFCs and drafts. Given the analytical approach however, many of the practical aspects that we already mentioned (e.g. scanning, packet loss and etc) have been neglected or not fully taken into account.

Authors of [84] analyze a combination of FMIPv6 and HMIPv6 and present an evaluation with NS2 [14] in the case of a TCP flow. Both HMIPv6 and FMIPv6 are reported to reduce MIPv6 handover latency 7 and 15 times respectively. The paper also uncovers a problem with the superposition of the two handover optimizations that consists in packet disordering due to ARs being often closer to the HMIPv6 MAP than to each other. This problem seems to be tampering with TCP flows and thus hindering the maximum potential handover performance. We believe that the document is quite useful and provides valuable insight on properly combining both optimizations. Yet it doesn't seem to have the objective of providing detailed and realistic protocol feedback. Values for L2 handover delay for example are fixed to 20 ms (which might correspond to a certain link layer technology but this could hardly be the case of IEEE 802.11).

Another performance analysis of FMIPv6 is provided in [149]. Authors focus on protocol overhead, wrongful anticipation, and "eating up buffer space" in access routers, and study how these problems relate to the "sensitivity" of L2 Triggers. They show that these vary largely depending on how close in time is the link layer trigger event to the actual connection disruption. They also give an optimal value for the time distance between this event and the disconnection (i.e. how long before the MN loses connection should it start its handover) and prove it analytically. We find, however, that this research is of little use for implementors and protocol designer since the exact moment in time that connection disruption happens is rarely (if ever) known in advance. We also believe that authors slightly overstate the problem with the protocol overhead since it is negligible compared to the data flows typical for VoIP, that are often used as a reference for handover performance evaluations.

Kempf, Wood and Fu provide in [108] yet another evaluation based upon experimentation with a proprietary implementation. Experiments make use of a wired handover emulator configured for 40ms link layer handover length (i.e. connection is unavailable

for 40 ms) and 10ms of packet delaying. The values were reportedly chosen to match those of 3G systems. Yet we find that this emulator adds a level of uncertainty and though useful to some extent from an analytical point of view - it (like all simulations and emulations) introduces some doubts as to the validity of the experimentation in real world deployments and usage of the protocol over wireless networks. Link layer triggers for example are (once again) configured to be sent at a predefined amount of time before the link layer handover is to occur. Access routers receive an instant link down trigger for mobile nodes belonging to their network which is not quite the case in reality and especially not for IEEE 802.11 that have not been taken into consideration in this work.

4.2 Testbed and Test Scenarios

The testbed that we have built and used both for the development of the implementation and for our experiments consists of three access routers a mobile node and a correspondent node (Figure 4.1). Two of the access routers, FMIP-AR1 and FMIP-AR2, both had a commercial AP connected on one of their wired interfaces. During the tests the mobile node FMIP-MN switched back and forth and performed handovers between those two routers. A third router FMIPNET was both serving the role of a home agent and interconnecting the rest of the testbed with the Internet. A correspondent FMIP-CN node, connected to the internet from a completely different IPv6 subnet in our campus, was used for generating flows destined to the FMIP-MN.

All three routers were running the Linux Operating System with the USAGI modified 2.6.8.1 kernel and the Quagga Routing Suite[154] (version 0.98.4). The routing protocol used both inside the testbed and on the interconnection link was RIPv3. FMIPNET was using the MIPL 2.0rc2 [9] Home Agent implementation from the Helsinki University of Technology.

FMIP-AR1 and FMIP-AR2 were also running the fmipv6-ar router implementation from fmipv6.org [66].

FMIP-MN was running the Linux Operating System with the USAGI modified 2.6.8.1 kernel and was equipped with a single wireless interface using an Atheros chipset and the MADWiFi [44] driver. The MADWiFi version running on the MN had minor modifications that optimized its behavior during handovers controlled by userland applications (i.e. the fmipv6 daemon) and thus allowed it to perform rapid L2 handovers (note that these modifications do not in the least deviate from standard IEEE 802.11

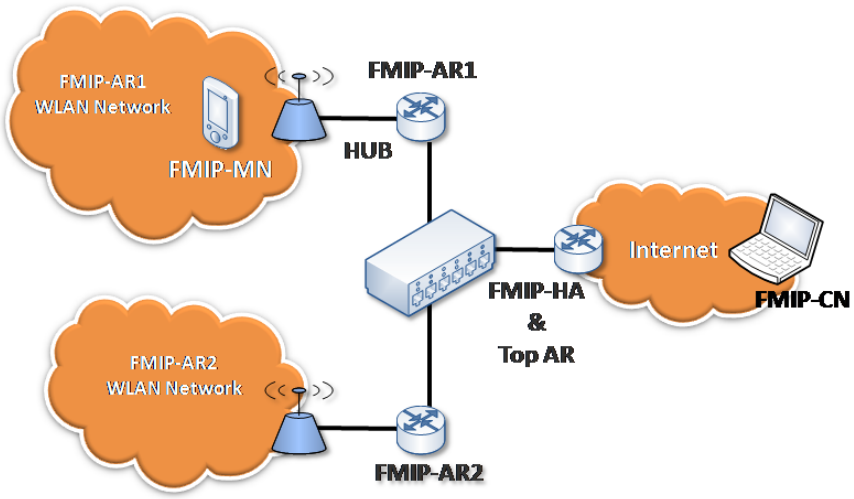


Figure 4.1: The fmipv6.org Experimental Testbed

practices). FMIP-MN was also running the fmipv6-mn mobile node implementation from fmipv6.org [66].

The reason for choosing this exact configuration is because we believe it's widely spread and Wireless LAN deployments often include one or more of its components. We therefore believe that results obtained in this experimentation set are of high interest and should be considered when implementing and/or improving the protocol.

Fig 4.1 provides a diagram describing the testbed and the way it's deployed.

For all predictive testing, the mobile node (FMIP-MN) was beginning the experiment attached to one of the FMIPv6 routers (FMIP-AR1 or FMIP-AR2). The Tx Power on that router was then manually lowered through a Wireless Extensions [193] ioctl call. The fmipv6.org MN implementation uses link layer information provided by the MADWiFi driver through the Wireless Extensions package. Whenever quality dropped below a configurable threshold (which is what happened upon modification of transmis-

sion power on the AP) the implementation performed a handover and switched to the other router.

Reactive handovers were tested by turning down the wireless interface of the router, that FMIP-MN was currently attached to and thus forcing it to associate with the other one.

We are aware that these scenarios do not represent all possible FMIPv6 use cases but we believe they are covering and put into use most key parts of the protocol semantics.

4.3 FMIPv6 evaluation results and analysis

4.3.1 Predictive Handovers

The predictive mode of operation of the FMIPv6 protocol is the one that best addresses handover issues and allows bringing connection disruption time and packet loss to levels that would satisfy most existing real time applications.

Figure 4.2 contains results from a (representative) experiment with one such predictive handover. Exactly 12.3 seconds after the beginning of the experiment the mobile node, by regularly scanning link layer quality, detects that it has crossed the predictive handover threshold and decides to begin a handover. It therefore sends the Fast Binding Update that we see as the first FMIPv6 message on Figure 4.2. In the experiment at hand, the FBU does not get immediately answered and the corresponding FBACK is only received after a retransmission of the FBU. This actually happens quite often since before sending an FBACK, the PAR needs to send a Handover Initiate message to the NAR, wait for a Handover Acknowledge and (in the case of fmipv6.org) create the forwarding tunnel so that it could be already operational once the MN has moved to NAR's network. Right after receiving the FBACK (at time 12.294), the MN starts an L2 handover which lasts for about 10 ms. After it arrives on the new link (at approximately 12.312s) it announces its arrival by broadcasting an ICMPv6 Neighbor Advertisement and sending a Fast Neighbor Advertisement to NAR. The FNA also allows routers to stop buffering packets and forward those that have been received through the tunnel prior to MN's arrival. In this case there is one such packet and it is the one immediately following the last packet received by the MN on PAR's link.

As shown on Figure 4.2 the MIPv6 implementation on the MN (MIPL v2.0 in our case [9]) sends a Binding Update upon detection of MN's new address, modifies its tunnel to match NAR's link and thus ends the usage of the FMIPv6 tunnel. In this

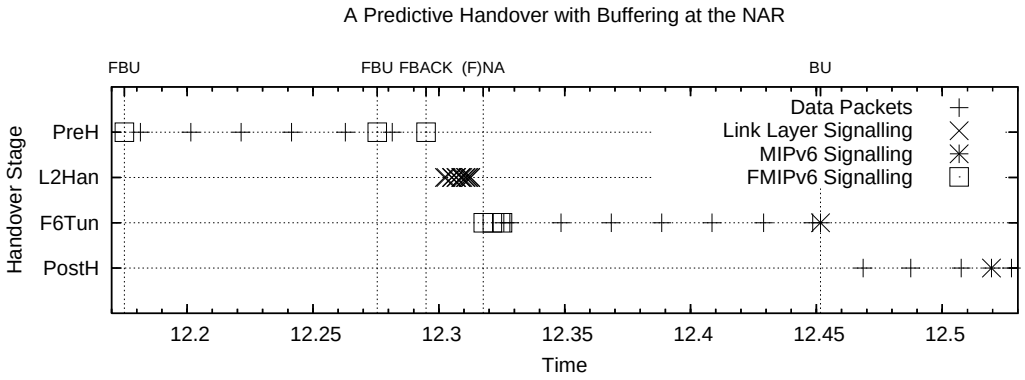


Figure 4.2: Packet loss and handover latency for a predictive handover

experiment that has happened 133 ms after the FNA has been sent. The reason for this delay comes from the fact that in order to confirm movement, MIPL had to send a Router Solicition and wait for the corresponding advertisement, which is by the way purposefully delayed by the NAR (see [141]) before sending the BU. The advantage that FMIPv6 offers in this case is the fact that an FMIPv6 implementation "knows" exactly what handover has taken place since it is the entity that has caused it and that controls it.

The time that the MN has suffered connection loss is equal to 10.42ms and there has been no packet loss as the one packet that arrived while the MN was out of reach, was buffered and later resent by the NAR.

4.3.2 Buffering issues

An interesting phenomenon that we have observed during our predictive experiments is the fact that L2 triggered events may sometimes be delayed significantly and thus have an impact on protocol performance. The standard mechanism used for L2 trigger event delivery to userland in Linux based Operating Systems is through RTNETLINK sockets. When and under what conditions these are sent is currently a matter of driver behavior and though there is intensive work on standardizing them this is not currently the case. When we were using a Wireless LAN card managed by the HostAP [83] driver for example, the events received indicating L2 handover end were received more than 100ms later than the MN had actually associated with the new AP. In such cases the MN's (Fast) Neighbor Advertisement on NAR's link suffered significant delay. And in the cases where NAR was providing an insufficient amount of buffering (such as

neighbor discovery's default 3 packets for example [141]), packets were being dropped while they could have been received by the MN, had they been sent.

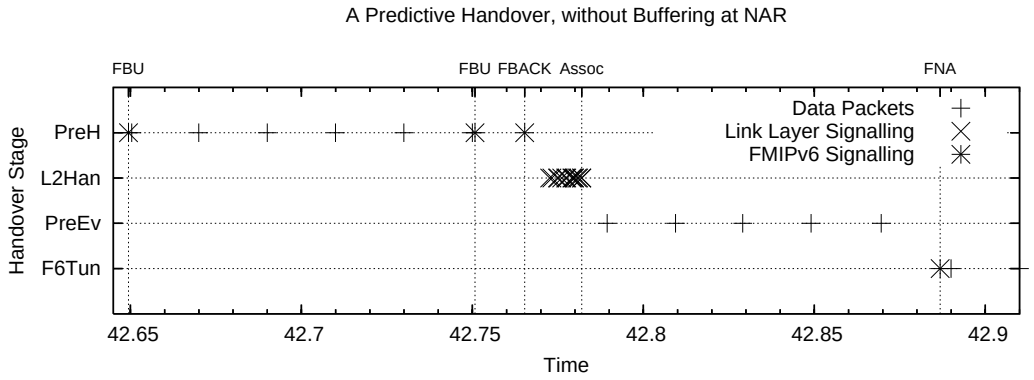


Figure 4.3: Packet loss and HO latency without buffering at NAR

We therefore conducted a set of experiments with the host AP driver on the MN side and without any buffering being done by the NAR. One such experiment, represented on Figure 4.3, shows that the MN receives 5 packets before it gets notified of L2 attachment and loses 1 because of the lack of buffering on the NAR. Using buffering would have saved that one packet (provided the configuration at NAR had allowed it to store a sufficiently large number of packets) but would have delayed the other 5 by more that 110ms. It is difficult to say whether it would have been better for the MN to receive all packets saved by the buffering but delayed by the L2 trigger delay or rather get them immediately but lose one. We believe that the MN alone could determine whether short handover latency or low packet loss is more critical to it as that depends on the types of applications being run on the mobile node. Yet the FMIPv6 protocol does not provide MNs with a way to indicate their preference to routers and this is one of the issues that we'd like to resolve in future work.

4.3.3 Reactive handovers

On figure 4.4 we see results from a reactive handover scenario. At a certain point of the experiment we turned down the accesspoint that FMIP-MN was associated with. The MADWiFi driver detects link loss through missed beacons. By default the number of beacons that need to be lost on a link for MADWiFi to declare it down is 7 which makes for at least a 700ms delay. We modified that number to 3 in order to achieve better performance for reactive handovers. Do notice that losing link layer connectivity generate a completely different event from the one caused by a drop in signal strength. Signal

strength is measured upon received packets and if there is no AP to send packets, no signal strength would be measured and no link quality event generated. Thus there is no risk of wrongfully beginning a predictive handover and waiting for the FBU transaction to expire (700 ms with default values from [70]) before initiating a reactive handover and thus losing time on a dead link.

2.289s after the beginning of the experiment or 375 ms after the last data packet received on the link the fmipv6.org daemon on the MN detected that the link was down and started an L2 handover to the access point that seemed to offer best quality during its last scan (results being stored by the FMIPv6 daemon itself). Right after it arrived on the new link the MN sent an FNA, to the NAR, followed by an FBU for the PAR.

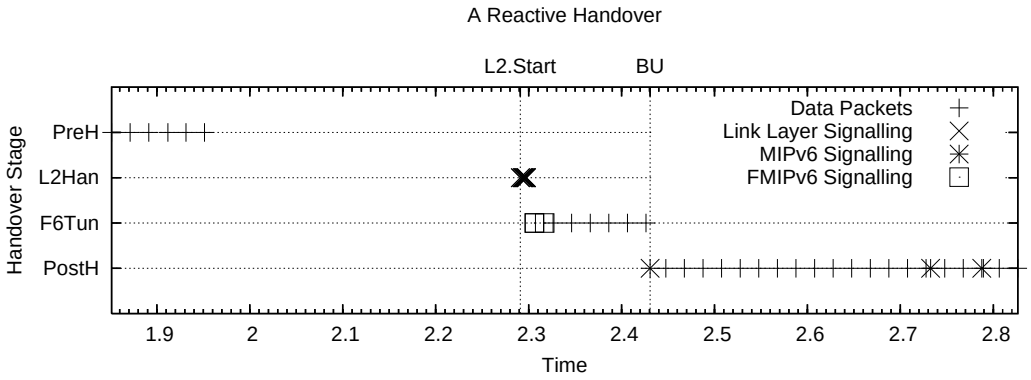


Figure 4.4: Packet loss and handover latency during a reactive handover.

Note that this is not really the standard protocol behavior since the FMIPv6 [70] protocol advises that FBUs in reactive handovers SHOULD be encapsulated in the FNA (as opposed to sending them separately). We didn't take this approach as it was causing problems with the Mobile IPv6 stack on the router which ignored mobility header packets with the "next header" value different from NONE. From that point on, the handover continued in the manner seen in previous sections - PAR started tunnelling packets destined to MN's PCoA to its NCoA and once MIPL detected the change as well, it modified the MIPv6 tunnel to point to NCoA. The handover has thus caused us a connection loss during approximately (less than) 343.53ms and caused us to lose 17 data packets.

4.3.4 Candidate Access Point Discovery

Figure 4.5 shows what we believe to be one of the currently most flagrant operational issues with the protocol. FMIPv6 does not provide a way for mobile nodes to discover

Candidate (neighbor) Access Points. Sending a wild card RtSolPr to PAR requests a list of Access Point link layer addresses known to the AR but it would by no means tell the MN whether these APs are within its coverage or either of their link characteristics such as channel/frequency, ESSID and others. A mobile node has therefore no other choice but to perform a wireless scan and only then send an RtSolPr demanding information on the discovered packets. The length of the scanning procedure may vary from one implementation to the other but is generally considered to be the heaviest part of a Wireless LAN handover. In our testbed we performed tests with two kinds of WLAN cards - one that supported IEEE 802.11a/b/g standards and one that only worked for IEEE 802.11b/g. Those supporting only the "a" and "b" standards were by default performing the complete scan procedure over the 13 standard channels for an average of 2670ms. The cards that also supported the "a" standard in addition to the "b" and "g" required more time in order to "sweep" through 802.11a's frequencies which made for an average of 15266ms. After modifying the MADWiFi variant of MaxChannelTime (called by the implementors as "dwelltime") from its default MADWiFi value of 200ms to the minimum allowed 100ms, times for b/g and a/b/g cards dropped to 1368ms and 4728ms respectively.

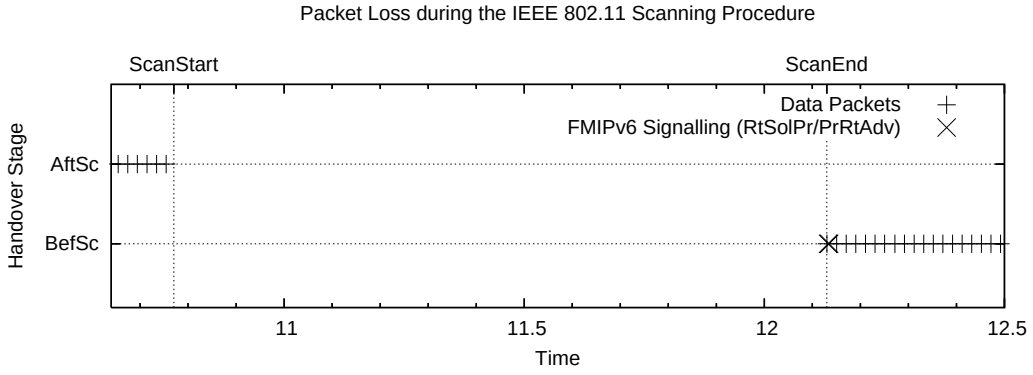


Figure 4.5: Packet loss and latency during scanning for candidate APs.

Figure 4.5 represents the impact that the shortest of the above times has had on packet loss. In that particular example 69 packets have been lost during a frequency scanning, that took 1368ms.

Note that performing this scan at an arbitrary point of FMIPv6's execution not only has a significant impact on ongoing communications but is by no means a guarantee that at the time the MN initiates a predictive or reactive handover, the set of candidate APs discovered during the last scanning procedure is still valid.

4.3.5 Results summary

Table 4.1 contains a summary of the results exposed in the previous subsections. One could easily see that predictive handovers are making for great handover performance by bringing packet loss and latency to a minimum. We do not provide a separate comparative "bare MIPv6" set since the purpose of this document is not to show the advantages of the FMIPv6 protocol over standard MIPv6. This has been already done far more than once in documents referenced in the 4.1 section. Furthermore, we believe that such comparisons are somewhat "unfair" as MIPv6 implementations do not generally provide any management of the handover destination selection process, the reason for that being the fact that MIPL's primary goal is to provide transparent mobility handling (i.e. avoid addressing and routing problems), and not protocol deficiencies. FMIPv6 on the other hand was designed to do just that. We have, however, included MIPv6 signalling so that the reader could could see the way both protocols interact.

Table 4.1 shows a summary of the losses during the different experiments and parts of FMIPv6 operation. The problem with candidate access point discovery is obvious - and losses in packet delivery caused by that problem greatly overpower whatever gains have been accomplished during the handover itself.

Table 4.1: Results from different handover scenarios

Test Scenario	Conn LossTime	Pack Loss	Pack Buff	FMIP Tunnelled	Mvmt Detect
Predictive	10.42ms	0	1	8	21.07ms
Buffer Issues	9.35ms	1	0	5	104.80ms
Reactive	343.53ms	17	0	0	332.21ms
Scanning	1368.11ms	69	NA	NA	NA

4.4 Drawing some conclusions from this evaluation

After studying the results shown in this chapter, we were quite convinced, that FMIPv6 has the potential of bringing considerable optimizations upon the use of Mobile IPv6 for packet loss sensitive applications such as VoIP. According to experimentation conducted in our testbed the protocol considerably reduces (and in some cases even eliminates) packet loss (during handover) and handover latency to a level acceptable for real-time communications.

Yet there are obviously still issues not addressed by the protocol that remain quite disturbing for media streaming, namely - candidate Access Point discovery. We have shown that the impact of a wireless scan, conducted for lack of alternative ways to discover neighboring APs, causes severe packet loss and lengthy connection disruption.

Another (rather minor, but still an) issue that we have demonstrated is the lack of a way for the mobile node to specify whether it wishes a New Access Router to buffer packets before the MN has had the chance of announcing itself on the new link.

Through the following chapters we have therefore presented work that we have completed on the subject, trying to resolve exactly that candidate Access Point discovery problem and that includes elaborating alternative, non-interrupting mechanisms. One possible way of doing so would be rendering possible parallel scanning and communication. This could be achieved through either the use of a secondary wireless interface, rapid alternation of frequencies on a single interface through temporal division of the medium access, or using multiple antennas on the wireless device.

We have also studied possible extensions of the protocol with new IEEE 802.11 specific options that allow access routers to send to mobile nodes, all details that they might need for rapidly associating with a new Access Point, such as frequency, ESSID, and authentication info.

Chapter 5

Optimizing IPv6 mobility

In this chapter we describe our work on optimizing transparent mobility with Mobile IPv6. In the following sections we would present a possible way of improving MIPv6 seamless mobility and two different approaches for extending the FMIPv6 protocol in a way that would resolve the problems related to the candidate access point discovery phase that we discussed in the previous Chapter 4.

In Section 5.1 we will be presenting a rather generic MIPv6 improvement (i.e., not related to one of its extensions) that allows IEEE WLAN 802.11 nodes to accomplish a softhandover (i.e., move from one IPv6 subnet to another without losing connectivity at all) through the use of multiple wireless interfaces, or in other words.

Next, in Section 5.2 we are using a similar technique to resolve the problems of the Fast Handovers for Mobile IPv6 protocol, that we discussed in Chapter 4 and that are related to the discovery of candidate access points.

Finally, in Section 5.3, we will be addressing the same FMIPv6 issue (i.e., access point discovery) only this time we present an alternative solution where the discovery procedure is aided by geographic positioning systems.

Acknowledgement: Work described in this chaper has been funded by a research contract with Orange Labs, France.

5.1 Multiple interfaces for IEEE 802.11 nodes

A major issue with IEEE 802.11 mobility is the fact that WLAN devices are not able to associate with more than one AP at a given moment. Most existing handover optimizations are therefore based on the assumption that the handover process is inherently accompanied by connection loss. In other words it is always considered that at a given moment a MN is bound to lose its network connectivity and would therefore become unreachable.

Soft Handover is the process where a MN will first connect at its new point of network attachment and will only then drop its previous link. In order to achieve this in an 802.11b environment we give MNs a second wireless interface. Furthermore, in order to eliminate power consumptions concerns, we try to avoid having both network interfaces functioning together for more than a couple of seconds (i.e. only during the handover process).

Compared to other mobility optimizations, the multi interfaces solution has the advantage of keeping to a minimum changes to existing protocols such as Mobile IPv6 and 802.11b and does not require changes in existing standard Mobile IPv6 network topologies (i.e. it does not define any new network entities).

During the rest of the document we will be using the following terminology and abbreviations:

- **Double Interface Mobile Node (*DIMoN*)** - A mobile node equipped with two (802.11b) network interfaces.
- **Access Router (AR)** - A *DIMoN*'s default router
- **Previous Access Router (PAR)** - The *DIMoN*'s default router prior to and during its handover
- **New Access Router (NAR)** The *DIMoN*'s default router during and subsequent to its handover
- **Bicasting** The splitting of a stream of packets destined for a MN into two or more streams, and the simultaneous transmission of the streams to PAR and one or more NARs. We use this technique to reduce packet loss during handover.

While not on the move, or rather during its stay under the coverage area by a single access point a *DIMoN* would behave as a standard mobile node. All application traffic

would go through the currently default network interface. During that period the node's second interface may be in a power save mode or completely turned off.

When a *DIMoN* starts moving the signal that the AP casts upon it would gradually fade away. When signal intensity goes below a certain threshold the *DIMoN* will "wake" its second interface and begin with it the 802.11b synchronization procedure. The procedure is slightly different in that the selection of an Access Point after the scan has been completed would ignore the AP the Old Network Interface is currently attached to. If no other AP has been found during the scanning process the New Network Interface is shut down and the *DIMoN* would behave as a standard MN.

Once synchronized with the newly found AP the *DIMoN* will form its New Care-of Address based on the prefix of the foreign link. In case of using stateless address autoconfiguration [192] it is highly advisable for the *DIMoN* to use at least one of the movement detection optimizations described in section 2.7 above since the time that network connectivity would be available through the Old Network Interface is likely to be relatively short.

Next the *DIMoN* would be supposed to perform Duplicate Address Detection (DAD) [142]. However, given the time that DAD normally takes to complete, it might be a better option to either skip that part or run it in a non-blocking fashion. It is important to notice that up to this point the Old Network Interface has remained operational and network connectivity has been uninterrupted.

Following address configuration, the *DIMoN* informs its home agent of its movement by sending a Binding Update (BU) message. The binding update message contains the mobile node's home address and its New and Previous care-of addresses. The home address is included in the home address option, the Previous Care-of Address is included in the source address field of the IP header and the New Care-of Address is recorded in an alternate-care-of address option. The bicasting flag of this message should be set to 1 as specified by [124]. From this point on, the home agent acts as a bicasting proxy for the *DIMoN*. It receives all packets addressed to the mobile node, and after duplicating them, sends a copy to both of the node's care-of addresses. The HA will then send a Binding Acknowledgement to the mobile node.

Data packets received by the HA from either of the *DIMoN*'s care-of addresses are handled as in standard Mobile IPv6 - the HA decapsulates the packet and forwards the contents to its destination.

During this part of the soft handover process (the time when the HA is doing the bicasting), some packets would be seen twice by the *DIMoN*'s upper layers. This should

not add any complications since packet duplication is expected to happen according to [60].

Once the BA has been received by the *DIMoN* it is up to it to determine when to stop the bicasting. It could do so right after receiving the acknowledgement or when signal intensity from one of the interfaces has gone above a certain threshold and may therefore be considered as reliable. At that stage the node will send a second BU without including the alternate address option so that the HA would stop duplicating packets. The node would then turn off its other interface and the soft handover procedure is considered terminated.

Figure 5.1 illustrates the *DIMoN* behavior that we have just explained.

Compared to other mobility optimizations such as FMIPv6 [70] and HMIPv6 [184] for example, the soft handover solution has the advantage of being extremely simple. It does not introduce any new entities in a network topology and completely reuses existing MIPv6 message semantics, header syntax and options. Furthermore it does not add enormous amounts of extra signalling traffic as in the case of MIPv6. The primary advantage of the soft-handover is that there is no actual disruption and no physical connection loss.

A major concern, often associated with multi interface nodes is elevated power consumption. We try to resolve that by only using both network interfaces for a very short interval - the time of the handover. In the following section we show that a worst case scenario would require both interfaces to stay simultaneously lit up for a period of approximately 2s. According to [115] a wireless device that discharges for 202 minutes without using a network interface, would lose approximately 67 minutes of battery lifetime when constantly transmitting at a maximum rate through its wireless interface. This means that using the wireless interface during one minute would "cost" around 30 seconds of battery lifetime, or in other words, having our second wireless interface lit up during 2 seconds, would shorten discharge time with approximately 1 second. Needless to say this could hardly be a cause for concern.

5.1.1 Analysis and evaluation

When an active *DIMoN* moves to a new IP subnet, it changes its point of attachment to the network through the following soft handover process. First Link-layer handoff occurs to (maybe) change the wireless AP to which the *DIMoN* is associated. According to evaluations such as [136, 137] the time needed for a node to attach to an AP is between 160 and 260 ms. This gives us:

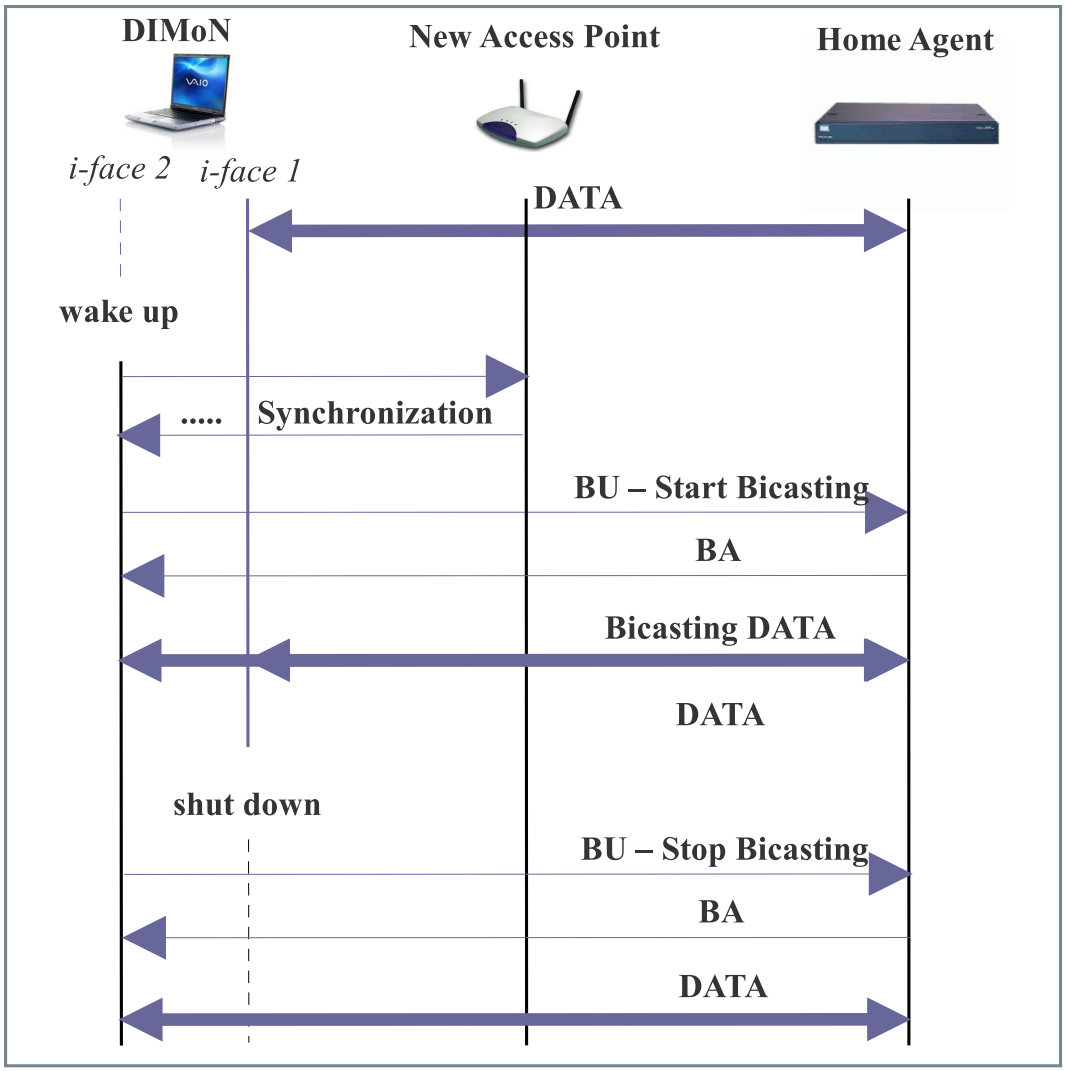


Figure 5.1: Typical behavior of a *DIMoN*

$$T(L2) \approx 200ms.$$

Where T(L2) is the time needed for Link-layer handover mechanisms to complete.

After a new Link-layer connection is established, Networklayer handover is performed, which broadly involves movement detection, IP address configuration and location update. The time this stage takes may vary depending on the movement detection scheme employed by the *DIMoN*. In the case of a New Access Router configured

in accordance with [103], it would take about 50ms (i.e., (MinRtrAdvInterval + MaxRtrAdvInterval)/2) until a RA is received by the node.

$$T(MIPv6MoveDetect) \approx 50ms.$$

If using a Link Layer Triggered RS against a standard RFC2461 [141] router the node would get a RA in about 250ms (the RFC mandates a random delay between 0 and 500 ms when sending RAs in response to a RS). We will label this time as T(LLTriggerMoveDetect).

$$T(LLTriggerMoveDetect) \approx 250ms.$$

Once the *DIMoN* discovers its New AR, it performs DAD. RFC2461 [141] default values stipulate a random delay between 0 and 1000ms before sending an initial NS and an extra delay of 1000ms before confirming address uniqueness. This would add another 1500 ms to the network layer handover.

$$T(DAD) \approx 1500ms.$$

Mobility signalling procedures are then started, with the *DIMoN* sending its first BU to the HA. The time needed for the first data packet to arrive after the BU leaves the mobile node may strongly vary depending on nodes' location. According to our experiments this interval comes close to 60 ms:

$$T(TrafficRedirection) \approx 60ms.$$

Keeping all the above in mind, our best case scenario would be one where no DAD is performed (or it is run in a non blocking manner) and the New Access Router is configured to send RAs at the interval defined in [103] or it implements the FastRA [107].

$$T(BestCase) = T(L2) + T(MIPv6Movement.Detection) + \\ + T(TrafficRedirection) \approx 310ms.$$

The slowest possible setup, where no movement detection optimizations are implemented by the New Access Router and where DAD is completed before using the auto-configured address:

$$T(WorstCase) = T(L2) + T(LLTriggerMovement.Detection) + \\ +T(DAD) + T(TrafficRedirection) \approx 2010ms.$$

Finally, a scenario that we find most realistic is where the *DIMoN* uses a link-up trigger for sending an RS and where DAD is run in parallel of network communication (or not at all). What makes this case interesting is that no special features are required from the network setup.

$$T(Optimal) = (L2) + T(LLTriggerMoveDetect) + \\ +T(TrafficRedirection) \approx 510ms.$$

According to [8] the average speed of a walking adult person may vary between 125cm/s and 150cm/s. The 510 ms from our third (optimal) case would thus require a minimal overlap interval length between $63.9cm$ and $76.9cm$ where network access will be available through both access points. For indoor areas this is close to 2% of a typical Access Point's range [2].

5.1.2 Simulation results

To be able to completely evaluate and experiment with Soft Handovers we have used the SimulX network simulator. SimulX (figure 5.2) was designed and developed by the Network Research Team at the Louis Pasteur University in Strasbourg.

At the time when we were working on the Soft Handover mobility optimization, the SimulX simulator did not support nodes with multiple network interfaces. We have therefore implemented the feature as well as the interface selection policies and the necessary behavior to be able to completely explore its potential.

In this section we present the results of a series of simulations run over the network topology and scenario presented on figure 5.3.

Typically, handover duration is a key characteristic of Mobile IPv6 optimizations since it suggests loss of network connectivity. When performing a Soft Handover a mobile node remains constantly connected to at least one network. Yet it is still advised to

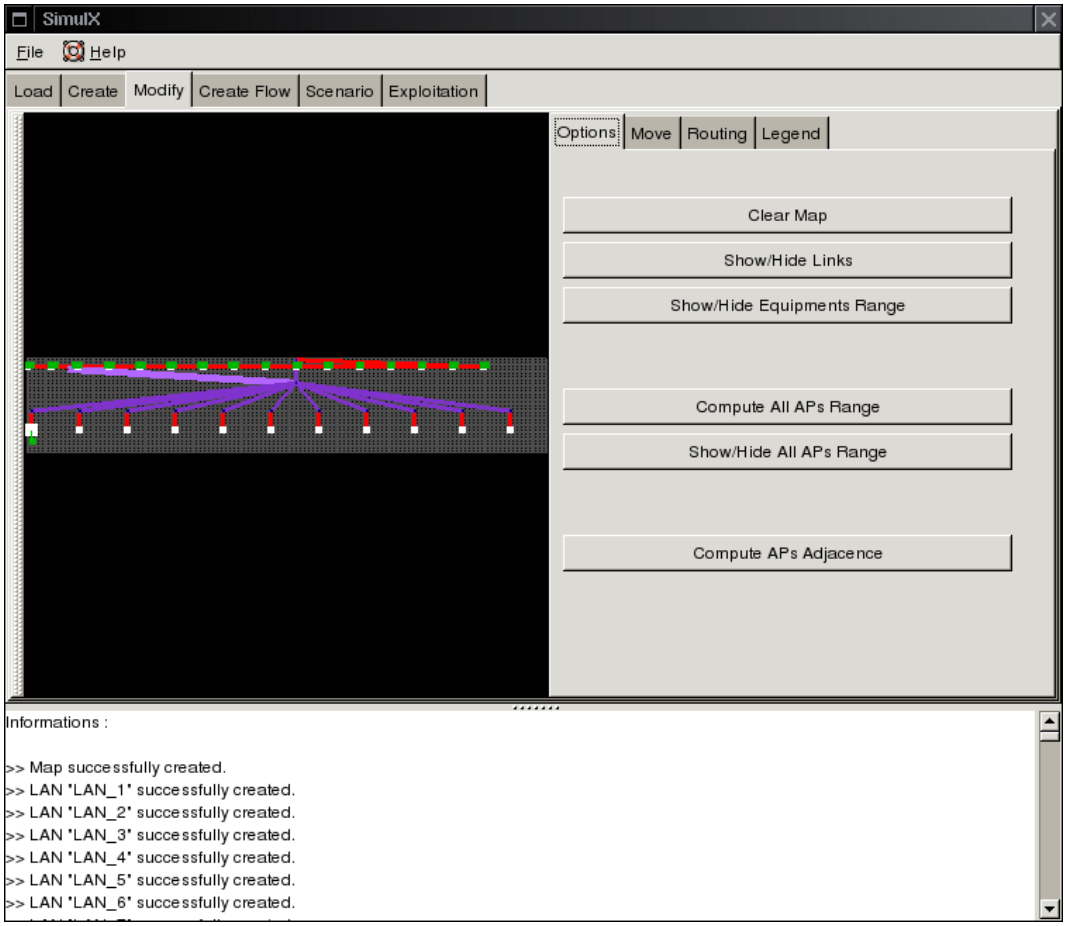


Figure 5.2: The SimulX Wireless LAN simulator

keep the process short to lower the risk of leaving the area covered by the AP before actually completing the handover (not to mention risks reasons like elevated consumption of resources, such as processor time, power consumption and network load).Therefore we have also added to the *DIMoN* the capability to receive and handle Link Layer Triggers by sending a Router Solicitation.

Figure 5.4 shows the following scenario: A mobile node would start moving at time 0 as shown on Figure 5.3. Exactly 1.242 seconds later it would lose the signal from its AP (i.e. signal intensity would drop below -82dbm). At that point signal from a second AP has already been available for some time. The same scenario is run for three mobile nodes, each handling mobility in a different manner.

- **Node 1**: implements Soft Handover and is located on a link with an RFC2461 compliant router. It is represented by the topmost bar on figure 5.4.

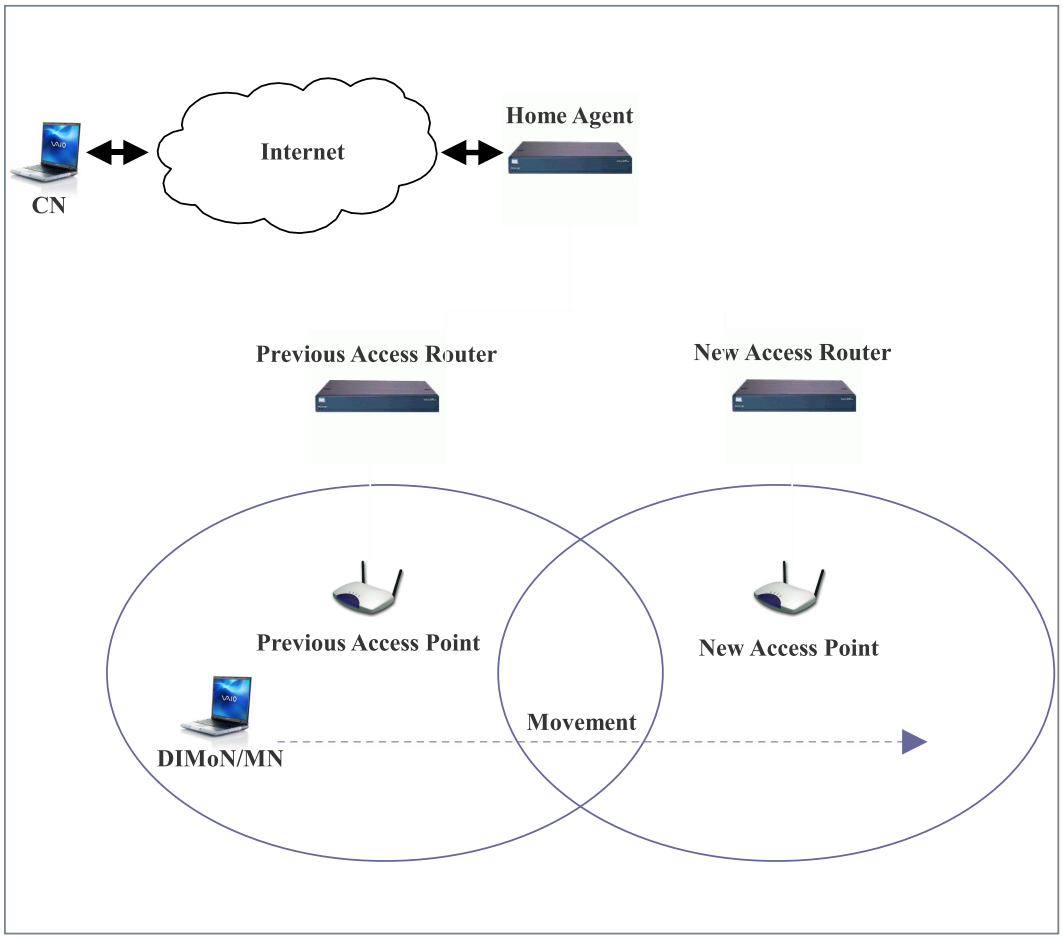


Figure 5.3: Scenario used in *DIMoN* evaluation tests.

- **Node 2**: implements standard MIPv6 and is on a link with an RFC3775 (MIPv6) compliant router (i.e. RAs are sent every 50 ms). The node is represented by the middle bar on figure 5.4.
- **Node 3**: implements MIPv6 mobility + the ability to send router solicitations. The node is represented by the bottom bar on figure 5.4.

As we can see from the figure, Node 2 (MIPv6 MN on a MIPv6 router's link) has managed to perform the fastest handover of all three nodes. Yet it was in a disconnected state for a period of about 330 ms.

The handover for Node 1 and Node 3 (see figure 5.4) lasted approximately the same time (Node1: 532ms and Node3: 515ms). Node 1, however started handover 694 ms

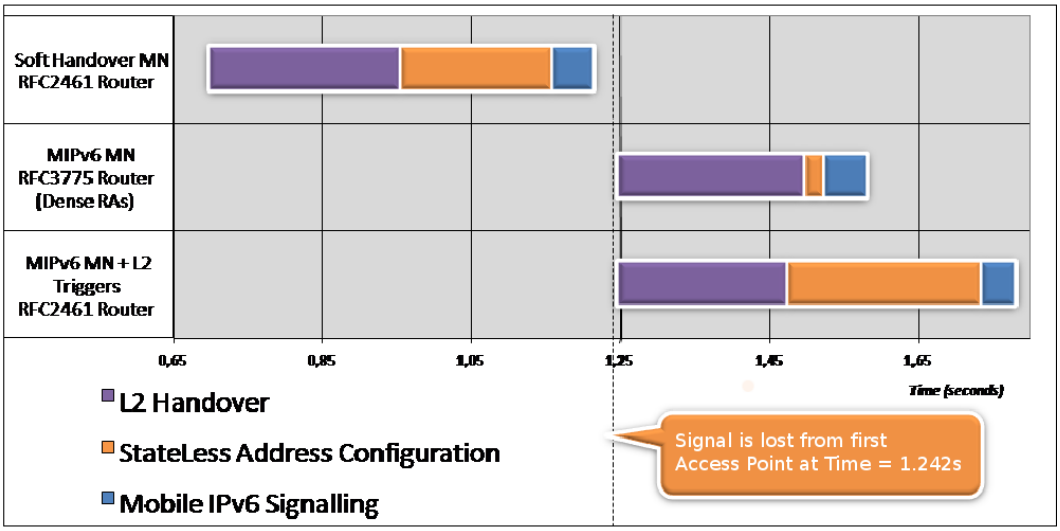


Figure 5.4: Duration of the handover process with a *DIMoN*.

after beginning its movement and at the time when the signal from the first AP faded away, it was already associated with the next one.

Next we analyze packet loss during handover. Figure 5.5 represents the same three nodes and the number of data packets that were emitted for each node and did not get to their destination due to connection loss. All three nodes are in the middle of a media session using a G.711 [91] stream.

Given the short handover duration for (the middle) Node2, packets that got lost on their way to it are fewer than in the case of (the leftmost) Node1. As expected, the (rightmost) *DIMoN*, going through a soft handover has not lost any data regardless of the relatively longer period needed to complete the transition.

Flow bicasting, inherent to Soft Handover algorithms, is often pointed out as increasing network load and therefore pointed out as a major disadvantage. Figure 5.6. shows the amount of extra network traffic caused by signaling or bicasting for the same three mobile nodes. Each column on figure 5.6. shows the number of extra packets during the handover and an enclosing second. As we can see the rapidity of the (middle)

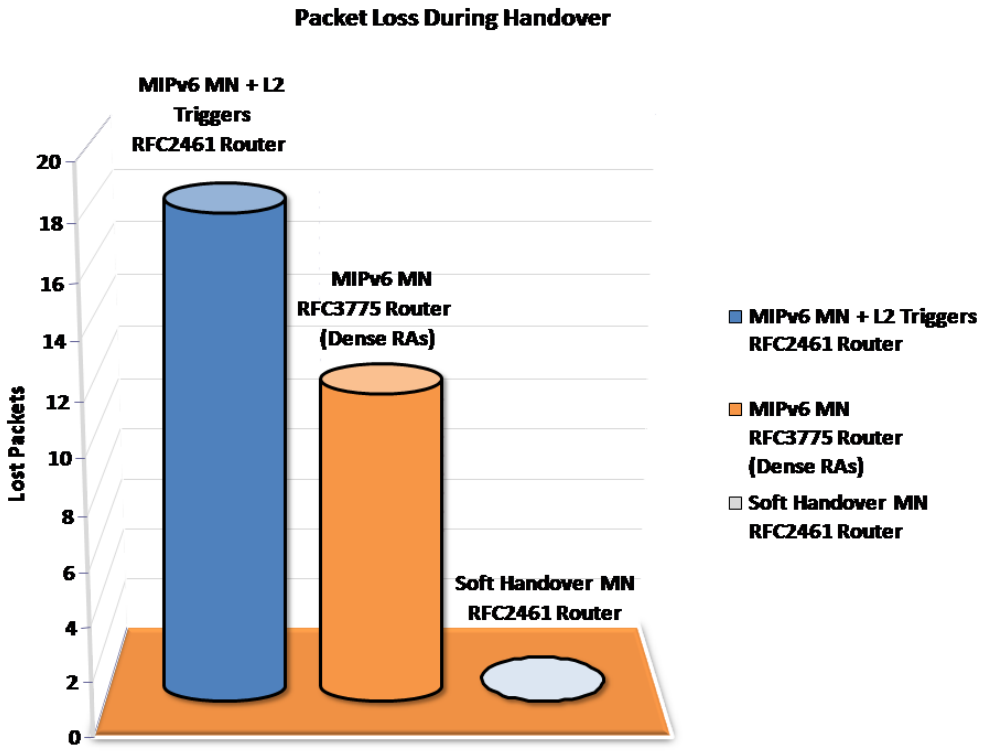


Figure 5.5: Packet loss during handover with a *DIMoN*.

Node2 comes at a cost. RAs broadcasted to aid movement detection generate excessive signaling traffic. Duplicating the data flow, quite on the contrary, turns out to be a more economical solution since it goes on during a relatively short interval.

5.1.3 Necessary further analysis

Though simple to implement, instantly switching to a new Access Point upon signal intensity drop, is not always the best strategy. It is necessary to explore further constraints. For example, prior to requesting ceasing of bicasting, a node should make sure that at least one of the two APs has signal intensity higher than a certain threshold. This may turn out to be crucial for certain network topologies and non-linear node movement. Another example constraint would be Access Point load. It would also be of interest to define a means for the Access Point or other network equipments to initiate handover as a way of implementing a load sharing policy.

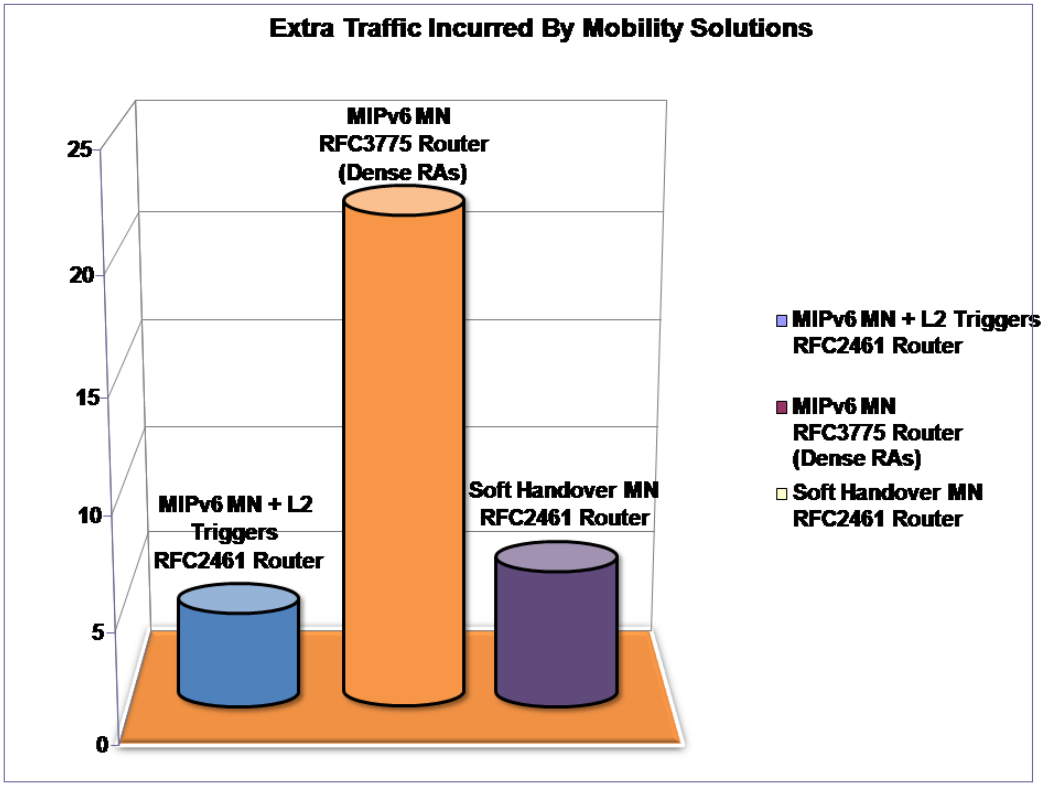


Figure 5.6: Extra traffic incurred by various hadover solutions.

Combining Soft Handover with existing MIPv6 extensions such as FMIPv6 [70] and HMIPv6 [184]is another promising field of research as it would allow for better centralized network control.

5.1.4 Summary conclusion on the soft handover approach

This section proposes an optimization of the handover process over WLAN, employing a model from cellular network architectures (soft handover) with the aid of an extra Wireless LAN interface. After an analytical evaluation and some simulation based performance testing it proved to be quite successful as it succeeds to virtually eliminate packet loss during handover in our simulation scenarios. Yet further work would have to prove the optimization merits by experimenting in real world conditions with an actual implementation. Existing mobility optimizations such as FMIPv6 would probably be needed to extend the applicability of the solution.

5.2 Double Wireless Network Interfaces with FMIPv6

In the previous section we have described a solution that allows a mobile node to use a secondary wireless interface in order to eliminate packet loss and reduce the connection disruption time during a handover.

The results from the evaluation of this approach have encouraged us to try and evaluate its impact when applied on existing mobility extensions such as the Fast Handovers for Mobile IPv6 protocol.

5.2.1 Solution Description

Applying the concept of multiple interfaces to the FMIPv6 protocol is quite straightforward and easy to understand. One of the protocol's strongest points is the fact that it completely separates the scanning phase, or in other words the discovery of nearby access points and routers, from the rest of the handover procedure.

We have already shown in Chapter 4 that the process of switching from one wireless network to another is quite optimal with FMIPv6. The protocol completely eliminates packet loss and keeps the connection disruption interval lower than the maximum which is generally allowed by jitter buffers.

We have also shown that the only limitation with the FMIPv6 protocol comes from candidate access point discovery since it is virtually unoptimized. Connection disruption time and packet loss accumulated during this phase are so high that it they greatly outweigh any benefits that the protocol may bring during the handover itself. It is therefore important that any optimizations and future work on that protocol focus on its discovery phase.

We have thus decided to limit use of multiple interfaces only to this phase. In other words, after every handover or on any other occasion that a node is to trigger a IEEE 802.11 scan, it would enable a secondary wireless interface and use it to perform the scanning procedure. In order to keep unnecessary energy consumption to a minimum, the node would disable the interface as soon as it has completed scanning and would keep it down until the next time it needs to scan its environment. Note that the node would only use its secondary WLAN interface for scanning and all other communication would continue uninterrupted through the primary interface of the node.

Through the rest of this section we will describe a comprehensive evaluation of this solution. We study behavior of a multi interface FMIPv6 node for both predictive and

reactive handover and we also analyze its advantages in a network initiated and a layer two only handover.

5.2.2 An experimental performance evaluation

The testbed

Our testbed is composed of three AR-s, three AP-s, one MN and one correspondent node. It also contains three IPv6 subnets. A top AR provides IPv6 Internet connectivity to the rest of the testbed and is also configured as a HA. AP1 is connected to one of the AR-s while AP2 and AP3 are connected to a second AR and hence a different IPv6 subnet. Figure 5.7 illustrates the testbed.

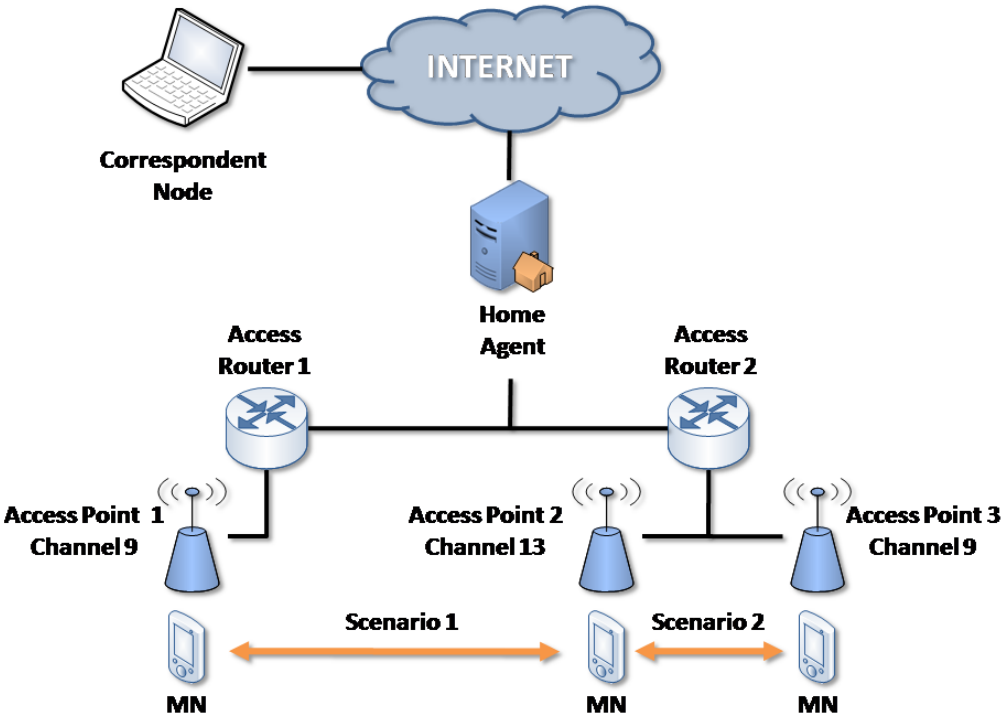


Figure 5.7: Testbed and scenarios used in the experiments

All devices are running the GNU/Linux operating system, except for the AP-s which are 802.11b Cisco AP 1200 devices. The HA is running the new MIPv6 daemon for the GNU/Linux operating system (MIPL-2 [143]). The MN and the AR-s (AR1 and AR2)

are using the FMIPv6 Open Source Implementation Suite (fmipv6.org [66]) which is based on MIPL-2. We are also using a legacy IPv6 node connected to the Internet from a point outside the testbed.

Due to the nature of the FMIPv6 protocol, the fmipv6.org implementation requires fine grained control over the behavior of the wireless card so that it could be efficient. We have therefore equipped the MN with a 3Com 802.11 a/b/g PCMCIA wireless card managed by the MADWiFi driver [44]. MADWiFi is an open source GNU/Linux kernel device driver for wireless LAN chipsets from Atheros. Our driver modifications address the periodic scans (occurring prior the RtSolPr/PrRtAdv exchange), the link-layer triggers (required for predictive handover) and the L2 handover itself.

It is important to note that for all testing we have been using an fmipv6.org specific feature that allows all scanning to be executed on a secondary wireless interface. This allows us to resolve (or rather circumvent) FMIPv6's inability to provide the Link-Layer details of candidate Access Points, such as their ESSID and frequency. By using a secondary interface for all scanning, communication through the primary interface is never interrupted. The candidate access point discovery problem is described in more detail in [94, 134].

Evaluation scenarios

In order to provide a complete and consistent evaluation of FMIPv6, we have run two distinct series of tests in two different scenarios. In Scenario 1, the MN would move between AP1 and AP2 and it would therefore also change its IPv6 subnet, moving from AR1 to AR2. This case represents a very common FMIPv6 demonstration scenario. Although FMIPv6 is often presented as a L3 solution, we have decided to evaluate its performance when optimizing L2 only handovers. In Scenario 2 the MN would therefore be moving between AP2 and AP3, without changing an AR in the process. (See Figure 5.7).

For both scenarios we have executed a series of tests in which the MN is in the process of having a video conversation with its correspondent (both hosts are equipped with a webcam and use the Gnomemeeting software [175]). We have also run a second series of tests in which the MN receives a video stream from the correspondent node (using VLC [42]). In the video stream case, data is encapsulated and sent through the Real-time Transport Protocol (RTP) [178]. RTP packets have an average length of approximately 1336 bytes and are sent every $30ms$. Gnomemeeting also uses RTP only this time the audio and video are sent in separate flows. Average audio packet size is

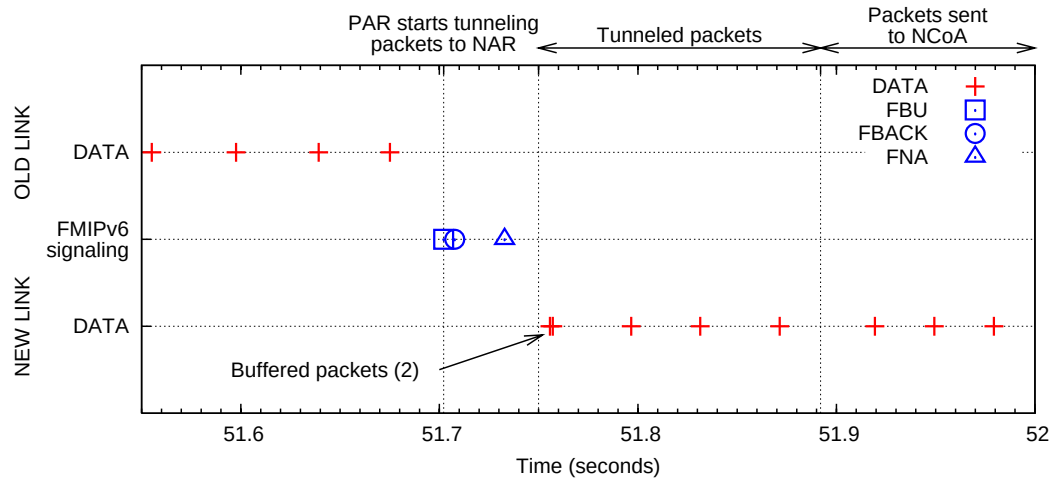


Figure 5.8: Impact of FMIPv6 Predictive HO on a video stream

approximately 70 bytes and packets are sent every $30ms$. Video packets have an average size of 950 bytes and are sent every $160ms$.

In Scenario 1, we perform an evaluation of all use cases defined by the FMIPv6 protocol: predictive handovers, reactive handovers and network-initiated handovers. Note that the FMIPv6 specification does not define network-initiated handovers as a separate mode of operation but we have decided to run separate tests for them as we believe, that due to their possibility to allow for network mobility control and load sharing, they might represent special interest to many readers and potential implementors. With the tests run for Scenario 2, we analyze the influence of predictive handover on L2 handover in order to evaluate the benefits of buffering (or lack of it) while performing L2 only handovers.

Evaluation results

Results presented in this section are obtained by running the Ethereal tool [45] on the MN, the correspondent node and the AR-s. Every distinct test case (e.g. a Gnomemeeting conversation during a Predictive Handover in Scenario 1, or a VLC streaming session in a reactive L2 only handover) was evaluated through a series of at least ten consecutive test runs, which gives us a total of 13 cases and more than 130 test handovers. Results shown on all following figures correspond to the test run whose L2 handover time represents the median for all L2 handovers in the corresponding test case.

Figures 5.8 and 5.9 present the results for FMIPv6 predictive handovers in Scenario 1. We can see, that the predictive mode allows MN-s to not lose even a single packet

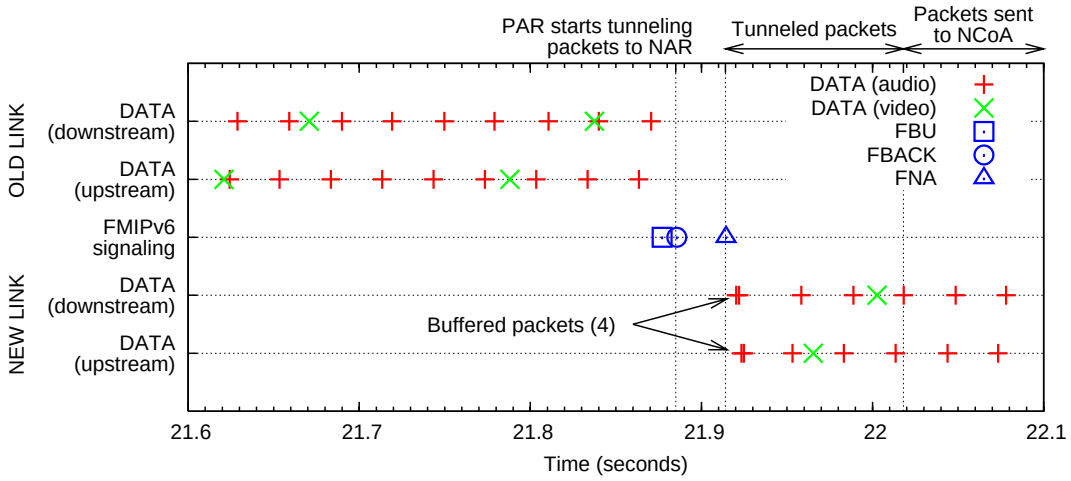


Figure 5.9: Impact of FMIPv6 Predictive HO on a video conference stream

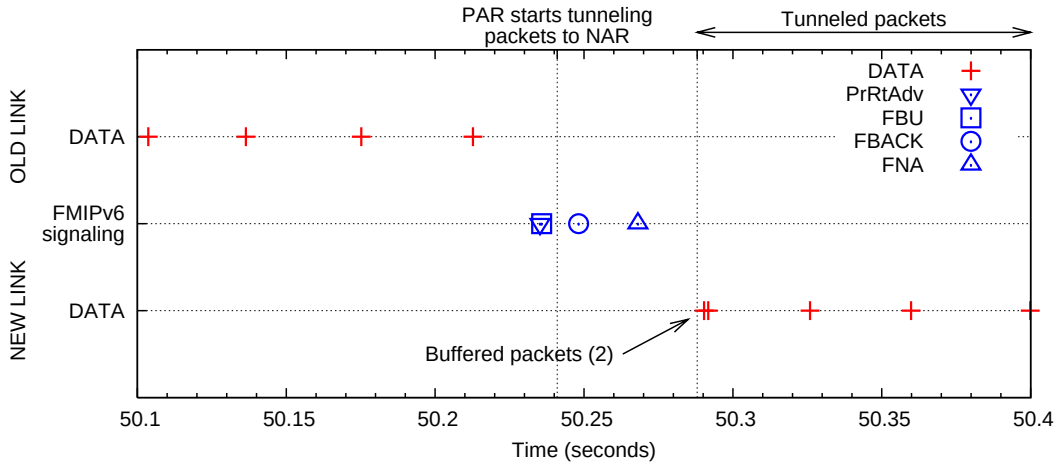


Figure 5.10: Impact of FMIPv6 network-initiated HO on a video stream

while performing the handover. Triggered by a link-layer event, the MN initiates the handover by sending to its current AR (i.e. the PAR) a FBU containing the identity of the AR that it plans to switch to (i.e. the NAR). After exchanging the HI and HACK messages with the NAR, the PAR starts tunneling packets bound to MN and routes them to the NAR. It then sends a FBACK back to the MN. Upon reception of the FBACK, the MN starts the L2 handover and associates with the new AP. Once on the NAR's link (after approximately $20ms$), the MN sends a FNA requesting the NAR to deliver all packets that it has buffered so far. At this point data packets are still being forwarded to the PAR by the HA and are therefore tunneled to the NAR through the FMIPv6 tunnel until the MN sends a BU from the NAR's link allowing the HA to update the corresponding binding cache entry. After the MIPv6 BU/BACK exchange, the HA starts forwarding data packets directly through the NAR. The tunnel between the PAR and the

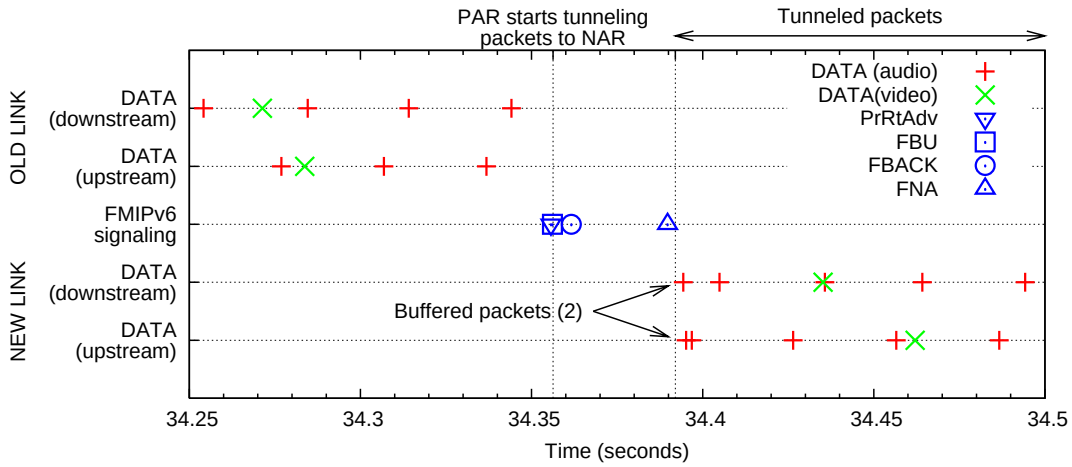


Figure 5.11: Impact of FMIPv6 network-initiated HO on a video conference

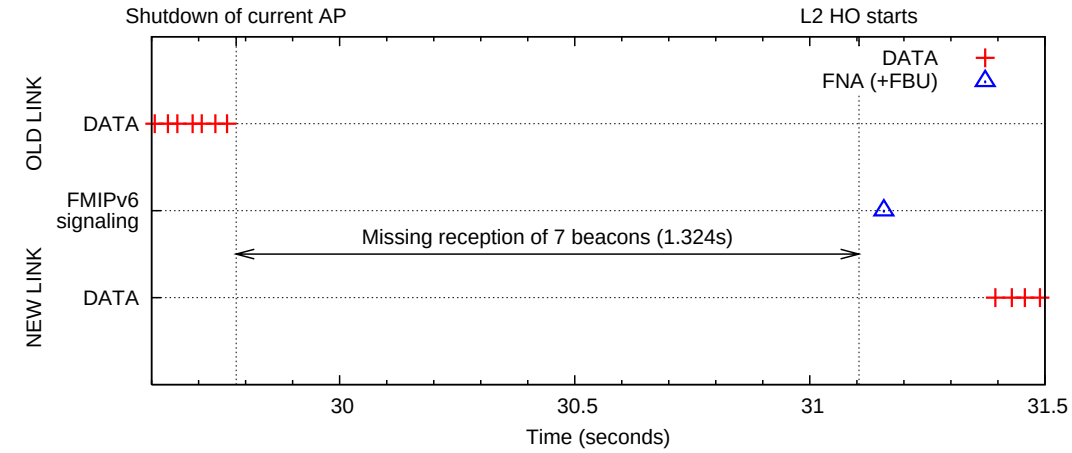


Figure 5.12: Impact of FMIPv6 Reactive HO on a video stream

NAR has been active (i.e. has been forwarding packets) for $190ms$ (Figure 5.8) and $134ms$ (Figure 5.9) on average, with respectively 5 and 12 packets forwarded through the FMIPv6 tunnel. The lifetime of the tunnel between PAR and NAR is specified by the MN at the transmission of the FBU. During our experiments, we configured the MN to request the minimal allowed tunnel lifetime, i.e. 4 seconds according to [70]. Note that the MN is not required to perform DAD before sending the BU as its NCoA is negotiated prior to the handover (through the FMIPv6 exchanges) which explains the relatively short interval between the end of the L2 handover and the transmission of the BU. We observe small delays during the reception (or transmission) of buffered packets, but these delays have no impact on the user application (neither on the MN nor on the correspondent node). As a result, the test runs for these cases are an example of completely seamless FMIPv6 predictive mode handovers.

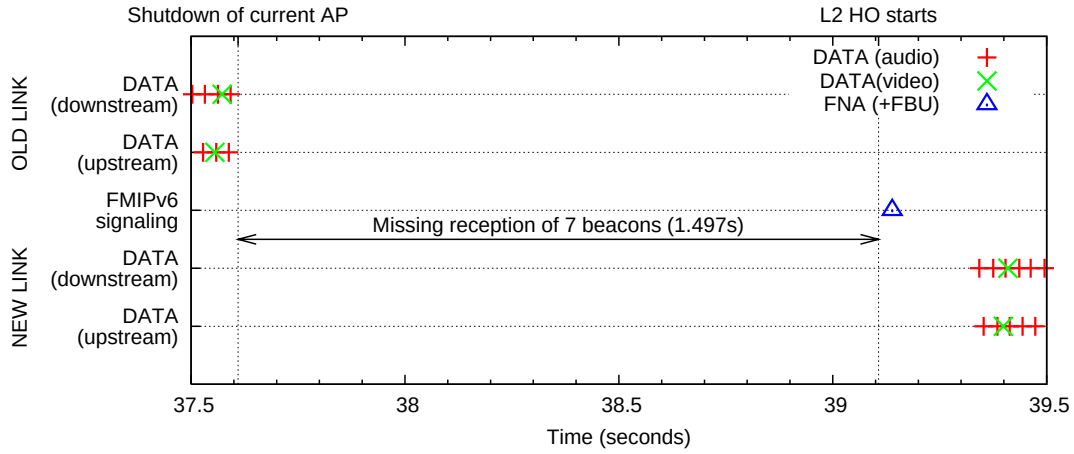


Figure 5.13: Impact of FMIPv6 Reactive HO on a video conference stream

Results related to FMIPv6 network-initiated handovers in Scenario 1 are shown on Figures 5.10 and 5.11. As expected we can see that FMIPv6 performance here is very similar to that observed during predictive handovers. In fact, the only thing that differentiates a network-initiated handover from a predictive one is that it is up to the PAR to select a destination AP and NAR for the MN, as well as the fact that the handover would start right after the MN has received the PrRtAdv (Figures 5.10 and 5.11). As a result FMIPv6 network-initiated handovers, like those initiated by the MN, are in most cases completely seamless. Note that packets sent directly to MN's NCoA (after the BU/BACK exchange between the MN and the HA) are not shown for readability reasons.

Figures 5.12 and 5.13 show the performance of FMIPv6 reactive handovers for Scenario 1. If a mobile node fails to anticipate a handover and does not sent a FBU from the PAR's link, it would perform a FMIPv6 reactive handover. To trigger reactive handovers, we manually shutdown the current AP of the MN (hard reset). As we can see from both Figures 5.12 and 5.13, it takes respectively $1.324s$ in the VLC test case and $1.497s$ when testing with Gnomemeeting for the wireless device (network card) to detect the link failure and starts the L2 handover. In wireless LANs, a link failure is generally detected through the Beacon messages that AP-s send periodically. After missing more than a certain number of such beacons a node would assume that it has lost its connection with the corresponding AP. Missing a single beacon frame does not necessarily imply connection loss as the event may be due to a link layer collision. The probability for such collisions to occur in wireless LAN-s is quite high, and wireless devices would generally wait a certain amount of time before giving up on the connection. When configured with default parameters the MADWiFi driver would wait for 7 times the default beacon retransmission interval before reporting that the link is down. In our testbed, the

AP-s are configured to send Beacon every $200ms$ which explains the amount of time necessary for the MN to initiate the L2 handover.

After completing the L2 handover, the MN sends a FNA and a FBU from its new link. The rest of the handover is not different from a standard predictive one (see Section 2.9.2). All packets sent between the shutdown of the AP and the reception of the FBU on the PAR are lost as there has been no buffering or tunneling (the PAR would tunnel packets for the MN only after receiving the FBU from the MN). During these delays, the MN has lost 60 packets for VLC (Figure 5.12) and 130 packets for Gnomemeeting (Figure 5.13) which leads to a relatively long interruption of media flows. For the Gnomemeeting tests the 130 packets can be divided to: 57 audio and 11 video packets (downstream) and 51 audio + 11 video packets (upstream).

As the lower performance of the FMIPv6 reactive handovers is mainly due to the slow link failure detection mechanism of the wireless device, we have decided to further evaluate them in some slightly different conditions. Instead of simply unplugging the AP we shut it down through its user interface. In this case the AP sends a Disassociation Request to the MN which is now able to immediately detect the link failure and start a L2 handover. Through the rest of this section we refer to this test case as "Fast Detection". We also include this test case when evaluating reactive handovers in Scenario 2 (i.e. when the MN only performs a L2 handover). The results have been obtained in the same circumstances as already explained (i.e. using VLC and Gnomemeeting). Figure 5.14 shows the results of these additional experiments. Each dot represents the reception of a data packet, at the time indicated in the X-axis. We have included the results for the FMIPv6 predictive handover in Scenario 1 as a reference. During a Fast Detection test, the MN loses an average of 8 packets, which is more than with a predictive handover, where there were no lost packets, but considerably less than the 60 packets of the standard reactive case. Using such fast detection of the link failure, the MN could initiate the handover sooner and therefore reduce the duration of the overall procedure. However, such amount of lost packets still introduces short flow interruptions in the video. In the case of a L2 only reactive handover, for standard and Fast Detection schemes, we have lost respectively 43 and 1 packets while the MN has been disconnected.

In Scenario 2 we analyze the benefits of the FMIPv6 predictive mode when the MN only performs a L2 handover. Due to the fact that FMIPv6 is often presented as a network layer solution, some might argue that using it in a Wi-Fi deployment that only uses a single subnet might be unnecessary. Therefore in order to demonstrate the benefits of FMIPv6 in L2 only handovers, we have completed a series of tests in such an environment. Figure 5.15 shows the results from these experiments for both video streaming and a videoconference. Each dot represents the reception (or transmission) of a data

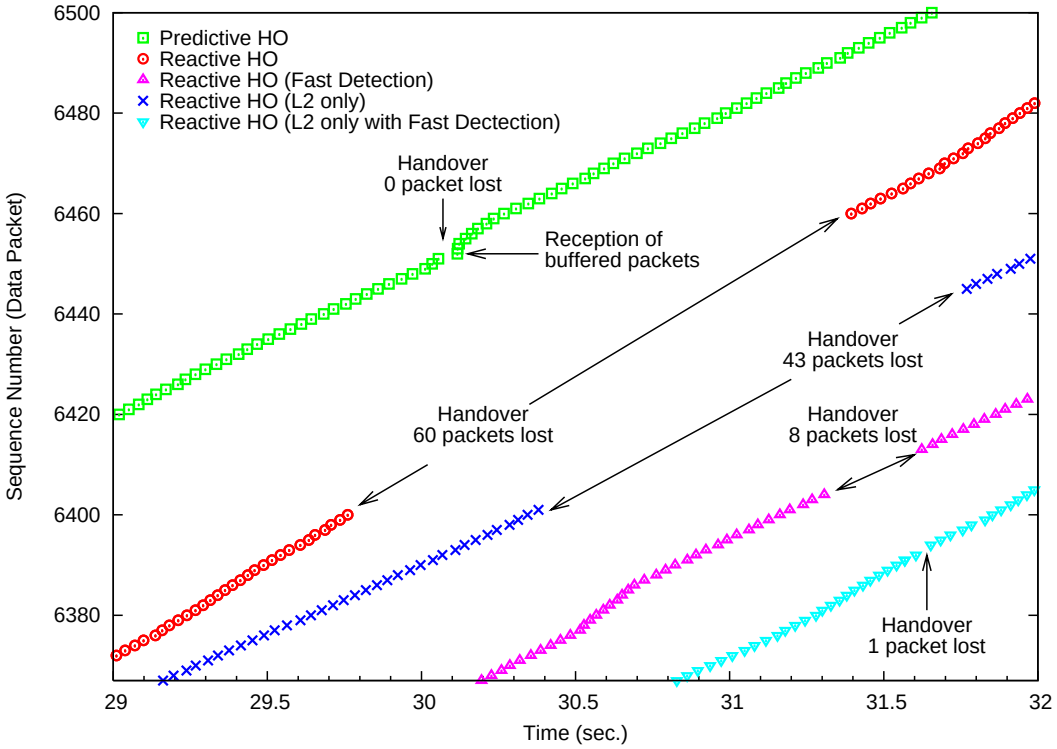


Figure 5.14: FMIPv6 Reactive Handover performance

packets at the time indicated on the X-axis. We can see that if the MN does not send a FBU for a L2 handover (the case is referred to as no-optim. on the figure) it loses 1 packet in the video streaming use case and 3 packets for the videoconference use case. This relatively small number of lost packets is explained by the short duration of the L2 handover. When using the information previously discovered through periodic scanning and the RtSolPr/PrRtAdv exchange, takes approximately $20ms$ to complete, as the MN could immediately send and receive packets after the L2 handover has completed. However, with even as few as 1-3 lost packets the MN would still experience noise in audio and video rendering for both VLC and Gnomemeeting applications. When applying the mechanism of L3 predictive handovers (cases shown as with-optim. on the figure), the MN does not lose a single data packet because of the buffering at both the MN and the NAR (which is also the PAR in this scenario). Despite the short delays observed in the reception (or transmission) of buffered packets, the handover remains seamless for the user in both the VLC and the Gnomemeeting use case. In other words it appears that a MN can earn a lot in terms of performance when using FMIPv6 even for L2 only handovers. Finally, Tables 5.1 and 5.2 present the average values for all results related to our experiments.

Table 5.1: Results for experiments related to VLC

Handover	Average HO Proc. Length (ms)	Average P. Loss	Average Packet Buffered	Average User Experience
Predictive	33.2	0	1.3	No interrupt.
Predictive (L2 only)	19.3	0	0.5	No interrupt.
L2 no opt.	18.5	0.6	0	Short flow interruption
Network Initiated	32.6	0	1.7	No interrupt.
Reactive	1518.9	52.6	0	Long flow Interruption
Reactive (Fast det.)	248.4	8.3	0	Long Flow Interruption
Reactive (L2 only)	1442.3	45.1	0	Long Flow Interruption
Reactive (L2 only +) Fast det.)	20.0	0.8	0	Short Flow Interruption

Table 5.2: Results for experiments related to Gnomemeeting

Handover	Average HO Proc. Len (ms)	Average P. Loss	Average P. Buffered	Average User Experience
Predictive	36.6	0	2.3	No interrupt.
Predictive (L2 only)	20.9	0	0.4	No Interrupt.
L2 no opt.	17.7	1.7	0	Short flow Interruption
Network Initiated	28.9	0	1.6	No Interrupt.
Reactive	1542.3	127.7	0	Long Flow Interruption

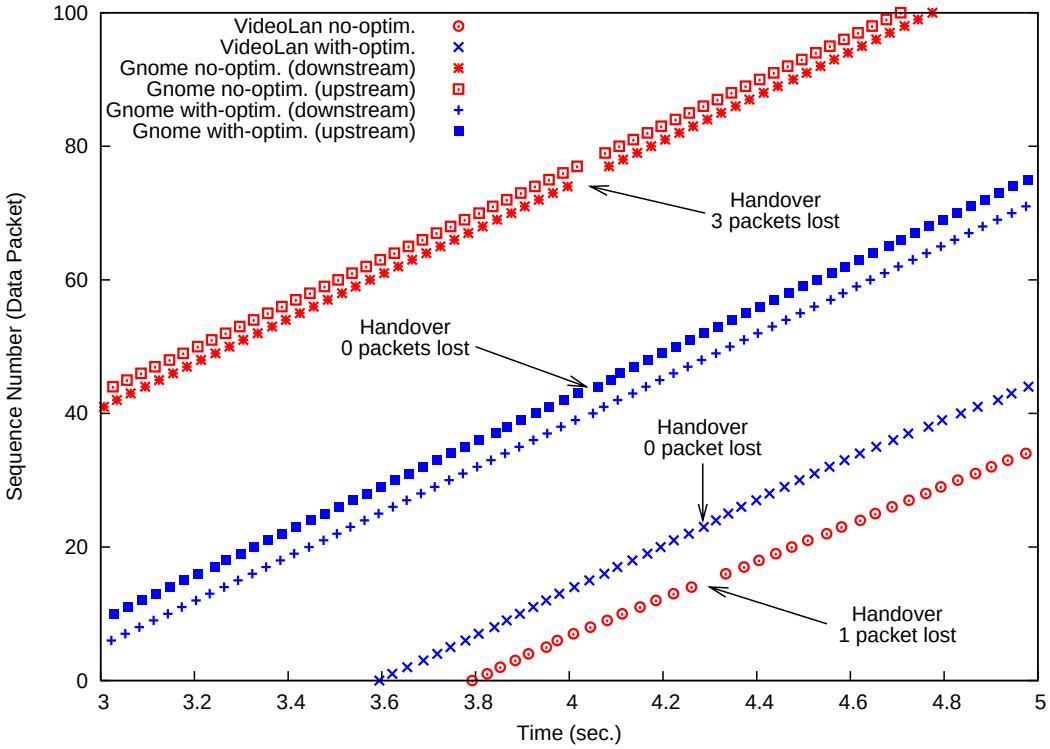


Figure 5.15: Benefits of FMIPv6 for IEEE 802.11 handovers (scenario 2)

5.2.3 Conclusions and future work

The results that we have presented in Section 5.2.2 show that the performance of a solution using a secondary interface to improve the FMIPv6 protocol is generally quite satisfactory, even though the exact duration could vary a lot depending on the employed FMIPv6 operation mode. The FMIPv6 predictive mode in this case has proved to be very efficient with as few as 0 lost packets during a handover and no user-perceptible cuts or delays at all. After successfully anticipating a movement event, every packet sent while the MN is performing handover is forwarded to and buffered by its Next Access Router (NAR). Once the MN is connected to its new subnet, the NAR could directly deliver buffered data packets. This mode is clearly the main advantage of the solution.

Network-initiated handovers, that we analyzed in order to get an idea on the possibility of using the protocol for purposes like load sharing, provide performance that is virtually the same as with predictive handovers.

Not surprisingly, the FMIPv6 reactive mode is less impressive and when MNs are not able to anticipate a handover it only allows to limit the damage to some extent. As

we can see from the results, the performance of reactive handovers is directly related to the time that is necessary for the wireless device to detect link failure. The longer this detection lasts, the more it is going to take for reactive handover to complete, and since no buffering has been started, all packets delivered during this period of time are lost. We have shown, however, that even in this case, a reactive mode handover could still be performed with satisfactory performance if we have a relatively fast link failure detection mechanism (see Figure 5.14). Note, that this problem is not related to the use of multiple interfaces or even FMIPv6 and is also an issue with the standard Mobile IPv6 protocol.

Finally, we have seen that using FMIPv6 with a secondary wireless interfaces (predictive mode in particular) could also reduce the number of lost packets during a layer 2 only handover (i.e. when the new access point of a MN is located in the same IPv6 subnet as the old one). Depending on the layer 2 handover latency and the rate at which data packets are sent (respectively every $20ms$ and $30ms$ on average in our experiments), the MN is likely to lose a substantial number of packets while performing a layer 2 only handover without any optimizations. FMIPv6 allows a MN to request packet buffering from its current access router prior to initiating its layer 2 handover, by simply notifying it that it is planning to move to another one of its access points (i.e. the PAR is the same as the NAR). In other words, while presented as a layer 3 specific solution, FMIPv6 could just as easily be used for layer 2 only handovers.

In conclusion, the FMIPv6 protocol combined with the use of a secondary wireless interface for scanning is particularly well suited for achieving seamless handovers over wireless LANs: predictive and network-initiated handovers prevent a MN from losing data packets and even with reactive handovers we are still able to achieve better handover performance than what we could hope for with the standard Mobile IPv6 protocol.

5.3 Optimizing FMIPv6 with geographical positioning information

The goal of the solution that we describe in this section is to provide an optimized method for discovering candidate APs and remove the delay inherent to the standard procedure (i.e. periodically performing complete or partial scanning [130]). This solution is an alternative to the one presented in the previous Section 5.2 and its use does not mandate the presence of a secondary wireless interface. It allows to take into account the geographical position of an MN in order to anticipate its trajectory and choose its next AP/AR couple accordingly. This way, based on its position, velocity and direction,

an MN would be able to request from its current AR the parameters of the AP it is most likely to switch to. We also define several ICMPv6 options that an AR could use with a PrRtAdv message in order to send L2 connection parameters to an MN so that it won't need to discover them through a scan.The actual handover (i.e. the way FBU, FBACK, HI, and HACK messages are exchanged) remains unchanged and follows the procedure defined by FMIPv6.

Our FMIPv6 extension requires MNs to be able to determine their geographical coordinates using a generic geographic positioning system (e.g. GPS for an outdoor environment). In order to do this , existing geolocation systems generally use triangulation based on radio signal strength as "seen" byfixed or mobile nodes with a known location. There are three major kinds of geolocation systems: mobile-based, mobile-assisted and network-based [185, 146]. We consider that for assisting mobility using a geolocation system in wireless LAN, it is best to use a mobile-based solution such as the GPS System or the wireless LAN infrastructure itself as suggested by [72]. This is the approach we have adopted for our experiments. In all tests scenarios MNs use a mobile-based geolocation system, i.e. each MN is aware of its own position.

5.3.1 Database extension for Access Routers

In order to support our new candidate AP detection mechanism, we had to extend FMIPv6 routers. As previously mentioned, FMIPv6 defines the semantics that allow sending to an MN the properties of IPv6 subnets behind the surrounding APs before it has actually engaged in a handover. For this purpose, every AR manages a database that maps APs to IPv6 subnets, including the layer 3 parameters of these IPv6 subnets. We have extended this database to include some extra layer 2 parameters related to APs. In addition to the link layer addresses (already stored in the database), the APs are registered together with their operating channel, Service Set Identifier (ESSID), radio range, and position (either with relative or geodesic coordinates). As this information does not change often, the database is populated manually.

5.3.2 Mobility cache

In addition to the database described in the previous section, every AR also monitors the MNs currently connected to its subnet and keeps their last known status in a so called Mobility Cache. Every record in this cache contains the link layer address of the corresponding MN, its last coordinates, the link layer address of its current AP, and a

context. A context is a collection of layer 2 and 3 data related to the APs that the MN may move to. Section 5.3.3 describes the way we create such contexts.

The Mobility Cache of an AR is dynamically updated by RtSolPr messages. In FMIPv6, an MN would generally send an RtSolPr message to its current AR when it needs to resolve one or more Access Point Identifier (generally the link layer address) to subnet specific information (see Section 2.9.2). We have slightly extended the purpose of RtSolPr-s and in our solution an MN would also use them to periodically update its geographic position with its AR. For this reason, we have defined a new ICMPv6 option, called Position Information Option, that allows the MN to also send its coordinates inside the RtSolPr. It contains the coordinates of the MN and the identifier of its current AP (i.e. the link layer address).

In order to limit the signaling overhead that may be generated by frequent transmissions of RtSolPr, the MN would only be sending periodic updates if the signal quality of its radio connection with the AP has fallen below a certain level. We define a first signal quality threshold S_a corresponding to an average quality. Whenever the signal quality drops below S_a, the MN would start sending, the results of its position checks to the AR which would update its Mobility Cache . The AR would then try to determin if the MN would soon have to perform a handover. This assessment is based on the AP range parameter R, that we store in the extended AR database. R corresponds to the maximum distance between the AP and an MN that still allows for a relatively good signal quality. As long as the distance D between the MN and its AP is shorter than R, we assume that the MN is still well covered by its current AP. Otherwise (i.e. when $R < D$), the AR would try to select a new AP for the MN according to its trajectory. The combination of signal quality and distance allows ARs to overcome temporary signal degradations or geographic positioning errors which may disturb the algorithm behavior when taken separately. According to the results presented in [138], we have set S_a between [-75dBm;-78dBm] and R to the half of the maximum range of the corresponding AP.

5.3.3 Selecting the next Access Point

As previously mentioned, the selection of the next APs is based on the trajectory of the MNs. To do this, we determine a trajectory by applying a linear interpolation on the two last known positions of the MN. This gives us the following parametric representation:

$$x(t) = O_x + t'U_x \tag{5.1}$$
$$y(t) = O_y + t'U_y \tag{5.2}$$

where O_x and O_y are the coordinates of a point, and U_x and U_y the coordinates of the director vector $\vec{U}$ of a line that passes through that point. Let (x_0, y_0) and (x_1, y_1) be the two last known positions of an MN. The equation of the trajectory would then be:

$$x(t) = O_x + t'(x_1 - x_0) \tag{5.3}$$
$$y(t) = O_y + t'(y_1 - y_0) \tag{5.4}$$

We assume that the coverage area of an AP is a circle. Let us denote the radius of this circle as G and let (a, b) be the coordinates of its center (the geographical position of the AP). The equation of the boundary of its coverage area would then be:

$$G^2 = (x - a)^2 + (y - b)^2 \tag{5.5}$$

Then, if we replace x and $y$in Eq. 5.5 by Eq. 5.3, we obtain a polynomial equation of second degree:

$$At^2 + Bt + C = 0 \tag{5.6}$$

in which:

$$A = ||\vec{U}||^2 \tag{5.7}$$
$$B = 2'\vec{MO}.\vec{U} \tag{5.8}$$
$$C = ||\vec{MO}||^2 - G^2 \tag{5.9}$$

where is the line's director vector for the MN's trajectory, is a point of the MN's trajectory, and is the center of the circle defined by Eq. 5.5.

In order to select a new AP for an MN, the AR will resolve Eq. 5.6 for every AP that is geographically adjacent to the current one (i.e. the zones covered by both APs intersect). When the discriminant is negative, the Eq. 5.6 has no real roots, this means that the MN is probably not headed for the coverage area of the AP. If the discriminant is zero, the Eq. 5.6 has exactly one root, this means that the trajectory of the MN is tangent to the coverage area of the corresponding AP. Finally, if the discriminant is positive, the

Eq. 5.6 has two distinct roots, this would mean that the MN seems to be moving right into the zone covered by the corresponding AP. When resolving Eq. 5.6, the AR will create a list of adjacent APs sorted in ascending order by the value of t. This way the first AP of the list would be the one that is likely to cover the MN for the longest distance according to its trajectory. Once the handover begins, the MN will try to connect to the first AP in the list. If it is not reachable it will go on with the second, and so on (see Section 5.3.4). As the selection of the next AP consumes resources on the AR, we try to maintain the AP list as long as possible. If a context is already set for the current association of the MN, the AR would check if the first AP of the list is still a valid target for the pending handover. If this is no longer the case, the AR has to select a new one as described previously.

Once the context is calculated, the AR would send it to the MN using a PrRtAdv message. We have also defined a new option for PrRtAdv, that we call the Access Point Information Option. It allows transporting the ESSID and the radio channel of an AP. Do notice that FMIPv6 already defines options to provide in PrRtAdv all other parameters necessary for the handover (link layer address of AP, new IPv6 prefix, new AR's link layer and IP address). The way options are ordered in PrRtAdv-s has to follow the AP order in the list (i.e. the first options are related to the first AP of the list). Upon reception, the MN will save the context until it starts a handover or until a new context is received.

5.3.4 Handover management

This section describes the handover process when initiated by MNs. Note that FMIPv6 also defines a mechanism for an AR to initiate a handover - a network-initiated handover [70], initially previewed for purposes like load sharing. We have defined a second signal quality threshold S_w, which corresponds to the lowest acceptable signal level. Once the signal quality drops under S_w, the MN would start the handover process. Layer 3 handover is carried out exactly as described in FMIPv6. If a context is set, the MN begins an FMIPv6 predictive handover by sending an FBU to its current AR including the parameters related to the first AP of the context. Once the MN receives the corresponding FBACK, it starts the layer 2 handover by switching its radio channel to that of the new AP and sends a Probe Request its ESSID. Upon reception of a Probe Response from the destination AP (identified by its link layer address), the MN moves directly to the authentication stage. If the authentication is successful, the MN goes through the association stage and completes the layer 2 handover. As a result, in case of a successful anticipation the delay of the layer 2 handover is significantly reduced , as the MN does

not have to scan several channels to discover its next AP and neither does it have to wait for MinChannelTime or MaxChannelTime before completing the handover process.

The rest of the handover follows the FMIPv6 specifications. During the layer 2 handover, the data packets sent to the MN are forwarded by the Previous AR (PAR) to the Next AR (NAR) which buffers them. After associating with its new AP, the MN sends an FNA to the NAR so that it would deliver all buffered packets and start routing the New CoA (NCoA) of the MN. Finally, the MN sends a BU to update its binding with the HA. Figure 5.16 illustrates the entire procedure.

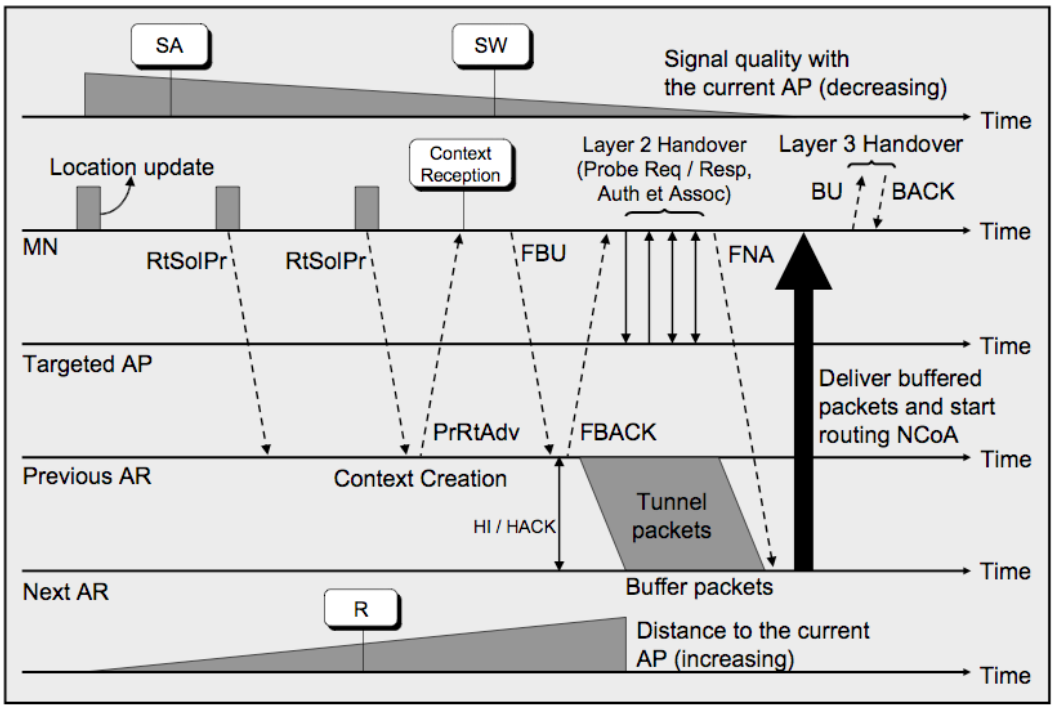


Figure 5.16: Protocol overview when the anticipation is successful

If the MN does not receive a Probe Response from the targeted AP, this could either mean that there was a collision on the wireless link, or that the AP was out of range. The latter may occur as a result of a wrongful projection of the MN trajectory (e.g. due to positioning errors or change of the direction of the MN). As the probability for an AP to fall out of range is quite high, we have decided to limit the time the MN waits for a Probe Response from an AP to 5ms. After 5ms, the MN considers that the target AP is not reachable and tries the next one in the AP list and so on. When a layer 2 handover completes, and the AP that the MN has attached to was the second one it tried (i.e. the attempt to connect to the first one has failed) two cases may occur. If the new AP is connected to the same IPv6 subnet as the first one it tried, no further

considerations are necessary and the MN can pursue the handover process as defined in the case of a successful anticipation. Otherwise, the MN would find itself in a situation where its packets are being tunneled to and buffered by a different router (referred to as AAR for Anticipated Access Router in) and not the NAR. In this case the MN would perform a specific reactive handover by sending to AAR an FNA which contains the FBU corresponding to its current location. As a result, two tunnels will be set up simultaneously, the first one from PAR to AAR, and the second one from AAR to the actual NAR. Upon reception of data packets addressed to the MN, the NAR delivers them directly to the MN. Finally, the MN updates its binding to the HA. This case is referred to as enhanced reactive mode.

5.3.5 Experimentation

The testbed

In order to evaluate the performance of our proposal, we implemented it and set up a testbed composed of two APs, one HA, two ARs, one MN and one correspondent node. The testbed contains three IPv6 subnets. The top AR provides IPv6 Internet connectivity to the rest of the testbed and is also configured as a HA. Each AP is connected to a different IPv6 subnet and AR. Figure 5.17 illustrates the testbed.

All network entities are running the GNU/Linux operating system, except for the APs which are 802.11b Cisco AP 1200 devices. The HA is running the new MIPv6 daemon for the GNU/Linux operating system (MIPL [143]). The MN and the ARs are running a modified version of the FMIPv6 Open Source Implementation Suite (fmipv6.org [66]). The modifications are related to our protocol specifics. In order to modify the behavior of the MN's wireless LAN device, we have used a 3com 802.11 a/b/g PCMCIA wireless card managed by the MADWiFi driver [44].

Concerning the geolocation system, the MN is equipped with a GPS device. We have chosen GPS for its ease of use and installation. Other systems with similar (or better) characteristics, such as [72], could also be used. As the experiments take place in an indoor environment, we first recorded GPS output generated by a moving MN in an open area. We then used the recorded GPS positions while performing the real experiments inside. The position and range of the APs are derived from indoor measurements conducted separately and are configured statically in all ARs.

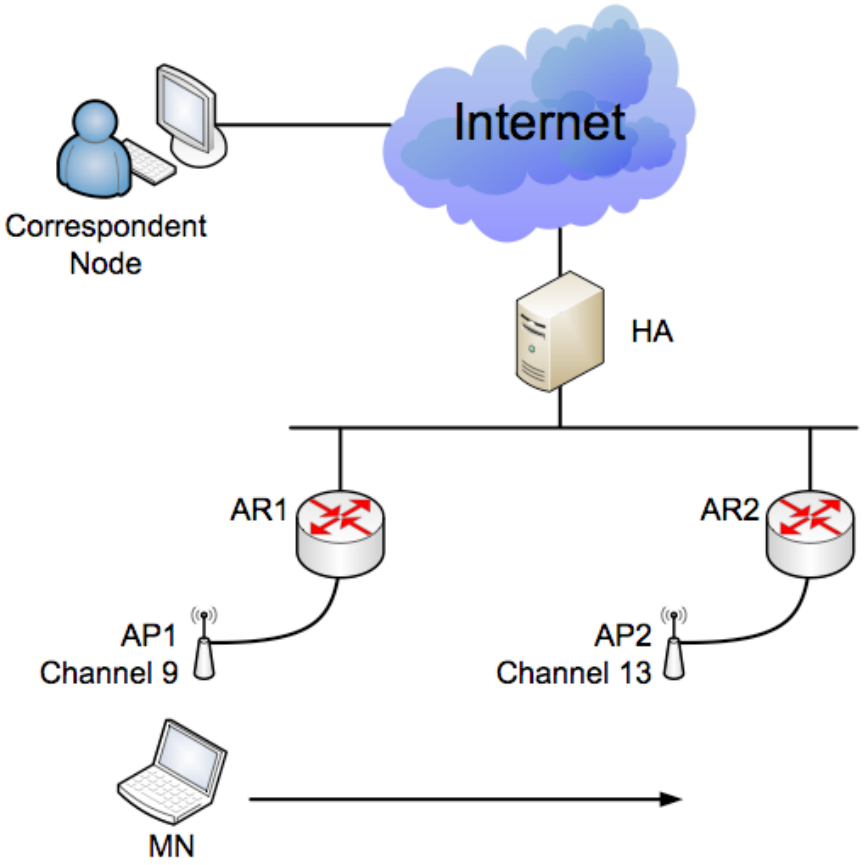


Figure 5.17: Testbed used in the geolocation experiments

Evaluation scenario

For the evaluation of our proposal we have defined the following scenario: the MN would move from AP1 to AP2 (see Figure 5.17). While the MN is moving, a correspondent node sends a video stream to it using the well known VideoLAN application [42]. Data is encapsulated and sent in Real-time Transport Protocol (RTP) [178] packets, with an average length of approximately 1336 bytes and sent every 30ms.

Evaluation results

Results presented in this section are obtained by running the network analyzer tool Wireshark [46] on the MN. Additional wireless sniffers are also used to collect the results. The evaluation scenario was run 10 times.

Figure 5.18 presents one significant run illustrating the entire handover procedure. When the signal quality of the MN is below the S_a threshold, it starts sending periodic RtSolPr messages to its AR in order to update its status and position. Upon reception of the fourth RtSolPr, the AR detects that the distance between the MN and its current AP is greater than the R threshold. Therefore, the AR calculates a new context and sends it to the MN using PrRtAdv. The reception of a new RtSolPr request (e.g. between seconds 9.5 and 12) does not initiate the calculation of a new context because the previous one remains valid until the actual handover occurs. Once the signal quality of the MN drops below S_w, the MN starts a predictive handover as described in the FMIPv6 specifications. The MN has thus obtained the parameters of the candidate APs (i.e. the context) without loosing a single data packet. Executing the standard scanning procedure [130] MN would have been scanning several 802.11 channels in order to discover candidate APs, which, as we have already mentioned , would have caused significant loss of data packets. In addition, regarding the time at which the scanning occurs, there is no guarantee that the set of candidate APs discovered during the last scan would still be pertinent (i.e. still in the radio coverage of the MN) when the MN actually beginsits handover. Furthermore, by sorting the list of candidate APs according to the length of the trajectory segment that they are expected to cover, our solution also reduces the number of performed handovers.

Figure 5.19 presents the complete handover procedure. Each dot represents the reception or the transmission of a packet by the MN at the time indicated on the X-axis. As previously mentioned, the MN moves while receiving a video stream (data packets are sent every 30ms). Once its signal quality drops below the S_w threshold, the MN starts the FMIPv6 procedure by sending a FBU to its current AR including the parameters of the NAR, i.e. the AR related to the first AP of the context. Then, it takes approximately 25.8ms for the current AR to get handover confirmation from the NAR (HI/HACK exchange) and to send the FBACK back to both MN and NAR. Upon reception of FBACK, the current AR starts tunneling data packets to the NAR while the MN starts the layer 2 handover procedure. As we can see, the MN completes the layer 2 handover in 20.4ms on average. This time is significantly reduced in comparison to the standard layer 2 handover latency which may vary between 58.7ms and 396.7ms according to the results described in [133]. Compared to the method defined in the IEEE 802.11 standard [17] in which MNs have to spend at least several hundred milliseconds scanning 802.11 channels to detect surrounding APs, our scheme allows the MN to directly send Probe

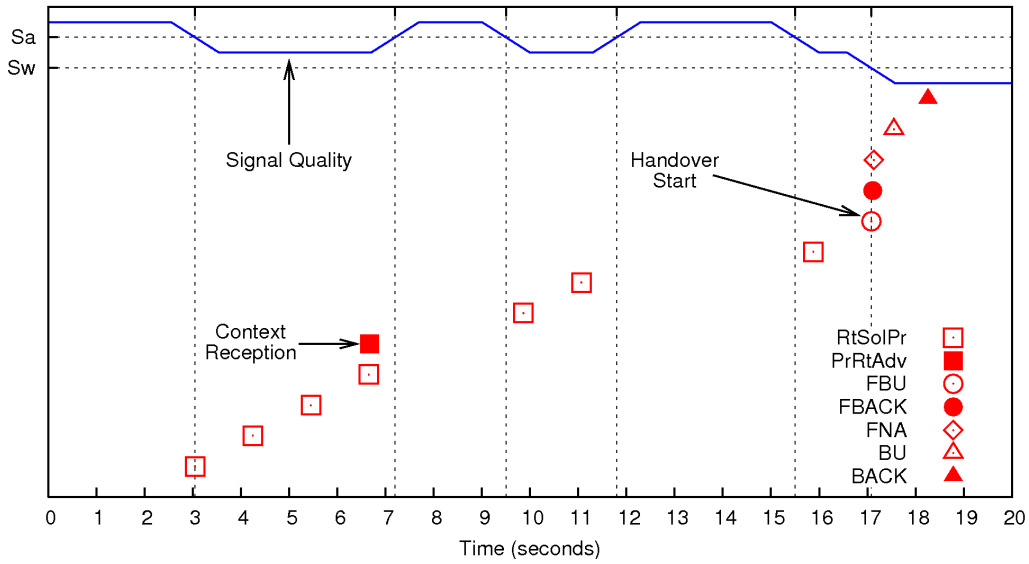


Figure 5.18: Candidate access points discovery followed by an FMIPv6 predictive handover

Requests over the radio channel of the pre-selected AP. When a Probe Response is received from the pre-selected AP, the MN proceeds with authentication and completes the layer 2 handover with the association stage. Do notice that the size of the NAR's buffer is limited and therefore such layer 2 optimization is necessary to avoid buffer overflow and packet loss. Simply augmenting the buffer size would not resolve this problem as real-time communications would still suffer the delays accumulated while packets were being in buffered.

Once the layer 2 handover is complete, the MN sends an FNA to the NAR. After receiving it, the NAR , delivers all buffered packets, and starts routing the NCoA. As we can see from Fig. 9, the NAR has only buffered two data packets during the entire procedure. Although there is a delay in the reception of buffered packets, this delay remains imperceptible to the application user and the handover is completely seamless with no packet lost and no perceptible delays.

While the MN does not update its binding with the HA, the data packets are still sent to the Previous CoA (PCoA) of the MN and are therefore forwarded through the FMIPv6 tunnel between the PAR and the NAR. The lifetime of this tunnel is specified by the MN at the transmission of the FBU. During our experiments, we configured the MN to request the minimal allowed tunnel lifetime, i.e. 4 seconds according to [70]. As shown in 5.19, the data packets are sent to PCoA until the time 310.6ms at which the MN sends a BU to the HA. After receiving it, the HA updates the tunnel endpoints with

the NCoA and sends back to the MN a BACK message with a successful status code which completes the MIPv6 signaling. Then, the following data packets are directly sent to the NCoA and thus do not use the FMIPv6 tunnel. 4 seconds later, the FMIPv6 tunnel between the PAR and NAR is removed.

Note that the procedure defined for the enhanced reactive mode (i.e. when the anticipation is not successful) is still being implemented and thus we can not yet provide results for this particular case.

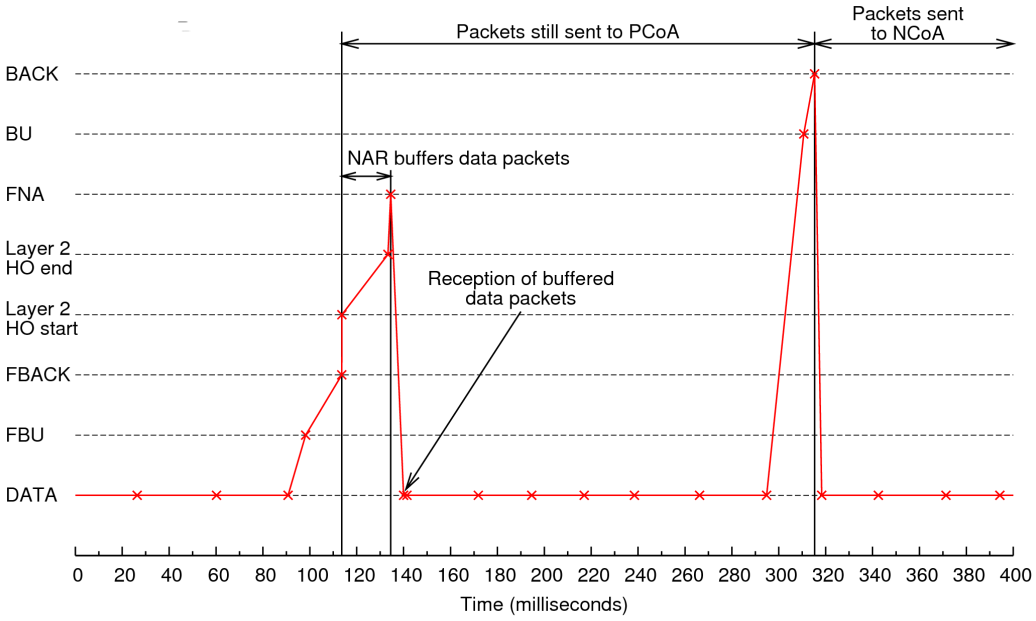


Figure 5.19: Impact of the Predictive FMIPv6 handovers on a video stream

5.4 Conclusion

The results presented in Section 5.3.5 have shown that our scheme allows the entire FMIPv6 procedure to remain imperceptible to users, from the discovery of candidate APs to the completion of the handover. Based on the trajectory of an MN access routers are able to select its next APs and provide the related parameters. The proposed method allows the MNs to avoid scanning prior to a handover and thus does not disturb communications. Moreover, use of our various thresholds on signal quality or geographical distance reduces the signaling overhead introduced by our additional mechanisms. The actual handover is also seamless due to the optimized layer 2 handover and the buffering

of data packets defined by the FMIPv6 specification. As a result, the quality of the video stream transmission is not affected by the movement of the MN.

Chapter 6

Concluding our work on mobility ... and moving on

In the previous sections we have presented our work on improving the handover procedure for Wi-Fi equipped mobile nodes, primarily in terms of rapidity, packet loss, and user experience. In Chapter 3 we have first presented our efforts to do so by only operating at the application layer of the OSI model. Our goal back then was to come up with a way for users that do not have control on the configuration of their network topology to still be able to benefit from acceptable handover performance. The SIP protocol defines a way for a node to re-initialize active communications sessions at any given time of a call. This mechanism is commonly used for adding new streams, modifying existing ones, or simple redirecting media flows to a new location. Knowing, however, that most SIP applications would react to movement only after detecting that the IP address has changed, SIP mobility often involves unacceptable delays that make it almost unusable in real-world conditions. We have therefore decided to evaluate the behavior of a SIP application where movement detection is also assisted by information obtained from lower layers (like for example a notification implying the end of the 802.11 association process). Such mechanisms are often referred to as link layer triggers. In the same Chapter 3 we have also described the results of our evaluation. They clearly show that it is possible to implement and deploy application layer mobility with acceptable quality without necessarily modifying the configuration of network topologies. Nevertheless, handover performance in all scenarios is far from being optimal, and users of such solutions would still experiences cuts and quality degradation during handovers.

Taking this into account, in chapter 4 we have shifted our attention to one of the most popular network layer mobility optimizations - the Fast Handovers for Mobile IPv6 protocol. Our work on the network layer has started with the first open source

implementation of this protocol - the fmipv6.org suite. Using it, we have been able to complete a comprehensive set of evaluation tests and determine that the protocol does indeed offer the possibility to completely eliminate packet loss and handover latency. However, being a generic layer three solution, the protocol does nothing to improve the Wi-Fi scanning process and as a result no improvement is visible to the user whatsoever. As indicated in Chapter 2 this comes from the fact that the IEEE 802.11 scanning is a blocking process that contrary to other solutions, devices are not capable of transmitting and/or receiving data on more than one channel at a time.

We have tried to come around this problem and implemented a procedure often referred to as a "soft handover" through the use of a secondary wireless interface. Following an implementation and an evaluation of this solution with the SimulX wireless simulator. The results that we obtained have shown that one could easily achieve completely seamless handovers through this technique, which is why we have tried to adapt the use of multiple interfaces to the standardized protocols existing at the time, or in other words, integrate it into our fmipv6.org implementation in order to resolve its scanning issues. After completing the implementation and the evaluation of this "combined" method, we have concluded that the use of multiple interfaces with the FMIPv6 protocol is indeed a solution that can be used to achieve 100% transparent handovers.

It is clear however, that few of today's mobile devices come equipped with two distinct Wi-Fi interfaces. Our next step has therefore been to work on a solution that allows mobile nodes aware of their geographic position to use it in retrieving a list of neighbor access points from the FMIPv6 access routers. The implementation and the experimental evaluation of this solution has shown that using positioning information could be quite beneficial when trying to improve the MIPv6 handover procedure.

6.0.1 Possible further improvement

Throughout the work that we describe in this part of the thesis, we have shown that implementation and experimental evaluations are something that we consider in important aspect of research in the field of IP mobility. Yet, adoption of any of the solutions proposed here would require completing even more tests in real-world scenarios. This is especially important for our geolocation proposal, which would require precise fine tuning depending on the deployment that it is being used for. Using it in order to provide Internet access for vehicles on a highway, for example, would require a completely different configuration from the one that would be used in a commercial center where users would primarily consist of pedestrians with erratic trajectories.

From an optimization point of view, we are convinced that we could improve the FMIPv6 protocol even further, and completely remove dependencies on an extra device (whether it is a GPS or a secondary Wi-Fi card). The MIMO technology for example, has made it possible for wireless interfaces to use two antennae. It might therefore be logical to try and use such a device rather than the two interfaces that we use in our multi interface version of FMIPv6.

Another possible direction that might be worth exploring is that of adapting the Cisco WDS solution (see Section 2.6.4) to FMIPv6. When studying WDS we noticed that most of the information that was being used by this solution was also exchanged between FMIPv6 mobile nodes and access routers. An FMIPv6 PAR, for example, is completely aware of the next point of attachment that a mobile node has moved to, just as a WDS access point is. During our evaluation of the WDS system we were impressed by its performance in link layer handovers, which makes us think that merging both technologies would probably be an idea worth working on.

6.0.2 Where to now ...

The worked that we have described in this part presents all our efforts in the field of IP mobility. We have shown numerous solutions that bring handover latency and packet loss down to an acceptable minimum. More importantly, we have implemented as many as three distinct optimizations and shown that they have quality which is completely satisfactory for industrial carriers. With some exceptions, we therefore consider further work in this field to belong to the world of the telecom industry and production implementations, and not really the research domain.

This is the reason why at that point we decided to focus on other problems that real-time communications are suffering today. An omnipresent issue that most Internet users are aware of, is that of how unreliable IP real-time communication are. Due to the Network Address Translation gateways, deployed everywhere today, it is almost impossible to use VoIP protocols the way they were originally defined. There is indeed a host of existing algorithms that describe various techniques for NAT and firewall traversal. However, very few of them are generic enough to work behind any NAT implementation and in most cases VoIP providers are obliged to relay most of the media that users exchange. This is obviously unacceptable since it brings to Internet communication the same constraints as those associated with conventional telephony providers such as high cost and difficulty of deployment.

Many expect IPv6 to resolve this issue, but we consider this to be an overly optimistic vision for several reasons. First of all, even though IPv6 is gaining more and

more popularity, IPv4 is quite unlikely to completely disappear in the following one or two decades. There are hundreds of millions of IPv4 devices, such as hardware phones, home gateways, set top boxes, and PCs equipped with older versions of popular operating systems, that have been already deployed and that users expect to be able to use for several more years. As a result, application developers of any non-research application would have to make sure it is IPv4 compatible for more than several years.

The situation is becoming even more complicated when trying the resolve the interconnection problems between IPv4 only and IPv6 only hosts where, once again, we are brought to relaying media through the network of the provider. Besides that, the fact that negotiating media sessions in popular signalling protocols such as SIP is not adapted to selecting between two distinct IP versions makes the problem valid even in cases where one of the nodes supports both IPv4 and IPv6.

All of the above, combined with the simple fact that we were interested in this domain, have made us continue our work in the field of Peer-to-Peer technologies and their applicability to real-time communications. With P2P one could hope to be able and implement NAT traversal algorithms and IPv4-IPv6 relaying while completely keeping a distributed network in accordance with the Internet “philosophy”.

Part II

Real-time communications using peer-to-peer overlays

Chapter 7

Introduction: Using peer-to-peer methods in real-time communications

During the last few years the use of Internet as a transport medium for real-time communications has become very popular. Thousands of providers are offering VoIP services and PSTN access, and the number of their users is constantly increasing. Yet, despite the migration toward the Internet we are still far from the liberty of usage and the prices that we see with other Internet technologies such as E-Mail and instant messaging. The services proposed by the new VoIP providers are not more or even much different than those that we have been proposed by conventional telecoms for the last twenty years. In this article we would first talk about VoIP today. We will describe what we believe to be the most common topologies and use cases that are deployed nowadays. We will then enumerate a set of problems associated with today's VoIP, such as NAT and firewall traversal, the high cost of the infrastructure necessary to deploy new services and others. Finally we explain why we believe that peer-to-peer overlays would help resolve these problems and why should devise ways for standard protocols to use and benefit from them.

Acknowledgement: Work described in this section has been completed in a joint effort with Telecom Italia Labs, Turin, represented by Enrico Marocco, and with Columbia University, NYC USA, represented by Dhruv Chopra and Henning Schulzrinne.

7.1 Common architectures

In this section we describe what we believe to be some of the most common ways that VoIP is used today. Depending on the users that the services are targetting we have decided to roughly split the various types of VoIP deployment into two different groups: relatively large scale providers targeting all users over the Internet (these are often referred to as hosted providers), and private installations meant for use by one specific company or organization.

We find this classification quite suitable for our needs since according to the users targeted by a VoIP deployment one would encounter support for different sets services and configurations.

It is probably worth mentioning that the same separation could be applied to deployments of other Internet technologies. With e-mail, for example, we have company mail servers on one hand and services like Hotmail, Yahoo!, and GMail on the other. We are also beginning to see the same situation with instant messaging where in addition to the large networks maintained by AOL, Microsoft, and Yahoo!, more and more companies are beginning to run their own instant messaging servers using Jabber, IRC or other protocols.

7.1.1 Hosted providers

Instant messaging networks such as AOL's ICQ, MSN and Yahoo! were among the first to offer VoIP near the end of 1990s. Back then the only service made available was one to one voice communication also referred to as voice chat. In almost all cases, establishing communication sessions was only possible between subscribers of the same network or a subscriber and the PSTN. Most of the above providers did not provide the possibility to communicate with users of other networks, but that was to be expected as almost all of them were using custom proprietary protocols without any possible interoperability.

A few years later, in the beginning of the 2000s, a new kind of providers began to emerge. That is when companies like Vonage, SIPphone and others started entering the scene. What was new about them was their use of standardized telephony protocols such as SIP. In addition to one-to-one communication, most of them were also offering common telephony services such as voicemail, conference bridges etc.. Still, despite the usage of a standard protocol the vast majority would continue working in a standalone manner without really trying to achieve, or even explicitly prohibiting interconnections.

Up to this date, users of different VoIP providers still need to cross the PSTN network in order to communicate with each other.

7.1.2 Private installations

Installing IP PBX servers in companies and organizations is arguably one of the first and the most widely deployed use cases for today's telephony protocols. After their appearance in the late 90s, many companies have adopted them instead of the more expensive conventional solutions.

With a typical private installation, an organization would install and maintain a VoIP server, or Application Layer Gateway (ALG) within its premises. All telephony equipment inside the company would then consist in hardware and software telephone clients. The VoIP server would typically have a single outward interface connecting it with the PSTN. The ALG is mainly used by clients within the company with some very rare exceptions.

In the rare occasions where conversations are routed over the Internet, it would rarely be to an arbitrary destination: In most cases inter-domain connections happen through preconfigured channels and are only allowed for a limited set of predefined locations. A widespread way of setting up such channels consists in configuring IAX connections between asterisk installations (e.g., between distant branches of one company).

7.2 Problems with existing topologies

In this section we try to present a number of problems characteristic of the deployments described in Section 7.1.

7.2.1 Difficult scalability and limited flexibility

Without a doubt one of the most recurring concerns associated with centralized systems is the relatively high cost that scalability comes for. Support for large numbers of users requires a heavy infrastructure, and large bandwidth which are both expensive and complex to setup and maintain.

NAT traversal issues

Some specific scalability problems are brought about by NAT traversal techniques. Deployment of Network Address Translation (NAT) devices has reached a point where almost all of the packages offered by internet service providers to home, as well as small and medium enterprise users rely heavily on it. There have been numerous reports explaining what kind of difficulties this represents for VoIP applications techniques to handle the issue. One of the implications coming with NAT, that remains despite the different traversal techniques is the fact that VoIP providers still need to install and maintain media relays capable of handling relatively high loads of traffic for clients that have been unable to establish direct connections. While this may be relatively trivial for existing operators, it still represents a significant entry barrier to the VoIP market. Any kind of media relays (regardless of the number of expected clients) require a dedicated infrastructure with guaranteed bandwidth which automatically excludes most of the existing adsl users for example and adds an expensive constraint to the establishment to any new VoIP based service.

Another consequence of common NAT traversal techniques is the increase of signalling traffic in the cases where signalling is based on UDP. In order to keep NAT bindings for signalling sockets alive, many solutions such as [37] increase traffic over them. Depending on the solution the increase might be with STUN, SIP OPTIONS, empty UDP or other messages but with either of these techniques, a standard registration exchange that occurs approximately once per hour [167] is augmented up to several hundred times. Needless to say this implies that a service provider would have to come up with even more bandwidth and resources in genereal.

7.2.2 Repetitive services

The fact that the infrastructure behind a service is expensive makes it obligatory to only use it for services that would be potentially interesting to a very large number of users (millions). As a result we keep seeing a relatively limited set of services such as voicemail, conference bridges, etc.. Most of them already existed with conventional telecom operators and even though some new ones such as the ring back tone have occurred lately we are still far from having the abundance typical for the Web for example.

Once again, the reason for this is primarily the fact that given their mastodon nature, telephony operators cannot address (at least not with reasonable prices) smaller communities with customized solutions based on professional needs or geographic location and offer services to them.

Let's take professional collaborative services as an example. The only collaborative tool that most of the small and medium enterprises have at their disposal today is phone or, generally speaking, voice, as well as mail, and instant messaging. Most of them would therefore tend to communicate in a rudimentary manner without any use of real-time collaborative services.

One of the reasons they don't use such services is because most of the time they don't exist, and when they do they often come at a price that is only affordable for big companies that can be regarded as isolated cases. Again this is because they represent a big community and they could justify the investment in the infrastructure necessary to provide customized collaborative tools.

7.2.3 Federation

Strangely enough, for a number of reasons, usage of SIP in particular and VoIP in general has so far failed to achieve the level of interoperability that we have with similar protocols like SMTP and XMPP. As mentioned in section 7.1 users subscribed for a particular VoIP provider are rarely able to make outgoing VoIP calls to other domains without first crossing the PSTN. This is valid for both hosted providers and (even more so) for the small private installations. The reason is most certainly not entirely technical because protocols like SIP use interdomain routing mechanisms very similar to those employed by SMTP. In section 7.2.1 we mentioned at least one reason that makes the VoIP market only accessible to enterprises with a structure similar to the traditional telecom operators, where the main service being offered, and billed to users, is the time spent communicating. In the case of VoIP, Internet conversations are often expected to be free of charge, but the "minutes" model applies to all conversations made through the PSTN. These calls are in most cases the major revenue source for the service providers which is in the least discouraging regarding an open interdomain call routing policy.

7.2.4 IP layer mobility

Work on IP mobility has been going on for a relatively long time today and there are already a number of existing solution that can be considered mature [103]. Still, apart from a number of research deployments, the only type of mobility we see in daily life today is the one provided by mobile phone operators with GPRS and 3G, with all the bandwidth and cost constraints that this implies. There are numerous reasons for this, and a major one is that almost all viable solutions require dedicated network entities

or system/kernel level modifications (i.e. home agents, mobility kernel patches, etc.) which are hard and costly to deploy.

7.3 The Peer-to-Peer Promise

In this section we address problems described in section 7.2 and describe ways of how each one of them could be resolved by P2P networks.

7.3.1 Scalability

One of the main advantages of peer to peer networks comes from the fact that resources are (in the common case) provided by the users themselves. This implies that a large number of users of a particular p2p based service would not automatically result in a need for the service provider to increase the capacity of a centralized infrastructure: the resources necessary to serve these users would be provided by the users themselves.

Apart from the cost of the infrastructure itself, this also decreases the expenses for maintenance and configuration that are necessary with centralized equipment.

Another important scalability advantage of P2P networks comes from the fact that they allow to gracefully handle usage peaks. In many cases when using real-time communications, service providers are required to handle usage peaks. Typical examples include morning hours - when many users get to work and turn on their VoIP clients for example, or during the hours after particular events such as New Year's Eve, where many users try to get in touch with each other at the same time.

It is particularly difficult to handle such scenarios in centralized environments as deploying the infrastructure necessary to resolve the issue is not feasible once it has been detected. P2P solutions on the other hand, could in theory help to gracefully handle such problems as, once again, the more users turn up, the more peers would be available, and the more they would provide resources.

NAT and firewall traversal

The primary advantage of using Peer-to-Peer overlays to assist NAT and firewalltraversal comes from the fact that it allows to reuse existing algorithms such as STUN [164], TURN [101], and ICE [163] in a distributed manner.

This is particularly advantageous for solutions based on TURN principle since relaying media from all conversations in a given network is often very expensive and a factor generally leading to severe quality degradation. One example of how this could be achieved is presented in [30]

7.3.2 Abundance of new, custom services

Generic Peer-to-Peer networks are expected to offer all the resources necessary for available users to benefit from all proposed services. This means that in order for a particular provider to offer a new service, all they would need to do before being able to propose it (if anything) would be to develop custom software if such is necessary to support the service. In other words, they would not have to also deploy their own infrastructure.

We expect this to instigate the provision of a large number of new services, such as real-time collaborative applications, targeting smaller groups of users based on profession or geographic location.

7.3.3 Federation

As mentioned in 7.3.2 we believe that P2P technologies would probably make the PSTN based business model obsolete and shift the accent to the diversity, quality, and number of other services being offered by a provider. We expect this to remove the incentive that providers currently have to try and funnel traffic over the PSTN.

The general opinion for P2P networks seems to be that the network operator of a particular overlay (if there is such an operator) would not necessarily be offering services to end users on that overlay. It is expected that one P2P overlay would probably host numerous service providers. Such providers are likely to strive to also make their services available outside of one particular domain which would be an extra incentive for network operators to open up and allow interdomain connectivity.

7.3.4 IP layer mobility

As we mentioned in Section 7.2.4, existing mobility solutions are relatively hard to deploy because of their low level. P2P overlays, on the other hand, are mostly application layer solutions, which gives them a great ease of deployment. We expect that generic

overlays would offer the possibility for devising and deploying new, higher level ways of handling mobility.

7.3.5 IPv4/IPv6 interoperability

P2P overlays could also help in the deployment of the new version of the Internet Protocol - IPv6. Just as peers in an overlay could be used for relaying media, they could also serve as IPv4 - IPv6 gateways. This would allow single stack, IPv4 users to access services provided on IPv6 only.

7.3.6 Sharing more than CPU and bandwidth

P2P networks today are commonly regarded as a means of sharing resources such as CPU, bandwidth, or data storage space. In the previous sections we have explained why we believe these to be very important features.

However, we believe that it is reasonable to expect that P2P overlays could also be very useful when sharing other resources. [123], for example, describes how a P2P technology could be used for sharing PSTN access.

Chapter 8

State of the art in the field of peer-to-peer overlays

8.1 Basic concepts of P2P architectures

Overlay networks provide an abstract view of a network environment. They are often designed for specific needs that do not require precise knowledge of the underlying infrastructure [87, 97]. In overlay architectures, a set of nodes (servers, services, end-user equipment etc.) and virtual links that do not directly match those of the underlying topology, are involved in specific applications. Data in such applications is routed according to these virtual links and an overlay network can therefore be viewed as a middle layer between them and the underlying topologies. Peer-to-peer overlays are a common implementation of overlay networks and as such they have received a lot of attention over the past several years [172].

Early work on P2P overlays (e.g., Gnutella [162] and GIA [43] has mainly focused on unstructured networks through the use of broadcasting as the primary means of propagation. Problems often associated with this approach include non-optimal resource consumption and lack of flexibility in policy management [121]. In order to address them .Solutions, such as CAN [86], Chord [186], and Pastry [170], have introduced the so called structured overlay networks. One of their key features is the fact they define and implement algorithms for self-organization and self-healing in specific network topologies which greatly facilitates navigation in the overlay. This allows for easy deployment of service components (e.g., content services, CDN - Content Distribution Networks [52]) in a flexible, scalable and decentralized way. Another key aspect of structured overlays is the fact that most of them employ distributed hash tables (DHT)

[57], which greatly optimize propagation of information through all clients. DHT based architectures have a flexibility that has made it possible to successfully implement them over grid networks [104] and not only Internet-like structures.

Overlay routing can also be used for obtaining better resilience, stronger security, or extra features like support for multicast. One such example is the usage of intermediate peers for improving firewall and NAT traversal when establishing end to end communication sessions between peers [171]. Such a technique is often used in VoIP peer-to-peer communication networks [13]. Another example is the usage of infrastructure overlay networks to enhance network routing. In these overlay networks, an overlay service provider places a number of overlay servers at strategic points in the network [24, 26] and these servers form a virtual network that can be used to provide alternative routes when errors occur in the network [23] and that can offer overlay multicasting support [25]. In addition to these components that are located in the network itself, these overlay networks can also contain proxy components at the edge of the network [122, 27]. These proxies are able to take both application and network information into account to perform high level services like trans-coding and bandwidth management.

Service discovery is another problem that occurs in several domains, especially web services. While the lookup service can be provided through a central UDDI server, new trends also suggest implementing it with peer-to-peer networks [73]. Overlay networks can also offer extra resilience. This has been shown in projects like RO [54] where the overlay network is used for the discovery of alternative routing paths. Other work in this domain includes [200], where paths are selected according to QoS metrics and overlay multicasting [199]. The rigid topology of structured overlay approaches, however, hardly fits the needs of modern networks with highly dynamic usage patterns. Therefore, some recent proposals suggest the adoption of overlays with adaptive topologies, in which the very structure of the network is determined at run-time, and based on the patterns of activity that occur inside it. Overlays could, for example, arrange nodes in way that places nodes with related patterns of data close to each other [55]; another option, suggested by [48] involves grouping reliable nodes into clusters and moving unreliable once towards the outside bounds of the network. What makes these techniques very interesting is their high adaptability, and the fact that they allow building robust and extremely scalable structures [96].

Adaptive structures together with the possibility of adopting gossip-based [20, 118] and probabilistic multicast methods [147] have recently brought research focus back to unstructured overlays based on random graphs [155]. Even though the lack of any deterministic structure is making search operations less efficient, the low diameter characteristic of such overlays could be successfully employed in the implementation of multicast/broadcast systems with low message propagation latency. Another important

property of these systems is their ability to provide a uniform peer sampling service [120]; this service can be leveraged to easily implement, very complex and large scale services like size estimation, publish/subscribe operations, or even structured topology bootstrap and management [119]. While providing powerful mechanisms to handle decentralized and self-organizing networks of services, P2P networks are very often designed for specific applications and their choice of architecture (e.g., structured vs. unstructured) strongly depends on the kinds of services that are meant to be supported by the overlay. GO, being aimed at using a P2P network as a generic substrate for general-purpose Telco services, requires a more flexible approach and the possibility of instantiating a "generic overlay" properly combining needed different solutions for different services. Foundational studies on overlay networks that have already been performed in the context of European projects such as DELIS and EVERGROW will be carefully analyzed.

It is now generally accepted that different applications may require different P2P overlay structures, and currently there are several proposals on instantiation and parameterization for specific overlays, as they are needed. One way to create multiple overlays is by inheritance, i.e., instances are generated from a parent virtual network by inheriting signalling protocols and communication services, as proposed in Genesis [39]. Overlays based on declarative logic, and that define their structure in a very compact and reusable manner, have been proposed in [36]. The proposed system, P2, is capable of directly parsing and executing specification from this language thus constructing and maintaining overlays. However, these works only focus on the issue of defining overlays, not on that (which is of interested to GO) to instantiating and integrating different overlays, and to make them dynamically parameterizable. A number of projects funded by the Global Computing II FP6 FET Initiative (e.g., SENSORIA, MOEBIUS and AEOLUS), focus on issues of formal theories and software engineering for large-scale distributed applications relying on global overlay network. In recent years we have also seen frameworks that define and instantiate parameterizable overlay networks, such as Opus [156], and JXTA [187]. Opus represents a large-scale overlay utility service that provides general abstractions and allows simultaneously hosting multiple distributed applications. It allocates individual nodes to competing applications and offers a high quality of services by using Service Level Agreements to dynamically distribute resources. The JXTA technology is a set of open, generalized peer-to-peer protocols that allows any connected device to communicate and collaborate. It has been introduced by Sun Microsystems and the high level vision it has been designed on, aims to increase interoperability among devices and networks. The IRIS project [5], supported by the US "National Science Foundation", proposes a framework that makes available a set of APIs abstracting the basic services of structured P2P overlays (i.e., key-based routing). On top of these a number of higher-level P2P service can be identified. Unfortunately,

these solutions do not efficiently address network and application dynamics, and mostly rely on rather static super-peer network structures [30].

8.2 Peer-to-peer Security

8.2.1 Introduction

Peer to Peer (P2P) overlays have become quite popular with the advent of file-sharing applications such as Napster [10], KaZaa [7] and Bittorrent [1]. After their success in file-sharing, P2P networks are now also being used for applications such as Voice over IP (VoIP) [13, 182] and television [6, 3]. However most of these systems are not purely P2P and have centralized components like the login server in Skype [30] or moderators and trackers in Bittorrent [153]. Securing pure P2P networks is therefore still a field of very active research. P2P overlays can be broadly classified as structured and unstructured. Unstructured overlays are often relatively simple but search operations in them tend to be inefficient. Structured P2P overlays use distributed hash tables (DHT) to perform directed searches which make lookups more efficient in locating data. Throughout this paper we will mostly focus on DHT-based P2P overlays.

Admission control is an important first step towards security [109]. It is important to restrict the number of malicious nodes in the overlay as most solutions rely on the assumption that they represent a small fraction of the overall number of peers. This also suggests the importance of identifying malicious nodes and disseminating this information. To understand the different attacks possible on a P2P system, it is important to know the motivation of the attackers and the resources (i.e. computation power, access to different IP subnets) that they would have to dispose with.

Other P2P specific security problems include attacks on the routing of queries, denial of service attacks and attacks on data integrity.

In this paper, after discussing main issues and proposed solutions for P2P systems in general, we focus on a particular application, real-time communication. The idea behind P2P real-time communication is using a DHT for services such as registration, location lookup, and NAT traversal assistance. We discuss how the above security aspects apply in this application and the solutions that seem appropriate.

The rest of this chapter is organized as follows. In section 8.2.2, we discuss admission control problems. In section 8.2.3, we identify the problem of *where* a node joins in the overlay. In section 8.2.4, we describe the problem of identification of malicious

nodes and the dissemination of this information. In section 8.2.5, we describe the issues of routing and data integrity in P2P networks. In section 8.2.6, we discuss the attackers. Specifically we discuss their motivation, the resources available to them, their victims and the attack timing. In section 8.2.7 we discuss how issues and solutions previously presented apply in P2P overlays for real-time communication, and in section 8.2.9 we conclude the paper and enumerate topics that would require future work.

8.2.2 Admission control

Admission control depends on who decides whether or not to admit a node and how this permission is granted. Kim et. al [109] answer these questions independently of any particular environment or application. They define two basic elements for admission in a peer group, a *group charter*, which is an electronic document that specifies the procedure of admission into the overlay, and a *group authority*, which is an entity that can certify group admission. A prospective member first gets a copy of the group charter, satisfies the requirements and approaches the group authority. The group authority then verifies the admission request and grants a group membership certificate.

The group charter and authority verification can be provided by a centralized certificate authority or a trusted third party, or it could be provided by the peers themselves (by voting). The former is more practical and tends to make the certification process simpler although it is in violation of the pure P2P model. The latter, the group authority could either be a fixed number of peers or it could be a dynamic number based on the total membership of the group. The authors argue that even if the group charter requires a prospective member to get votes from peers, the group membership certificate must be issued by a distinct entity. The reason for this is that voters need to accompany their votes with a certificate that proves their own membership. Possible signature schemes that could be used in voting such as plain digital signature, threshold signature and accountable subgroup multisignature are also described. Saxena et. al [176] performed experiments with the different signature schemes and suggest the use of plain signatures for groups of moderate size and where bandwidth is not a concern. For larger groups and where bandwidth is a concern, they suggest threshold signature [110] and multisignature schemes [144].

Another way of handling admission would be to use mechanisms based on trust and recommendation where each new applicant has to be known and vouched for by at least N existing members. The difficulties that such models represent include identity assertion and preventing bot/worm attacks. A compromised node could have a valid certificate, identifying a trustworthy peer and it would be difficult to detect this. Possible solutions include versions of CAPTCHA or sending simple logic puzzles [19].

8.2.3 Determining the position in the overlay

For ring based DHT overlays such as Chord [186], Kademlia [129] and Pastry [170], when a node joins the overlay, it uses a numeric identifier (ID) to determine its position in the ring. The positioning of a node determines what information it stores and which nodes it serves. To provide a degree of robustness, content and services are often replicated across multiple nodes. However it is possible for an adversary with sufficient resources to undermine the redundancy deployed in the overlay by representing multiple identities. Such an attack is called a sybil attack [63]. This makes the assignment of IDs very important. One possible scheme to tackle such attacks on the ID mapping is to have a temporal mechanism in which nodes need to re-join the network after some time [47, 177]. Such temporal solutions, however have the drawback that they increase the maintenance traffic and possibly deteriorate the efficiency of caching. Danezis et. al [59] suggest mechanisms to mitigate the effect of sybil attacks by reducing the amount of information received from malicious nodes. Their idea is to vary the nodes used for routing with time and thus avoid a trust bottleneck. Other solutions suggest making the joining process harder by introducing cryptographic puzzles as suggested by Rowaihy et. al [169]. The assumption is that the adversary has limited computational resources which may not be true if the adversary has control over a botnet. Another drawback of such methods is that non-malicious nodes would also have to perform the extra computations before they can join the overlay.

A possible heuristic to hamper sybil attacks is to employ redundancy at nodes with diametrically opposite IDs (in the DHT ID space) instead of successive IDs as in Chord. The idea behind choosing diametrically opposite nodes is based on the fact that a malicious peer can grant admission to others as its successor without them actually possessing the required IP address (whose hash is adjacent to the former's), and then they can cooperate to control access to that part of the ring. If however admission decisions and redundant content (for robustness), also involve nodes which are the furthest away (diametrically opposite) from a given position, then the adversary would require double resources (IP addresses) to attack. This happens because the adversary would need presence in the overlay at two independent positions in the ring.

It is also worth mentioning that in DHT overlays using different geometric concepts, (e.g., hypercubes instead of rings), peer positions are usually not related to identifiers. In the content addressable network (CAN) [159], for example, the position of an entering node may be either selected by the node itself, or, with little modification to the original algorithm, assigned by peers already in the overlay. However, even when malicious nodes do not know their position before joining, the overlay is still vulnerable to sybil attacks.

8.2.4 Identification and dissemination

Making overlays robust against even a small percentage of malicious nodes is difficult [41]. It is therefore important for other peers to identify such nodes and keep track of their number. There are two aspects to this problem. One is the identification itself and the second is the dissemination of this information amongst the peers. Different metrics need to be defined depending on the peer group for the former and reputation management systems are needed for the latter.

Identification

For identifying a node as malicious, malicious activity has to be observed first. This could be done in either a proactive way, or a reactive way.

- Proactive identification - When acting proactively, peers perform periodic operations with the purpose of detecting malicious activity. A malicious node could prevent access to content it is responsible for (e.g., by claiming the object doesn't exist), or return references to content that does not match the original queries [183]. With this approach, publishers of content can later perform lookups for it at periodic intervals and verify the integrity of whatever is returned. Any inconsistencies could then be interpreted as malicious activity.
- Reactive identification - In a reactive strategy, the peers perform normal operations and if they happen to detect some malicious activity, then they can label the responsible node as malicious. In a file-sharing application for example, after downloading content from a node, if the peer observes that data does not match its original query it can identify the corresponding node as malicious. Poon et. al [150] suggest a strategy based on the forwarding of queries. If routing is done in an iterative way, then dropping of packets, forwarding to an incorrect node and delay in forwarding arouse suspicion and the corresponding peer is identified as malicious.

Reputation management systems

Reputation management systems are used to allow peers to share information about other peers based on their own experience and thus help in making better judgments. Most reputation management systems proposed in the literature [194, 58, 112, 105] are for file-sharing applications. In reputation systems, it should not be possible for a

misbehaving peer with low reputation to simply rejoin the network with a different ID and therefore start from a clean slate. To counter this, Kwon et. al [112] store not only the reputation of a peer but also the reputation of files based on file name and content to avoid spreading of a bad file. Another method is to make the reputation of a new peer the minimum possible [105]. Kamvar et. al [105] define five design considerations for reputation management systems;

- Self policing.
- Anonymity.
- No profit to new comers.
- Minimal overhead.
- Robustness to malicious peers.

- Unstructured reputation management - Unstructured reputation management systems have been proposed by Aydin et. al [194] and Milano et. al [58]. The basic idea of these is that each peer maintains information about its own experience with other peers and resources, and shares it with others on demand. In the system proposed by Aydin et. al [194], each node maintains trust and distrust vectors for every other node that it has interacted with. When reputation information about a peer is required, a node first checks its local database, and if insufficient information is present, it sends a query to its neighbors just as it would when looking up content.
- Structured reputation management - One of the problems with unstructured reputation management systems is that they either take the feedback from few peers, or if they do from all, then they incur large traffic overhead. Systems such as those proposed by [112, 105] try to resolve it in a structured manner. The idea of the eigen trust algorithm [105] for example, is transitivity of trust. If a node trusts peer X then it would also trust the feedback it gives about other peers. A node builds such information in an iterative way. The algorithm has fast convergence properties [81]. For maintaining this information in a structured way, the authors use a content addressable network (CAN) DHT [159]. The information of each peer is stored and replicated on different peers to provide robustness against malicious nodes. They also suggest favoring peers probabilistically with high trust values instead of doing it deterministically, to allow new peers to slowly develop a reputation. Eventually, they suggest the use of incentives for peers with high reputation values.

8.2.5 Integrity in P2P networks

Preserving integrity of routing and data, or, in other words, preventing peers from returning corrupt responses to queries and routing through malicious peers, is an important security issue in P2P networks. The data stored on a P2P overlay depends on the applications that are using it. For file-sharing, this data would be the files themselves, their location, and owner information. For real-time communication, this would include user location bindings and other routing information. We describe such data integrity issues separately in Section 7.

Data integrity

For file-sharing applications, pollution with corrupt or bogus files is a problem. BitTorrent uses voluntary moderators to weed out bogus files and the SHA-1 algorithm to determine the hash of each piece of a file to allow verification of integrity. If a peer detects a bad chunk, it can download that chunk from another peer. With this strategy, different peers download different pieces of a file before the original peer disappears from the network. However, if a malicious peer modifies the pieces that are only available on it and the original peer disappears, then the object distribution will fail [201]. An analysis of Bittorrent in terms of integrity and performance can be found in the work of Pouwelse et. al [153].

Routing integrity

To enhance the integrity of routing, it is important to reduce the number of queries forwarded to malicious nodes. Marti et. al [128] developed a system that uses social network information to route queries over trusted nodes. Their algorithm uses trusted nodes to forward queries (if one exists and is closer to the required ID in the ID space). Otherwise they use the regular Chord routing table to forward queries. While their results indicate good average performance, it can not guarantee $\log N$ hops for all cases. Danezis et. al [59] suggest a method for routing in the presence of a large number of sybil nodes. Their method is to ensure that a peer queries a diverse set of nodes and does not place too much trust in a node. Both the above works have been described based on Chord. However, unlike Chord, in DHTs like Pastry and Kademlia there is flexibility in selecting nodes for any row in a peer's routing table. Potentially many nodes have a common ID prefix of a given length and are candidates for routing a given query. To exploit the social network information and still guarantee $\log N$ hops, a peer should

select its friends to route a query, but only when they are present in the appropriate row selected by the DHT algorithm.

8.2.6 The attackers

Incentive of the attacker

Attacks on networks happen for a variety of reasons such as monetary gain, personal enmity or even for fame in the hacker community. There are quite a few well known cases of denial of service attacks for extortion in the client-server model [131]. One of the salient points of the P2P model is that the services it provides have higher robustness against failure. However, such attacks are still possible against individuals within the overlay if the attackers possess sufficient resources. For instance a botnet of malicious nodes could simultaneously bombard lookup queries for a particular ID in the DHT. The peer responsible for this ID would then come under a lot of load and could crash [183]. However with replication of key-value pairs at multiple locations, such threats can be mitigated.

Attackers may also have other incentives apart from money. With the growth of illegal usage of sharing files with copyrights, record companies have been known to attempt polluting content in the overlays by putting up nodes with corrupt chunks of data but with correct file names to degrade the service [114] and in hope that users would get frustrated and stop using the service. Attacks can also be launched by novice attackers who are there attacking the overlay for fun or fame in a community. These are perhaps less likely to be successful or cause damage, since their resources tend to be relatively limited.

Resources available to the attacker

Resource constraints play an important role in determining the nature of the attack. An attacker who controls a botnet can use an Internet relay channel and launch distributed denial of service attacks against another node. With respect to the sybil attack, IP addresses are also an important resource for the attacker since in DHTs such as Chord [186], the ID is determined by using a base hash function such as SHA-1 [16] on the node's IP address. The cryptographic puzzles [169] that are sometimes suggested as a way to deter sybil attacks by making the join process harder are futile against an attacker with a botnet and virtually unlimited computation power. Doucer [63] proves that even

with the assumption that attackers only have minimum resources at their disposal, it is not possible to defend against them in a pure P2P system.

Victim of the attack

The victim of an attack could be an individual node, a particular content or the entire overlay service. If malicious nodes are strategically placed in the overlay, they can block a node from using its services. Attacks could also be launched against specific content [183] or even the entire overlay service. For example, if the malicious nodes are randomly placed in the overlay and drop packets or upload malcontent, then the quality of the overlay would deteriorate.

Time of attack

A malicious node could start misbehaving as soon as it enters the overlay or it could follow the rules of the overlay for a finite amount of time and then attack. The latter could prove to be more harmful if the overlay design suggests accumulating trust in peers based on the amount of time they have been present and/or not misbehaving. In Kademlia, for instance [129], the routing tables are populated with nodes that have been *up* for a certain amount of time. While this provides some robustness from attacks in which the malicious nodes start dropping routing requests from the moment they enter, it would take time for the algorithm to adapt to nodes which start misbehaving in a later stage (i.e., after they have been recorded in routing tables). Similarly for reputation management systems, it is important that they adapt to the current behavior of a peer.

8.2.7 P2P in Real-Time Communication

The idea of using P2P in real-time communication boils down to distributing centralized entities from conventional architectures over peer-to-peer overlays and thus reducing the costs of deployment and increasing reliability of the different services. Initiatives such as the P2PSIP working group in IETF [12] are working on achieving this by using a DHT for services such as registration, location lookup, and support for NAT traversal, which are normally performed by dedicated servers. Solutions that currently emerge in the working group try to achieve such a distribution adopting three different approaches: P2PP [31] and XPP-PCAN [127] use the overlay only for storing and retrieving user locations (or location of the peers acting as proxies for them), RELOAD [35] and HIP-HOP [49] route SIP signaling over the mesh of connections between peers, while in

ASP [102] peers tunnel both SIP and media flows in end-to-end encrypted connections established with ICE [163] and often relayed by other peers.

Even if based on the same technology, overlays used for real-time communication differ from those used for file sharing in at least two aspects:

- Resource consumption. Contrary to file sharing systems where the DHT is used to store huge amounts of data (even if the distributed database is used only for storing file locations, each user usually indexes hundreds or thousands of files), real-time communication overlays only require a subset of the resources available at any given time as users only register a limited number of locations (rarely more than one).
- Confidentiality. While in file sharing applications, where shared files are supposed to be made publicly available, eavesdropping and identity theft do not constitute real threats, in real-time communication, since exchanges of data are usually meant to happen privately, it is essential to have mechanisms to assert identities and to guarantee confidentiality.

In this section we go over the admission issues, and security problems discussed in previous sections, and discuss solutions that would be applicable to real-time communication in P2P.

Admission

In order to keep as much compatibility with existing user agents as possible, nodes in P2P communication architectures would probably have to participate as either peers or clients. If a node participates as a client, then it would use the overlay network by simply attaching to a peer or a proxy instead of registering with a server. In most cases users would be able to benefit from the overlay by only acting as clients. However, in order to keep the solution scalable, at some point clients would have to be promoted to peers (admission to the DHT). This requires addressing the following issues:

Active vs. passive promotions

Most existing P2P networks [7, 1, 6], would generally make it the responsibility of clients to determine if and when they would apply for becoming peers. A well known exception to this trend is the Skype network [13], arguably one of the most popular

overlay networks used for real-time communications today. Instances of the Skype application are supposed to operate as either super-nodes or ordinary-nodes and the "promotions" are decided by the higher levels of the hierarchy [30]. Even if there is not much difference for a client whether it has to actively ask for authorization to join an overlay, or passively wait for an invitation, the latter approach has some advantages which fit well in overlays where only a subset of the peers is required to provide the service (as in real-time communication):

- An attacker cannot estimate in advance when and if it would be invited to join the overlay as a peer.
- Allows peers to perform long-lasting measurements on sets of candidates, in order to accurately select the most appropriate for upgrading and only invite it when they are "ready" to do so. The opposite approach, that is when clients initiate the join themselves, adds an extra constraint for the peer that has to act upon the request since it doesn't know if and when the peer would attempt to join again.
- Discourages malicious peers from attempting sybil and, more generally, brute force attacks, as only a small ratio of clients has chances to join the overlay (possibly after an extensive examination).

When to upgrade

In order to answer this question one would have to define some criteria that would allow to determine the load on a peer and a reasonable threshold. When the load exceeds this threshold, a client is invited to become a peer and share the load. The criteria for determining load can be:

- Number of clients attached.
- Bandwidth usage for DHT maintenance, forwarding requests and responses to and from peers and from the attached clients.
- Memory usage for DHT routing table, DHT neighborhood table, application specific data and information about the attached clients.

Which clients to upgrade

Selecting which clients to upgrade would require defining and keeping track of new metrics. The exact set of metrics and how they influence decisions should be the sub-

ject of serious analysis and experimentation. These could be based on the following observations:

- Uptime. A peer could easily record the amount of time that it has been maintaining a connection with a client and take it into account when trying to determine whether or not to upgrade it.
- Level of activity. It is reasonable to assume that the more a client uses the service (e.g. making phone calls), the less they would be willing to degrade it.
- Keeping track of history. Peers could record history of the clients they invite and the way they contribute to the overlay.

Other metrics such as public vs. private IP addresses, computation power, and bandwidth should also be taken into account even though they do not necessarily have a direct impact on security.

Incentives for clients

Clients need to have incentives for accepting upgrades in order to prevent excessive burden on existing peers. One way to handle this would be to maintain separate incentive management through the use of currency or credits. Another option would involve embedding these incentives inside the protocol itself:

- Peers share with clients only a fraction of their bandwidth (uplink and downlink). This would result in higher latency when using the services of the overlay as a client and better service quality for peers.
- Peers could restrict the number or types of calls that they allow clients to make.

Introducing such incentives, however, may turn out to be somewhat risky. Differences in quality would probably be perceptible for end users who would not always be able to understand the difference between the roles that their user agent is playing in the overlay. Such behavior may therefore be interpreted as arbitrary and make the service look unreliable.

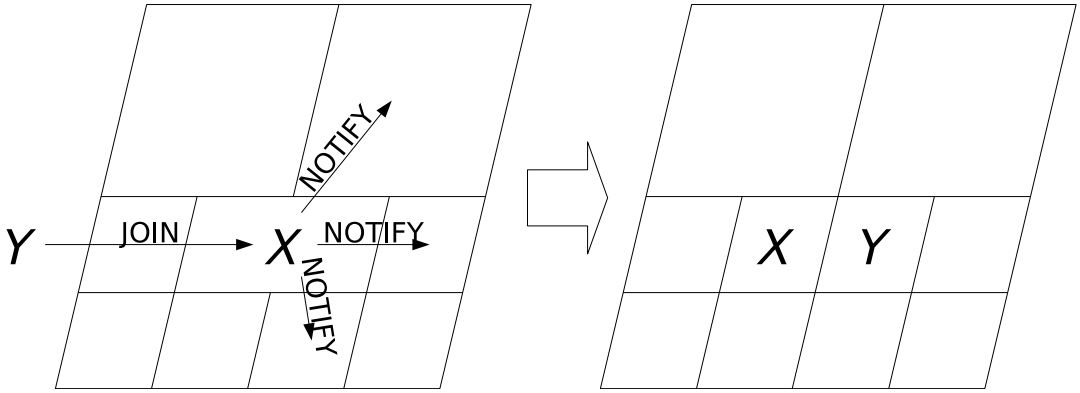


Figure 8.1: Node join in an XPP-PCAN [127] overlay: neighbors of the new peer are notified by the admitting peer.

8.2.8 Security

Denial of Service

In addition to bombardment with queries as described in section 8.2.6, the denial of service attack against an individual node can be conducted in DHTs used for real-time communications if the peers which surround a particular ID are compromised. These peers which act as proxy servers for the victim, can fake the responses from the victim by sending fictitious error messages back to peers trying to establish a session. Danezis et al's [59] solution can also provide protection against such attacks as in their solution peers vary the nodes used in queries.

Man in the middle attack

The man in the middle attack described by Seedorf [180] is an attack that exploits the lack of integrity when routing information in P2PSIP. A malicious node could return IP addresses of other malicious nodes when queried for a particular ID. The requesting peer would then establish a session with a second malicious node which would again return a "poisoned" reply. This could go on until the TTL expires and the requester gives up the "wild goose chase" [59]. A simple way for entities to verify the correctness of the routing lookup is to employ iterative routing and to check the node-ID of every routing hop that it is returned and it should get closer to the desired ID with every hop. However, this is not a strong check and can be defeated [180].

Trust between peers

The effect of malicious peers could be mitigated by introducing the concept of trust within an overlay. This can be done in different ways:

- Using certificates assigned by an external authority. The drawback with this approach is that it requires a centralized element.
- Using certificates reciprocally signed by peers. This mechanism is quite similar to PGP [202]; every peer signs certificates of "friend" peers and trusts any other peer with a certificate signed by one of its friends. However even though it might be theoretically possible, in reality it is extremely difficult to obtain long enough trust chains.
- Spreading the information of each trusted peer to its future neighbors, as shown in Figure 8.1. This approach, described in [127], works well with some DHTs like CAN, when it is possible to base the trust on some sort of mutual relationship (e.g., neighborhood in CAN [159]).

Routing call signaling

One way for implementing real-time communication overlays (as we have mentioned in earlier sections) would be to simply replace centralized entities in signalling protocols like SIP [167] with distributed services. In some cases this might imply reusing existing protocol mechanisms for routing signalling messages. In the case of SIP this would imply regarding peers as SIP proxies. However the design of SIP supposes that such proxies are trusted, and makes it possible for them to fork requests or change their destination, add or remove header fields, act as the remote party, and generally manipulate message content and semantics

However, in a P2P environment where messages may be routed through numerous successive peers, some of which might be compromised, it is important not to treat them as trusted proxies. One way to limit what peers can do is by protecting signalling with some kind of end-to-end encryption, as proposed in ASP [102].

Another option would be to extend existing signalling protocols and modify the way they route messages in order to guarantee secure end-to-end transmission. Gurbani et al. define a similar mechanism for SIP called SIPSEC [77]. It allows nodes to establish a secure channel by sending a CONNECT SIP request, and then tunnel all SIP messages through it.

Integrity of location bindings

It is important to ensure that the (URI, IP) pair that a node registers with is what is returned to a requester. Or the entities that issue the lookup request must be able to verify the integrity of this pair. A pure P2P approach to allow verification of the integrity of location binding information is presented in [181]. The idea is for an entity to choose an asymmetric key pair and hash its public key to generate its URI. The entity then signs its present location with its private key and registers with the quadruple (URI, IP, signature, public key). Any entity which looks up for the URI and receives such a quadruple can then verify its integrity by using the public key and the certificate. Another possible merit of such an approach could be that it is possible to identify the malicious nodes and maintain a black list. However, the resulting SIP URIs are not easy to remember and associate with entities. Discovering these URIs and associating them with entities would therefore require some sort of a directory service. The authors suggest using existing authentication infrastructure for this such as a certified web service using SSL which can publish an "online phone book" mapping users to SIP-URIs.

Encrypting content

Using P2P overlays for real-time communication implies that content is likely to traverse numerous intermediate peers before reaching its destination. A typical example could be the use of peers as media relays as a way of traversing NATs in VoIP calls.

Contrary to publicly shared files, communication sessions are in most cases expected to be private. It is therefore very important to make sure that no media leaves the client application without being encrypted and securely transported through a protocol like SRTP [32].

Other issues

Identifying more specific threats related to the P2P real-time communications, would require a clearly defined economic model. Answers to the following questions would be helpful.

- To whom do the users pay?
- Do the users only pay for services across the PSTN gateway?
- Is the billing done per call or is it fixed?

For instance, the implications of an attack such as taking control over another's user agent or its identity and using it for outbound calls would depend on whether or not this would be economically advantageous for the attacker. Baumann et. al [33] suggests that to prevent unwanted communication costs, the PSTN gateway should only be accessible via an authenticated server and dialing authorizations should be enforced. Also it seems that it would be difficult to do billing in a pure P2P manner as it would mean keeping the billing details with untrusted peers.

8.2.9 Conclusion and future work

We have discussed problems in peer to peer security and some solutions at the state of the art, mentioning the suitability and drawbacks of the different schemes. We looked at security keeping the attackers into perspective, their motivation, their restrictions and their targets. Existing security solutions do not seem mature enough to be deployed in pure peer to peer networks. Specifically secure ID assignment and entity-identity association seem the most challenging of problems. Future work on the subject would therefore need to analyze emerging implementations and the way the suggested security solutions perform with them.

Throughout the document we have been insisting that in P2P overlays both signaling and content would have to be encrypted in an end-to-end manner. Further research on the topic would have to investigate possible ways to distribute/exchange the keys necessary for such encryption.

Chapter 9

The Extensible Peer Protocol

9.1 Introduction

In this chapter we define the Extensible Peer Protocol (XPP), which provides end-to-end delivery services for data among peers in distributed overlay networks. The work described in here has been completed in the context of a collaboration between the Louis Pasteur University and Telecom Italia Labs, Turin, represented by Enrico Marocco.

Given the popularity and wide use of firewalls and NATs in most existing network configurations, one of the main goals of this protocol is to provide support for them. XPP is therefore using UDP as a transport protocol following guidelines provided in [71], and defines a way for sessions to be simultaneously initiated by both endpoints in pretty much the same way that standard media sessions are negotiated with SIP [167] or XMPP [174]. This makes possible the establishment of direct connections between peers even if both endpoints are located behind RFC 4787 [22] compliant NATs.

The semantics that XPP uses for session initiation and their resemblance with standard call negotiation allow the use of tools like ICE [163] and STUN [168] that further facilitate session establishment.

We also define rudimentary mechanisms for fragmentation and reliability. They are, however, not well suited for large amounts of data and may require further work like for example the definition of ACK rolling windows.

XPP is a simple protocol designed in a way that makes it easy to implement and extend; it is explicitly meant to be used as the P2PSIP peer protocol described in [56].

Acknowledgement: Work described in this chapter has been completed in a joint effort with Telecom Italia Labs, Turin, represented by Enrico Marocco. The XPP protocol specification has also been contributed to the P2PSIP working group of the Internet Engineering Task Force.

9.1.1 Why UDP

The main reason for the choice of UDP as a transport protocol is related to NAT traversal. When two peers behind different NAT devices want to establish a connection and exchange data flows, they have to start sending packets simultaneously, as opposed to waiting for one of the peers to initiate the session. This way, when their NATs receive such packets, they would eventually match them to previous outgoing packets belonging to the same session and forward them to the corresponding peer.

It is true that the definition of three-way TCP [152] handshake, also provides semantics that could be used for simultaneous connection establishment; however, this mechanism is defined for resolving race conditions and is not meant for use as a common practice.

In fact, Berkeley sockets, the standard interface applications use to access network functionalities exposed by the operating system, are designed around a client/server model and do not natively allow to initiate connections simultaneously.

Furthermore, other than requiring a correct implementation of the full TCP state machine on both endpoints, for the simultaneous establishment to succeed it is necessary that all traversed network elements (especially NATs and firewalls) are compliant with the best practices specified in [76]. Such requirements, at the time of writing, are certainly much less likely to be satisfied by common devices than those necessary for UDP NAT traversal [22].

Another weakness of TCP is its inability to handle address changes within established connections. While in normal environments it is possible to handle mobility at the network layer, in some scenarios specifically addressed by P2PSIP maintaining the same IP address after a handover or a NAT reboot is often not an option.

Using TCP would thus make mandatory the usage of ICE-TCP [100], at least for handling of simultaneous session establishment and mobility. Given the higher level of complexity inherent to ICE-TCP compared to ICE Lite and even standard ICE, using it would make XPP a lot more difficult to implement.

On the other hand, using UDP as the transport protocol would also give us the possibility to "switch off" reliability if necessary. This is sometimes necessary when using DHT algorithms based on frequent optional routing table updates.

9.1.2 Relation with other Proposals

Since we started specifying and implementing XPP there have been two other proposals for peer protocols: dSIP [51] and P2PP [29].

While dSIP – a textual protocol substantially based on SIP – is pretty different, P2PP has many things in common with XPP; mainly, it is binary and uses a very similar encoding for parameters based on type-length-value (TLV) fields. However, it misses a mechanism for simultaneously initiating sessions, which is one of XPP's most important features.

9.2 Terminology

- XPP Session : a logical relationship between two peers required for transmitting XPP Messages.
- XPP Message : either an XPP Operation Request or a XPP Operation Response. An XPP Message can be transmitted as one or more XPP Fragments.
- XPP Fragment: a segment or a whole of an XPP Message not exceeding the path maximum transmission unit (MTU).
- XPP Operation : a logical relationship between an XPP Operation Request and zero or more XPP Operation Responses.
- XPP Operation Request : an XPP Message requesting the execution of a given operation.
- XPP Operation Response : an XPP Message reporting the result of an operation.

9.3 Overview

9.3.1 XPP Sessions

XPP sessions are logical end-to-end relationships between pairs of peers. A session is identified by a pair of tags univocally generated by the peers and encoded in Local ID and Remote ID fields of each fragment.

For example, an XPP session between Alice and Bob, where Alice's generated tag is LA and Bob's generated tag is LB, will be identified by the pair <LA, LB> by Alice and by <LB, LA> by Bob.

Peers must discard fragments with values in Local ID and Remote ID fields not matching an active session, correctly established as described in Section 9.4.1.

It is important to note that received fragments will have inverted Local ID and Remote ID depending on whether they are sent by one side or the other.

9.3.2 XPP Operations

An XPP operation contains one operation request and zero or more operation responses. Requests and responses of the same operation must be sent in opposite directions. That is, if one side has sent the operation request, it cannot send responses in the same operation. Similarly, if a side has received a request it can only send responses for the corresponding operation and any new request must be sent in a new operation.

Every XPP operation has a sequence number and an operation type that serve as a way to identify and order operations. The operation number and type for all operation responses must match those of the corresponding request.

We do not define any specific operation type. Such types are to be defined by extensions of the XPP protocol according to their needs.

The lifetime of an operation (i.e. the amount of time that the sender of a request must keep the context associated with it) is also left to documents extending this specification as it may vary according to the operation type and the purpose it serves.

9.3.3 Requests and Responses

Requests and responses can be distinguished by the value of the Operation Type field. Senders must set this value to the to 0x00000000 for all outgoing responses. Operation type values for requests must be registered with IANA.

Requests and responses may have any number of parameters as specified in extension documents.

9.4 Use Cases

9.4.1 Session Establishment

Currently XPP only supports a single mode for session initiation which we refer to as simultaneous session establishment. Prior to the establishment, all parameters of the session need to be negotiated using external rendezvous and negotiation mechanisms such as those provided by SIP [167] and SDP [79] as defined in RFC 3264 [165].

9.4.2 A Sample XPP Operation Scenario

XPP Operations are initiated by an Operation Request that can be followed by an arbitrary number of responses. In the scenario presented in Figure 9.2 Alice is the sender of the operation request. Bob ACKs receipt of the request as soon as he gets it. At some point Bob would send an Operation response that in this case would get lost before reaching Alice. According to the XPP retransmission mechanisms described in Section 9.6.2, Bob would resend the response upon expiration of the corresponding timer and Alice would acknowledge reception as soon as the response has reached her.

9.4.3 Message fragmentation

When storing data in an overlay, or when simply exchanging information on neighboring zones, P2P applications are likely to have to exchange data chunks exceeding the path MTU. XPP therefore defines mechanisms for fragmentation that allow sending long XPP messages over multiple XPP fragments (see Section 9.6.1).

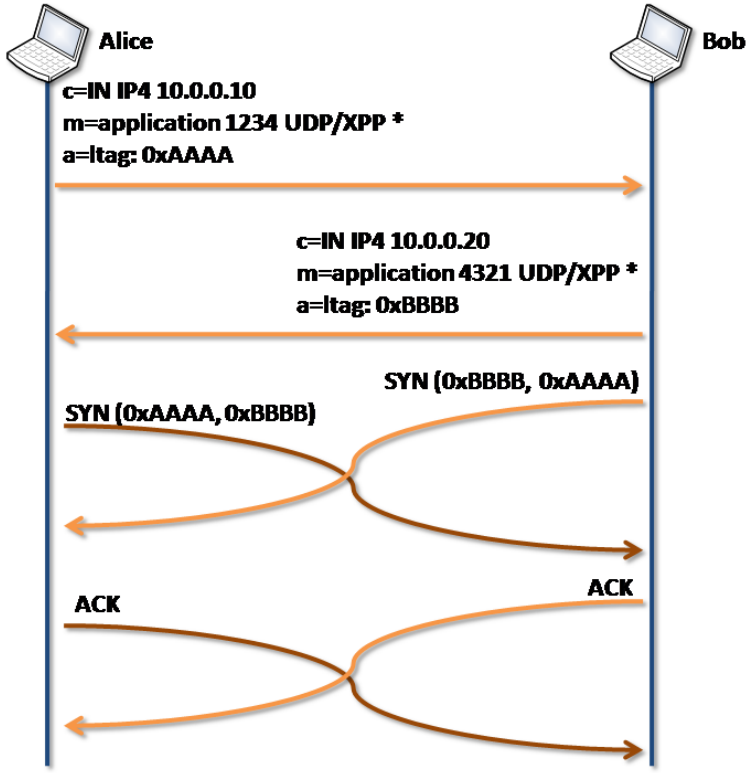


Figure 9.1: Simultaneous establishment of an XPP Session.
UDP packets are sent between endpoints 10.0.0.10:1234 and 10.0.0.20:4321. Session is identified by ID pairs (0xAAAA, 0xBBBB) and (0xBBBB, 0xAAAA) on Alice and Bob respectively.

Figure 9.3 describes a scenario where Alice is sending to Bob an XPP operation request with a size greater than that of the path MTU.

9.5 Protocol Details

All XPP messages are encoded using binary fields. All integer fields are carried in network byte order, that is, most significant byte (octet) first. This byte order is commonly known as big-endian. The transmission order is described in detail in Appendix B of RFC 791 [151].

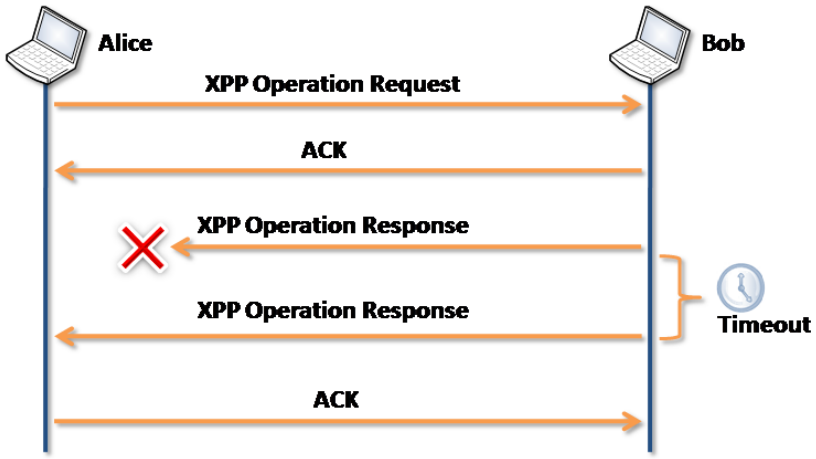


Figure 9.2: A sample XPP operation.
Alice sends an Operation request to Bob. Bob confirms reception and later sends a response. The response does not reach Alice, so Bob retransmits until the corresponding ACK is received.

9.5.1 XPP Fragment Header

All XPP messages start with an 8 byte message header represented on the following figure (Figure 9.4):

Fields:

- Ver : 3-bit XPP version number = 1.
- Reserved : Flags reserved for use by future versions or extensions of the protocol. must be set to zero by the sender and ignored by the receiver.
- SYN : Session synchronization flag. Set to 1 if this is the first message in a session and 0 otherwise.
- ACK : Fragment acknowledgment flag. Set to 1 if this message is sent to acknowledge receipt of a previously sent fragment or 0 otherwise.
- FIN : Session close flag. Set to 1 if no more non-ACK messages will be sent in this direction.
- REL : Fragment reliable flag. Set to 1 if the remote party is to send an acknowledgment upon receipt of this fragment and 0 otherwise.

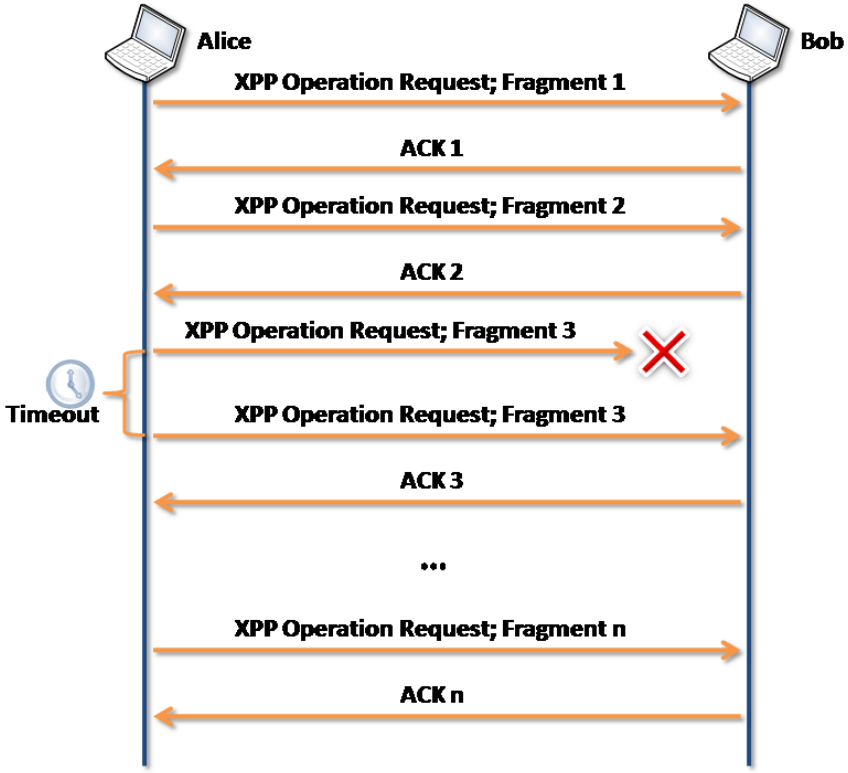


Figure 9.3: Fragmentation
Alice is sending to Bob a message larger than the current path MTU. Fragments are transmitted one by one, and every time Alice sends a packet, she would wait for Bob to respond with an ACK before proceeding. In a case when no ACK is received, Alice would resend the last packet.

- LFR : Last fragment flag. Indicates whether this is the last of a series of fragments (1 or more). If a message only consists in a single fragment this flag is to be set to 1.
- KPA : Keep alive flag. Set to 1 if this is a keep alive packet sent with the sole purpose of maintaining session state in intermediate routing devices.
- Sequence Number : Contains the sequence number of reliable fragment. Set to a random integer between 0 and 65535 (inclusive) for a first fragment and incremented by 1 for every next fragment.
- Local ID : The local identifier of an XPP session.
- Remote ID : The remote identifier of an XPP session.

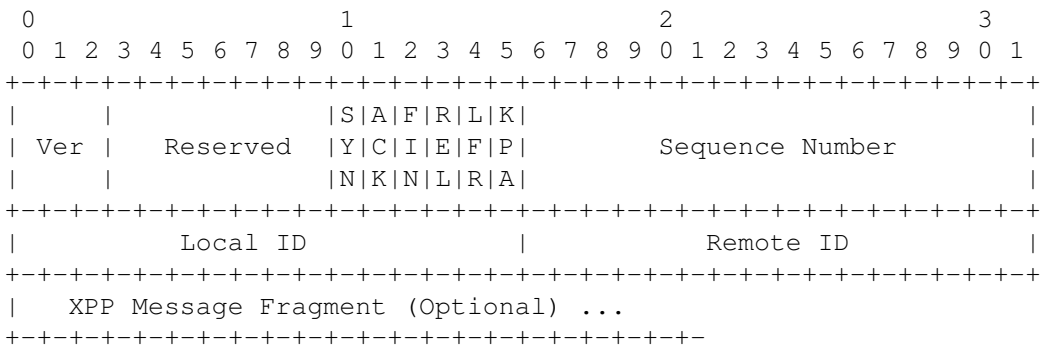


Figure 9.4: XPP fragment header.

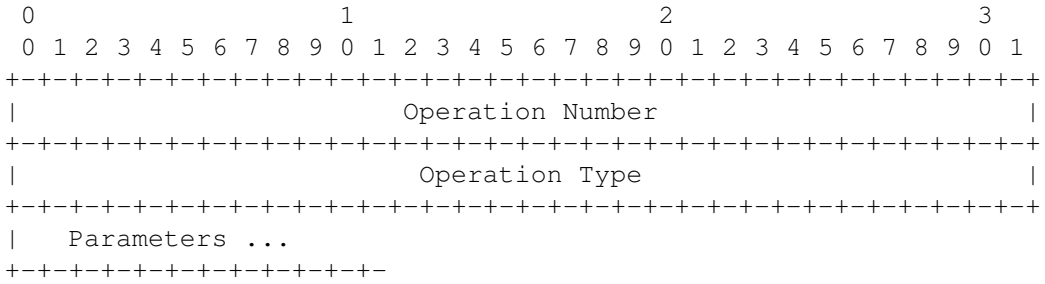


Figure 9.5: XPP Message.

9.5.2 XPP Message

Depending on their size, XPP messages can be transmitted in one or more XPP fragments, one fragment per UDP packet. Every XPP fragment would start with an XPP fragment header, but only the first one would also contain the XPP Message Header specifying the operation number.

- Operation Number : The number identifying an operation within a session.
- Operation Type : A token identifying the operation type. The field is to be set to 0x00000000 for operation responses and to the corresponding value for operation requests.
- Parameters : A concatenation of parameters, as defined below (Section 9.5.3).

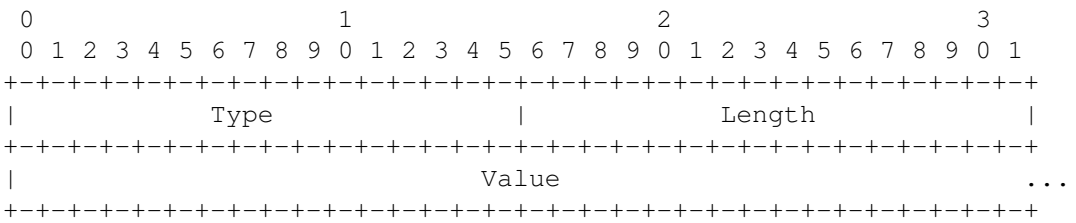


Figure 9.6: Parameter.

9.5.3 Parameters

- Type : A token identifying the parameter type.
- Length : The length of the value element, expressed as an unsigned integral number of bytes.
- Value : The value of the parameter. If the length reported in the Length field is not a multiple of 4, a padding is added so that total parameter length would always be a multiple of 4 bytes.

9.5.4 Session Establishment

Currently XPP only supports a single mode for session initiation which we refer to as simultaneous session establishment. Prior to the establishment, all parameters of the session need to be negotiated using external rendezvous and negotiation mechanisms such as those provided by SIP [167] and SDP [79] as defined in RFC 3264 [165].

Figure 9.7 shows the session establishment process in detail.

9.5.5 Session Teardown

Teardown of a particular session must be initiated by the signalling protocol used for the establishment of the session (a BYE request in the case of SIP) and is then followed by acknowledged transmission of two XPP messages with the FIN bit set by the two endpoints.

The transmission of a message with the FIN bit set explicitly indicates that the sender is not going to send any more messages and, from that point in time on, it will only

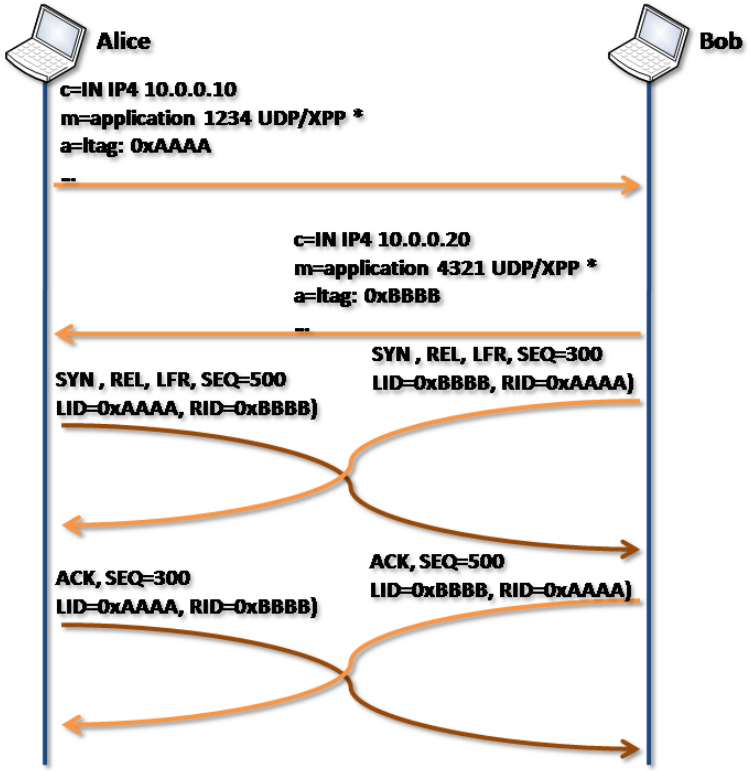


Figure 9.7: Simultaneous establishment of an XPP Session.
UDP packets are sent between endpoints 10.0.0.10:1234 and 10.0.0.20:4321. Session is identified by ID pairs (0xAAAA, 0xBBBB) and (0xBBBB, 0xAAAA) on Alice and Bob respectively.

ACK received fragments. Such a mechanism is required to let both endpoints to finish transmitting messages already scheduled for sending before the session is definitively destroyed.

After communicating her intention to close the session, Alice sends a fragment with the FIN bit set and stops sending non-ACK messages. On his side Bob agrees to the session teardown and sends the last message he had in his local queue (consisting of two fragments with sequence number 600 and 601) and then closes the session sending a fragment with the FIN bit set.

The complete session teardown process is displayed on figure 9.8

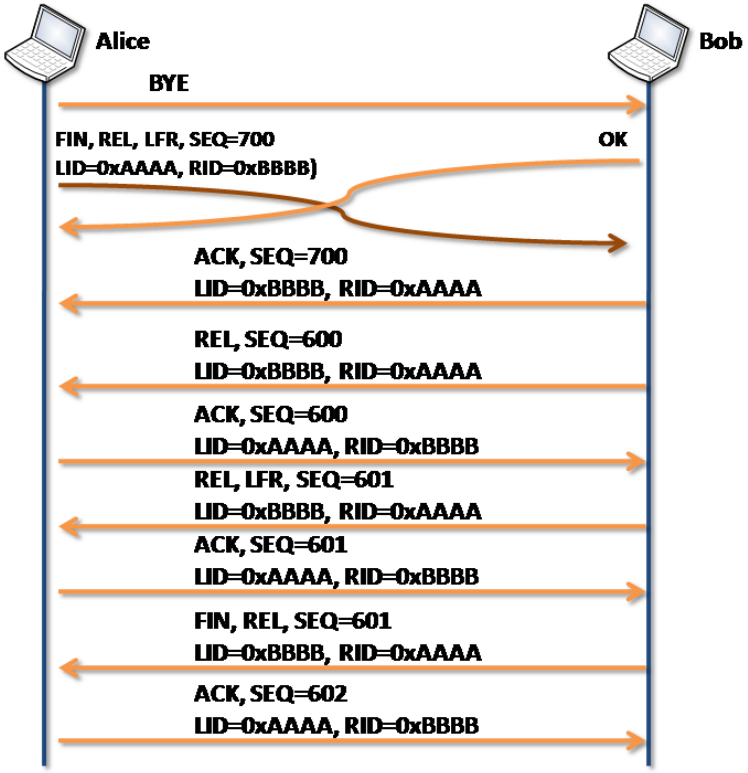


Figure 9.8: Session teardown

.

9.5.6 Session Failure

Session failure must be reported to the application by the XPP protocol stack when detected. The stack may detect such failure upon expiration of a keep alive timeout, or loss of network connectivity.

Once failure has been detected, an XPP protocol stack should stop keeping information about the state of the session and ignore any incoming XPP fragment belonging to that session (just as it would do for non-existing ones).

9.5.7 Managing XPP Operations

It is the responsibility of the XPP protocol stack to keep track of currently active XPP operations. An operation is created when the first request that belongs to it has been sent.

Implementations should provide to the application a means of specifying an expiration delay ("D") for every request being sent.

The protocol stack would consider an operation terminated "D" ms after the last response for that operation has been received. The stack should also provide the application with a means of manually ending an operation (e.g. an end Operation(operationID) method).

Once an operation has been ended, the protocol stack may either ignore all incoming responses belonging to the operation, or pass them to the application without a context associated.

9.6 Transport

All XPP messages are transported in UDP datagrams. Depending on its size a single XPP message may be transported in one or more datagrams using the XPP fragmentation mechanisms defined in Section 9.6.1.

Depending on their purpose XPP messages can be transported in both a reliable and unreliable way. Senders must set the REL bit (Section 9.5.1) of the fragment header to 1 and apply the retransmission mechanisms described in section Section 9.6.2.

Unreliable messages must be transmitted in a single fragment and any attempt from the application to send data exceeding the size of the current path MTU must result in an error.

9.6.1 Fragmentation

If a reliable message cannot fit the path MTU (fragment header included), it must be split in as many fragments as necessary. Each fragment must have the REL bit set to 1 (see Section 9.5.1). The value of the Sequence Number in the fragment header must be incremented for every following fragment.

The LFR bit (see Section 9.5.1) must be set to 1 for only in the last fragment of a message as well as for messages that are not fragmented. It must be set to 0 in all other cases.

9.6.2 Retransmissions

XPP fragments are transmitted one at a time. Using UDP as a transport implies that some fragments may be dropped by intermediate devices. Reliability is therefore accomplished through XPP fragment retransmissions. Sending party should retransmit the request as soon as timer T1 fires. Values for T1 would vary across retransmissions starting with an interval of t0 for the first one. t0 is an estimate of the round-trip time (RTT), and it defaults to 100 ms. The interval would double every retransmit until it reaches t1 (1.6s by default), and retransmissions would then continue with intervals of t1 until an XPP ACK with the matching sequence number is received, or a total of r (9 by default) fragment retransmissions have been sent. If no response is received by t1 seconds after the last fragment retransmission has been sent, the sending party should consider the transmission unsuccessful and report failure to the application.

In other words, when using default values (i.e. t0=100ms, t1=1.6s and r=9) fragments would be sent at times 0ms, 100ms, 300ms, 700ms, 1500ms, 3100ms, 4700ms, 6300ms, 7900ms, and 9500ms. At 11100ms, the sender considers the transaction to have failed.

Note that the retransmission mechanisms that we've just described must be used for all messages that require reliability (i.e. those with the REL bit set to 1 in the fragment header) and must not be applied to those that do not.

9.6.3 Keep-alive

In order to guarantee session persistence XPP uses periodically sent keep-alive messages.

Every time a fragment is received within a session, timer T2 for that session is set to t2 (default: t2 = 5 sec). When T2 fires, a keep- alive message is sent. The message only contains the XPP fragment header with both the REL and KPA bits Section 9.5.1 set to 1. The Sequence Number for keep alive messages must be incremented just as it would for any other request.

When a keep-alive fragment is received, it is acked as usual, but, since it doesn't carry any data, it is not reported to the application. If a keep-alive fragment transmission fails (i.e. if no ack is received after applying the retransmission mechanisms from section Section 9.6.2) the corresponding session is to be considered inactive and a session failure is to be reported to the application.

9.7 Conclusion

In this chapter we have presented the specification of the eXtensible Peer Protocol (XPP). The purpose of this protocol is to provide a transport mechanism for data that needs to be exchanged during the maintenance of a distributed hash table.

During the design of XPP we have strived to make it as friendly as possible to existing Network Address Translation devices and firewalls. We have therefore implemented the possibility for simultaneous session establishment and used UDP as a transport protocol.

Another important feature of this protocol is the fact that it's design does not make any assumptions whatsoever about the nature of the DHT that it will be used with. As a result, in addition to classic ring based algorithms like Chord, Kademlia, or Pastry, XPP is also compatible with others, like CAN, using hypercube models. In the following chapter we would show one possible XPP extension, called XPP-PCAN, which allows creating and maintaining CAN-like overlays.

Chapter 10

XPP-PCAN: XPP Extensions for CAN DHT overlays

10.1 Introduction

This chapter describes a possible solution for a P2PSIP Overlay Network [56] currently implemented in the SIPDHT [125] open source project. The work described in here has been completed in the context of a collaboration between the Louis Pasteur University and Telecom Italia Labs, Turin, represented by Enrico Marocco.

The passive approach that the solution adopts is relatively uncommon in related work. Yet it seems to be well-suited for addressing issues arising from the wide spread deployment of NAT devices, high churn rate and possible participation of malicious peers. This solution is intended to:

- easily support deployments where many peers are located behind NAT boxes, adopting NAT traversal mechanisms such as STUN [164] and ICE [163];
- provide mechanisms for keeping overlay size to an optimal minimum. Allowing peers located behind NAT devices to become members of the overlay network implies frequent exchange of keep alive messages. XPP-PCAN addresses this issue by only allowing the best available clients (i.e. those offering most significant resources) and doing so only when their participation is really necessary.
- provide mechanisms for limiting the effects of malicious nodes attempting to degrade the service.

XPP-PCAN is based on the following key points:

1. the overlay is organized as a distributed database or hash table based on the CAN [158] algorithm and it only offers functionalities for storing and retrieving data and for routing SIP [167] messages;

2. the P2P overlay is transparent to clients. A client can maintain SIP outbound flows [37] and register its location with any peer (causing the insertion of a routing record on the peer authoritative for its address);

   It would also be possible for clients, not participating in the overlay, to allow others to maintain outbound flows with them, as this would significantly lighten the load of the overlay. At the moment such an option has been put aside for simplicity, but is listed as an open issue.

3. Peers use XPP [126] for performing maintenance operations;

4. Peers use SIP (and possibly ICE) for establishing XPP sessions;

5. For every client using the overlay there is at least one registrar peer that the client uses as point of attachment to the overlay, and one authoritative peer (the one keeping location and routing information for the client (in some cases the two may coincide).

6. A peer can invite any client it is authoritative for to join the overlay.

Once again, one of the main characteristics of XPP-PCAN is the fact that it is not possible for a node to decide whether it would simply use the overlay network as a client or become a part of it as a peer. One could therefore imagine the network as a free service which, in some cases, could invite any of its clients to provide resources for use by others. Answering positively to an invitation is the only way for a client to become a peer.

It is worth noting that points 1 and 4 allow peers and clients to use the SIP routing functionality provided by the overlay for establishing XPP sessions, which, in turn, are needed for maintaining the overlay itself.

The remainder of this document, after giving a short overview of the CAN-based algorithm, specifies XPP extensions and defines operations each peer must perform in order to consistently maintain the overlay. However, it does not specify how a peer decides to invite a client to join the overlay; implementations must apply their own criteria for deciding both when and which node to invite.

Acknowledgement: Work described in this chapter has been completed in a joint effort with Telecom Italia Labs, Turin, represented by Enrico Marocco. The XPP-PCAN protocol specification has also been contributed to the P2PSIP working group of the Internet Engineering Task Force.

10.1.1 The Passive Approach

Networks using XPP-PCAN would manage joins in a "passive" way, and only allow nodes to become members of the overlay once they have been invited by existing peers. Overlays, used by file-sharing applications, would generally adopt the opposite approach and let clients decide whether or not to become peers. A well known exception to this trend is the Skype network [30], arguably one of the most popular overlay networks used for real-time communications today. Instances of the Skype application may operate as super-nodes or ordinary-nodes and the "promotions" are decided by the higher levels of the hierarchy.

The passive approach seems appropriate for P2PSIP because:

- all peer candidates are actually clients that have already registered with the service and are therefore known to the overlay.
- the overlay only needs to provide a limited amount of functionalities to clients and these could be handled by a relatively small subset of the nodes that actually use them.
- providing SIP connectivity to clients, storing their location, routing messages,in addition to maintaining tens or even hundreds of connections with neighbor peers could be very resource consuming. It is therefore important to confirm availability of such resources prior to inviting a client to join the overlay as a peer.

An additional advantage of passive joins is the fact that they allow the overlay to select peers among candidates based virtually any kind of performance and security characteristics.

It is possible, for example, to limit the effects of malicious peers by only inviting trusted nodes to join the overlay. Assessing the level of trust could be handled in many different ways like provider maintained white-lists or social networks.

10.1.2 Why CAN?

The choice of the distributed hash table algorithm has been heavily influenced by our goal to have robust overlay networks even in cases when many peers are behind a NAT. This requires maintaining persistent connections between peers and CAN fits this requirement quite well, because:

1. It is symmetric [29] unlike Chord [186]. Without such a property, each connection is efficiently used by only one of the endpoints;
2. Peers maintain a stable routing table with a limited number of entries (which is not the case with Pastry [170] and Kademlia [129]).

A possible drawback when using the CAN algorithm is its relatively low performance. While other popular algorithms claim to always be logarithmic in complexity, overlays based on CAN cannot scale indefinitely. CAN based overlays need to be configured during deployment with parameters (e.g the number of dimensions for the hash space) depending on the size of the intended network.

However, in the case of P2PSIP, the service could be provided by a subset of interested nodes. This and the possibility to use delay measurements between peers when selecting best routes within the overlay, could help achieving acceptable performance even with a limited number of peers.

10.2 Algorithm Overview

The algorithm implemented by the overlay is a version of CAN [158] slightly customized to fit the "passive" approach. This section briefly describes the overall design and the procedures for routing, joining the overlay and recovery after failures. [34] and Section 10.4 will then define how clients and peers establish and use XPP sessions for implementing these procedures.

10.2.1 General Design

The functionality on which the overlay is based, like all other distributed hash table algorithms, is the mapping of keys over the peers. Such a functionality, unlike in algorithms based on the concept of consistent hashing [186], [170], [129], is implemented in a virtual d-dimensional Cartesian coordinate space on a d-torus.

The d-torus is the generalization of the Chord ring in d dimensions. In fact, while in Chord keys are uni-dimensional and can be represented on a ring (i.e. a circular line), in CAN – and so in the algorithm described here – they are at least bi- dimensional and thus require a torus (i.e. a circular plane).

A d-dimensional key can easily be obtained by splitting in d parts a uni-dimensional value returned by a common hash function.

Any peer is assigned a distinct zone of the overall space and is authoritative for all the keys falling in it. Zones are always defined by the coordinates of their lower left and top right corner, and contain the area locked between the borders including the bottom and left edges (thus excluding the right and top edges). At any point in time, the overlay is consistent if the whole space is completely covered. Figure 10.1 shows a bi-dimensional space partitioned among 5 peers.

A peer maintains XPP sessions with all its immediate neighbors, along with information about the zones they are authoritative for. When a peer receives an operation request for a key which falls out of its zone, it routes the request to the most appropriate neighbor.

The most appropriate neighbor is generally the one whose center of mass is closer to the requested key; however, an implementation may also take into account link speed and so select not just according to geometric distance. In any case, in order to avoid loops, peers must not route messages to neighbors which are not geometrically closer to the targeted key.

10.2.2 Peer Join

The decision when to invite a client to join the overlay is always taken by existing peers. After choosing the best candidate to "promote", the inviting peer would either select a fragment recovered from a dead neighbor (see Section 10.2.3) or obtain a new one by splitting its zone through the longest edge into two equal parts. It would then assign the new fragment to the entering peer.

The selection process of the candidate to invite may be based on the evaluation of parameters like bandwidth, connectivity, uptime and trust; such a process strongly depends on the deployment and is outside the scope of this document.

It is worth noting that, in the case of P2PSIP, a peer may choose the most appropriate node to invite among the clients it stores a registration for, that is the client whose address fall under its authority.

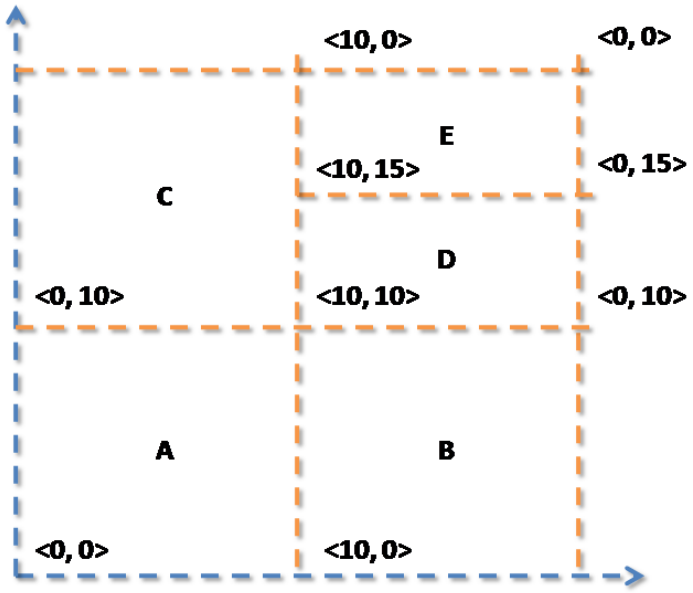


Figure 10.1: A bi-dimensional space with 5 peers.
For simplicity, the space is shown in two dimensions. In reality the figure is actually a torus.

After identifying the zone and the node to invite, the inviting peer sends a SIP INVITE request to the new peer and establishes an XPP session. Once the XPP session established, the join is completed in the following six steps:

1. The inviting peer transfers to the new peer all data bound to keys located in the new zone;
2. The inviting peer notifies all the neighbors of the zone being transferred that a peer is about to join, and that they should be ready to establish XPP sessions with their new neighbor;
3. If the zone of the inviting peer has been split, it notifies all its current neighbors of its new coordinates;

4. The inviting peer sends to the invitee the list of all neighbors it will have after joining the overlay;

5. The entering peer establishes XPP sessions with all its new neighbors using SIP;

6. The inviting peer updates its neighbor list and ends all XPP sessions with peers which are no longer its neighbors.

10.2.3 Failure Recovery

When a peer fails, one or more zones of the overlay need to be recovered through a distributed election. Elections are run by peers neighboring orphaned zones. The exact algorithm is describe in Section 10.4.2.

When a peer loses connectivity with one of is neighbors, it starts a timer waiting for other neighbors to also detect the failure. The exact value of the timer depends on XPP timers and must therefore be greater than the maximum allowed interval between two subsequent keep alive responses plus the time necessary for neighbors to complete retransmissions of their last keep alive message. When the timer fires, the peers would start the election algorithm.

At this point all neighbors of the failed peer start exchanging messages for electing the one to take over the orphaned zone. Every one of them would store the identity of the most appropriate candidate it sees and, after a stabilization interval Section 10.4.2, consider it authoritative on that zone.

When a peer is elected for taking over a zone, it is likely that it does not know all of its new neighbors; if this is the case, it must query the neighbors it knows in order to discover new ones and establish XPP connections with them.

10.2.4 Stabilization

Race conditions, temporary transport failures and misbehaving peers may cause inconsistencies in the topology of the network. Such inconsistencies most often result in neighbor peers have different representations of the geometry of a certain number of bordering zones.

A simple yet efficient stabilization procedure to realign the internal state of a peer with that of its neighbors is accomplished querying all peers in the neighborhood and checking that their geometrical view is consistent with that of the querying peer. Any

time an inconsistency is detected (i.e. the querying peer discovers a zone whose owner differs from the one it had recorded or whose coordinates overlap with some zones it is aware of), it is resolved on a popularity basis, aligning the internal state with the information reported by the highest number of peers.

10.2.5 Data Replication

Persistence of data against peer failures is achieved by making each peer replicate its local hash table on an arbitrary number of neighbors. Upon a peer failure, right after the recovery procedure described in Section 10.2.3 has completed, all peers which where neighbor of the failed peer check in their replica tables if they have records which are under the authority of the peer elected as the new owner of the recovered zone; if they find any, they immediately send records back to the new peer.

As long as peers consistently choose their replica holders (i.e. they send records bound to the same keys always to the same peers whenever this is possible), such a mechanism can recover the data stored by a failed peer unless all its neighbors storing a replica of that data fail at the same time.

10.3 Client Behavior

The overlay is intended to be transparent to conventional SIP user agents. Once such a client has discovered the location of one or more peers (how exactly it has done so is outside the scope of this document), it may set any of them as its outbound proxy. It should then register using the peer as a registrar and following the registration procedures described in RFC3261 [167] and, if needed, the SIP client may direct SIP outbound flows [37] to this peer in order to allow NAT traversal for SIP messages.

If the client wants to be able to join the overlay, it must use a SIP contact that is routable from any point in the overlay (e.g. a global scope IP address); if it does not have such a contact, it may request a public GRUU [99] from the peer (or peers) that it is currently registered with.

When additional resources are necessary for the maintenance of the overlay, a client may receive a SIP INVITE request asking to join the overlay, as described in Section 10.4.6. Since such a request must list the 'pcan' tag in the 'Require' header field, clients not implementing this specification, will answer with a 420 (Bad Extension) error message, as specified in RFC3261 [167].

10.4 Peer Behavior

10.4.1 Key-Point Mapping

At any point in time, every peer is authoritative for one or more geometric zones, which means that it is responsible for storing all data bound to keys that correspond to points in these zones.

In order to obtain the coordinates of a d-dimensional point, one has to split into 'd' equal components the 20 byte-long hash string returned by applying the SHA-1 [68] function on the key stripped of any URI parameters (i.e. all characters up to the first occurrence of a semi-colon - ';').

The number of components must be equal to the number of dimensions indicated in the initial JOIN request; allowed values are 2, 4, 5 and 10.

10.4.2 Internal State

The state of a peer participating in an overlay can be represented as a list of zones the local peer is authoritative for or which border on zones the local peer is authoritative for. For each zone, a peer must store the SIP address-of-record and contact of the responsible peer, the coordinates of the geometric area and a state variable which can be set according to the state diagram in Figure 10.2. When keeping track of the states of all neighbor zones (explained in the following paragraphs), a peer may assign each one of them one of three timers – timer TC1, timer TC2 or timer TC3.

Timers are named TC1, TC2 and TC3 for differentiating from XPP timers T1, T2 and T3.

At any point in time a peer should have active XPP sessions with all its neighbors. It must handle failures occurring in such sessions as explained in the following paragraphs.

Zone States

Every peer maintains a list of all zones bordering on its own. Every such zone may be in one of the following states.

The states have the following meaning:

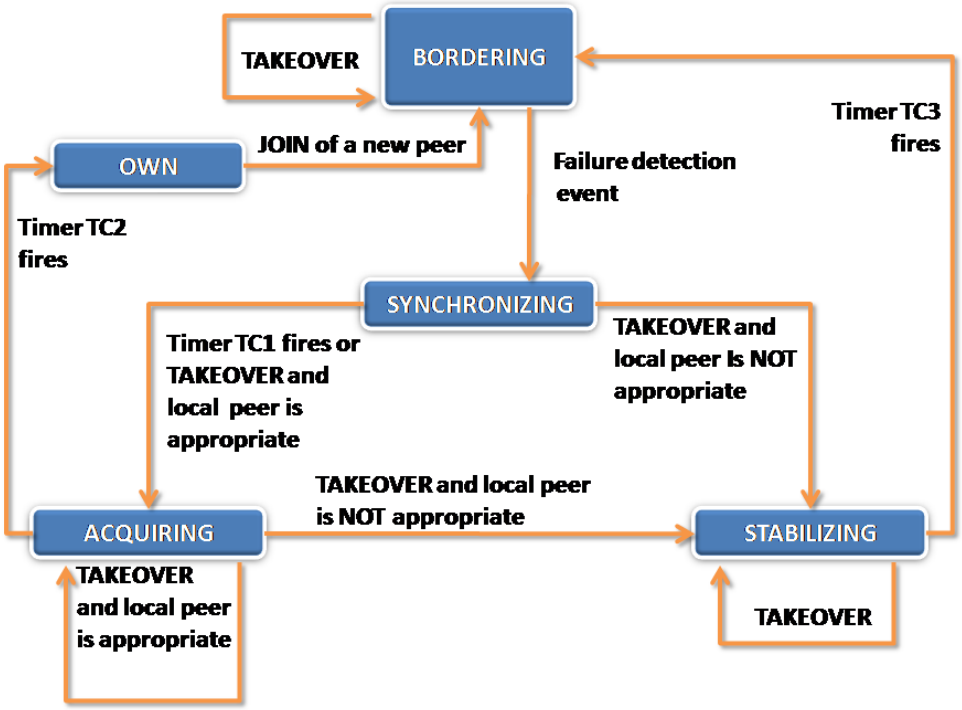


Figure 10.2: Zone state diagram.

- BORDERING: a neighbor peer is authoritative for the zone and the XPP session with this peer is active.
- SYNCHRONIZING: the neighbor authoritative for the zone is unreachable. The peer waits until all neighbors detect the failure.
- STABILIZING: a neighbor has been elected to take over the orphan zone. The local peer is waiting to make sure that no one else claims ownership of this zone.
- ACQUIRING: the local peer is about to take over a zone but it is still waiting in case a more appropriate neighbor would claim it.
- OWN: the local peer is authoritative for the zone.

In most cases, zones would be in either an OWN or a BORDERING state. States like SYNCHRONIZING, STABILIZING and ACQUIRING are only entered when a failure is detected and a zone needs to be taken over by another peer. The recovery algorithm

uses three timers – TC1, TC2, and TC3 – and is completed through several exchanges of TAKEOVER messages. A peer must therefore be able to handle five different events for each of its neighbor zones, as defined in the remainder of this section. The transition from state OWN to BORDERING, which occurs only when a new peer joins the overlay, is separately described in Section 10.4.6.

A complete description of all state transitions defined by this protocols is provided in section 11.12

Neighbor Evaluation for Takeover

In order to determine whether one peer is more appropriate than another for taking over a zone, a list of conditions need to be taken into account in the following order:

1. If one of the peers is the current owner (e.g. the failure was temporary or just limited to some connections), it wins the election;
2. If only one of the peers can merge the uncovered zone with its own, it wins the election. Note: two zones can be merged if they share a border component (an edge for 2d, a surface for 3d and so on);
3. If the sums of the zone volumes are different for the two peers, the one with the smallest value wins the election;
4. If none of the above produces a winner, the peer with the first identity in lexicographic order wins the election.

Stabilization Procedure

Since misbehaving peers, race conditions and temporary network failures may cause inconsistencies in their internal state, peers should periodically execute the following procedure in order to be sure it is coherent with that of their neighbors:

1. Send a QUERY operation request targeted to each zone bordering on any of its own zones;
2. On reception of a QUERY operation response, update a temporary list for each of the reported zones:

if an equivalent zone with the same owner cannot be found in the list, insert it and assign a default initial score value;

if an equivalent zone with the same owner can be found in the list, increase its score value;

3. When QUERY operation responses have been received for all sent requests, resolve conflicting zones (i.e. zones with equal coordinates but different owners or zones with overlapping coordinates) keeping in the temporary list the one with the higher score value and dropping the other;

4. Remove from the temporary list all zones conflicting with any zone in the effective list whose state is not BORDERING;

5. Update the internal state replacing in the effective zone list all zones in BORDERING state with those in the temporary list.

10.4.3 Routing XPP Messages

When a peer receives a GET, a PUT or a QUERY operation request for a key or a target which is not contained in any of its zones, it must route it to the most appropriate neighbor.

The most appropriate neighbor is generally the one whose center of mass is closest to the target; however, an implementation may also take in account link speed and so select not just according to geometric distance. In any case, in order to avoid loops, peers must not route messages to neighbors which are not geometrically closer than them to the destination zone.

XPP responses must be returned through the path that the originating request came through. When routing an XPP request, a peer must therefore locally store information for routing responses back on the same session it was received. Neither the XPP specification [126] nor this document currently define for how long peers should keep state information about routed requests (at the moment this is still an open issue). However, it is still worth noting that as GET and PUT operations are supposed to be used for handling SIP transactions, it is recommended that such an interval is not shorter than 32 seconds.

10.4.4 Handling SIP Registrations

All peers must be able to process SIP REGISTER requests sent directly by clients or routed by other peers in the overlay's domain, and update routing records in the distributed database with XPP PUT operations (see Section 10.4.7). Moreover, to overcome issues related to connectivity restrictions, such as NAT devices, peers must support the SIP outbound [37] and GRUU [99] extensions.

If the Contact header field in the incoming REGISTER contains the 'reg-id' parameter, the connection from which it was received must be kept alive as described in [37] and the routing record must consist of a path where the first element is the URI identifying the peer (a SIP contact or a GRUU), the last element is the value in the Contact header field, and the elements in between are copied from those read in the Path header field [198], if one is set. Otherwise, if the registration does not require outbound support, the record must only contain the first value in the Contact header field.

If the REGISTER request has a Supported or Require header field containing the 'gruu' tag, the peer must generate a GRUU appending a 'gr' parameter with a value equal to the instance ID (reported in the '+sip.instance' parameter in the Contact header field) to the address-of-record of the registering peer. In such cases, the peer must insert records for both the address-of-record and the GRUU into the distributed database, and return the GRUU in the SIP response as specified in [99].

It is worth mentioning that, according to mapping rules in Section 10.4.1, the same peer will be authoritative on both the GRUU and the address-of-record.

In general, a peer handling a REGISTER request is not necessarily the one storing the corresponding routing record; A registrar peer must send an XPP PUT operation request as described in Section 10.4.7 and wait for the XPP response to generate the SIP response. Since the XPP request could silently fail, if a response is not received after a "reasonable" time, it should close the SIP transaction with a 504 "Server Time-out" error code.

The "reasonable" time period depends on several parameters, such as the overlay size and the transport protocol used for the SIP registration. It is recommended that, if the REGISTER request was sent over UDP, such period is shorter than SIP's Timer F value (default: 32 seconds) [167]. The matter is still considered an open issue (Section 8).

10.4.5 Routing SIP Requests

When a peer receives a SIP request not addressed to itself and with no Route header field set, it must first determine if the target (the request URI) belongs to the overlay domain. If not, it should route it according to rules defined in [166]. Otherwise, if the target of the request is a SIP URI in the overlay domain, it must retrieve the routing record bound to the URI in the request line, generating an XPP GET operation request as defined in Section 10.4.9.

The corresponding GET operation response, returned by the peer authoritative for the destination URI, must contain the path SIP requests need to follow to reach the target. After receiving the GET operation response, the peer must recursively resolve all path components that are represented as URIs belonging to the overlay domain. Next, it must add Route header fields reflecting the resolved path and use it to route the request.

If the peer does not receive a GET response after a "reasonable" amount of time, it should close the SIP transaction with a 504 "Server Time-out" error code. As already mentioned , this "reasonable" amount of time is currently considered an open issue (Section 11).

As noted in Section 10.4.4, the "reasonable" time interval depends on several parameters, such as the overlay size and the transport protocol used for sending the SIP message. It is recommended that, if the SIP request was sent over UDP, such period is shorter than 32 seconds.

10.4.6 Inviting a Client to Join

When a peer needs to invite a new client to join the overlay it first needs to select the "best" available among those it stores an active SIP registration for. This document does not specify how it could be done; implementations will apply their own criteria.

After identifying the client to invite and a zone to transfer (either one recovered from a dead peer or one obtained by splitting its own), the inviting peer must send a SIP INVITE request to the joining client in order to establish an XPP session with it. The INVITE request must have a Require header field containing the 'pcan' extension tag and will be routed as specified in Section 10.4.5.

In network environments where it is expected that peers might be located behind NAT devices, the session negotiation should be completed using the ICE [163] mechanism.

If the client answers positively and after the XPP session has been correctly established as described in [126], the join procedure would continue through the following steps:

1. The inviting peer sends a PUT request generated as described in Section 10.4.7 to the entering peer for transferring all the data bound to keys located in the new zone;
2. The inviting peer sends an UPDATE request to all its neighbors. Generated UPDATE requests is described in Section 10.4.15. This request would report all zones of authority of the inviting peer as well as the one transferred to the entering peer. Peers receiving an UPDATE request must take into account any changes of the overlay topology and, in case they are also going to be neighbors of the new peer, prepare to establish XPP sessions with it.
3. The inviting peer then sends a JOIN request generated as described in section 10.4.17 to the entering peer. The request contains coordinates of the zone being transfered and the list of its neighbors;
4. The entering peer establishes XPP sessions with all its new neighbors using SIP;
5. The inviting peer updates its zone list and closes sessions with peers that, as a result of the transfer, are no longer its neighbors.

10.4.7 Generating PUT Operation Requests

PUT operations insert key-value pairs in the distributed database. In general, such requests would be used when storing SIP registrations. Every such request is composed of one or more tuples containing:

- **KEY**: a SIP URI, usually an address-of-record or a GRUU [99];
- **VALUE**: the route-path SIP messages addressed to the user should follow;
- **BINDING-ID**: the token identifying the key-value pair instance. Two subsequent PUT operations with same KEY and same BINDING-ID overwrite the same record; two PUT operations with same KEY and a different BINDING-ID would cause the insertion of two different records. For PUT requests generated upon SIP registrations, the BINDING-ID should contain the value of the Call-ID header field in the REGISTER request;

- **EXPIRES**: the amount of time (in seconds) that the record needs to be kept.
- **It is recommended** that requests containing more than one tuple are generated only if all tuples fall under the authority of the same peer; this could be the case, for example, during a join (Section 4.6) or when handling a registration requesting a GRUU (Section 10.4.4).

10.4.8 Handling PUT Operation Requests

Upon reception of a PUT operation request, a peer must first check if it is responsible for the KEY parameter contained in the first tuple. If it is not, it must route the request as defined in Section 10.4.3; otherwise, for each tuple in the request it is responsible for, it must update or insert a record in its local table, respectively if one with same KEY and BINDING-ID previously existed or not.

Once stored records it is responsible for, the peer must return in a PUT operation response, all records in its local table bound to keys matching one of those contained in the initial request. PUT operation responses are thus syntactically identical to PUT requests (see Section 10.4.7).

In general, according to replication policies, after storing data a peer peer should replicate it on some of its neighbor peers generating a REPLICA operation request as described in Section 10.4.11.

Even if the selection of the neighbor peers to send REPLICA requests to is not important per se, it needs to be consistent in a way that, whenever possible, REPLICA operation requests for records bound to the same keys are sent to the same peers.

10.4.9 Generating GET Operation Requests

GET operation requests retrieve data in the distributed database. Most often, GET requests would be used when querying the location of clients and peers for routing SIP messages. Such requests contain one or more KEY parameters, typically a SIP address-of-record or a GRUU [99].

It is recommended that requests containing more than one KEY parameter be generated only when all keys fall under the authority of the same peer.

10.4.10 Handling GET Operation Requests

Upon reception of a GET operation request, a peer must first check if it is responsible for the first KEY parameter. If it is not, it must route the request as defined in Section 10.4.3; otherwise, it must return in a GET operation response, all records in its local table bound to keys matching one of those contained in the initial request.

GET operation responses are syntactically identical to PUT responses (see Section 10.4.8).

10.4.11 Generating REPLICA Operation Requests

REPLICA operations insert key-value pairs in the local replica table of a neighbor peer. In general, such requests would be generated as a result of handling PUT operation requests. Every such request is composed of one or more tuples containing:

- **KEY: a SIP URI**, usually an address-of-record or a GRUU [99];
- **VALUE**: the route-path SIP messages addressed to the user should follow;
- **BINDING-ID**: the token identifying the key-value pair instance. Two subsequent REPLICA operations with same KEY and same BINDING-ID overwrite the same record; two REPLICA operations with same KEY and a different BINDING-ID would cause the insertion of two different records.
- **EXPIRES**: the amount of time (in seconds) that the record needs to be kept.

10.4.12 Handling REPLICA Operation Requests

Upon reception of a REPLICA operation request, a peer must update or insert a record in its local replica table, respectively if one with same KEY and BINDING-ID previously existed or not.

An entry in the local replica table is stored until either it expires or a new neighbor peer becomes responsible for it. In the latter case, other than removing from its replica table all entries the new neighbor is responsible for, the local peer must send a PUT operation request for all the purged entries, as described in Section 10.4.7.

10.4.13 Generating QUERY Operation Requests

QUERY operations are used by peers to gather information about the overlay topology, for example, for discovering new neighbors after a recovery. QUERY requests consist of one TARGET parameter used for routing the operation.

10.4.14 Handling QUERY Operation Requests

Upon reception of a QUERY operation request, a peer must first check if the TARGET parameter belongs to one of the zones it is authoritative for. If not, it must route the request as explained in Section 10.4.3; otherwise, it must return a representation of the zones it is aware of in a QUERY response containing:

- A list describing zones the answering peer is authoritative for, each one containing:

  OWN-CONTACT: contact of the answering peer;

  OWN-AOR: address-of-record the answering peer is registered for;

  OWN-ZONE: coordinates of the zone;

- A list describing zones neighboring those of the answering peer. Every list entry contains:

  PEER-CONTACT: contact of the neighbor peer;

  PEER-AOR: address-of-record the neighbor peer is registered for;

  PEER-ZONE: coordinates of the zone.

10.4.15 Generating UPDATE Operation Requests

UPDATE operations are used for advertising changes in the overlay topology, such as joins, recoveries or zone merges. UPDATE requests must report information related to zones the sending peer is authoritative for or is currently transferring. They contain the following parameters:

- a list describing zones the sending peer is currently authoritative for, each one containing:

  OWN-CONTACT: contact of the sending peer;

OWN-AOR: address-of-record the sending peer is registered with;

OWN-ZONE: coordinates of the zone;

- a list containing all zones neighboring those of the sending peer. The list may also contain zones that used to be under the authority of the sending peer but have just been transfered to a joining node. List tuples must contain the following parameters:

  PEER-CONTACT: contact of the neighbor peer;

  PEER-AOR: address-of-record the neighbor peer is registered for;

  PEER-ZONE: coordinates of the zone.

Contrary to QUERY responses, UPDATE requests must only report details that the sending peer is authoritative for, or zones that it is in the process of transfer to a joining peer after inviting it.

10.4.16 Handling UPDATE Operation Requests

Upon reception of an UPDATE operation request, a peer must first remove from its zone list all zones overlapping with any of those reported in the request. It must then insert all zones from the UPDATE request that border zones it is authoritative for. UPDATE is a responseless request and must not be routed.

10.4.17 Generating JOIN Operation Requests

JOIN operations are used to pass the data that a client needs in order to join the overlay and become a peer. JOIN requests are generated by the inviting peer as described in Section 10.4.6 and contain the following parameters:

- SPACE: coordinates of the whole space;
- YOUR-ZONE: coordinates of the zone the entering peer will be authoritative for;
- a list describing zones the inviting peer is authoritative for, each one containing:

  OWN-CONTACT: contact of the inviting peer;

  OWN-AOR: address-of-record the inviting peer is registered for;

  OWN-ZONE: coordinates of the zone;

- a list describing zones bordering on the one the entering peer will be authoritative for, each one containing:

  PEER-CONTACT: contact of the neighbor peer;

  PEER-AOR: address-of-record the neighbor peer is registered for;

  PEER-ZONE: coordinates of the zone.

10.4.18 Handling JOIN Operation Requests

JOIN operation requests can only be received during the join phase, as described in Section 10.4.6. JOIN is a responseless end-to-end operation and must not be routed; a peer receiving an unexpected JOIN request must ignore it.

10.4.19 Generating TAKEOVER Operation Requests

TAKEOVER operations are used for electing the most appropriate peer to take over a zone left by a peer that has become unreachable. TAKEOVER requests are generated and exchanged by neighbors of the dead peer and contain the following parameters:

- DEAD-CONTACT : contact of the dead peer;
- DEAD-AOR : the address-of-record the dead peer was registered with;
- DEAD-ZONE : coordinates of the dead zone;
- PEER-CONTACT : contact of the peer candidate to take over the dead zone;
- PEER-AOR : address-of-record the peer candidate to take over the zone is registered for;
- PEER-ZONE : coordinates of a zone the peer candidate to take over the zone is authoritative for. If one or more of the zones the candidate peer is authoritative for is mergeable with the dead one, it should be reported in this parameter as it would increase the probability for the peer to be elected (see Section 10.4.2);
- PEER-VOLUME : the total volume of the zones the candidate peer is authoritative for.

10.4.20 Handling TAKEOVER Operation Requests

TAKEOVER operation requests are received only when recovering from failures and are handled according to the state diagram defined in Section 10.4.2.

Unexpected TAKEOVER requests, for example referring to non-bordering zones, must be ignored.

10.5 XPP Extensions

As mentioned earlier in this chapter, the XPP-PCAN protocol extends the XPP specification [126]. The exact syntax of all message extensions, XPP parameters and operations is provided in an annex in section 11.16.

10.6 Security Considerations

It is possible to identify (at least) four different categories of security issues:

- XPP transport issues, that can be addressed by securing the transport channels using a mechanism like DTLS [160];
- Eavesdropping and message forgery of SIP and media traffic by seemingly benign peers. While media can be encrypted using SRTP [32], SIP end-to-end security is still an open issue;
- Unsolicited XPP session establishments - Procedures for join and recovery are defined so that any peer needs to allow session establishments to peers whose identity have already been presented by one of its neighbors. Such a mechanism builds a sort of "overlay trust", which requires further investigation, because, while it seems one of the most powerful techniques for managing authorization in peer-to-peer environments, it could be exploited by malicious peers that have entered the overlay through an error or deceit;
- Overlay damages caused by malicious peers - This kind of issue is characteristic for peer-to-peer systems and, in general, is the hardest to deal with. However, with the "passive" approach, a malicious node cannot determine when it will become peer, which zone of the overlay it will be assigned and can only cause damages

proportional to the area under its authority; moreover, implementations can apply their own methods, possibly based on both performance and social information, to filter malicious nodes and select the best ones to upgrade.

Chapter 11

P2P Conclusion and future work

In the this part we have presented our work on the adaptation of Peer-to-Peer overlay networks to real-time communications. The reason to head in this direction consists mainly in the fact that services such as VoIP, VOD, etc, generally require substantial resources in terms of bandwidth and processing power. We have also considered interesting the opportunity of distributing NAT and firewall traversal algorithms so that they would allow for a larger number of providers and a innovative collaboration possibilities.

In the last two chapters we have described our work on two protocols meant for creating and maintaining Peer-to-Peer overlays - XPP and XPP-PCAN.

The eXtensible Peer Protocol (XPP) is a transport protocol that was especially designed for use with distributed hash table maintenance algorithms. It contains a number of features, such as simultaneous session establishment and UDP transport, that make it particularly friendly to existing algorithms for traversal of Network Address Translation gateways and firewalls. Another important characteristic of this protocol is the fact that it would work with any DHT algorithm regardless of its nature, including ring-based ones like, Chord, Kademlia, and Pastry, but also algorithms like CAN based on hypercubes.

The second protocol, called "XPP Extensions for Implementing a Passive P2PSIP Overlay Network based on the CAN Distributed Hash Table" is an extension of XPP that specifies its use with the CAN DHT algorithm. One of its most important characersistics is the fact that the decisions of which clients would be joining the network as contributing peers, belong to the overlay itself. As a result, XPP-PCAN allows for creating overlay networks where any peer, at any given time has trust based relations with all its neighbors.

11.1 Future work

An implementation of our work on Peer-to-peer is already available under an open source license [125], however as of today we have not yet completed a comprehensive evaluation in real world scenarios. Our future work would therefore most certainly start with a second implementation in the Java programming language, its integration in the SIP Communicator communications client and the evaluation of the whole solution by standard users and a relatively large scale.

Other points that also require future work include the definition of mechanisms to select peers that are most suitable for joining an overlay. Some of the directions that first come to mind include exchange of peer processing and bandwidth characteristics.

In addition to this we need to devise ways for peers to announce authentication credentials upon the join of new neighbor in addition to simply broadcasting its identity to all future neighbors.

We also need to determine whether it is a good idea for clients to allow other clients to establish SIP outbound flows with them. In such a case, they would probably need to implement the mechanisms for registering non-adjacent contacts defined in RFC3327 [198].

Finally, the current version of this protocol does not define the amount of time that peers should keep state information about routed requests, and whether this should be indicated in the requests. We would need to determine whether they have to do so and if yes how.

Part III

Overall conclusions and future work

Overall conclusions and future work

The work described in this thesis has been completed with the purpose of optimizing two different aspects of real-time communication over the Internet protocol. In Part I we have focused on solutions for achieving seamless terminal mobility. Part II then presents our research on the subject of using Peer-to-Peer networks in real-time communications.

11.2 Seamless mobility

When a mobile node disconnects from one Wi-Fi network and connects to a new one, the resulting handover may have an impact on different layers of the ISO OSI model. Depending on the network configuration the transition could span through the link, network and application layers. Work on managing and optimizing mobility could therefore be done on each of these layers.

Our work has started at the application layer in an effort to provide a solution that offers reasonable handover performance and that could be easily deployed over most existing network topologies without modifying them. The IETF Session Initiation Protocol provides mechanisms for handling terminal mobility at the application layer. After moving to a new network a node can reinitialize its ongoing SIP sessions by sending reINVITE SIP requests to all its correspondents. However a standard implementation of these mechanisms would react to movement only after detecting change in the IP address through the socket APIs exported by the network layer. In most cases, this detection, would suffer delay accumulated by lower layers which would result in degraded quality when rendering media flows and hence a poor user experience. We have therefore designed, implemented, and evaluated a solution that uses link layer triggers to assist movement detection and hence speed up the handover process. The results of the evaluation have shown that it is possible to achieve satisfactory, though not exceptional, performance of application layer mobility, suitable for an interim solution in cases where modifying the network topology is impossible.

We have therefore switched our focus to the network layer with the goal of achieving an optimal solution that allows for 100% transparent mobility. After a comprehensive study of existing mobility optimizations our attention has been drawn by the Fast Handovers for Mobile IPv6 protocol (FMIPv6). We have created one of its first implementations (now available under an open source license on http://fmipv6.org) and used it to evaluate the protocol in real world scenarios (as opposed to the simulational evaluations existing at the time). Our analysis have shown that even though the protocol allows for a very smooth transition between networks, it does little to improve the discovery process of Wi-Fi access points that a node can use to connect to. As a result, even if handovers themselves are completely seamless, user experience is still suffering during the scanning procedure performed by the wireless interface.

The main reason for these problems come from the blocking nature of the scanning procedure and the fact that contrary to other technologies, Wi-Fi devices are only able to transmit and receive data on one frequency at a time. In order to resolve this problem we have devised a mechanism that allows WLAN nodes to use a secondary wireless interface when switching from one access point to the next. This kind of handovers are often referred to as "soft handovers". We have evaluated the solution using the SimulX wireless simulator and the results were quite encouraging. Using soft handovers allows to completely eliminate packet loss when moving between access points.

Our next step has been to try and adapt the use of multiple interfaces to something which would better match mechanisms being worked on by standardization bodies such as the IETF. We have already mentioned that the most problematic point in FMIPv6 was the fact that it did nothing to assist the mobile in avoiding the loss of connectivity during a wireless scan. In the same time, our evaluation of the multi interface soft handovers approach had clearly qualified it as a good way to address this same issue and finding a way to integrate it inside the FMIPv6 platform was definitely worth the effort. Following an implementation and its very complete evaluation we have concluded that the combination is indeed quite efficient.

There is of course one aspect of the above solution that could be considered "uncomfortable" in certain cases and it is the fact that few of today's mobile devices have more than one wireless interface. It was therefore important for us to work on an alternative way of providing non-disruptive discovery of surrounding access points that does not have that constraint. We have therefore also worked on a solution which allows a mobile to use a geographic positioning device in order to obtain its coordinates, feed them through the FMIPv6 protocol to a geographically aware system and receive in return the access points available in the surroundings. After evaluating the solution, we have obtained results which are quite similar to those in the multi interface solution, or in other words one could easily replace the constraint of a second interface with that

of a GPS device - something which is more commonly found in most handheld devices currently available on the market.

11.2.1 Future work

Given the nature of the problems associated with IP mobility, we are convinced that any solution proposing improvement of existing techniques should be evaluated primarily through implementation and experimentation. We have kept this in mind during the last several years and tried to apply it to most of our work. Yet, bringing it to a stage close to that necessary for production deployment would require even more experimentation and evaluation of other aspects such as scalability and its behavior in different use cases. A GPS assisted mechanism, for example, would behave differently when used from within vehicles on a highway or pedestrians in the city.

We also believe that we could further improve IP mobility solutions described in this thesis by completely removing the need of having an extra device assisting the handover. Porting the multi interface extension of FMIPv6 to a single IEEE 802.11n (a.k.a. MIMO) card, for example, should be a fairly straightforward process. Such an improvement would make the solution a lot more realistic since most new wireless devices come with network cards with integrated support for this protocol.

Concerning, our GPS assisted FMIPv6 extension, we plan on studying alternative geographic positioning systems that allow a node to discover its position by only using its wireless interface.

Finally, in Section 2.6.4 we have described a proprietary mobility solution that allows the optimization of the handover process by enabling access points to keep track of the movement of mobile nodes and use it to assist them in choosing their next point of attachment to the network. The FMIPv6 protocol is actually equipped with most of the mechanics which are necessary for the implementation of the same solution with the advantage that it would also be able to operate over multiple domains and be based on open standards.

11.3 P2P networks in real-time communications

In the second part of this thesis we have been concentrating on the possibility of using Peer-to-Peer networks in order to improve certain aspects of services such as voice over IP, videoconferencing, and real-time collaboration. The benefits of this association

come from the fact that quality real-time services require significant resources, mostly in terms of bandwidth. Furthermore, distributing registration services algorithms for traversal of Network Address Translation devices should significantly lower the requirements for their support and allow for a larger number of providers and a richer set of services.

We have presented our work on the design of two new protocols that would allow to build and maintain peer-to-peer overlay networks. The first of these, the eXtensible Peer Protocol (XPP), offers a transport mechanism suitable for the maintenance of any Distributed Hash Table (DHT) algorithm. It uses standard mechanisms for session establishment through the Session Initiation Protocol. This makes it particularly friendly to existing Network Address Translation algorithms such as STUN, TURN, and ICE, that can be reused without modification. Another advantage of the protocol comes from the fact that it makes no assumptions on the nature of the DHT and can be used with anything starting from classic ring based ones like Chord to hypercube based like CAN in our case.

The second protocol is an extension of XPP defining its use with the CAN DHT algorithm, called "XPP Extensions for Implementing a Passive P2PSIP Overlay Network based on the CAN Distributed Hash Table". One of the most interesting and original points in this suggestion is the fact that it is up to the network to select the users that would contribute to it as peers, and to decide when exactly this is going to happen. This allows for establishing an overlay network where any peer, at any given time has trust based relations with all its neighbors.

11.3.1 Future work

Our work on using Peer-to-Peer networks in real-time communication has started only recently. An open source implementation of the protocols that we have defined is already available on the SourceForge hosting site [125]. Yet, we have not had the opportunity to fully evaluate its behavior and determine points that need further work. This is where we are heading.

We are planning on creating a second implementation of the protocols in the Java programming language and integrate them into the SIP Communicator communications client. Next we will concentrate on a complete evaluation in real-world conditions and determine points that need further work.

Apart from this we would also have to devise algorithms that allow existing peers to determine which overlay clients should join the overlay network and contribute to

it, without compromising network security at any point. Possibilities for doing this include exchanging peer processing and bandwidth characteristics and, in order to do so securely, providing centralized lists of pre-authorized clients possibly established through a process with human participation.

We also have to work on mechanisms that would allow peers to communicate to all neighbors the authentication credentials of a newly joining peer in addition to simply informing them of its identity.

Determining whether clients should allow other clients to establish SIP outbound flows with them is another important task. If we do decide to allow this, we would also have to choose the mechanism for doing it. One possible option is support for standardized registration of non-adjacent contacts defined in RFC3327 [198].

Finally, the current version of the XPP-PCAN protocol does not define for how long peers should preserve state information about the requests they route. One way of resolving this would be to indicate time in the requests themselves.

Part IV

Annex: The SIP Communicator VoIP and instant messaging client

SIP Communicator (http://sip-communicator.org) is a free/open source application (distributed under the LGPL license), whose development was started at the University Louis Pasteur, Strasbourg, and is now maintained by a thriving community of over 800 members throughout the world. The application allows users to establish audio and video conversations through the Internet via the SIP protocol. It also supports the exchange of instant messages by using some of the most popular IM protocols such as Jabber, AIM/ICQ, MSN, Yahoo! Messenger, Hello or IRC.

In this chapter we describe the history, the features, and the architecture of SIP Communicator. We also enumerate some of its advantages, which according to us make it an excellent tool for both deployment in real-world environments (such as research labs and universities) as well as a very useful tool for the evaluation and testing of solutions produced by research in the field of networking.

11.4 Introduction

Today, an ever growing number of research labs and universities are planning the migration of their communication services over solutions using IP architectures. In addition to the extensibility and ease of configuration that such a migration would give to telephony, it would also make it possible to support new services such as video conversations and real-time collaboration, which are being increasingly demanded in the research community.

Apart from the infrastructure which must be deployed in order to support IP telephony, it is also necessary to select a set of real-time communication applications to use. Quite obviously, conventional services such as voice can be supported almost exclusively by dedicated devices such as hardware phones or VoIP adapters for example. However, in order to fully benefit from all possibilities that come with VoIP and in order to be able to use all available services, it is very important to also deploy and maintain a set of software applications.

Currently SIP Communicator has the complete set of characteristics features that are necessary for a viable communications application. Among others it supports audio and video conversations with SIP [167], most of the widely used communications protocols, and it runs on all operating systems that a Java virtual machine is available for. It is also developed under an free/open source licence. SIP Communicator is also among the very few such applications that support the IPv6 protocol and uses an architecture that allows for a great ease of deployment, update and development of new extensions.

The rest of this chapter is organized as follows: Section 11.5 briefly presents the history of SIP Communicator. Sections 11.6 and 11.7 describe its existing functionalities as those which are under development. Section 11.8 described the software architecture. Section 11.9 exposes the points which, according to us, make SIP Communicator, a client adapted to the universities and research laboratories. Section 11.10 finishes the article by a short conclusion.

11.5 A short history

Work on SIP Communicator started out in the context of my master thesis at the Louis Pasteur University in Strasbourg. I had been interested in IP telephony for several years already so I jumped at the occasion and started working on a VoIP client application.

At the end of my master thesis, encouraged by my advisor at the time - Thomas Noel, professor at the ULP, I decided to continue work on the project and we released it as free software (Open Source) under the LGPL licence. This is how the SIP Communicator project was created on the Java.NET hosting site. Several months later it was also made available through its own domain name: www.sip-communicator.org.

This happened more than four years ago and today it is relatively known and used by many. Some of it users are for example universities and companies such as France Telecom, British Telecom, Alcatel-Lucent, etc.. In the beginning SIP Communicator was mostly a proof-of-concept application and it was therefore mainly used by researchers and developers. Back at the time, very few of its users were non-professional.

In November 2005 we started work on a new version, in order to make the application stabler, more flexible and easily extensible. Work on this version changed both the architecture and the features of the application, as well as its graphic interface and usability.

SIP Communicator is no longer the work of a single person, today a community of developers is thriving around it. The open source model that we have selected has made it possible to rapidly extend the application with many functionalities essential for it success with both business and home users.

The human and hardware resources provided by the Louis Pasteur University have also given us a great liberty and a solid development team working full time on the implementation.

After a little more than a year of intensive development (based on a new modular architecture, also used by large commercial companies such as Mercedes), SIP Communicator now supports most of the instant messaging and telephony protocols in use today, like for example Jabber, ICQ, AIM, GoogleTalk, MSN, Yahoo! Messenger, Bonjour, IRC, RSS and SIP. The application also includes a large number of important features such as the possibility to manage all the contacts of a user (regardless of their protocol), make audio and video calls and record history of calls and messages.

Furthermore, SIP Communicator is much more than a simple application with a set of commonly used services. It is also a modular software architecture developed with the idea of supporting many new services. Such new extensions could be added without the necessity of a complex and heavy development. Finally, SIP Communicator can be regarded as a dynamic plugin architecture, capable of integrating new services on the fly.

As I have already mentioned, a thriving community has developed around the project and an ever growing number of people contribute to it different ways. Over the 800 members of the java.net project, some parts intervene as observers, participate in the discussions suggested on the mailing lists, and share opinions on the design, the development or the planning of the project. Other people contribute in a more specific way by submitting bug reports or fixes. Finally certain members go even further and regularly contribute code and documentation to the project.

During 2007, SIP Communicator was selected as one of the projects allowed to take part in the "Google Summer of Code" program [4]. Needless to say, for us this was a great success and a true recognition by the Open Source community given the other projects that accepted in the program (e.g. Apache, Firefox, OpenOffice, Gimp or Gaim).

11.6 Existing features

After two years of active development, SIP Communicator has reached a level of maturity that is completely sufficient for every day use as a default communications client. Regardless of its experimental stage (alpha version) it already integrates the majority of the features which a user expects to find in this kind of application.

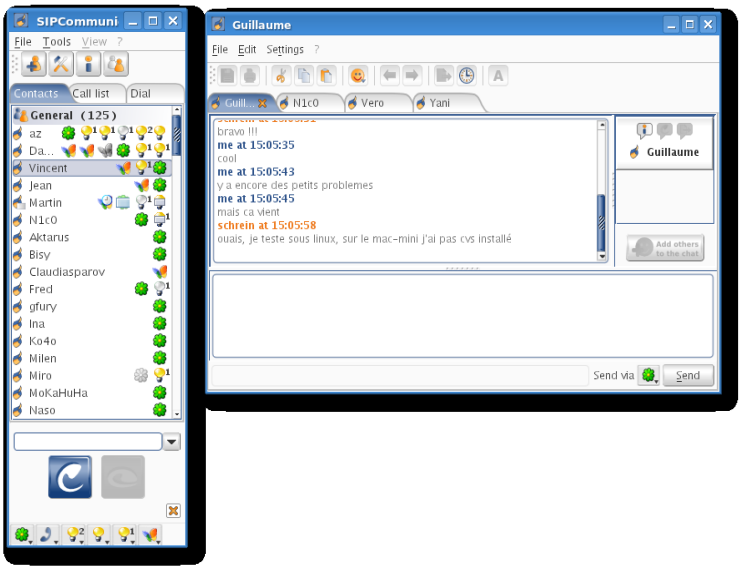


Figure 11.1: SIP Communicator screen shots

11.6.1 Audio video conversations

Since its very beginning in 2003, SIP Communicator has always provided support for audio and video conversations via the SIP protocol. The protocol has now become the de facto standard for voice over IP communication. Nowadays a large variety of products (ex: software phones) and commercial services are using it. Through its support for the SIP protocol, SIP Communicator (as its name indicates it) is compatible with all of these products and is hence usable with the majority of the existing hardware devices.

Recently, in addition to the SIP protocol, SIP Communicator has also integrated, support for audio calls through the XMPP/Jabber extension - Jingle [116].

11.6.2 Instant Messaging

In spite of its popularity for telephony, the SIP protocol is not very popular for instant messaging. Therefore, in addition to the SIP protocol SIP Communicator also supports instant messaging through the following protocols:

- Jabber/GoogleTalk
- AIM

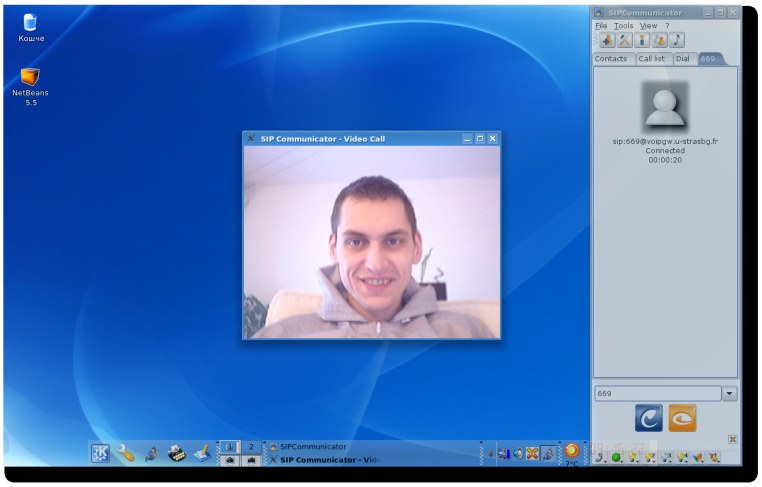


Figure 11.2: A video conversation with SIP Communicator

- ICQ
- MSN
- Bonjour
- Yahoo! Messenger

It allows users to simultaneously access all their instant messaging accounts through the same graphic environment and without losing the functionalities of any of them.

11.6.3 Multiplatform support

SIP Communicator is entirely written in JAVA and can work on most of the major operating systems such as : Linux, BSD, Windows and Mac OS X. Native installers for all of them are available on http://sip-communicator.org

11.6.4 Multichat support with Jabber and IRC

Multichat is, arguably, one of the most popular applications on the Internet since its early age. Today it represents much more than simple entertainment and it is used in remote educational systems or collaborative applications.

Today SIP Communicator allows its users to take part multichat sessions (or chat-rooms) through the Jabber and IRC protocols.

11.6.5 Plugin management

SIP Communicator is built on a modular architecture which makes it possible to activate or disable certain features as well as add new ones, not-provided with the original version or produced by third parties. The current application interface allows the management of all existing modules as well as the installation of new features.

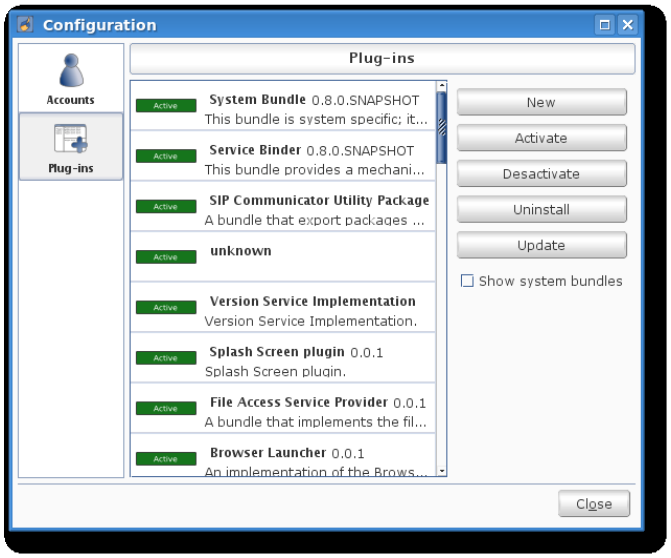


Figure 11.3: Plugin management interface

11.6.6 And other goodies ...

The architecture of SIP Communicator allows the development and the integration of services which are not directly related to interactive communications.

It is for example possible to subscribe for RSS flows and to manage them as if they were contacts. Every new item in this flow is then presented to the user as a new instant message.

Another functionality implemented during Google Summer of Code 2007 makes it possible to add machines in the contact list. The messages which they send and receive are interpreted as SSH commands.

SIP Communicator also supports a so called Meta Contact List. It allows users to gather under the same contact all the addresses that they know a person to use over various transport protocols or accounts.

11.7 Features currently in development

The community of SIP Communicator is increasingly active, which enables us to concentrate on a greater number of functionalities. Here are some, currently developed by the various contributors.

11.7.1 Shared white boards

This functionality proposes to users a shared drawing environment or white board that can simultaneously and interactively modified by several people. It was developed in the context of a student project in the 2007 edition of the Google Summer of Code program. It is currently in the stage of integration and testing.

The shard whiteboards are based on an XMPP extension [90] which is used to transport the contents of the whiteboard using an SVG derived format.

11.7.2 Peer-to-peer communication

Communication services such as audio and video calls, collaboratives applications and multiparty conferences are in general particularly expensive in terms of resources necessary for their deployment. For this reason the SIP Communicator team is currently investigating the possibilities of developing mechanisms that would allow the distribution of these services over overlay peer-to-peer networks.

Such an architecture would make it possible to reduce the costs necessary for the deployment of IP telephony services by using in a distributed way the resources of all participating users.

11.7.3 Secure communication

In most cases, the telephone conversations are intended to remain confidential and their content is supposed to be protected. Security and confidentiality is a feature that is not always offered by IP telephony solutions.

During the summer of 2007 we therefore started the integration of the SRTP protocol in SIP Communicator. Its use will make it possible for users to have the same level of security at the time of the calls with SIP Communicator as the one they are used to with traditional telephony.

The role of SRTP is even more important in a peer-to-peer environment where media flows could be forwarded to third party users, not participating in the call. The lack of encoding in such cases would very seriously compromise the confidentiality of the calls.

11.7.4 Reliable and robust connectivity with the ICE protocol

In the early days of Internet telephony, signalling protocols and their first implementations had to handle much less constraints than today. Network Address Translation devices (NATs) were not common and the machines of the users had in most cases only one IP address (generally public).

Today there are almost no users with public IPv4 addresses, and NATs are present in most network environments. It is increasingly common for a machine to have several IP addresses: IPv4 and IPv6 addresses, VPN addresses or Mobile IPv6 tunnels etc..

This is why the Interactive Connectivity Establishment (ICE) [163] protocol was created. It provides communications clients with a reliable way to negotiate the addresses that will be selected for the exchange of media flows and also allows for a transparent transition towards IPv6.

The development of an ICE stack (the protocol being rather recent, there are currently no Java implementations) and its support in SIP Communicator is one of our priorities.

11.7.5 Support for new communication protocols

In addition to the protocols currently supported by SIP Communicator we are actively working on adding new ones. The Inter-Asterisk eXchange (IAX2) [117] protocol for

example is becoming more and more popular in the world of internet telephony mainly due to its simplicity and its capacity to be adapted to NAT-ed environments. This protocol is intended for client server communication as well as server to server connections (hence its name). It was conceived for the control and transmission of multi-media flows, contrary to the SIP protocol which only handles signaling. Having this double functionality IAX allows for a relatively easy deployment in NATed networks.

Regardless of its youth and current lack of standardization by the IETF, IAX is becoming more and more popular which makes it an excellent candidate for support in SIP Communicator.

11.7.6 Exchanging geographic location

The integration of the geographic location support will allow users of SIP Communicator to visualize the geographical position of the members of their contact lists.

The implementation of this functionality is developed according to the XMPP extension - XEP-0080 [82].

11.7.7 Centralized bundle repositories

As we already mentioned, SIP Communicator allows for fine grained management of the features currently in use as well as the installation and the updates of new plug-ins. We currently work on the installation and the maintenance of an official bundle/plug-in repository which will allow for automatic updates and checks for new versions of the software in the same way as other applications like Firefox or Debian do it.

This repository would also be used to store and distribute modules developed by third parties and not included in the main SIP Communicator version.

11.7.8 The architecture

The main idea behind the architecture of SIP Communicator is allowing for a maximum level of extensibility. For this purpose we have based its development on the modularity concepts developed by the Open Services Gateway initiative (OSGi Alliance) [145] and we use one of its most popular open source implementations, known as Felix [78] created by Richard S. Hall and now maintained by the Apache Software Foundation.

The various modules in the OSGi architecture are called bundles. A bundle can export services (Java interfaces) and thus make them available to the rest of the application modules.

In SIP Communicator, most of the existing features are integrated in the application in the form of OSGi bundles.

This segmentation brings a great flexibility and allows for inexpensive branding and creation of SIP Communicator versions customized for a particular deployment. It is possible to imagine for example that the installation of the software in a particular laboratory only requires support of the Jabber and SIP protocols. Site administrators would thus be able to easily remove the parts not related to these protocols, such as Yahoo! Messenger or MSN, in order not to present users with an overloaded graphic interface.

The rest of this section describes in a more technical detail the various concepts and services in the SIP Communicator architecture.

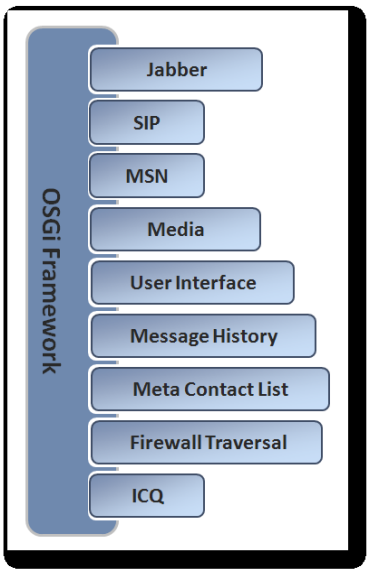


Figure 11.4: Simplified representation of the SIP Communicator architecture.

11.8 The different services

An OSGi service represents a set of functionalities provided by a bundle and available to the rest of the modules in the application. SIP Communicator defines many such services some of which are:

- **ProtocolProviderService**

  Provides an interface to the functionalities of a communications protocol such as, for example, sending and receiving instant messages, subscribing for notifications upon a change in the presence status of other users, initiate, accept or refuse phone calls, etc.. This service is used by the graphic interface, the modules that handle message and call history, and many other plugins.

- **ConfigurationService**

  Gives the possibility to the other parts of the application to record in a persistent way configuration data such as, for example, the size of the various windows of the user interface, the accounts of the user, and other preferences.

- **FileAccessService**

  This service gives the possibility to the other modules of the application and external plug-ins to create temporary, public or private files following the SIP Communicator file storage policy. This makes it possible for example to make sure that different plug-ins which will need to create files that would store details specific to a particular user, would place this files in

  $$/home/username/.sip - communicator$$

  on Linux like systems and in

  $$C : \backslash DocumentsandSettings\backslash username\backslash .sip - communicator$$

  on Windows boxes, without the developer having to handle the 2 cases separately.

- **NotificationService**

  Manages the configuration and the execution of all the notifications following events that occur in SIP Communicator. The service gives users the possibility to personalize the way they are being notified for the arrival of new messages, calls or other events as well as to choose between a sound notification, a pop-up box in the system notification tray or by the execution of a shell command.

- **MetaContactListService**

  Gives access to a joint list of all the contacts available in all the protocols currently configured by the application user. By using this service, the graphic interface or other modules, can easily show to users all their contacts and to modify their properties independently of the underlying protocol.

- **VersionService**

  Allows modules like the startup splash screen the dialog shown by the Help->About menu to query and display the current version of SIP Communicator.

- **GeolocationService**

  Provides information on the current geographical position of the user. According to implementations of this service, this position can be recovered from a configuration file, a GPS device or another means of geographical positioning.

- **MediaService**

  Gives access to the management of the sound and video capture devices and allows for initiation, reception and control of the media flows that are sent or received during a call.

- **MessageHistoryService et CallHistoryService**

  These services provide interfaces which allow other modules to recover the history of the messages sent or received by SIP Communicator as well as that of the incoming and outgoing audio/video calls. These services also allow to execute search queries over the message and call logs, based on key words or time periods.

- **UIService**

  This service represents a means for external plugins to integrate their graphical components into the main user interface of SIP Communicator. It also gives the possibility to subscribe for and receive notifications for certain events triggered by the user such as opening a chat window with a contact, modifying the main contact list window, etc..

The design and implementation of an OSGi service happens through the following two phases:

- the definition of the service. The service itself is quite abstract and quite often a service contains only Java interfaces or abstract classes;

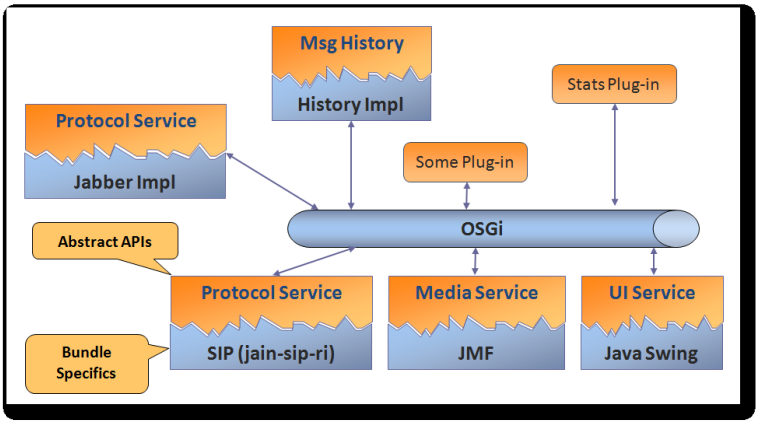


Figure 11.5: A representation of the notion of services and service implementations in SIP Communicator.

- one or more implementations of the service.

A service can have several implementations that could run in a parallel or exclusive manner. The ProtocolProviderService for example is implemented for each protocol supported by SIP Communicator. This way, once the application has been launched, this service will be instantiated once per every protocol account configured by the user.

Other services such as for example the GeolocationService, it is possible to consider different implementations based on alternative positioning systems like GPS, Galileo, or Wi-Fi. According to the devices available on a user station one or the other will be used.

In addition to services and their implementations, one can also have bundles which do nothing but use the existing services without defining new ones. In SIP Communicator, we are using such bundles for all account creation and configuration wizards, the splash screen or the utility that allows to visualize the history of the calls.

11.9 Advantages for research labs and universities

As we already mentioned, SIP Communicator is entirely written in JAVA and is based on a modular architecture (OSGi) which is easy to extend and enrich with new functionalities. This architecture facilitates the evolution of the software and its customization for the specific needs of a research experiment, a lab, or a university deployment. SIP Communicator is developed as an open source project which further guarantees com-

plete freedom in usage and modifications, and also answers common security concerns because it does not have the obscurity often associated to alternative softphones such as Skype and X-Lite. The project is maintained by an ever growing community of developers which reacts quickly to important bug reports and which continuously improves the application.

These and other characteristics make it an ideal candidate for use in research experiments and deployment in universities or laboratories. Following are some more in details on its various advantages:

11.9.1 Support for heterogeneous platform envorinments

As we have mentioned previously, native fitters of SIP Communicator are available for multiple operating systems. Currently we maintain native installers for:

- Debian
- Fedora
- Generic Linux installers
- Mac OS X ;
- Windows ;
- A Java jar installer for any operating system that supports the Java virtual machine.

11.9.2 Cost free use

SIP Communicator is developed and distributed under the open licence source GNU Lesser General Public Licence (LGPL) [113]. This implies that any use of this application in a commercial or research context, is subject to no cost whatsoever.

11.9.3 An open development process

Nowadays most of us know, if only by name, the Skype communications application. Among its numerous advantages one counts the almost no-cost phone calls, a very good

audio quality during conversations and, above all, a very high resilience and tolerance to the various kinds of network infrastructures, which enables the application to work in most of the existing network configurations and in particular from within subnets located behind a Network Address Translation (NAT). However, the creators of Skype have never expressed any will of revealing the code of their application or the mechanics of the network protocol that it uses, thus making its use incompatible with most of the official security policies. Its use has therefore been completely forbidden within most government organizations in both Europe and the United States.

SIP Communicator is completely open source and it is developed under the LGPL licence [113]. With its source code available for everyone to query, its use does not suffer the same black box effects as Skype. Its operation is transparent and secure. It thus answers the security requirements of most institutions and its use is entirely compatible with most of the security policies in use today.

11.9.4 Ease of deployment and maintenance

The deployment of a communications solution based on SIP Communicator only requires a minimal amount of effort for its installation and maintenance. It is enough to install a SIP server in order to handle audio video calls.

Furthermore, with an advanced support for OSGi repositories (currently being developed), an administration team would be able to make all updates automatic by simply installing one such repository on the local network. This would allow for very low intervention delays in urgent situations (e.g., the installation of security updates).

11.9.5 Compliance with existing standards

Within the SIP Communicator community, we strive to achieve a maximum level of compliance with standards for the various functionalities of the application. Our support and development processes give priority to protocols such as Jabber and SIP, that are strictly defined by organizations such the IETF or The XMPP Standards Foundation. This guarantees a significant level of independence for universities or research laboratories, compared to proprietary protocol and gives freedom to network administrators to decide and constantly their usage policies.

11.9.6 Support for a large number of protocols

Today SIP Communicator supports most of the popular communications protocols (the complete list is described in section 11.6.2). It is therefore unlikely for any conflicts to arise when trying to impose and deploy a communications policy in a particular company or institution. In other words, the employees of this company will be able to benefit from the services suggested by the local infrastructure and in the same time continue using their private accounts without having to use two different applications.

11.9.7 Easy to customize and extend

Thanks to the extension mechanisms provided by the OSGi architecture (described in section 11.8), it is particularly easy to modify SIP Communicator in order to adapt it to a particular use case or simply add new features.

A company, an university, or some other institution which wishes to offer tighter integration of their communications application with its existing services can add the missing modules for a negligible development cost.

For example, if the network administrators of a university would like the employees and the students in that university to be able to benefit from the IP voice services and have access to an LDAP directory server (SIP Communicator does not currently support the LDAP protocol), they could develop a lightweight plug-in which is able to handle LDAP research queries and would give users the possibility to use the results returned by these queries for making calls. This plug-in would require a minimal amount of development efforts and would be easily achievable within the scope of a simple student project.

11.9.8 Support for IPv6

SIP Communicator could be deployed in both IPv4 and IPv6 networks. It is therefore compatible with the deployment in most existing research laboratories and universities that are currently migrating to the next generation Internet.

Furthermore, the efforts which are currently being invested in the implementation of the ICE protocol and its integration in SIP Communicator, would guarantee a transparent migration for the user between these two protocols.

11.10 Conclusion

In a context of limited resources typical for the world of research laboratories and universities, IP telephony is drawing more and more interest of the publicly-owned companies and institutions. Most of these organizations and bodies are making short term plans for a migration towards this type of communications solutions.

Taking into account the importance of communication, the set of the tools that are to be deployed in the various installations will have to be maturely considered. They would have to be selected with a number of constraints and in accordance with existing security policies for computer and information systems.

Their deployment and maintenance will have to guarantee the same quality and ease of use as the one offered by conventional telephony. Moreover the migration to this new technology should offer new services in the fields of real-time communication and on-line collaboration.

As an open source project SIP Communicator guarantees a perfect transparency of its operation and thus allows to control the security of all information exchange.

The simplicity of its architecture facilitates the development of extensions and gives the possibility to all kinds of institutions (i.e., companies, universities, and organizations) to easily adapt it to their needs.

The open source community, supporting the project has already proved its reactivity and is thus guaranteeing a follow-up and a continuous evolution of the project.

The usability and the functionalities of the software also make it very attractive for the end-user.

SIP Communicator thus constitutes with our direction a serious candidate as customer of communication within the framework of a deployment of an architecture of voice on IP.

11.11 Acknowledgements

I would personally like to thank all members of the SIP Communicator community for their continuing efforts and for allowing it to become the mature application that it is today.

Since the early days of SIP Communicator, many people have assisted us in many different ways. I would therefore also like to express my gratitude towards Thomas Noel of the Network Research Team in the Louis Pasteur university, and the members of the Center Reseaux Communication in the Osiris network for their constant support.

Part V

Annex: XPP-PCAN details

11.12 Zone state transitions

This section provides a complete description of all zone state transitions that XPP-PCAN peers are expected to implement.

11.12.1 Failure Detection Event

The event is fired when an XPP session between the local peer and one of its neighbors has failed. Upon According to the state that the neighbor zone had in the local zone list the local peer would do one of the following:

State: BORDERING

Set the zone state to SYNCHRONIZING; Start Timer TC1 for this zone and set it to 'tc1'. The 'tc1' value is proportional to the total volume of the zones under the authority of the local peer and always greater than the maximum time required for detecting a failure. Such a value must be calculated using the following formula (where r, t0, t1 and t2 are values used for handling retransmissions and keep alive in XPP [126], v is the total volume under the authority authority of the local peer and V is the volume of the whole hash space):

$$r' = min(ceil(log2(t1/t0)), r)$$
$$tc0 = t0 * 2powr' + (r - r') * t1 + t2$$
$$tc1 = tc0 * (1 + v/V)$$

State: SYNCHRONIZING, STABILIZING, ACQUIRING or OWN

Not possible.

11.12.2 Timer TC1 Event

State: SYNCHRONIZING

Set the zone state to ACQUIRING; start Timer TC2 with value tc2 (default: 6 seconds) and send a TAKEOVER operation request to all neighbors which are also neighbors

of the zone to recover. The Takeover request must be generated as defined in Section 10.4.19.

State: BORDERING, STABILIZING, ACQUIRING or OWN

Not possible.

11.12.3 Timer TC2 Event

State: ACQUIRING

Set the zone state to OWN; the local peer is authoritative for the orphaned zone and the recovery algorithm is terminated. All neighbors are aware of the new owner, but the local peer may need to discover and establish XPP sessions with new neighbors acquired as a result of the recovery, and to notify its neighbors about possible changes in the geometry due to one or more merges between its previous zones and the recovered one.

New neighbors must be discovered sending QUERY operation requests to known neighbors, as defined in Section 10.4.13. The local peer must establish XPP sessions with all discovered neighbors using SIP.

A peer must not establish XPP sessions with peers it does not know; however, after the recovery process, all neighbors of the dead zone will know the identity of the new peer and must accept incoming SIP requests from it.

If the recovered zone is mergeable with any of the zones previously owned by the local peer (i.e. they share an edge in 2-dimensional spaces, a plane in 3-dimensional ones and so on), all the neighbors must be notified of the merge through an UPDATE operation requests, as defined in Section 10.4.15.

State: BORDERING, SYNCHRONIZING, STABILIZING or OWN

Not possible.

11.12.4 TAKEOVER Operation Request Event

State: BORDERING

Leave the zone in a BORDERING state; send back a TAKEOVER operation request on behalf of the peer authoritative for the zone. The TAKEOVER request must be generated as defined in Section 10.4.19.

This event would only occur when the failure is only detected by some of the neighbors, while others are still able to communicate with the peer.

State: SYNCHRONIZING

If, according to the rules defined in Section 10.4.2, the local peer is more appropriate than the one sending the TAKEOVER request, set the zone state to ACQUIRING; cancel Timer TC1, set Timer TC2 with value tc2 (default: 6 seconds) and send a TAKEOVER operation request to all neighbors that are also neighbors of the zone to recover. The TAKEOVER request must be generated as defined in Section 10.4.19.

Otherwise, if the local peer is less appropriate than the one sending the TAKEOVER request, set the zone state to STABILIZING; cancel Timer TC1, set Timer TC3 with value tc3 (default: 4 seconds), temporary store the identity of the peer candidate to take over the zone and send a TAKEOVER operation request to all neighbors which are also neighbors of the zone to recover on behalf of such peer. The TAKEOVER must be generated as defined in Section 10.4.19.

State: ACQUIRING

If, according to the rules defined in Section 10.4.2, the local peer is more appropriate than the one reported in the TAKEOVER request, keep the zone state as ACQUIRING; reset Timer TC2 to value tc2 (default: 6 seconds) and send a TAKEOVER operation request to all neighbors that are also neighbors of the zone to recover. The TAKEOVER request must be generated as defined in Section 10.4.19.

Otherwise, if the local peer is less appropriate than the one reported in the TAKEOVER request, set the zone state to STABILIZING; cancel Timer TC2, start Timer TC3 for a period of tc3 (default: 4 seconds), store the identity of the peer as the best candidate to take over the zone and send a TAKEOVER operation request to all neighbors which

are also neighbors of the zone to recover on behalf of such peer. The TAKEOVER must be generated as defined in Section 10.4.19.

State: STABILIZING

If, according to rules defined in Section 10.4.2, the peer previously stored as the best candidate is more appropriate than the one reported in the TAKEOVER request, keep the zone in a STABILIZING state; reset Timer TC3 to tc3 (default: 4 seconds) and, on behalf of the old candidate, send TAKEOVER requests to all neighbors that are also neighbors of the zone to recover. The TAKEOVER request must be generated as defined in Section 10.4.19.

If the peer previously stored as the best candidate is less appropriate than the one reported in the TAKEOVER request, stay in STABILIZING; reset Timer TC3 with value tc3 (default: 4 seconds), store the identity of the latter as the best candidate and send a TAKEOVER operation request to all neighbors which are also neighbors of the zone to recover on behalf of such peer.

Otherwise, if the peer previously stored as the best candidate is the same as the one reported in the TAKEOVER request, keep the zone in a STABILIZING state and do nothing further.

State: OWN

Not possible.

11.12.5 Timer TC3 Event

State: BORDERING, SYNCHRONIZING, ACQUIRING or OWN

Not possible.

STABILIZING

Set the zone state to BORDERING; set the peer that has qualified as the best candidate as the new owner of the zone.

The local peer may or may not have a connection with the new neighbor; in case it doesn't, it must be ready to accept a request for establishing one.

11.13 XPP Extensions

11.14 Parameters

This section describes the exact encoding, based on the use of XPP TLV options, of all parameters used in the XPP-PCAN protocol.

KEY

Code: 0x8001.

Format: String (Section 11.16.2).

Semantic: the key identifying a record to insert or to retrieve.

VALUE

Code: 0x8002.

Format: String list (Section 11.16.3).

Semantic: the set of peer URIs to traverse for reaching a client or a peer, as described in RFC 3327 [198]. Every URI must be encoding according to the rules defined in [34].

BINDING-ID

Code: 0x8003.

Format: string (Section 11.16.2).

Semantic: the token identifying a key-value pair.

EXPIRES

Code: 0x8004.

Format: integer (Section 11.16.1).

Semantic: the expiration time of a key-value pair, in seconds.

TARGET

Code: 0x8005.

Format: point (Section 11.16.4).

Semantic: the point in the overlay that a request is addressed to.

SPACE

Code: 0x8006.

Format: zone (Section 11.16.5).

Semantic: the bounds of the space used in the current overlay.

OWN-AOR

Code: 0x8007.

Format: String (Section 11.16.2).

Semantic: the SIP URI identifying the user who owns the sending peer, as defined in RFC 3261 [167] and RFC 3986 [34].

OWN-CONTACT

Code: 0x8008.

Format: String (Section 11.16.2).

Semantic: the SIP URI identifying the sending peer.

It is worth noting that, when the peer has restricted connectivity (e.g. it is located in a NAT-ted network), the value of this parameter should be a GRUU [99].

OWN-ZONE

Code: 0x8009.

Format: zone (Section 11.16.5).

Semantic: a zone the sending peer is authoritative for.

YOUR-ZONE

Code: 0x800A.

Format: zone (Section 11.16.5).

Semantic: a zone the receiving peer is going to be authoritative for (usually sent to joining peers).

PEER-AOR

Code: 0x800B.

Format: String (Section 11.16.2).

Semantic: the SIP URI identifying a user whose peer is known by the sending peer, as defined in RFC 3261 [167] and RFC3986 [34].

PEER-CONTACT

Code: 0x800C.

Format: String (Section 11.16.2).

Semantic: the SIP URI identifying a peer known by the sending peer.

It is worth noting that, when the peer has restricted connectivity, the value will be a GRUU [99]. URI values must be encoded as specified in RFC3986 [34]

PEER-ZONE

Code: 0x800D.

Format: zone (Section 11.16.5).

Semantic: a zone a peer known by the sending peer is authoritative for.

PEER-VOLUME

Code: 0x800E.

Format: integer (Section 11.16.1).

Semantic: the total volume owned by a peer known by the sending peer.

DEAD-AOR

Code: 0x800F.

Format: String (Section 11.16.2).

Semantic: the SIP URI identifying a user whose peer is known by the sending peer, as defined in RFC 3261 [167]. URI values must be encoded as specified in RFC3986 [34]

DEAD-CONTACT

Code: 0x8010.

Format: String (Section 11.16.2).

Semantic: the SIP URI identifying a peer known by the sending peer.

It is worth noting that, when the peer has restricted connectivity, the value will be a GRUU [99]. URI values must be encoded as specified in RFC3986 [34]

DEAD-ZONE

Code: 0x8011.

Format: zone (Section 11.16.5).

Semantic: a zone a peer known by the sending peer is authoritative for.

11.15 Operations

This section describes the exact encoding, based on the use of XPP TLV options and the parameters presented in Section 11.14, of all operations used in the XPP-PCAN protocol.

PUT

Code: 0x80000001.

Request parameters:

+[KEY VALUE BINDING-ID EXPIRES

Response parameters:

+[KEY VALUE BINDING-ID EXPIRES

Semantic: insert one or more records in the distributed database.

Routing: requests must be routed to the peer responsible for the first key; responses must follow the reverse path of requests.

GET

Code: 0x80000002.

Request parameters:

+[KEY

Response parameters:

*[KEY VALUE BINDING-ID EXPIRES

Semantic: retrieve one or more records in the distributed database.

Routing: requests must be routed to the peer responsible for the first key; responses must follow the reverse path of requests.

REPLICA

Code: 0x80000003.

Request parameters:

+[KEY VALUE BINDING-ID EXPIRES

Response parameters:

+[KEY VALUE BINDING-ID EXPIRES

Semantic: insert one or more records in the local replica table.

Routing: end-to-end, must NOT be routed.

Query

Code: 0x80000004.

Request parameters:

[TARGET

Response parameters:

+[OWN-CONTACT OWN-AOR OWN-ZONE

*[PEER-CONTACT PEER-AOR PEER-ZONE

Semantic: query the status of the peer authoritative for the zone a given point falls in.

Routing: requests must be routed to the peer authoritative on the zone which include the point encoded in TARGET parameter; responses must follow the reverse path of requests.

Update

Code: 0x80000005.

Request parameters:

+[OWN-CONTACT OWN-AOR OWN-ZONE

*[PEER-CONTACT PEER-AOR PEER-ZONE

Semantic: status update upon a change.

Routing: end-to-end, must NOT be routed.

Join

Code: 0x80000006.

Request parameters:

[SPACE YOUR-ZONE

+[OWN-CONTACT OWN-AOR OWN-ZONE

*[PEER-CONTACT PEER-AOR PEER-ZONE

Semantic: join the overlay becoming authoritative on a given zone.

Routing: end-to-end, must NOT be routed.

Takeover

Code: 0x80000007.

Request parameters:

[DEAD-CONTACT DEAD-AOR DEAD-ZONE

[PEER-CONTACT PEER-AOR PEER-ZONE PEER-VOLUME

Semantic: candidate a peer for taking over a zone.

Routing: end-to-end, must NOT be routed.

11.16 Parameter Formats

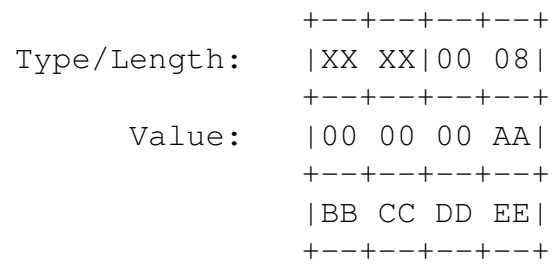
For convenience purposes we define a non-exclusive set of formats that we later use when defining PCAN related XPP parameters.

11.16.1 Integer

Length: the total number of bytes transported in the value. The length must always be divisible by 4.

Value: numeric, encoded in network byte order.

Example:



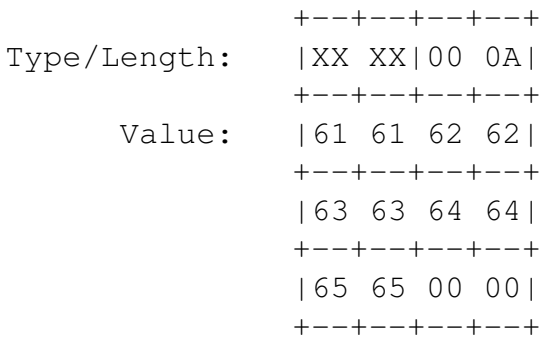
Encoding of an integer parameter with value 0xAABBCCDDEE.

11.16.2 String

Length: the total number of characters. If the length is not divisible by 4; the value field must be padded as defined in [126].

Value: a sequence of ASCII characters.

Example:



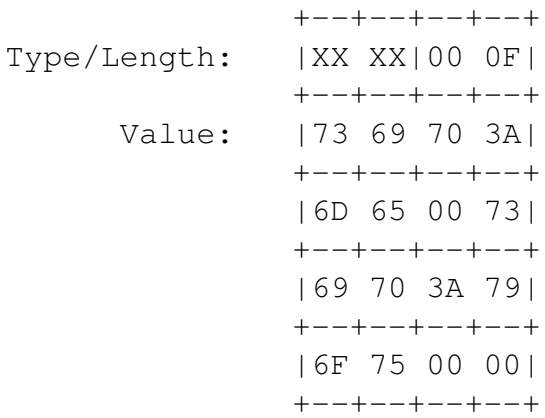
Encoding of a string parameter with value "AABBCCDDEE".

11.16.3 String List

Value: a list of ASCII strings terminated by a byte with value zero.

Length: the total number of characters, including the terminator bytes. If the length is not divisible by 4 it must be padded as defined in [126].

Example:



Encoding of a URI list parameter with value "sip:me", "sip:you".

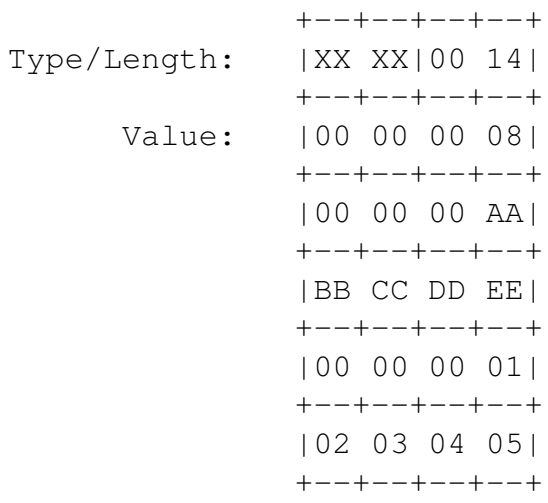


Figure 11.6: Encoding of a point parameter with components 0xAABBCCDDEE and 0x0102030405

11.16.4 Point

Value:

- One 4 byte-long numeric value encoded in network byte order, representing the number of bytes used for encoding each one of the following components. The value of this field must always be divisible by 4.
- A sequence of numeric values representing the components of a multidimensional point, each one encoded in network byte order and taking the number of bytes reported as the the first value of the parameter.

Length: the total number of bytes occupied by the length value and by the components. Such a value must be divisible by 4.

Example:

11.16.5 Zone

Value:

- One 4 byte-long numeric value encoded in network byte order, representing the number of bytes used for encoding each one of the following components. The value of this field must always be divisible by 4;
- A sequence of numeric values representing the components of two multidimensional points, each one encoded in network byte order and taking the number of bytes reported as the the first value of the parameter. The point defined by the first half of values represents the start vertex (the bottom-left corner of a rectangular zone in a bi-dimensional space) and the other defines the end vertex (the top-right corner, in a bi- dimensional zone).

Length: the total length of the Value field. Values of the length field must always be divisible by 4.

Example:

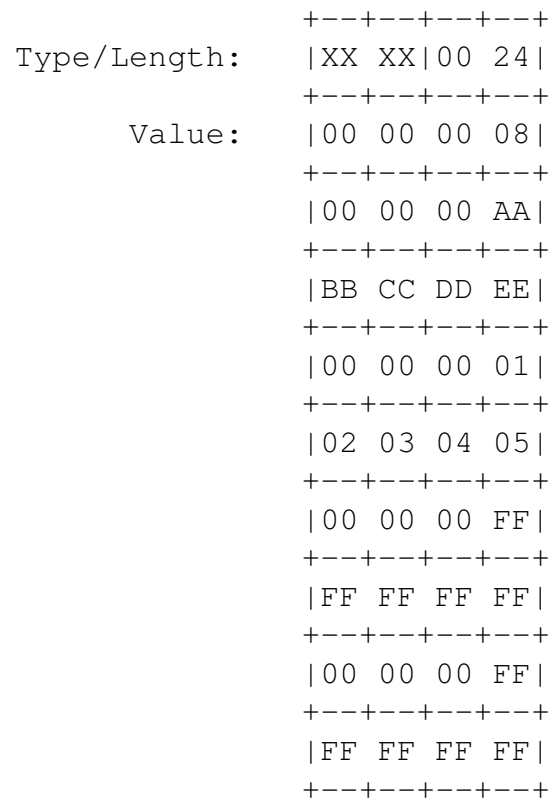


Figure 11.7: Encoding of a zone parameter
The parameter is represented by points (0xAABBCCDDEE, 0x0102030405) and (0xFFFFFFFFFF, 0xFFFFFFFFFF).

Bibliography

[1] Bittorrent–what is a tracker? http://support.bittorrent.com/.

[2] Cisco Aironet 350 Series Access Points Data Sheet (http://www.cisco.com).

[3] Coolstreaming. http://www.coolstreaming.us.

[4] Google Summer of Code (TM), http://code.google.com/soc.

[5] The iris web project. `http://project-iris.net/`.

[6] Joost. http://www.joost.com.

[7] Kazaa. http://www.kazaa.com/.

[8] Manual on Uniform Traffic Control Devices - US Department of transportation. "Federal Highway Administration".

[9] Mobile IPv6 for Linux. http://mobile-ipv6.org.

[10] Napster. http://www.napster.com/.

[11] One Step Closer to IPv6. `http://it.slashdot.org/article.pl?sid=08/02/05/1740221`. Slashdot.

[12] Peer-to-peer session initiation protocol (P2PSIP) IETF working group.

[13] Skype. http://www.skype.com/.

[14] The Network Simulator ns-2. http://www.isi.edu/nsnam.

[15] The SIP Communicator Project `http://sip-communicator.org`.

[16] FIPS 180-1. Secure hash standard. *US Department of Commerce / NIST, National Technical Information Service*, Apr 1995.

[17] IEEE Std. 802.11. 1999 Edition (R2003) (ISO/IEC 8802-11), IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

[18] IEEE Std. 802.1X-2004. IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control, 2004.

[19] L. Ahn, M. Blum, and J. Langford. Telling humans and computers apart automatically. In *Commun. ACM*, volume 47, pages 56–60, New York, NY, USA, 2004. ACM Press.

[20] A. Allavena, A. Demers, and J. Hopcroft. Correctness of a gossip based membership protocol. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 292–301, New York, NY, USA, 2005. ACM.

[21] J. Arkko, V. Devarapalli, and F. Dupont. Using IPsec to Protect Mobile IPv6 Signaling Beteween Mobile Nodes and Home Agents, Internet Engineering Task Force Request for Comments (RFC) 3776, June 2004.

[22] F. Audet and C. Jennings. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787 (Best Current Practice), January 2007.

[23] B. De Vleeschauwer and F. De Turck and B. Dhoedt and P. Demeester. Dynamic algorithms to provide a robust and scalable overlay routing service. Lecture Notes in Computer Science 3961, Proceedings (on CD-ROM) of ICOIN 2006, the International Conference on Information Networking 2006, Vol. LNCS 3961, pp. 945-954.

[24] B. De Vleeschauwer and F. De Turck and B. Dhoedt and P. Demeester. On the construction of QoS enabled overlay networks. Lecture Notes in Computer Science (LNCS 3266), Fifth International Workshop on Quality of Future Internet Services, QofIS 2004, ISSN 0302-9743, ISBN 3-540-23238-9, Springer-Verlag Berlin Heidelberg 2004, J. Sole-Pareta et al (Ed) Vol. LNCS 3266, pp. 164-173.

[25] B. De Vleeschauwer and F. De Turck and B. Dhoedt and P. Demeester. Online management of QoS enabled overlay multicast services. In *IEEE GLOBECOM 2006, the Global Telecommunications Conference*, San Francisco, CA, USA, December 2006.

[26] B. De Vleeschauwer and F. De Turck and B. Dhoedt and P. Demeester. Server Placement and Path Selection for QoS Enabled Overlay Networks. European Transactions on Telecommunications, 2007.

[27] B. De Vleeschauwer and F. De Turck and B. Dhoedt and P. Demeester and M. Wijnants and W. Lamotte. End-to-end QoE Optimization Through Overlay Network Deployment. International Conference on Information Networking ICOIN 2008.

[28] B. Pentland and G. Daley and J. Choi. Router Advertisement Link Identification for Mobile IPv6 Movement Detection - Internet Draft - Work in Progress, Internet Engineering Task Force, draft-pentland-mobileip-linkid-03, October 2004.

[29] S. Baset. A Protocol for Implementing Various DHT Algorithms, Work in Progress, Internet Engineering Task Force, draft-baset-sipping-p2pcommon, June 2006.

[30] S. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. April 2006.

[31] S. Baset and H. Schulzrinne. Peer-to-peer protocol (P2PP). Internet-Draft draft-baset-p2psip-p2pp-00 (Work in Progress), July 2007.

[32] M. Baugher, D. McGrew, M. Naslund, , E. Carrara, and K. Norrman. The secure real-time transport protocol (SRTP). RFC 3711 (Draft Standard), March 2004.

[33] R. Baumann, S. Cavin, and S. Schmid. Voice over IP - security and SPIT. *Swiss Army, FU Br 41, KryptDet Report, University of Berne*, Sept 2006.

[34] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986 (Standard), January 2005.

[35] D. Bryan, M. Zangrilli, and B. Lowekamp. Resource location and discovery. Internet-Draft draft-bryan-p2psip-reload-01 (Work in Progress), July 2007.

[36] B.T. Loo and T. Condie and J. M. Hellerstein and P. Maniatis and T. Roscoe and I. Stoica. Implementing Declarative Overlays. 20th ACM Symposium on Operating Systems Principles (SOSP).

[37] C. Jennings and R. Mahy. Managing Client Initiated Connections in the Session Initiation Protocol (SIP) Work in Progress, Internet Engineering Task Force, draft-ietf-sip-outbound, January 2007.

[38] P. Calhoun, B. O'Hara, R. Suri, N. Cam Winget, S. Kelly, M. Williams, and S. Hares. Light Weight Access Point Protocol, Work in Progress, Internet Engineering Task Force draft-ohara-capwap-lwapp-03.txt, June 2005.

[39] A. T. Campbell and et al. The genesis kernel: A virtual network operating system for spawning network architectures. In *Open Architectures and Network Programming Proceedings, 1999. OPENARCH '99. 1999 IEEE Second Conference*, New York, USA, March 1999.

[40] C. Castelluccia. HMIPv6: A hierarchical mobile IPv6 proposal. In *ACM Mobile Computing and Communication Review (MC2R)*, April 2000.

[41] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. *5th Usenix Symposium on Operating Systems Design and Implementation*, Dec 2002.

[42] A. Cellerier and Al. VideoLAN - VLC media player. `http://www.videolan.org`.

[43] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 407–418, New York, NY, USA, 2003. ACM.

[44] G. Chesson, M. Renzmann, and Sam Leffler. Multiband Atheros Driver for Wi-Fi (MADWIFI), `http://madwifi.org`.

[45] G. Combs and al. The Network Protocol Analyser Ethereal, `http://www.ethereal.com`.

[46] G. Combs and al. Wireshark Network Protocol Analyzer, `http://www.wireshark.org`.

[47] T. Condie, V. Kacholia, S. Sankararaman, J. M. Hellerstein, and P. Maniatis. Maelstorm: Churn as shelter. *University of California Berkeley, Technical report UCB/EECS-2005-11*, Nov 2005.

[48] T. Condie, S. Kamvar, and H. Garcia-Molina. Adaptive peer-to-peer topologies. In *International Conference on Peer-to-Peer Computing (P2P)*, Zurich, Switzerland, 2004.

[49] E. Cooper, A. Johnston, and P. Matthews. A distributed transport function in P2PSIP using HIP for multi-hop overlay routing. Internet-Draft draft-matthews-p2psip-hip-hop-00 (Work in Progress), June 2007.

[50] S. Corner. Verizon business plans 18 month transition to IPv6. IT Wire, http://www.itwire.com/content/view/14635/127/.

[51] D. Bryan and B. Lowekamp and C. Jennings. dSIP: A P2P Approach to SIP Registration and Resource Location, Work in Progress, Internet Engineering Task Force, draft-bryan-p2psip-dsip-00, 2007.

[52] D. C. Verma. *Content Distribution Networks: An Engineering Approach*. John Wiley & Sons, Inc.

[53] D. Chopra and H. Schulzrinne and E. Marocco and E. Ivov. Security Issues and Solutions in P2P Networks and Their Applicability in Communication Overlays. In submission for IEEE Surveys and Tutorials, 2008.

[54] D. G. Andersen and H. Balakrishnan and M. Frans Kaashoek and R. Morris. Resilient overlay networks. Symposium on operating Systems Principles (SPECTS), pages 131-145.

[55] D. Hales and S. Arteconi. SLACER: A Self-Organizing Protocol for Coordination in P2P Networks. IEEE Intelligent Systems, Vol. 22, No. 2.

[56] D. Willis and D. Bryan and P. Matthews and E. Shim, Work in Progress, Internet Engineering Task Force, draft-willis-p2psip-concepts-01. Concepts and Terminology for Peer to Peer SIP, 2006.

[57] F. Dabek, M. Frans Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, October 2001.

[58] E. Damiani, D. C. Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. *Conference on Computer and Communications Security*, Nov 2002.

[59] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson. Sybil-resistant DHT routing. *Tenth European Symposium on Research in Computer Security*, 3679, Sep 2005.

[60] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFC 5095.

[61] Free deploie l'IPv6. `http://www.iliad.fr/presse/2007/CP_IPv6_121207.pdf`. www.iliad.fr.

[62] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. Network Mobility (NEMO) Basic Support Protocol, Internet Engineering Task Force Request for Comments (RFC) 3963, January 2005.

[63] J. R. Douceur. The sybil attack. *Revised paper, 1st International Workshop. Peer-to-Peer Systems, Lecture Notes in Computer Science*, 2429, Mar 2002.

[64] R. Droms. Dynamic host configuration protocol. RFC 2131 (DRAFT STANDARD), March 1997.

[65] E. Ivov and J. Montavont and T. Noel. Thorough Empirical Analysis of the IETF FMIPv6 protocol over IEEE 802.11 networks. IEEE Wireless Communications Magazine, Special issue on "Architectures and Protocols for Mobility Management in All-IP Mobile Networks", April 2008.

[66] E. Ivov and M. Andre. The FMIPv6 Open Source Implementation Suite. http://www.fmipv6.org, 2005.

[67] E. Ivov and T. Noel. Optimizing SIP Application Layer Mobility over IPv6 Using Layer 2 Triggers. IEEE Vehicular Technology Conference VTC'04, September 2004.

[68] D. Eastlake 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1). RFC 3174 (Informational), September 2001. Updated by RFC 4634.

[69] C. Perkins (éditeur). IP Mobility Support for IPv4, Internet Engineering Task Force Request for Comments (RFC) 3344, August 2002.

[70] R. Koodli (éditeur). Fast Handovers for Mobile IPv6, Internet Engineering Task Force Request for Comments (RFC) 4068, June 2005.

[71] L. Eggert and G. Fairhurst. UDP Usage Guidelines for Application Designers, Work in Progress, Internet Engineering Task Force, draft-ietf-tsvwg-udp-guidelines-01, 2007.

[72] Ekahau. Ekahau Positioning Engine for WLAN based navigation, `http://www.ekahau.com`.

[73] F. Forster and H. DeMeer. Discovery of Web Services with a P2P Network. Computational Science - ICCS 2004, 4th International Conference.

[74] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE), Internet Engineering Task Force Request for Comments (RFC) 2784, March 2000.

[75] G. Daley and R. Nelson. Duplicate Address Detection Optimization using IPv6 Multicast Listener Discovery - Internet Draf - work in progress - draft-daley-ipv6-mcast-dad-02.txt, February 2003.

[76] S. Guha. NAT Behavioral Requirements for TCP, Work in Progress, Internet Engineering Task Force, draft-ietf-behave-tcp-06, 2007.

[77] V. Gurbani, F. Audet, and D. Willis. The SIPSEC uniform resource identifier (URI). Internet-Draft draft-gurbani-sip-sipsec-01 (Work in Progress), June 2007.

[78] R. S. Hall. Apache felix `http://felix.apache.org`. Apache Software Foundation.

[79] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. RFC 4566 (Proposed Standard), July 2006.

[80] H. Hartenstein, M. Liebsch, X. Perez-Costa, and R. Schmitz. A MIPv6, FMIPv6 and HMIPv6 handover latency study: analytical approach. In *IST Mobile & Wireless Telecommunications Summit*, Thessaloniki, Greece, 2002.

[81] T. H. Haveliwala and S. D. Kamvar. The second value eigenvalue of the google matrix. *Technical Report, Stanford University*, 2003.

[82] J. Hildebrand and P. Saint-Andre. XEP-0080: User Location. XMPP Standards Foundation.

[83] HostAP. Host ap - linux driver for prism2/2.5/3. http://hostap.epitest.fi/.

[84] R. Hsieh, A. Seneviratne, H. Soliman, and K. El-Malki. Performance analysis on hierarchical mobile IPv6 with fast-handoff over end-to-end tcp. In *GLOBECOM*, Taipei, Taiwan, 2002.

[85] R. Hsieh, Z.G. Zhou, and A. Seneviratne. S-MIP: A Seamless Handoff Architecture for Mobile IP. In *Proceedings of the the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocomm'03)*, San Francisco, USA, April 2003.

[86] I. Abraham and B. Awerbuch and Y. Azar, Y. Bartal, D. Malkhi and E. Pavlov. A Generic Scheme for Building Overlay Networks in Adversarial Scenarios. Proceedings of the 17th International Symposium on Parallel and Distributed Processing.

[87] I. Clarke and O. Sandberg and B.Wiley and T.W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. Workshop on Design Issues in Anonymity and Unobservability, pages 311-320.

[88] IEEE. Draft 5 Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation. IEEE Draft 802.1f/D5, January 2003.

[89] The Internet Engineering Task Force. `http://www.ietf.org`.

[90] H.J. Imbens. XEP-0113: Simple Whiteboarding. XMPP Standards Foundation.

[91] International Communications Union (ITU). Transmission Systems and Media, Digital Systems and Networks, Recommendation G.711: Pulse Code Modulation (PCM) of Voice Frequencies, 1988.

[92] E. Ivov and T. Noel. Soft Handovers over 802.11b with Multiple Interfaces. In *Proceedings of the 2nd International Symposium on Wireless Communication Systems (ISWCS'05)*, pages 549–554, Siena, Italy, September 2005.

[93] E. Ivov and T. Noel. An Experimental Performance Evaluation of the IETF FMIPv6 Protocol over IEEE 802.11 WLANs. In *Proceedings of the IEEE Conference on Wireless Communications and Networking (WCNC'06)*, Las Vegas, USA, April 2006.

[94] E. Ivov and T. Noel. An Experimental Performance Evaluation of the IETF FMIPv6 Protocol over IEEE 802.11 WLANS. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'06)*, volume 1, pages 568–574, April 2006.

[95] J. Choi and D. Shin. Fast Router Discovery with RA Caching in AP - Internet Draft - work in progress. draft-jinchoi-mobileip-frd-01.txt, February 2003.

[96] J. Doyle and et al. The Robust Yet Fragile Nature of the Internet. Proceedings of the National Academy of Science, Vol.102, No. 41.

[97] J. Gao and P. Steenkiste. Design and evaluation of a distributed scalable content discovery system. IEEE Journal on Selected Areas in Communications, Vol. 22, No. 1, pp. 54-56.

[98] J. Montavont and E. Ivov and T. Noel and K. Guillouard. Analysis of a Geolocation-based FMIPv6 Extension for Next Generation Wireless LANs. Ubiquitous Computing And Communication Journal [ISSN 1992-8424], Volume 2, Number 5, October 2007.

[99] J. Rosenberg. Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP), Work in Progress, Internet Engineering Task Force, draft-ietf-sip-gruu-12, March 2007.

[100] J. Rosenberg. TCP Candidates with Interactive Connectivity Establishment (ICE), Work in Progress, Internet Engineering Task Force, draft-ietf-mmusic-ice-tcp-05, 2007.

[101] J. Rosenberg and R. Mahy and P. Matthews. Traversal Using Relays around NAT (TURN): Relay Extensions to "Session Traversal Utilities for NAT (STUN), Work in Progress, Internet Engineering Task Force, draft-ietf-behave-turn-06, January 2008.

[102] C. Jennings, J. Rosenberg, and E. Rescorla. Address settlement by peer to peer. Internet-Draft draft-jennings-p2psip-asp-00 (Work in Progress), July 2007.

[103] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6, Internet Engineering Task Force Request for Comments (RFC) 3775, June 2004.

[104] K. Keahey. Computational Grids in Action: The National Fusion Collaboratory. Future Generation Computer Systems, Vol. 18, No. 8, pp. 1005-1015.

[105] S. D. Kamvar, H. Garcia-Molina, and M. T. Schlosser. The eigentrust algorithm for reputation management in P2P networks. *12th international conference on World Wide Web*, May 2003.

[106] G. Karlsson. Quality Requirements for Multimedia Network Services. Lulea, Sweden, June 1996.

[107] J. Kempf, M. M Khalil, and B. Pentland.
IPv6 Fast Router Advertisement - Internet Draft - work in progress, Internet Engineering Task Force, draft-mkhalil-ipv6-fastra-05.txt. `http://tools.ietf.org/html/draft-mkhalil-ipv6-fastra-05`, 2004.

[108] J. Kempf, J. Wood, and G. Fu. Fast Mobile IPv6 Handover Packet Loss Performance: Measurement for Emulated Real Time Traffic. In *Proceedings of the IEEE Conference on Wireless Communications and Networking (WCNC'03)*, March 2003.

[109] Y. Kim, D. Mazzocchi, and G. Tsudik. Admission control in peer groups. *Second IEEE International Symposium on Network Computing and Applications*, Apr 2003.

[110] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for MANET. *International Conference on Network Protocols*, Nov 2001.

[111] Rajeev Koodli and Charles E. Perkins. Fast handovers and context transfers in mobile networks. *SIGCOMM Comput. Commun. Rev.*, 31(5):37–47, 2001.

[112] S. Y. Lee, O-H. Kwon, J. Kim, and S. J. Hong. A reputation management system in structured peer-to-peer networks. *Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, 2005.

[113] GNU Lesser General Public License (LGPL). `http://www.gnu.org/licenses/lgpl.html`. Free Software Foundation.

[114] J. Liang, R. Kumar, Y. Xi, and K. Ross. Pollution in P2P File Sharing Systems. In *IEEE Infocom*, Miami, FL, USA, March 2005.

[115] J. Lorchat and T. Noel. Power Performance Comparison of Heterogeneous Wireless Network Interfaces. *VTC 2003 Fall*, 2003.

[116] S. Ludwig, J. Beda, P. Saint-Andre, R. McQueen, S. Egan, and J. Hildebrand. XEP-0166: Jingle. XMPP Standards Foundation.

[117] M. A. Spencer and B. Capouch and E. Guy and F. Miller and K. C. Shumard. IAX2: Inter-Asterisk eXchange Version 2. (Internet Draft) Internet Engineering Task Force.

[118] M. Jelasity and A. Montresor and O. Babaoglu. Gossip-based Aggregation in Large Dynamic Networks. In *ACM Transactions on Computer Systems, 23(3):219-252*, August 2005.

[119] M. Jelasity and O. Babaoglu. T-Man: Gossip-based Overlay Topology Management. In *Proceedings of Engineering Self-Organising Applications*, 2005.

[120] M. Jelasity and R. Guerraoui and A.-M. Kermarrec and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, 79-98*, 2004.

[121] M. Stokes. Gnutella2 specification document - first draft, Gnutella2 Web site `http://www.gnutella2.com/gnutella2_draft.htm`, 2003.

[122] M. Wijnants and B. Cornelissen and W. Lamotte and B. De Vleeschauwer. An Overlay Network Providing Application-Aware Multimedia Services. 2nd International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA 2006).

[123] O. Malik. Ooma wants voice to be free. http://gigaom.com/2007/07/19/ooma/, July 2007.

[124] K. El Malki and H. Soliman. Simultaneous Bindings for Mobile IPv6 Fast Handovers, Work in Progress, Internet Engineering Task Force draft-elmalki-mobileip-bicasting-v6-06.txt, July 2005.

[125] E. Marocco. Sipdht: A sip-based distributed hash-table. http://sipdht.sourceforge.net.

[126] E. Marocco and E. Ivov. XPP: Extensible peer protocol. Internet-Draft draft-marocco-p2psip-xpp-01 (Work in Progress), December 2007.

[127] E. Marocco and E. Ivov. XPP extensions for implementing a passive P2PSIP overlay network based on the CAN distributed hash table. Internet-Draft draft-marocco-p2psip-xpp-pcan-01 (Work in Progress), June 2007.

[128] S. Marti, P. Ganesan, and H. Garcia-Molina. SPROUT: P2P routing with social networks. *First International Workshop on Peer-to-Peer and Databases*, Mar 2004.

[129] P. Maymounkov and D. Mazi. Kademlia: A peer-to-peer information system based on the xor metric. *First International Workshop on Peer-to-Peer Systems*, Mar 2002.

[130] P. McCann. Mobile IPv6 Fast Handovers for 802.11 Networks. RFC 4260 (Informational), November 2005.

[131] A. McCue. Bookie reveals 100,000 cost of denial-of-service extortion attacks. *silicon.com*, june 2004.

[132] D. L. Mills. Network Time Protocol (Version 3), Specification, Implementation and Analysis, Internet Engineering Task Force Request for Comments (RFC) 1305, March 1992.

[133] A. Mishra, M. Shin, and W. Arbaugh. An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process. *SIGCOMM Comput. Commun. Rev.*, 33(2):93–102, 2003.

[134] J. Montavont, E. Ivov, and T. Noel. Analysis of Mobile IPv6 Handover Optimizations and their Impact on Real-Time Communication. In *Proceedings of the IEEE Conference on Wireless Communications and Networking (WCNC'07)*, Honk Kong, China, March 2007.

[135] J. Montavont, J. Lorchat, and T. Noel. Deploying NEMO: A Practical Approach. In *Proceedings of the 6th International Conference on ITS Telecommunications (ITST'06)*, Chengdu, China, June 2006.

[136] N. Montavont and T. Noel. Handover Management for Mobile Nodes in IPv6 Networks. *IEEE Communication Magazine*, 40(8):38–43, August 2002.

[137] N. Montavont and T. Noel. Analysis and Evaluation of Mobile IPv6 Handovers over Wireless LAN. *Mobile Networking and Applications (MONET), special issue on Mobile Networking through IPv6 or IPv4*, 8(6):643–653, November 2003.

[138] N. Montavont and T. Noel. Anticipated Handover over IEEE 802.11 Networks. In *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2005)*, Montreal, Canada, August 2005.

[139] N. Moore. Optimistic Duplicate Address Detection (DAD) for IPv6, Internet Engineering Task Force Request for Comments (RFC) 4429, April 2006.

[140] N. Nakajima, A. Dutta, S. Das, and H. Schulzrinne. Handoff Delay Analysis and Measurement for SIP based mobility in IPv6. In *IEEE International Conference on Communications, ICC'03*, volume 2, pages 1085–1089, May 2002.

[141] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). RFC 2461 (Draft Standard), December 1998.

[142] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6), Internet Engineering Task Force Request for Comments (RFC) 2461, December 1998.

[143] V. Nuorvala, H. Petander, and A. Tuominen. Mobile IP for Linux (MIPL), `http://mobile-ipv6.org`.

[144] K. Ohta, S. Micali, and L. Reyzin. Accountable subgroup multisignatures. *ACM conference on Computer and Communications Security*, Nov 2001.

[145] Open services gateway initiative (osgi alliance). `http://www.osgi.org`.

[146] P. Prasithsangaree and P. Krishnamurthy and P. Chrysanthis. On indoor position location with wireless LANs, 2001.

[147] P. T. Eugster and R. Guerraoui and S. B. Handurukande and P. Kouznetsov and A. M. Kermarrec. Lightweight Probabilistic Broadcast. ACM Transanctions on Computer Systems, 21(4): 341-374, 2003.

[148] S. Pack and Y. Choi. Fast inter-ap handoff using predictive-authentication scheme in a public wireless lan. In *IEEE Networks, (Joint ICN 2002 and ICWLHN 2002)*, pages 15–26, Atlanta, USA, August 2002.

[149] S. Pack and Y. Choi. Performance Analysis of Fast Handover in Mobile IPv6 Networks. In *The Eighth International Conference on Personal Wireless Communications (IFIP PWC 2003)*, Venice, Italy, 2003.

[150] W-K. Poon and R. K. C. Chang. Robust forwarding in structured peer-to-peer overlay networks. *SIGCOMM*, August 2004.

[151] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[152] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFC 3168.

[153] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The bittorent P2P file-sharing system: Measurements and analysis. *4th International Workshop on Peer-to-Peer Systems*, February 2005.

[154] QUAGGA. Quagga routing software suite, `http://www.quagga.net`.

[155] R. Albert and A. Barabasi. Statistical Mechanics of Complex Networks. Reviews of. Modern. Physics, 74(47).

[156] R. Braynard and D. Kostic and A. Rodriguez and J. Chase and A. Vahdat. Opus: an Overlay Peer Utility Service. Proceedings of the 5th International Conference on Open Architectures and Network Programming (OPENARCH).

[157] I. Ramani and S. Savage. SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, pages 675–684, Miami, USA, March 2005.

[158] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000.

[159] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. *SIGCOMM*, 2001.

[160] E. Rescorla and N. Modadugu. Datagram Transport Layer Security. RFC 4347 (Proposed Standard), April 2006.

[161] RIPE Network Coordination Centre. RIPE Community Resolution on IPv4 Depletion and Deployment of IPv6. RIPE 55 Meeting, October 2007.

[162] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. Technical Report, University of Chicago, 2001.

[163] J. Rosenberg. Interactive connectivity establishment (ICE): A protocol for network address translator (NAT) traversal for offer/answer protocols. Internet-Draft draft-ietf-mmusic-ice-18 (Work in Progress), September 2007.

[164] J. Rosenberg. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). draft-ietf-behave-rfc3489bis (Work in Progress), 2007.

[165] J. Rosenberg and H. Schulzrinne. An Offer/Answer Model with Session Description Protocol (SDP). RFC 3264 (Proposed Standard), June 2002.

[166] J. Rosenberg and H. Schulzrinne. Session Initiation Protocol (SIP): Locating SIP Servers. RFC 3263 (Proposed Standard), June 2002.

[167] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session initiation protocol. RFC 3261 (Draft Standard), June 2002.

[168] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489 (Proposed Standard), March 2003.

[169] H. Rowaihy, W. Enck, P. McDaniel, and T. La Porta. Limiting sybil attacks in structured peer-to-peer networks. *Technical Report NAS-TR-0017-2005, Network and Security Research Center, Department of Computer Science and Engineering*, Jul 2005.

[170] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, pages 329–350, Heidelberg, Germany, 2001.

[171] S. A. Baset and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol, 2004.

[172] S. Androutsellis-Theotokis and D. Spinellis. A Survey of P2P Content Distribution Techniques. ACM Computing Surveys, 36(4):335-371.

[173] S. Shin and A. Forte and A. Singh Rawat and H. Schulzrinne. Reducing MAC layer handoff latency in IEEE 802.11 wireless LANs. In *Proceedings of the 2nd International Workshop on Mobility Management & Wireless Access Protocols (MobiWac'04)*, pages 19–26, Philadelphia, USA, 2004. ACM Press.

[174] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920 (Proposed Standard), October 2004.

[175] D. Sandras and Al. Gnomemeeting: an open source VoIP and video conferencing application for GNOME, `http://www.gnomemeeting.org`.

[176] N. Saxena, G. Tsudik, and J. H. Yi. Admission control in peer-to-peer: Design and performance evaluation. *Security of Ad Hoc and Sensor Networks*, 2003.

[177] C. Scheideler. How to spread adversarial nodes?: Rotate! *Thirty-Seventh Annual ACM Symposium on Theory of Computing*, May 2005.

[178] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, Internet Engineering Task Force Request for Comments (RFC) 3550, July 2003.

[179] H. Schulzrinne and E. Wedlund. Application-layer mobility using SIP. *ACM SIGMOBILE Mobile Computing and Communications Review*, 1(2), Feb 2001.

[180] J. Seedorf. Security challenges for peer-to-peer SIP. *IEEE Network*, 20, 2006.

[181] J. Seedorf. Using cryptographically generated SIP-URIs to protect the integrity of content in P2P-SIP. *VoIP Security Workshop*, June 2006.

[182] K. Singh and H. Schulzrinne. Peer-to-peer internet telephony using SIP. *International Workshop, Network and Operating System support for Digital Audio and Video*, June 2005.

[183] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. *1st International Workshop on Peer-to-Peer Systems*, March 2002.

[184] H. Soliman, C. Catelluccia, K. El Malki, and L. Bellier. Hierarchical Mobile IPv6 Mobility Management (HMIPv6), Internet Engineering Task Force Request for Comments (RFC) 4140, August 2005.

[185] S. S. Soliman and C. E. Wheatley. Geolocation Technologies and Applications for Third Generation Wireless. Wireless Communications and Mobile Computing, Vol. 2, No. 3, pp. 229-251, 2002.

[186] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.

[187] SUN Microsystems. JXTA Technology `http://www.sun.com/software/jxta/`, March 2005.

[188] Cisco Systems. Cisco Catalyst 6500 Series Wireless LAN Services Module: White Paper, http://cisco.com.

[189] Cisco Systems. Cisco Fast Secure Roaming Application Note, http://cisco.com.

[190] Cisco Systems. Configuring WDS, Fast Secure Roaming and Radio Management, http://cisco.com.

[191] F. Teraoka, K. Gogo, K. Mitsuya, R. Shibui, and K. Mitani. Unified L2 Abstractions for L3-Driven Fast Handover, Work in Progress, Internet Engineering Task Force, draft-koki-mobopts-l2-abstractions-05.txt, June 2006.

[192] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration, Internet Engineering Task Force Request for Comments (RFC) 2462, December 1998.

[193] J. Tourrilhes. The Linux Wireless Extensions and Wireless Tools. http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux, 2003.

[194] E. Uzun, M. R. Pariente, and A. A. Selpk. A reputation-based trust management system for P2P networks. *International Symposium on Cluster Computing and the Grid*, Apr 2004.

[195] J. Vatn. An Experimental study of IEEE 802.11b Handover performance and its Effect on Voice Traffic. In *Technical Report TRITA-IMIT-TSLAB R 03:01, Telecommunication Systems Laboratory, Department of Microelectronics and Information Technology, KTH, Royal Institute of Technology*, Stockholm, Sweden, July 2003.

[196] H. Velayos and G. Karlsson. Techniques to Reduce the IEEE 802.11b Handoff Time. In *Proceedings of the IEEE International Conference on Communications (ICC'04)*, volume 7, pages 3844–3848, Paris, France, June 2004.

[197] R. Vida and L. Costa. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. RFC 3810 (Proposed Standard), June 2004. Updated by RFC 4604.

[198] D. Willis and B. Hoeneisen. Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts. RFC 3327 (Proposed Standard), December 2002.

[199] Y. Chu and S. Rao and H. Zhang. A case for end system multicast. SIGMETRICS 2000: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, 2000, pp. 1-12.

[200] Z. Li and P. Mohapatra. QRON: QoS-aware routing in overlay networks. IEEE Journal on Selected Areas in Communications 22 (2004) 29-40.

[201] X. Zhang, S. Chen, and R. Sandhu. Enhancing data authenticity and integrity in P2P systems. *Internet Computing*, September 2005.

[202] P. Zimmermann. Pretty good privacy: public key encryption for the masses. In *Building in big brother: the cryptographic policy debate*, pages 93–107, New York, NY, USA, 1995. Springer-Verlag New York, Inc.

Chapter 12

List of publications

International journals

[1] **Security Issues and Solutions in P2P Networks and Their Applicability in Communication Overlays**, Dhruv Chopra, Henning Schulzrinne, Enrico Marocco, Emil Ivov, In submission for IEEE Surveys and Tutorials, 2008

[2] **Thorough Empirical Analysis of the IETF FMIPv6 protocol over IEEE 802.11 networks**, Emil Ivov, Julien Montavont, Thomas Noel, Accepted for publication in IEEE Wireless Communications Magazine, Special issue on "Architectures and Protocols for Mobility Management in All-IP Mobile Networks", April 2008

[3] **Analysis of a Geolocation-based FMIPv6 Extension for Next Generation Wireless LANs**, Julien Montavont, Emil Ivov, Thomas Noel and Karine Guillouard, Ubiquitous Computing And Communication Journal [ISSN 1992-8424], Volume 2, Number 5, October 2007.

International Conferences

[4] **Analysis of Mobile IPv6 Handover Optimizations and their Impact on Real-Time Communications**, Julien Montavont, Emil Ivov and Thomas Noel, IEEE Conference on Wireless Communications and Networking (WCNC'07), Honk Kong, Chine, 11-15 Mars 2007.

[5] **An Experimental Performance Evaluation of the IETF FMIPv6 Protocol over IEEE 802.11 WLANs**, Emil Ivov and Thomas Noel, Wireless Communications and Networking Conference.(WCNC'06), Las Vegas, USA, April 2006.

[6] **Soft Handovers over 802.11 with Multiple Interfaces**, Emil Ivov and Thomas Noel,2nd International Symposium on Wireless Communication Systems 2005 (ISWCS2005) Siena, Italy, September 2005.

[7] **Optimizing SIP Application Layer Mobility over IPv6 Using Layer 2 Triggers**, Emil Ivov and Thomas Noel, IEEE Vehicular Technology Conference VTC'04 Los Angeles, USA, September 2004.

National Conferences

[8] **SIP Communicator - Un outil open source de communication sur IP adapte a nos laboratoires et a nos universites**, Emil Ivov, Jean-Marc Muller, Journees Reseaux, Novembre, 2007, Strasbourg.

Internet Drafts

[9] **XPP Extensions for Implementing a Passive P2PSIP Overlay Network based on the CAN Distributed Hash Table**, Enrico Marocco and Emil Ivov, Work in Progress, Internet Engineering Task Force, draft-marocco-p2psip-xpp-pcan-01, December 2007.

[10] **Extensible Peer Protocol (XPP)**, Enrico Marocco and Emil Ivov, Work in Progress, Internet Engineering Task Force, draft-marocco-p2psip-xpp-01, December 2007.

Internal Reports

[11] **Current Optimization Techniques of the Mobile IPv6 Handover Process**, Emil Ivov and Thomas Noel, August 2004.

[12] **Dossier d'Experimentation Fast Mobile IPv6**, Emil Ivov, and Thomas Noel, August 2006.

[13] **Fast IPv6 Handovers for Multimedia Terminals, Project Closure Report**, Emil Ivov, and Thomas Noel, August 2006.