ISCTE IUL
Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

Simultaneous Multi-Access in Heterogeneous Mobile Networks

Hugo Miguel Almeida Alves

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia de Telecomunicações e Informática

Orientador:

Professor Doutor Rui Marinheiro, Professor Auxiliar,
ISCTE-IUL

Setembro 2015

Abstract

The exponential growth of the number of multihomed mobile devices is changing the way how we connect to the Internet. Unfortunately, it is not yet easily possible to a multihomed device to be simultaneously connected to the network through multiple links.

This work enhances the network access of multihomed devices. This enhancement is achieved by using simultaneously all of the mobile devices interfaces, and by individually routing each data flow through the most adequate technology. The proposed solution is only deployed at the network core and it does not depend on the mobile devices, i.e., it's transparent to the mobile devices. This work gives the necessary tools to reuse the already deployed technologies like WiFi or 3G/LTE. Moreover, it is also possible to extend the network by using femtocells which support multi access technologies. This work is also integrated with IEEE 802.21 standard to improve the handover mechanisms in the network. Additionally, we also propose an integration with a broker that can manage all the data flows individually.

The proposed solution improves the quality of service of the users while not overloading the operator infrastructure. Evaluation results, obtained from the developed prototype, evidence that the overhead for using the proposed solution is very small when compared to the advantages.

Keywords: flow mobility; PMIPv6; femtocell

Resumo

O crescimento exponencial do número de equipamentos móveis com múltiplas tecnologias de acesso à rede está a mudar a maneira como nos ligamos à Internet. Infelizmente, ainda não é possível usar simultaneamente todas as interfaces de rede de um equipamento móvel.

Este trabalho melhora o acesso à rede a partir de dispositivos móveis com múltiplas interfaces de rede. Para alcançar esta melhoria todas as interfaces de rede dos dispositivos móveis podem ser usadas simultaneamente, e os fluxos de tráfego são encaminhados individualmente através da tecnologia mais conveniente. A solução proposta apenas é instalada na rede *core*, ou seja, é transparente para os equipamentos móveis. Este trabalho desenvolveu as ferramentas necessárias para reutilizar as tecnologias existentes que já estão disponíveis em larga escala, como o WiFi ou o 3G/LTE. É também possível usar femto-células com suporte a múltiplas tecnologias de acesso para expandir mais rapidamente a rede. Este trabalho criou também uma integração com a norma IEEE 802.21 para melhorar os processos de *handover*. Adicionalmente propomos a integração com um *broker* externo para uma melhor gestão dos fluxos de tráfego.

A solução proposta melhora a qualidade de serviço dos utilizadores sem sobrecarregar a infra-estrutura do operador. Os resultados obtidos a partir dos testes realizados ao protótipo desenvolvido mostram que o impacto na performance ao usar esta solução é extremamente reduzido quando comparado com as suas vantagens.

Palavras chave: mobilidade de fluxo; PMIPv6; femto-células

Acknowledgments

Agradeço aos meus amigos e colegas o apoio dado ao longo do meu percurso académico.

Agradeço também a todos os professores do ISCTE-IUL, em particular ao meu orientador, Prof. Rui Marinheiro.

Este trabalho foi apoiado pelo projeto da FCT: PTDC/EEA- TEL/120666/2010

Este trabalho foi apoiado pela Fundação para a Ciência e Tecnologia (FCT), Portugal no âmbito do projeto UID/EEA/50008/2013.

Contents

List of figures

List of Tables

List of Acronyms

AAA	Authentication, Authorization and Accounting
AP	Access Point
API	Application programming interface
BCE	Binding Cache Entry
BID	Binding Identifier
BU	Binding Update
CBR	Constant Bit Rate
CN	Correspondent Node
CoA	Care of Address
D-GW	Distributed Gateway
DNS	Domain Name System
DPI	Deep Packet Inspection
DSMIPv6	Dual Stack Mobile IPv6
HNP	Home Network Prefix
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IF	Interface
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
LIF	Logical Interface
LIFO	Last In First Out
LMA	Local Mobility Anchor
LMD	Localized Mobility Domain
LTE	Long-Term Evolution
LTE-A	Long-Term Evolution Advanced

MAC	Media Access Control
MAG	Mobile Access Gateway
MIH	Media Independent Handover
MIHF	Media Independent Handover Function
MIIS	Media Independent Information System
MIPv4	Mobile IPv4
MIPv6	Mobile IPv6
MN	Mobile Node
NAI	Network Access Identifier
NAPI	New API
NAT	Network Address Translation
OS	Operating System
PBA	Proxy Binding Acknowledge
PBU	Proxy Binding Update
PMIPv6	Proxy Mobile IPv6
QoE	Quality if experience
QoS	Quality of service
RFC	Request For Comments
RSS	Received Signal Strength
SAP	Service Access Point
SOHO	Small Office Home Office
TCP	Transmission control protocol
TLV	Type-Length-Value
UDP	User Datagram Protocol
USB	Universal Serial Bus
VM	Virtual Machine
WG	Working Group

1. Introduction

According to Cisco [1], mobile traffic is growing very fast, and in 2014 the number of mobile connected devices exceeded the world population. Also according with this report, by 2019 three-quarters of mobile traffic will be generated by smartphones. Moreover, in 2016 more than half of the traffic generated each month by mobile devices will be offloaded to the fixed network by means of WiFi or femtocells. The traffic generated by mobile phones will exceed the traffic generated by laptops, tablets and any other devices [2].

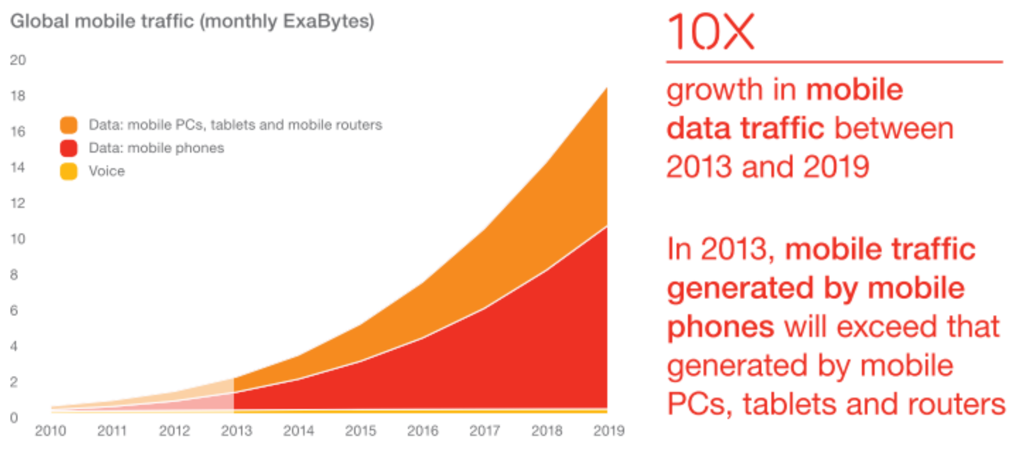


Figure 1 - Global mobile traffic grow [2]

To support the growth of the traffic consumed by Mobile Nodes (MNs), new technologies, like LTE-A, are being developed and deployed. The LTE-A allows MNs to transmit up to 300 Mbit/s, but this throughput is very dependent on the cell congestion.

The cell congestion problem is basically solved by deploying more cells. However, it is desirable to reduce the cost of deploying more equipment in the network. To avoid this cost the operator can use the already deployed infrastructures like WiFi, WiMAX or others that can be deployed at a low cost. We have already the required technology available and deployed, 90% of the world population has 2G coverage, and 45% has 3G coverage [3]. The number of public WiFi hotspots is growing for the next years, and in 2015 there will be about 5.8 million hotspots [4].

Despite the possibilities already offered by the existing technologies, operators are not using all of the available network capacity, e.g. in practice they aren't yet efficiently using the WiFi infrastructure to alleviate mobile data traffic. This could be better achieved if operators could improve the support for mobility protocols in their network infrastructure. The mobility consists on the ability of MNs to move across the operator network while keeping their reachability and continuity of sessions, independently of the technology that is in use.

The mobility can be host based or network based. There are some differences between the two approaches, namely the responsibility of the MN tracking, the place where the routing decision is taken and the MNs technological requirements.

Aside the differences, both approaches have a common element, the Home Agent (HA). The HA is an entity that resides at the MN home network and it's responsible to maintain a bridge between the MN home address and the MN foreign address. The HA must maintain an association between the MN home address and the foreign address.

In host based protocols the MN must keep the HA informed about the changes in its foreign address. In these type of solutions all of the logic resides in the MNs, the network must only assure coherent routes from the MN home address to the MN foreign address.

On the other hand, the network based protocols don't require the intervention of the MN. The tracking of the MN movements is implemented at the network side. This is done by deploying new entities in the network edge that have the capability of tracking the MNs movements and keep the HA informed about them.

Host based solutions have the disadvantage of requiring extra software/hardware deployed in the MNs. It's not easy to an operator to install new software/hardware in all the user's equipment, mainly because of the costs involved. Another disadvantage in host based protocols is that the MN can have the final decision in the mobility process, so, the operator can't fully control the mobility process according with its policies. Nonetheless, one advantage of host based solutions is the scalability, each MN is responsible for its own mobility, so the network isn't overloaded with extra tasks.

Contrary to the host based solutions the network based solutions have shifted all the logic to the network. In these networks, the modifications required in the MNs are inexistent or null. The network based solutions are more appealing to the operators, since they can now offer a service that can be used by a vast majority of their clients. However, contrary to the host based solutions, network based solutions can have some problems in large networks. The network must be capable of supporting the management of all the MNs that are connected, and it is more challenging to scale than the host based solutions.

Additionally, it is important to notice that the type of MNs have been evolving over the years, and now the operators have to deal with: more traffic, more connected devices, multiple network interfaces for each device, multiple access technologies available and distinct Quality of Service (QoS) requirements depending on the application being used by the user.

These new requirements demand a new paradigm to route the traffic, and this work addresses this challenge by adopting the flow Mobility concept [5]. The flow mobility concept, is basically applying the mobility concept for each individual traffic flow i.e., each traffic flow can be routed individually through the available routes. In this scenario the MN can be connected to the network through multiple interfaces and it can be receiving/sending traffic simultaneously in all of its interfaces. Each flow can be routed according to some policy defined by the operator with or without the user intervention. In this document the term "flow mobility" means the possibility of moving a specific IP flow from one access link to another. This definition implies that the MN has the capability of being connected to at least two access points simultaneously. The concept of an equipment being connected to more than one network at the same time is called multihoming.

To better illustrate this concept a flow mobility scenario is represented in Figure 2, where a MN may use all the available interfaces simultaneously. For example, a MN that has a WiFi and a 3G interface can be connected with both interfaces while receiving traffic by both.

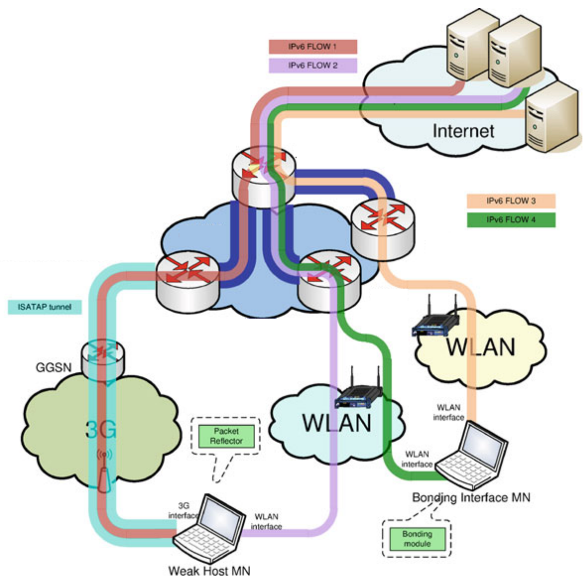


Figure 2- Flow mobility network (adapted from [6]*)*

With a flow based mobility strategy, the operator can positively discriminate more critical traffic with the QoS available in the 3G/LTE networks when necessary, and simultaneously provide cheaper rates for non-critical traffic. For example, in a congested 3G/LTE cell an operator may only accept voice traffic and offload non-critical traffic, like HTTP or Email, to the WiFi infrastructure. This strategy brings advantages to both operators and clients.

There are solutions that in part permit the mobility of traffic flows either host based [7], [8], or network based [6], [9]. The focus of the work presented in this thesis is to extend a network based mobility approach, and to propose a solution to solve the main problems in the flow mobility process, namely:

- The mobility process must be transparent to the MN;
- The flows can be dynamically routed without impacting the transport or application layers protocols;
- The operator can make routing decisions for each individually flow;

Additionally, this work proposes and implements three new features that will benefit both the operator and the users:

- Improved detection of MNs attachments/detachments to/from the network. This objective is accomplished by integrating the flow mobility software with an implementation of IEEE 802.21 [10]. IEEE 802.21 brings several advantages to the proposal: improve the mechanism of MNs tracking on the network; provides a robust framework to monitor the network and link status;

makes the access technology transparent to the mobility logic; and allows for a easier to scale to large networks;

- Integration with an external broker that may take decisions based on real-time QoS parameters, requirements and policies. This broker can receive as input the business rules of the operator and manages routes for the flows accordingly;
- The use of femtocells with multi-access technologies. It will study the feasibility and performance of having MNs simultaneously connected to multiple technologies in the same femtocell.

As this work has a requirement of not requiring modifications in the MNs, a network based mobility solution had to be adopted. The host based solutions aren't adequate to the topic of this work because they all require mobility software installed in the MNs.

To accomplish the proposed goals this work has followed a straightforward methodology in Figure 3.

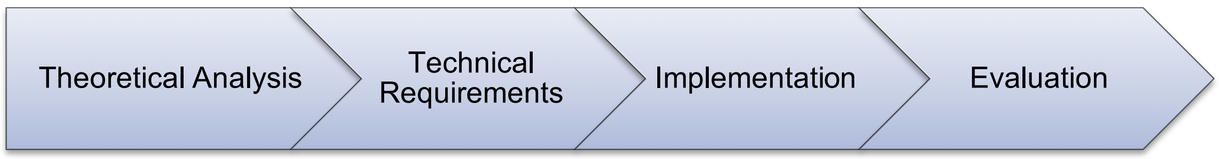


Figure 3 –Work methodology

1. In theoretical analysis phase, the problem was better identified and studies regarding available solutions and possible implementations were conducted.
2. The technical Requirements phase consisted in identifying the necessary tools to solve the problem identified in the previous step. In this phase, a couple of mobility software implementations were analysed and tested. One was chosen to be used as starting point to implement this work proposal.
3. The Implementation phase corresponds to the coding of the defined requirements in a real prototype.
4. Finally, in the Evaluation phase, the output of the implementation phase was tested in a real scenario using real traffic with real equipment.

The accomplished results of this research have already been partially published [11], the next five chapters of this work explain in more detail the proposal. Chapter 2 describes some of the existing protocols and presents them in more detail. The Chapter 3 presents the proposal to solve the problems stated above. The proposal contains a detailed description of the necessary hardware/software to deploy in the network. Chapter 4 shows how the proposed solution was implemented. This chapter describes the choices taken during the implementation phase and enumerates the necessary equipment to deploy in a network to achieve the stated objectives. The proposed solution is evaluated at chapter 5. Chapter 5 contains the results collected from the evaluation phase. Finally, Chapter 6 contains the conclusions about this work and some future work that can be done to improve the developed solution.

2. Mobility protocols

Today mobile networks are facing new challenges: increasing of traffic, increasing of number of MNs connected simultaneously, MNs connected through multiple interfaces and technologies and the new type of apps requiring specific QoS. These challenges require the development of new solutions to support users mobility.

The term “mobility” is normally associated to the physical movement of the client with his/her mobile equipment. In a mobility scenario the user can move his/her equipment while keeping its reachability and sessions continuity. While the MN is moving it cannot lose the already established connections, and the mobility process must be transparent to the transport protocols.

The mobility can be achieved at different layers of the network stack. It can be provided at: physical layer, link layer, network layer, transport layer or application layer. Implementing mobility at physical or link layer can provide fast and transparent handovers, but a solution at this level needs to be developed specifically for each technology. On the other end of the stack a solution at the application layer would be easier to develop, but it would require changes in all the applications [9].

From a performance perspective the mobility at physical layer is the most advantageous solution, but it’s the solution that involves more costs. A solution implemented at application layer would be cheaper to develop but it would require a huge effort from all developers.

This work considers mobility at the network layer. Since the Internet Protocol (IP) is the most used protocol for network layer, a solution at this layer will require only changes to one protocol. All the applications that are deployed over IP would automatically benefit from mobility.

There are already mobility protocols for the network layer. These mobility protocols can be classified in two main groups [12]:

Host Based Protocols:

In this type of protocols, the MN must participate in the mobility process. When the MN moves to a foreign network it must inform its anchor point about its foreign IP address. The anchor point is responsible to forward the traffic to the clients.

Network Based Protocols:

These types of protocols don't require the intervention of the mobile node in the mobility process. The MN movements are detected by an entity in the network, and the mobility process takes place without the MN intervention.

Unfortunately, the flow mobility protocols are still in development state. There are some implementations of flow mobility either network based [6], [9], [13] or host based [7], [8]. However these implementations are still in a development/research state, and will be better described and compared in the next chapters.

As the flow mobility protocols are in development state, there isn't a final Request for Comments (RFC) to work on. The most detailed documentation, until this date, is being published by the NETEXT Working Group (WG) from IETF[1]. This group is working on the future RFC that will describe the necessary extensions to support flow mobility on the IETF network based mobility protocol, the PMIPv6.

The following sub chapters describe some of the available mobility solutions, either with or without support for flow mobility.

2.1. Hercules network stack

The Hercules stack [7], [8] is a mobility solution where a new network stack is deployed on the client and in its peer. This solution is suitable for devices that have multiple network interfaces available. The author has developed a prototype for Linux and Android.

Hercules Stack proposal consists in refactoring the network stack of all the nodes that are involved in a communication. The Hercules new network stack, see Figure 4, manages the physical interfaces and provides a single virtual interface to the protocols above IP layer. This solution creates an abstraction layer that hides the complexity of network management to the applications running on the node. The management of the flows is made by a control plane that was added to the network stack.

The Hercules stack can be separated in three modules:

- **Virtual Interface**
  This module provides to the protocols above network layer a single network access. This interface has an arbitrary IP address that only exists inside the device.
  When an application wants to send a packet it will use the address provided by this virtual interface. The IP address assigned is a local IP address, and it's only valid inside the node. Before sending the packets to the network the IP address must be translated to the real IP address that is being sent to the network.
- **Control Plane**
  This module makes the decisions about the packets routing. The decisions can be based on static policies or can be configured in runtime using an Application Programming Interface (API) developed by the author.
  The proposal doesn't restrict the control to the node. The control plane can be located anywhere in the network. This control plane can also be used by the applications that have specific routing requirements.
- **Switch**
  This is the module that forwards the packets according to the control plane rules. The switch must multiplex/demultiplex the packets to/from the correct physical interface.

[1] http://datatracker.ietf.org/wg/netext/

This module works like a Network Address Translator (NAT). It must translate the private IP addresses that are provided to the upper layers, to the real IP address assigned to the MN. When it receives a packet from the network it must translate the destination address to the address assigned to the network layer.

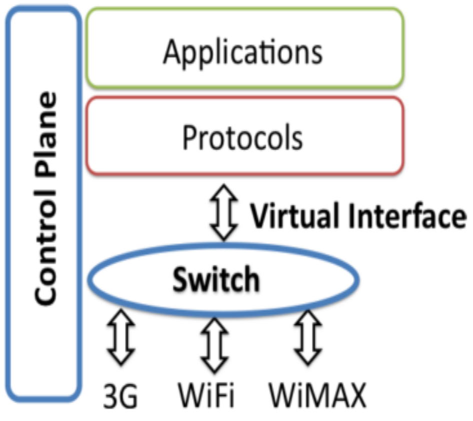


Figure 4 – Hercules Stack [7]

This solution has been implemented and tested in Linux and Android Operative Systems (OSs). Also according with a presentation made by the author in 2011 [14], the Android prototype should become available to the public "soon". Unfortunately, the prototypes aren't yet available publicly.

Flow Mobility Support:

The Hercules stack is already prepared for flow mobility. The flow mobility can be exploited by the applications using the API to manipulate the control plane. Alternatively, the user can also configure static policies in the control plane.

For the Transmission Control Protocol (TCP) the Hercules stack must assure that the session is kept alive between handovers. The session continuity is more problematic in connection oriented protocols like TCP. In TCP, a session is identified by the pair of IP addresses and the transport ports. If one of the IP addresses changes during the handover the TCP session will be lost. The Hercules stack must ensure that after a handover the TCP layer continues to see the same pair of IP addresses that was being used before the handover. As said by the author it is not easy to migrate a flow from an oriented connection protocol like TCP. In a normal situation this would require that the TCP protocol is modified or the packets headers that arrive to the TCP layer are modified before being delivered, so that the TCP layer always see the same endpoint. The Hercules uses the last option, it modified the IP packets so that the transport layer doesn't detect changes at the network layer.

Figure 5 represents three different phases of the flow migration process, one for when the client is behind a NAT (1), another when client isn't behind a NAT (2) and another for when exists a middlebox between the client and its peer (3). In all the situations the nodes are using the Hercules stack. During the flows migrations the MN must exchange signalization messages with its peer, this signalization is responsibility of the control plane.

The right image of Figure 5 shows the address translation procedure. Independently of the network where the MN is connected the application running in host A thinks that the communication is between the IP addresses A' and B. At the other side the application running in host B thinks that the communication is between the addresses A1 and B'. To make this "illusion" possible the switch module must translate the addresses accordingly before sending the packets to the network and before delivering the packets to the transport layer.

This solution was also tested with a middlebox that also uses the Hercules stack. The middlebox must do the same procedure of address translation as in an end to end connection. According to the author this configuration can be useful for introducing a firewall or a Deep Packet Inspection (DPI) tool.

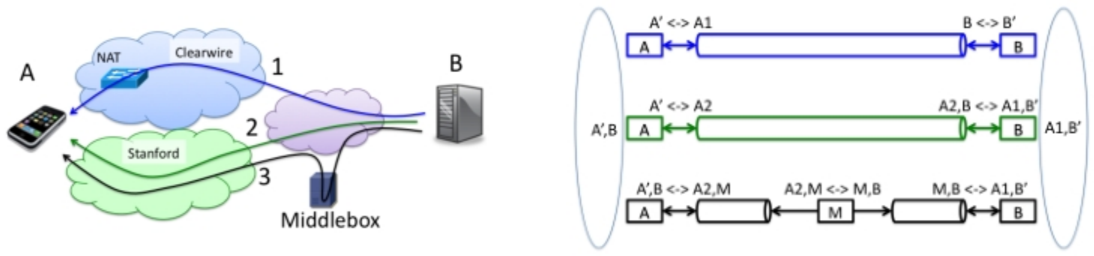


Figure 5 - Hercules stack address translation [7]

To handover a flow extra signalization between the participants is required. The sender must inform its peer about the new address that it will start to use to send the packets. This step is required so that the receiver can associate the traffic coming from the new IP address to an existing flow.

Final Remarks

One key aspect of the Hercules stack it's the control plane. The control plane makes the Hercules a very configurable solution. The control plane can be used by external applications to provide QoS based routing and handovers. An operator that can configure the control plane of all of its users will have a powerful tool to optimize its network. The Hercules is the adequate solution to deploy in an environment where it's impossible to modify the network core equipment.

Hercules is prepared for a full flow mobility environment. The API introduced in Hercules can become very useful for introducing external flows schedulers. With this API the system administrator can configure dynamic policies to best serve their clients. Another advantage is the abstraction that the Hercules creates for the applications, this stack configures an internal IP address that the applications can use. The mapping between this internal IP address and the real IP address it is responsibility of the switch module.

As a host based mobility protocol Hercules introduces modification in the MNs. Hercules in specific, requires a complete new network stack. This is a very expensive operation to perform for all the MNs of an operator network. Even though, the author has already developed the stack and all the dependencies for Linux and Android OS, they still need to be installed in all the MNs. This kind of operation involves changing the whole OS.

The Hercules stack has problems when dealing with midleboxes that don’t have the Hercules stack installed. This is a common situation in the operators networks, sometimes there are statefull firewalls in the network core. When the Hercules stack makes a handover it will change the packets IP addresses. The firewalls in the packets route could drop these packets if they can’t detect that the packets belong to an previously established flow.

This solution has a very restrict set of pre-conditions that must be met so that the protocol works as expected. This solution only supports the TCP and User Datagram Protocol (UDP) as transport protocols. If the user wants to use another transport protocol it won’t work. Also, the solution does not explicitly explains the compatibly with the existing NATs.

The Hercules stack provides to the application running on the host a private IP address. This IP address is a virtual address, it must be translated before sending the packets to and from the network. This translation mechanism is identically to the standard NAT behavior in IPv4 networks. This solution inherits the problems that are usually related to the NATs. Like the difficulty to provide public services in a known port. Another problem of this solution is the fact that it isn’t a standard.

2.2. Mobile IPv6

The MIPv6 [15] is a host based mobility protocol that allows mobile nodes to move around an IPv6 network while keeping their reachability and session continuity in transport layer.

In a MIPv6 scenario, the MN is always identified and reachable by the same address, the home address. As the MN keeps the same address while moving, the movement is transparent to the higher layers of the stack, namely the transport and application layer. The home address is an address that it’s assigned to the MN by an anchor point, the anchor point is located in the MN home network.

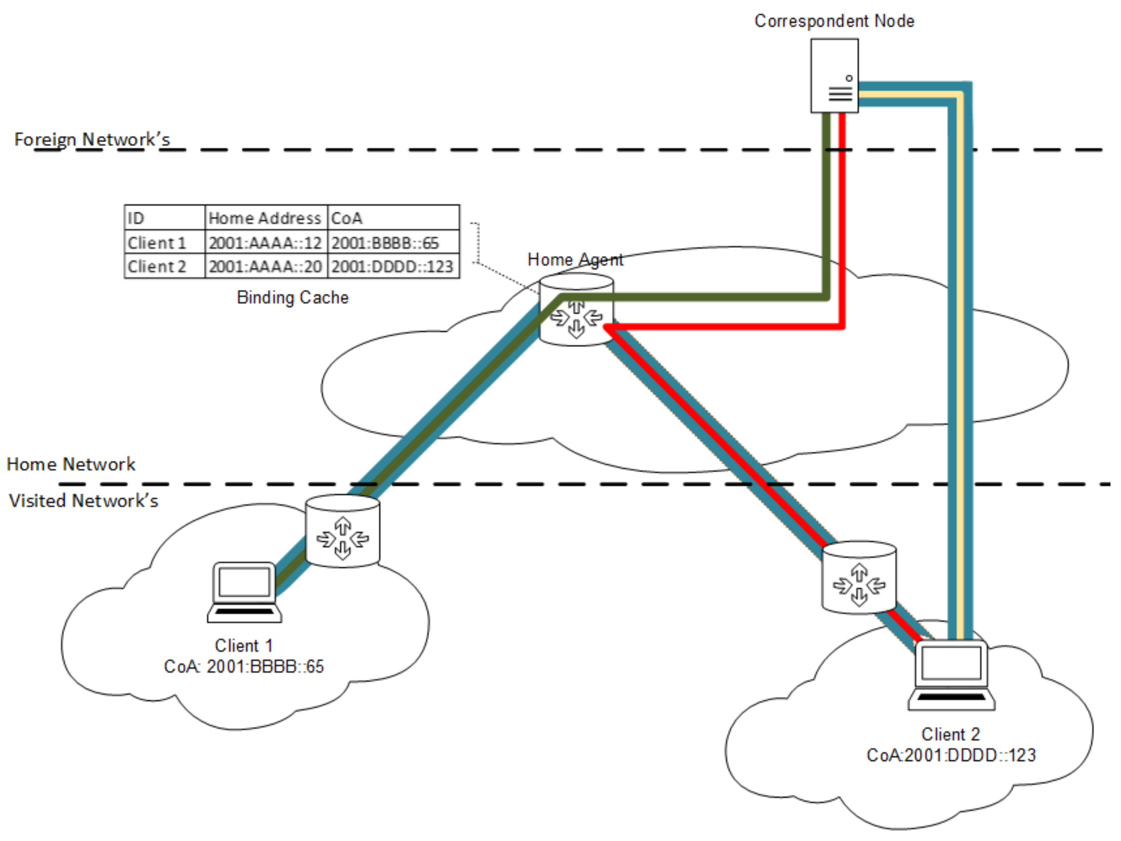
This protocol introduces a new entity in the network, the HA. The HA is located in the MN home network and it acts as an anchor point. This entity must assure the route to the MNs, even if they aren’t at the home network. To assure the route the HA must maintain a structure that maps the MNs home address with their real IP address, the foreign address. With this information the HA can keep a bi-directional tunnel to the MN and forward all the traffic through this tunnel.

When the MN moves to a foreign network it will have to configure a local address with the available mechanisms, either static, stateless [16] or statefull [17] configuration. The address obtained it’s the Care-of Address (CoA). After getting a CoA the MN must register it in its HA. With this information the HA can create a bidirectional tunnel to the MN. With this tunnel the HA can forward all the traffic received that has the MN home address as destination. If the MN has available more than one CoA it can register all in the HA, but it has to select one primary CoA. The MN must ensure that it’s always reachable by the its primary CoA.

To store the MNs location, the HA maintains a structure called Binding Cache. This structure stores a set of bindings. A binding is an association between a home address and one or more CoA. This structure has the necessary information to reach the MN. This cache consists in a binding cache entry for each connected MN. Each binding cache entry must contains at least the home address and the respective CoA.

The HA only maintains the binding cache with the information that it receives from the MN. The MN is responsible to keep its location updated in the HA binding cache.

As seen in Figure 6, there are two possible solutions to the communication between the MN and its peer, the Correspondent Node (CN). The bidirectional Tunneling or the Route Optimization mode.



ID	Home Address	CoA
Client 1	2001:AAAA::12	2001:BBBB::65
Client 2	2001:AAAA::20	2001:DDDD::123

Figure 6 - MIPv6 forwarding mechanisms

Bidirectional Tunneling:

This mode does not require MIPv6 support in the CN.
In this mode all the traffic travels by the HA. The packets addressed to the MN are routed to the HA and then forwarded to the MN through a secure tunnel. The packets from the MN to the CN are sent through the tunnel and then forwarded by the HA to the CN. This mode of operation has a non-optimal route path. In a scenario where the MN is at the same network than the CN, the packets sent to each other always traverse the HA, even if the HA is at a different network.
The advantage of this mode is the transparency to the CN, the real location of the MN is irrelevant to the CN. In this scenario the CN doesn't need to support MIPv6.

Route Optimization:

In this scenario the first messages are exchanged in bidirectional tunneling mode. After the first message both correspondents can reach each other. Now the MN can register its CoA in the CN, like it was the HA. Now the packets are routed directly between the MN and the CN. This also alleviates the congestion in the HA, making this approach more scalable that the bidirectional tunneling. The possible downside is that it is no longer transparent to the CN, in this scenario the CN must implement the MIPv6 protocol.

In MIPv6 the signalization messages are vital to keep the MN location always updated in the HA. The signalization messages are encapsulated in a new IPv6 extension header [18][19], the mobility header.

This header was specifically created to transport messages related to the mobility management. The MIPv6 defines a set of messages to serve multiple purposes on the mobility management process. Each of these messages can contain one or more mobility option encoded in Type-Length-Value (TLV) format. These options contain additional information that can be reused between messages. From the options defined in the standard the most important to consider for this work is the Home Prefix Option, this is the option that carries the CoA.

Flow Mobility support:

The MIPv6, as defined in RFC 6275 [15], cannot provide flow mobility to the MNs. The bindings that the HA stores are always a binding between an IP address and one or more CoA. Even when a MN registers more than one CoA it must choose one primary CoA.

Due to the importance of flow mobility, the MIPv6 standard has been updated to accommodate the necessary changes to provide the flow mobility to the clients. The first change [20] was to give the possibility to the client register multiple active CoAs. If the MN has more than one access link, for example two WiFi interfaces, it can obtain two distinct CoAs for its interfaces and register both in the HA. A HA compliant with this update can forward traffic to the MN based on a specific forwarding policy. The forwarding policy is dependent on the HA implementation, so it doesn’t consider the user preferences.

A later update [21] to MIPv6 added the necessary signalization messages so that the MNs can bind a specific flow to a CoA. The MN now have the capability of installing routes in the HA to forward each flow through the most adequate route.

With the last two updates the MIPv6 became ready to provide flow mobility to its users.

Remarks:

The MIPv6 is a simple and mature protocol that is widely deployed. It also has the advantage that even low end devices can support MIPv6 without requiring significant modifications. With MIPv6 it’s possible to deploy a mobility enabled network without significant hardware requirements in the network core. It’s even possible to obtain mobility without permanently using a centralized HA. If both participants in the communication support MIPv6 they can route the first messages through the HA and then the CN can act as the HA for the MN.

As stated before, sometimes the session continuity can be more important than the mobility. In MIPv6 the MN is always identified by the same address. Even when the MN moves to another network the transport and application layers will always see the same IP address. The MIPv6 assures that the mobility is transparent to the MN.

One downside of MIPv6 is that not always the route to the CN is the best. To get the best route both ends of the communication must exchange signalization messages directly. This is only possible if both ends are MIPv6 compliant devices. Even in the route optimization scenario the HA has to exist. The HA assures that the CN can reach the MN for the first message. After reaching the MN the CN can exchange MIPv6 signalization directly with the MN so that they can optimize the route.

Another problem is the lack of control that the operators have. In a MIPv6 network the only entity managed by the operator is the HA. Moreover, this HA can only act as simple router that it’s configured directly by the clients. In most of the time, the operator has more information about the whole network than the client. With this information the operator could take better handover decisions than the client.

2.3. Dual Stack Mobile IPv6

Dual stack mobile IPv6 [22] is a set of modifications to the MIPv6 protocol. Since IPv6 isn’t wide deployed it isn’t possible to assure mobility only with standard MIPv6. Even when the MNs support IPv6, it can move to other networks that don’t support IPv6. This protocol gives the possibility of registering an IPv4 as CoA in the HA. This protocol also supports private addressing that can exist behind a NAT in an IPv4 network.

Although the nodes involved in the mobility must support both IPv4 and IPv6, only MIPv6 is used for the mobility management. The standard MIPv4 is not used because MIPv6 offers more benefits. For example, route optimization and the dynamic HA discovery are only available in MIPv6. IPv6 also offers a large address space, this makes possible to assign a global unique addresses to the MNs. The large address space removes the necessity of using NATs, avoiding all of the forwarding problems that they imply.

DSMIPv6 specifies two use cases: one when the foreign network supports IPv6 (1), and another when the foreign network only supports IPv4. When the foreign network only supports IPV4 two situations can occur: the network can provide a public IPv4 address (2) or it can provide private addresses that it’s behind a NAT (3).

When the foreign network supports IPv6 (1) the DSMIPv6 works like the standard MIPv6. The only difference is that if the MN also has an IPv4 address assigned, that it can also register it as its CoA. In this situation the HA will have two binding entries, one for the IPv4 address and another for the IPv6 address. If the HA has to forward traffic to the IPv4 CoA it will encapsulate the packets in IPv6 and send them through a tunnel, as the standard MIPv6 does.

The process for when the MN gets a public IPv4 address (2) it’s identical. The MN encapsulates the MIPv6 binding update in an IPv4 packet and addresses it to the HA IPv4 address.

If the MN obtains a private IPv4 address as the CoA (3), it will have to traverse the NAT. To traverse the NAT, the MN encapsulates the IPv6 packets in UDP datagrams and then in a IPv4 packet. In both cases after the Binding Update (BU) message, the communication between the MN and the HA it’s made through a secure tunnel, in this tunnel the IPv4 packets are encapsulated in IPv6 packets.

Flow Mobility

Conceptually DSMIPv6 is just a set of extensions to MIPv6, so most of the RFCs are shared between both protocols. Inclusively for the specification of multiple CoAs and flow bindings [20], [21].

Remarks

When the objective is to use a host based mobility protocol the DSMIPv6 is the most universal solution. DSMIPv6 has all of the features of the MIPv6 and the capability of also supporting IPv4 nodes. IPv6 is not widely deployed, there still are a lots of networks that only provide IPv4 addresses. Limiting the mobility to IPv6 networks would not make the solution appealing to deploy in large scale.

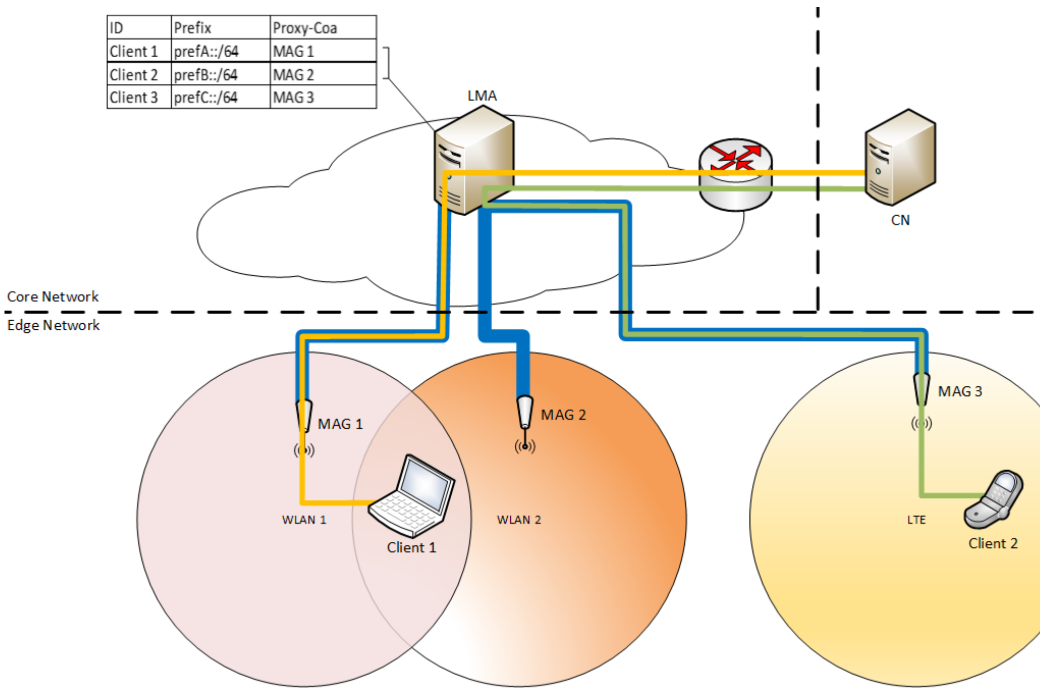
One problem of this protocol is that with the support of IPv4 it will also have to deal with the existing NATs. The NATs have some inconveniences to the incoming traffic [23], especially for the connections originated outside the private network. In a flow mobility scenario, the NAT also introduce some problems. If an already established flow starts being routed through a new network, the NAT on that network will not recognize the flow. As the NAT doesn’t have a bind for that flow it will have to discard all the packets.

2.4. Proxy Mobile IPv6

PMIPv6 [24]–[26] is a network based mobility protocol developed by IETF. Since it’s a network based mobility protocol it doesn’t require the participation of the MNs in the mobility process.

The MIPv6 protocol required the participation of the MN in the mobility process. The PMIPv6 removes that limitation. In PMIPv6 the functions that were assured by the MN were transferred the network edge.

The PMIPv6 reuses most of the concepts defined in MIPv6. As can be seen in Figure 7, a PMIPv6 Localized Mobility Domain (LMD) is assured by two entities. The Local Mobility Anchor (LMA) is the entity located in the network core and it’s responsible for anchoring the MNs locations in the LMD. The Mobile Access Gateway (MAG) is deployed at the network edge and it must track the MNs movements and keep the LMA informed about them.



ID	Prefix	Proxy-Coa
Client 1	prefA::/64	MAG 1
Client 2	prefB::/64	MAG 2
Client 3	prefC::/64	MAG 3

Figure 7 - PMIPv6 reference diagram

The LMA is an extension of the HA used in MIPv6. The LMA is responsible to maintain routes to the MNs that are using the mobility domain. The LMA must be compatible with the standard HA functions defined in MIPv6. Normally there is only one LMA in a LMD.

The MAG is the entity that it's located at the network edge and it responsible to track the MNs movements and inform the LMA about them.

This separation between core and edge makes possible to provide mobility between different access technologies. The LMA is independent of the access technology, for the LMA it's indifferent if the MN is connected through a LTE network or through an IEEE 802.11x network. The LMA only needs to forward the traffic to the MAGs. Each MAG will have the necessary technology to forward the traffic to the final destination, the MN.

The LMA must be able to forward all the traffic addressed to its clients, it must act as an anchor point. When the LMA wants to forward traffic to its clients it must route it through a secure tunnel that connects to the most adequate MAG. The MAG is responsible by the last hop. Contrary to MIPv6, the secure tunnel ends at the network edge, it doesn't reach the MN.

Another responsibility of the PMIPv6 entities is the assignment of the Home Network Prefixes (HNPs) to the MNs. The actual specification supports only the per-MN prefix model. In this model the HNPs assigned to one MN can't be shared with another MN. If the MN attaches to the network with multiple interfaces simultaneously, each one will be assigned a unique HNP or set of HNPs. In this situation the prefixes that aren't assigned to the same interface aren't managed under the same mobility session, i.e., each MN interface will have an independent mobility session.

In a typical scenario, the association of a MN to the network is made in four steps, as shown in Figure 8. The MN starts by attaching its interface to an Access Point (AP), this attachment occurs at the link layer. Then, the MN will send a router solicitation message (1) to obtain an IP address. When the MAG detects a client connection it will identify the client and check in its policy database if the client is authorized to use the mobility service. If the client has authorization, the MAG will send a Proxy Binding Update (PBU) message (2) to the LMA. The PBU message contains the information about the MN, namely the HNP assigned and its identification. The LMA will then check the validity of the PBU and if everything is correct it will respond with a Proxy Binding Acknowledge (PBA) (3) message. After receiving the PBA message the MAG can send a router advertisement message (4) with the assigned HNPs to the client.

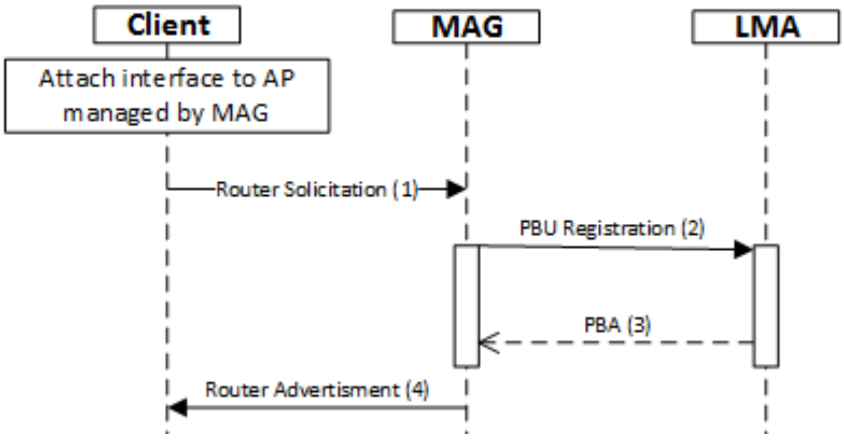


Figure 8 - PMIPv6 MN attachment simplified diagram

The LMA must be backward compatible with the MIPv6 HA functionalities. To avoid creating new data structures the binding cache structure defined in MIPv6 was reused in PMIPv6. In PMIPv6 the binding cache entry must also store:

- A flag that indicates that the entry is from a proxy registration;
- An identifier of the MN. This identifier is implementation dependent but it is recommended to use the Network Access Identifier (NAI) [27];
- The link-layer identifier of the MN;
- The link-local address of the MAG on the point-to-point link shared with the MN;
- The list of all IPv6 HNPs assigned to the MN;
- The identifier of the bidirectional tunnel between the LMA and the MAG;
- The access technology type that the MN is using;
- A timestamp of the last accepted PBU message.

The LMA operation is similar to the MIPv6 HA. Its main objective is to manage the MNs locations and provide a reliable route to the traffic addressed to the MNs. The main difference to the HA, is that it no longer exchanges signalization with the MN. In a PMIPv6 network the responsibility of keeping the LMA updated is transferred to the MAG. The LMA only has to guarantee a route to the MN by using the information that it receives from the MAG.

The MAG is a new entity introduced specifically for PMIPv6. The MAG resides on the access links. It has the responsibility of tracking the MNs movements and keep the LMA informed about them. The MAG has to maintain a Binding Update List updated with the information of the clients attached to it. The PMIPv6 extends the Binding update list defined in MIPv6 with the following extra fields:

- The identifier of the MN;
- The link layer address of the MN;
- A list of HNPs assigned to the MN;
- The IPv6 address of the LMA managing the MN;
- The interface identifier of the point-to-point link between the MN and the MAG;
- The identifier of the bidirectional tunnel that connects the MAG and the LMA.

With PMIPv6 the mobility is independent of the technology being used at the access links. The network can provide mobility in a heterogeneous network scenario. The only requirement is to have a MAG in the access links with the capability of tracking the MNs movements. The MAG also removes the necessity of the client having to update its location.

Flow Mobility:

The PMIPv6 as defined in its specification does not have support for flow mobility. The IETF has created the WG NETEXT[2] to standardize a solution to extend the PMIPv6 so that it can support flow mobility. This group has already created multiple drafts about flow mobility extensions for PMIPv6 [28]. Even though this is a very active WG, at the time of the writing of this work there wasn't any final RFC published.

There is however a embryonic implementation of flow mobility on PMIPv6 [6], [9]. This is also based on PMIPv6 protocol, and in the documents published by NETEXT WG. This work has shown that it is possible to have flow mobility without the intervention of the MN, and those different types of MNs can be supported without requiring modifications on them. Another important aspect of this work was the integration with a 3G network, the tests shown that it's possible to provide network based flow mobility between different technologies. With the right choice of routing policies it's possible to obtain gains for the operator and for the users. The operator can use its knowledge of the status of the network to provide better QoS to the users and to alleviate its infrastructure. Unfortunately, the software and the prototype produced aren't publicly available and their research hasn't progressed significantly.

Remarks

A downside of PMIPv6 when compared to the MIPv6 protocol is the requirement of having MAGs in all access links that are under the same LMD. In a MIPv6 scenario, it's only necessary to have a single

[2] http://datatracker.ietf.org/wg/netext/charter/

entity in the network, the HA. The deployment of MAGs at the network edge it's the price to pay when compared to the cost of upgrading all of the MNs for a host based mobility scenario. The MAG can be as simply as a software entity running in the APs.

PMIPv6 is the IETF protocol for network based flow mobility. This protocol has been updated over time to accommodate the new developments in flow mobility research area. Contrary to client based protocols this protocol does not impose any changes to MNs. This is very appealing for network administrators and operators. Modifications in MNs hardware or software aren't easy to implement and definitely they would represent a huge monetary cost to the operators.

Being a protocol that doesn't require modification in the MNs, makes it a first choice to be deployed in large scale. This protocol also gives the control of the mobility process to the network administrator.

2.5. IEEE 802.21 Media Independent Handover

IEEE 802.21 [10], [29] is a standard that defines mechanisms to improve the handover between IEEE 802.x networks or cellular networks. This standard is not a mobility protocol like the ones presented before, but it provides useful mechanisms to improve the handover process regardless of the mobility protocol being used.

IEEE 802.21 is a framework that has the objective to provide intelligence to the layers above link layer. IEEE 802.21 can be separated in three main components:

a) A framework that facilitates the handover process between heterogonous networks. This framework is implemented as a stack deployed in all devices that are involved in the handover;
b) A new entity called Media Independent Handover Function (MIHF) that provides mobility related functions to the users. And the necessary primitives to the communication;
c) A Media Independent Information System (MIIS) and its primitives so that all the users can communicate with it. This entity works like a central repository from where the users can request information that will help to realize a handover;
d) A Link Layer Service Access Point (Link SAP) and its primitives. The Link SAP is the bridge between the MIHF and the link layer of the network stack. The link SAP implementation is specific for each link layer technology.

Figure 9 shows a simplified architecture of an IEEE 802.21 deployment. In a simple deployment there are two entities, the MIHF and one or more Link SAPs. The Link SAP is the entity responsible to communicate directly with a specific technology at the link layer of the network stack. For example, IEEE 802.11 and IEEE 802.3. The information collected by the Link SAP is sent to the MIHF so that it can forward it to its clients.

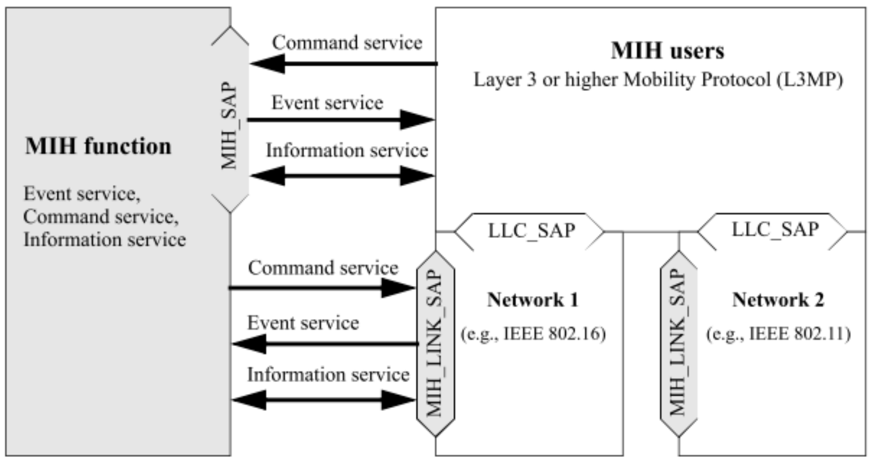


Figure 9 – IEEE 802.21 generic architecture [10]

The Link SAP is the entity responsible to detect events at the link layer level. Because of the multiple link layer technologies existent, the IEEE 802.21 doesn't define a protocol for the communication between the Link SAP and the link layer technology. The Link SAP implementation is dependent of the link layer technology. The information provided by Link SAP can be events, for example MNs attachments or detachments, or parameters, like Received Signal Strength (RSS) and QoS.

The MIHF is the entity that provides services to the Media Independent Handover (MIH) users. The services provided can be used to optimize the handover process. For the communication with the MIHF the 802.21 defines the MIH protocol.

The Media Independent Handover (MIH) user is any entity that may perform handover related functions. The MIH users rely on the information provided by the MIHF to improve the handover process.

Most of the network mobility protocols don't define how the tracking of the MNs movements should be done. For these cases IEEE 802.21 provides a reliable and efficient way to do it. There are proposed solutions [30][31], [32] to use IEEE 802.21 together with a mobility protocol, the first two papers propose an integration with PMIPv6. However there are no solutions that use IEEE 802.21 when there is flow mobility with PMIPv6, as it is proposed in the present research [11].

2.6. Distributed Mobility Management

Distributed Mobility management is a paradigm that has the objective of improving the existing mobility protocols. The mobility protocols presented before relies on a centralized entity, the HA. The HA is located at the network core and it redirects the traffic destined to the MNs to the real location of the MNs. The standard mobility protocols, as anything that relies in a centralized entity, has some inconveniences in terms of scalability and fault tolerance.

Distributed mobility management concept could be applied to the exiting mobility protocols [12], either client based protocols or network based protocols. The distributed mobility management solution has the purpose of deployment mobility anchors closer to MNs.

There is one proposal to include a distributed management approach in PMIPv6 [33]. This proposal extends the PMIPv6 protocol so that it follows the distributed mobility management paradigm. The proposed solution replaces the exiting MAG and LMA by a new entity, the Distributed Gateway (D-GW). This new entity is located at the network edge and it must implement the functionalities of: a regular router, a MAG and a LMA. The D-GW will behave like one of these entities depending on the situation. The D-GW is the first hop that the MN sees when it joins the network.

In the proposal [33] the attachment of a MN to the network followed by a handover can be described as follow:

- When a MN connects for the first time to a D-GW, let's call it D-GW1, it will receive a HNP, in this situation the D-GW is working as a regular router. The MN will then use the HNP to configure its address, let's call it addrA. In this situation there isn't any mobility related function being used.
- When the MN moves to another network or wants to do a handover, it wants to keep its reachability by maintaining its current address, addrA. Once the MN connects to the new D-GW, it will receive a new HNP.
- So to continue to being reachable by the address addrA, the new D-GW must act as a MAG and send a PBU to the LMA. The LMA in this case, is the D-GW1. The D-GW where the MN was previously connected.
- After the standard PMIPv6 procedure, the D-GW1 will route the traffic addressed to the addrA to the new D-GW. The procedures to transfer the previous assigned HNPs to the new D-GW is not defined in the proposal. The author suggests that it could be used layer 2 signaling.

The distributed mobility management solves some of the problems of the actual protocols, like the scalability or the route optimization problems. Despite being a proposal in a development phase, there is already one implementation for PMIPv6 and MIPv6 from the Open DMM project[3].

The work developed for this thesis is compatible with the distributed mobility management paradigm. With some extra work, the distributed mobility management paradigm can be easily implemented in the proposed solution.

[3] http://www.odmm.net/

3. Solution Description

This chapter describes a novel proposal to allow for network based flow mobility in the operator network. In this proposal the mobility process is transparent to the MN. The MN can move in the network while keeping its reachability, and the transport protocols will keep their sessions state. The proposed solution is fully network based, and the MNs don't have to be involved in the mobility process. The handover decisions and the MN tracking in the network is implemented by entities managed by the network administrator.

The proposal extends PMIPv6 and considers documentation published by the NETEXT WG. This group has already began the process of standardization of the flow mobility extensions to PMIPv6 [28]. Figure 10 shows a typical scenario where this proposal can be deployed. Here the Localized Mobility Domain (LMD) has one Local Mobility Anchor (LMA) and multiple Mobile Access Gateway (MAGs). These entities are meant to be deployed at network core and at the edge respectively.

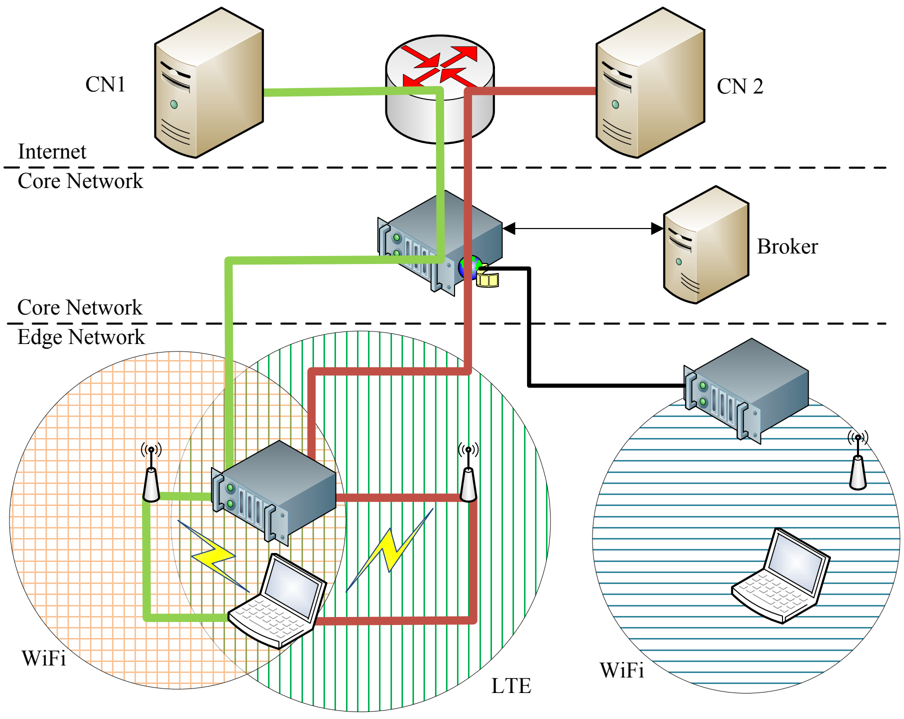


Figure 10 – Proposal for solution deployment

The network core has the LMA installed. The LMA acts as the anchor point for the MNs. The LMA has the traditional functionalities of mobility management and the necessary extensions to also support flow mobility. In this proposal, the LMA is able to forward each client flow individually.

The network core architecture also includes a broker. The broker is an entity that receives inputs about the network status to calculate the best routes for the flows. The result of this choose is then passed to the LMA so that it can install the routes. The broker can be either a proactive entity or a passive entity. When the broker is a passive entity it merely reacts to events in the network and suggest handover decisions for the necessary flows to the LMA. On the other hand, when in proactive mode it will be constantly monitoring the network status to make timely handover suggestions to the LMA, when necessary.

The MAGs are the components located at the network edge that are responsible for the tracking of the MNs movements in the network. To improve the MNs detection and the handovers process, this work proposes the integration with the IEEE 802.21 standard.

This proposal may also be deployed in femtocells scenarios where a single network equipment can provide multiple access networks though distinct technologies. For example, in Figure 10 there is one cell that provides access to the network through two different networks, in this case it’s WiFi and LTE.

This proposal doesn’t impose any restriction in the edge network technologies, and it works with any heterogeneous scenario, where flow mobility is available, even if connected through multiple access technologies simultaneously. For example, a MN can be connected to the LMD through WiFi and LTE simultaneously.

This independency of the network edge technologies is achieved in part by using IEEE 802.21. With IEEE 802.21 the MAG just has to register adequate link events in the MIHF. In this architecture each access point has a specific Link Sap adapted for access technology that it provides. The Link Saps will be monitoring the activity in the network, sending to the MIHF all the relevant events that are detected. For example, a MN attachment or detachment event. Then, the MIHF will forward the events received to the MAGs.

As above mentioned, this solution has three key entities, the MNs, the MAGs and the LMA. The next chapters will then describe the proposed and implemented modifications, relative to the standard PMIPv6, to each one of these entities.

The section 3.1 shows the two types of MNs supported by this solution. When a MN has multiple interfaces it can manage that interfaces in several ways. This chapter describes the most common approaches taken by operating systems to manage multi-homed MNs.

The section 3.2 describes the behaviour expected from the MAG, it describes how the MNs movement is detected and how the MAG will forward the traffic to and from the MNs.

Finally, the section 3.3 proposes the solution to be adopted by the LMA, it describes the core of the solution. This chapter explains the decisions and technologies chosen to have network based flow mobility.

3.1. Mobile Nodes

There are multiple types of MNs, and this work will support some of them. When a multihomed equipment configures its interfaces some problems may appear. After configuring all the interfaces, the node will have two sets of parameters, these parameters can be of two types. The ones that are bounded to the interface, like IP address and link layer address, and global parameters like Domain Name Servers (DNS) and gateways. The main problem here is how the node will merge and use all the parameters available from the different interfaces. In an scenario like this an host can behave like a weak host, a strong host it may has a LIF (Logical Interface) [34][35].

The strong host model is the most restrictive model. In this model the host can only send packets on an interface if the interface is assigned the source IP address of the packet being sent. And the same logic applies for the incoming packets. For example, a host has two interfaces, one with IP address “A” and another with IP Address “B”. If the interface “B” receives a packet addressed to the interface “A” it will drop the packet.

On the other hand, the weak-host model is less restrictive at receiving and sending packets. A weak host accepts packets in any interface Wen sending traffic the only requirement is that the source address is assigned to any of the interfaces of the node. And when receiving traffic, the destination address must be assigned to any of the interfaces of the node.

In addition to this two host models a host can also implement a LIF [36]. A LIF is mechanism provided by the operating system to hide the complexity of the network management. The LIF objective is to hide from the IP Layer the details of the available physical interfaces. In a LIF setup the IP layer is on top of a virtual interface created by the operating system, see Figure 11.

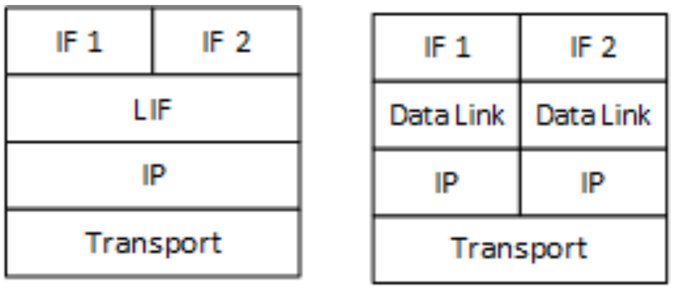


Figure 11 - Client stack implementation types. LEFT: logical interface. RIGHT: weak or strong host model

The model that the node uses depends on the stack implementation, i.e., it depends on the OS. Each OS vendor can choose one of these solution or it can use a custom solution. Fortunately, the most common operating systems, Widows, Linux variants and OSx, use one of these three approaches.

Only the weak-host and the LIF can be used in a flow mobility environment. The strong-host model is very restrictive about the packet addressing and it won’t be considered for this work.

The goal of the proposed solution in this work is to not require any modification on the client equipment, and for this reason, this proposal supports MNs that are weak-hosts or have a LIF.

When a MN has multiple routes available for the outgoing traffic it must install forwarding rules that can take advantage of all of routes. This behavior is dependent on the equipment/software manufacturer implementation [35]. This work does not have any requirement for MN outgoing traffic. In a later update to this work, the incoming traffic from the client can be used to help to make handover decisions.

However, the MN must install routes that minimize the asymmetries in the traffic paths. The MN should ensure that it sends the traffic by the same interface where it was received, with the exception when it wants to signal a handover to the LMA. The asymmetrical routes can degrade the flow performance.

This phenomenon is especially relevant for protocols that are dependent on feedback sent in acknowledgment messages to calculate the ideal throughput, like TCP [37]. Despite of being a recommendation this work does not impose any traffic outgoing policy in the client. It is up to the client to minimize traffic asymmetries.

3.2. Mobile Access Gateway

In this proposal the MAG implements the standard PMIPv6, however PMIPv6 doesn't define how the MAG should detect the MNs movements. For this reason, this work proposes to use the IEEE 802.21 framework to decouple the client management from the PMIPv6. The MAG will communicate with a MIHF deployed in the network. This communication is based on triggers, and to receive these the MAG will make a registration in the MIHF for attachment and detachment events generated on the APs. These events will be detected by Link SAPs, which will forward them to the MIHF that will forward them once again to the MAGs that had subscribed them.

The integration with IEEE 802.21 requires the inclusion of new entities in the network, at least one MIHF and one Link SAP for each AP. These new entities will deal with all the aspects of monitoring the MNs movements in the network.

Figure 12 shows the high level architecture of the MAG. The MAG is composed by 5 modules: the Finite State Machine (FSM) (1), the events handler (2), the MIHF (3), the Link Saps (4) and the Binding cache (5).

Figure 12 represents an overview of the MAG software modules. At the bottom, there are the Link SAPs. The Link Sap is an entity from the IEEE 802.21 standard. Each Link Sap interacts with one network physical interface and it detects events that are occurring in the network. In this work the events are the link up and link down, i.e. attachments and detachments of MNs. Each MAG will have one Link Sap for each network interface that it manages. The Link Sap software is specifically developed for each technology, for example, the Link Sap for IEEE 802.11 isn't the same as the Link Sap form IEEE 802.3.

The MIHF is another entity from the IEEE 802.21 framework. This entity acts as a central point that receives events from the Link Saps and forward those to other entities that previously asked to be notified. The MIHF acts as an abstraction layer to the applications that want to manage heterogeneous network interfaces.

At the top layers there are the PMIPv6 standard entities. The event handler listens for events on the network. It listens for PBA messages, ICPMv6 messages and IEEE 802.21 messages. During the setup phase this module asks to the MIHF to be notified about Link up and Link down events that occur in the network interfaces under the domain of this MAG.

When the Event handler receives an event it passes it to the finite state machine. In this work the finite state machine handles PBA messages, the finite state machine applies the rules defined in PMIPv6 standard to process the PBA messages.

Finally, the binding cache is a data structure that stores information related with the MNs. This table stores all the necessary information about the MNs. From all the information that it stores, for each MN, the most relevant information is the HNPs assigned to the MN.

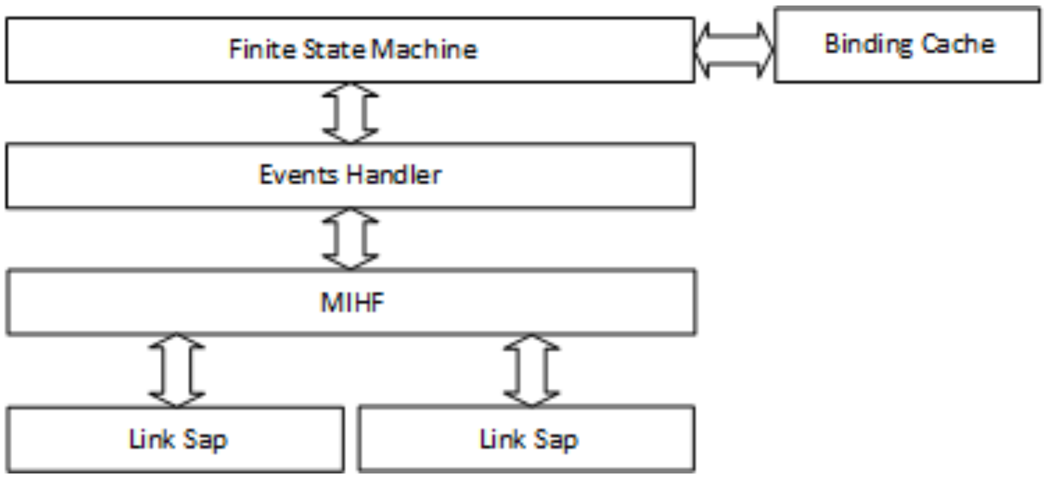


Figure 12- MAG Software Modules

Figure 13 shows the process of setting up the MAG movement detection routine. The MAG registration with MIHF is made with three messages. The first message registers the MAG as a client in the MIHF. This message has an optional acknowledgment response. After a successfully registration, the MAG will send a capability discover request message (3) to the MIHF. The MIHF will respond with the interfaces that it manages and the available events for each one (4). When the MAG receives the capability discover response (4) it will check if the interface that it should manage is under that MIHF supervision. If it is, it will send an event (link up and link down) subscribe request (5) to MIHF, this message requests to the MIHF to send back a message when it detects an attachment or a detachment on that specific link. After this setup phase, the MAG will be waiting for event messages from the MIHF (9).

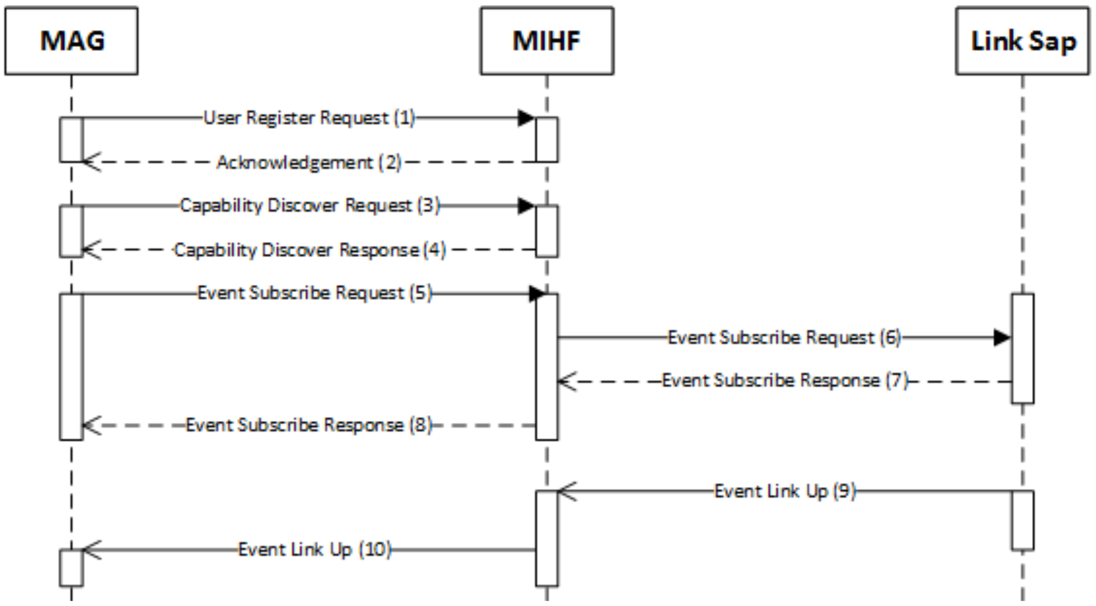

MAG
MIHF
Link Sap
User Register Request (1)
Acknowledgement (2)
Capability Discover Request (3)
Capability Discover Response (4)
Event Subscribe Request (5)
Event Subscribe Request (6)
Event Subscribe Response (7)
Event Subscribe Response (8)
Event Link Up (9)
Event Link Up (10)

Figure 13 - Interaction between MAG and ODTONE

Figure 14 shows how the MAG processes a MIHF event, in this example the link up (10) from Figure 13. When the MAG receives the link up event it only knows the link layer address of the MN that has been attached. To obtain the MN ID the MAG must convert the MN link layer address to the EUI-64 format and send an authentication message to the (Authentication, Authorization and Accounting) AAA server (11). The AAA server will then return the MN authorization, the MN ID and the HNP that should be assigned (12).

To finalize the attachment, the MAG will send a PBU message (13) to the LMA with the information obtained from the AAA server. After receiving the PBA (14), the MAG can advertise the prefixes assigned to the client by sending a unicast Router advertisement (15) message to the MN.

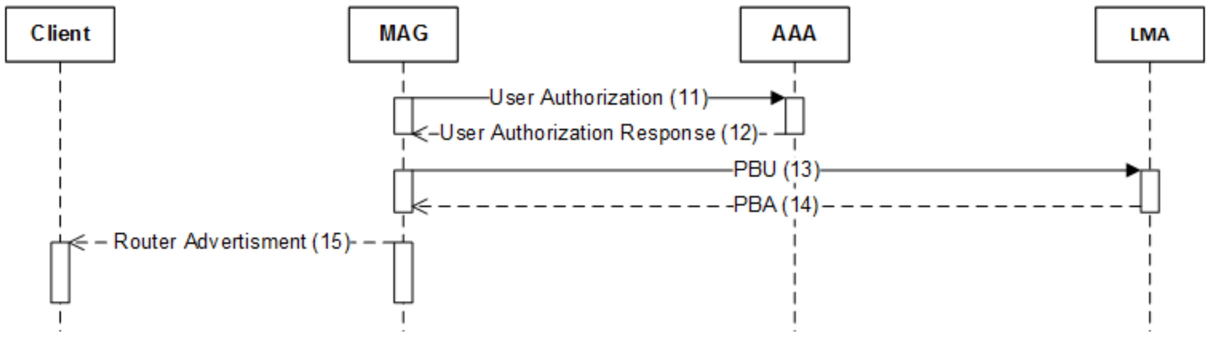


Figure 14 - MAG processing a user attachment

Integrating IEEE 802.21 with PMIPv6 improves the MNs movements detection mechanism. It also opens the door to future upgrades that can take advantage of all the functionalities that IEEE 802.21 has to offer. For example, in an IEEE 802.21 scenario the MIHF can be used to detect situations of an eminent link down. This information can be passed to the LMA so that it can start preparing a handover for the flows that will be affected by the link down.

Figure 15 shows a simplified version of the MAG finite state machine. This state machine represents the MAG logic to process the events that are generated during the PMIPv6 operation. There are four events: attachment (1), detachment (2), a PBA response to a register PBU (3) and a PBA response to a deregister PBU (4). They indicate when a client connects a new interface (1), when it disconnects the interface (2), and when the MAG receives a PBA, either related with a registration (3) or with a deregistration (4).

The attachment and detachment events (1) and (2) can be generated by multiple sources, in proposed solution they can have origin in a MIHF message or in a syslog message. Only one of the last two mechanisms can be active at a time.

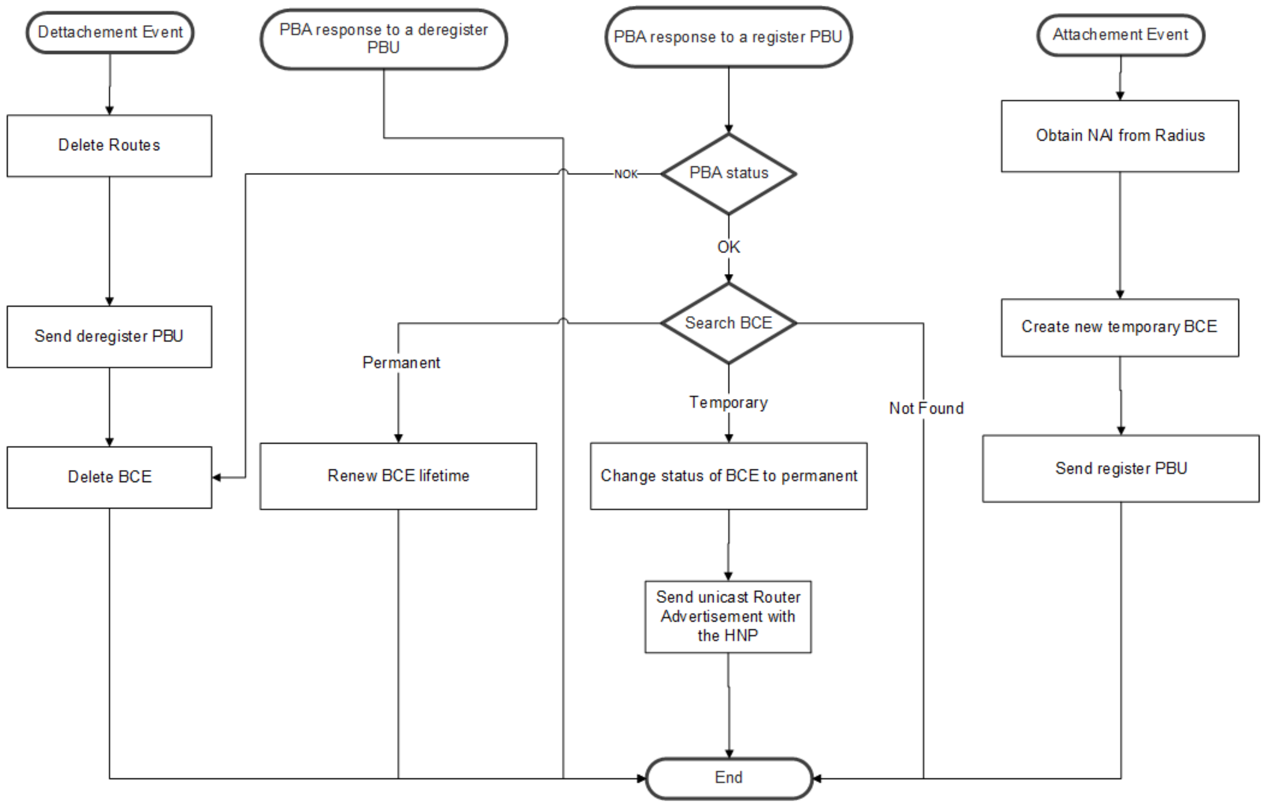


Figure 15 - MAG Finite State Machine

When the MAG receives an attachment event it means that the MN is connecting for the first time. In this situation the MAG gets the MN Network NAI from the Radius server and creates a temporary Binding Cache Entry (BCE). Next it sends a PBU register message to the LMA.

The previous PBU message will trigger a PBA in response, when the MAG receives it, it tries to find a BCE for the MN to what that event refers to. The search can return or not a BCE, if it returns a BCE it can either be a temporary entry or a permanent entry.

The temporary entry is the entry that exists while the MAG hasn't received the PBA message in response to the PBU that originated that entry. When the MAG receives the PBU with status success, it will change status of the entry from temporary to permanent and sends a unicast router advertisement containing the HNP to the MN. It concludes the registration process. If the PBU is received has an error status code, it will delete the BCE and ends the process.

If the MAG finds a permanent BCE it means that the PBA is the response to a previous renewal request, it will update the BCE lifetime.

Lastly, the MAG can also receive a detachment event. This situation occurs when the MN disconnects from the network. In response to this event the MAG deletes the routes and the BCE that are related with that MN. Finally, it sends a deregister PBU to LMA to inform about the MN detachment.

Another task of the MAG is the management of the binding cache table. Each BCE has a lifetime, when the BCE lifetime is about to expire the MAG must check if the MN is still reachable. This is done by sending a neighbor solicitation message to the MN. If the MN is still active, i.e. it responds back with a neighbor advertisement, the MAG will renew the BCE. In this proposal the lifetime is a static value, that it's equal for all the BCEs. This implementation can be changed so that the lifetime is more adaptable. For example, the lifetime value can be calculated based on the volatility of the MN. If the MN is constantly moving, the lifetime should be a small value. On the other side, if the MN has a stable behavior the lifetime should have a higher value to avoid unnecessary signalization messages.

The two supported types of MNs, weak host and LIF as described in chapter 3.1, have some particularities during HNP assignment. In a flow mobility scenario, the MN will have to have multiple prefixes for its interfaces. NETEXT draft specifies three use cases for the prefixes attribution:

1. When the MN attaches a new interface it receives the same prefix or set of prefixes that were already assigned to the other interface;
2. At attachment the MN receives a new prefix or set of prefixes;
3. The MN will receive new prefixes and prefixes that are already assigned to other interface. This is a hybrid of the two scenarios stated above.

When the MN receives a common prefix or set of prefixes for its interfaces (1) additional signalization to install routes in MAGs isn't necessary. This happens because the MAGs where the MN is connected has the necessary information to route the packets to the MN. For example, a MN with two interfaces, each one connected to a different MAG, will have the same prefix configured on both. Each MAG can receive traffic addressed to any of the MN interfaces because they all share the same HNP.

The proposal of this work adopts the 2nd case, where the MN receives a unique prefix for each interface. If each MN interface has a unique prefix or set of prefix, MAGs won't be able to route packets to the MN, if they are addressed to a prefix that they don't manage. In this case, a MAG cannot forward packets addressed to a prefix that it doesn't know. In this situation the LMA must explicitly inform MAGs with all the prefixes that are assigned to the MN, so that they can install routes to the MN.

To inform MAGs about changes in the mobility sessions, the LMA must send an Update Notification Message [38] to the MAG to inform it about new prefixes that the MAG should route. This proposal makes a simplification and does not use this message. Since all the traffic addressed to the MNs is tunneled from the LMA the MAG, each MAG can simply forward all the traffic coming from the tunnel to the wireless medium.

3.3. Local Mobility Anchor

The LMA is the entity that is responsible for anchoring MNs. This entity must be able to provide a reliable route to traffic addresses to each MN.

Figure 16 shows the overview of the LMA architecture. The figure is subdivided in two groups, the kernel space and the user space. Each one indicates where the software is running, either in kernel or in the user space.

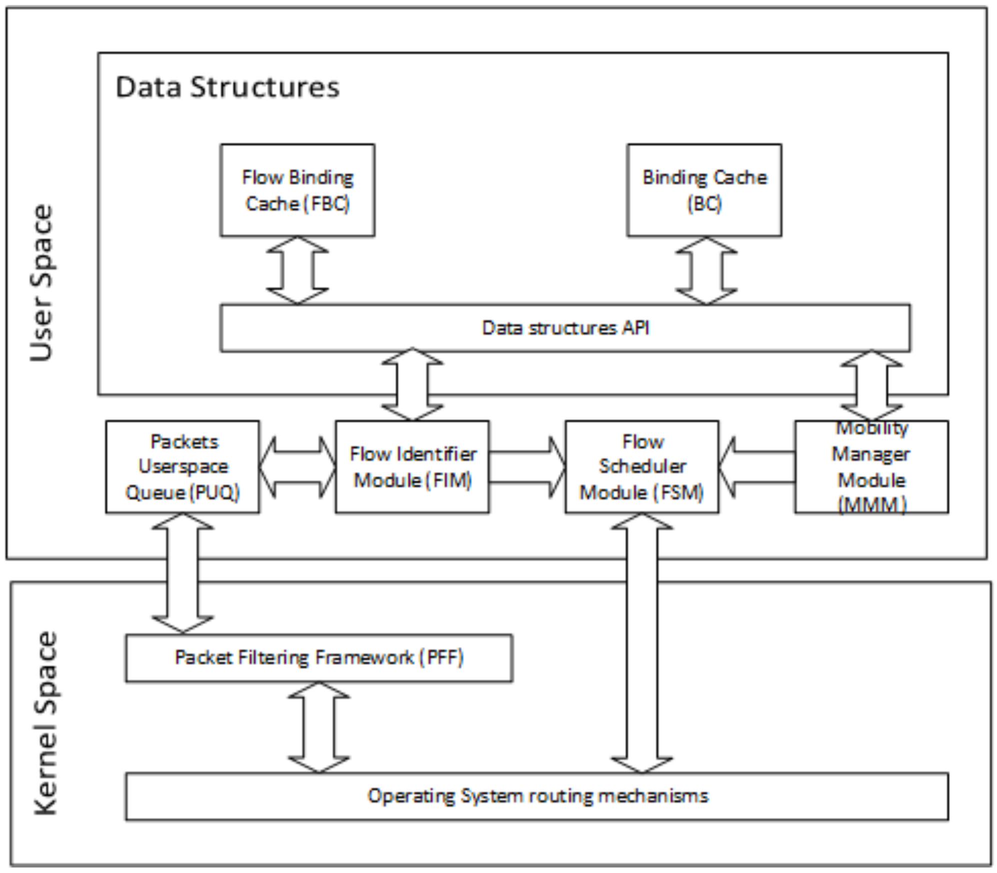

User Space
Data Structures
Flow Binding Cache (FBC)
Binding Cache (BC)
Data structures API
Packets Userspace Queue (PUQ)
Flow Identifier Module (FIM)
Flow Scheduler Module (FSM)
Mobility Manager Module (MMM)
Kernel Space
Packet Filtering Framework (PFF)
Operating System routing mechanisms

Figure 16 - LMA software modules

In this solution the core of the protocol runs in user space. The kernel space is used to install routing rules to forward the traffic.

In the first row there is the Binding Cache (BC) and the Flow Binding Cache (FBE). These are conceptual data structures that store the necessary information to keep the traffic flowing to the MNs.

Bellow the data structures are the four core modules of the proposed solution. Each one of these modules has a specific task in the FMPMIPv6 protocol.

The Packet Userspace Queue (PUQ) is a queue where the packets belonging to flows that don't have a forwarding rule are put before being routed. These packets are handled by the Flow Identifier Module (FIM). The FIM identifies the MN to where the packets must be routed. The information collected by FIM is then passed to to the Flow Scheduler Module (FSM). The FSM module will choose a route according to its policies and install it in kernel space.

The Mobility Manager Module (MMM) handles all the events related to the PMIPv6 protocol, as PBUs and IEEE 802.21 events.

The kernel space has all the software responsible to route the network packets according to the pre established routes. It also has a packet filtering framework that can divert packets from the kernel to the user space for further processing by the PUQ before being routed.

Data Structures

Because of MN multihoming, the LMA binding cache structure must be changed. With multihoming, the binding cache must store an association between each MN prefix and its proxy CoAs, i.e., the MAG that is serving the HNP.

Table 1 shows a simplified LMA Binding cache. The BC data structure has now two new fields: The Binding Identifier (BID) and the MN-ID. The BID is an identifier that uniquely identifies one entry of the table. The MN ID is the value used to identify a MN. It's important to reinforce that the MN-ID identifies the MN not a specific interface. This example contains two MNs and three routes. The MN 1 could be reached either through MAG 1 or MAG 2. This MN has unique prefixes assigned to its interfaces.

Table 1 - Example of a Binding Cache

BID	MN-ID	HNPs	Proxy-CoA
1	MN 1	HNP1,HNP2	MAG 1
2	MN 1	HNP3	MAG 2
3	MN 2	HNP4	MAG 1

A MN with multiple interfaces will have multiple entries in the binding cache, each entry will have a different BID but they will all share the same MN-ID.

The BC only gives support for multihoming, and it doesn't enable the solution to identify specific flows. The binding cache supports the management of the MNs locations. When the LMA receives a PBU from a MAG it must be able to check if it corresponds to a new attachment or to a handover. In this implementation the LMA only deletes a route to a client in two situations, when it receives a PBU asking for deletion or when the lifetime of the binding entry expires. So if a MAG doesn't detect the MN detachment and later the LMA receives PBU from a different MAG, the LMA will insert a second entry in the BC. Until the old BCE expires, or the MAG sends a de-registration message, the LMA will have two routes, even if one of them is a dead end. This is minimized with a reliable MN movement detection mechanism, like IEEE 802.21, or with an adequate lifetime for the BCEs

Table 2 shows the data structure of the flow binding cache. This is a new data structure implemented in the LMA that stores the information about each individual flow. This table is composed at least by a traffic selector and a BID.

The traffic selector is a value that uniquely identifies a flow. In this proposal, the traffic selector is composed by values of the transport and network headers. From the network header it uses the destination and source IPv6 address, the flow label, the traffic class and the next header value [19]. From the transport header it will use the source and destination ports.

This solution is compatible with all packets that have a IPv6 network header, and a transport header that has the source and destination port in the first 4 bytes, for example TCP and UDP. If a flow doesn't meet these requirements it will follow the standard routing mechanisms bypassing the flow mobility process. This strategy avoid routing conflicts for protocols that are interface aware, like ICMPv6 or DHCPv6.

All the values required to construct the traffic selector are retrieved directly from each packet transport and IPv6 headers, they aren't manipulated or generated by the software.

Finally, the BID in the flow mobility cache. The BID points to an entry in the BC, i.e., identifies the route that the flow must follow. For example, Table 2 shows that the packets that are addressed to TCP port 80 of the MN 1 will be routed through the MAG 1. And all the flows destined to the TCP port 5060 of the MN 1 will be routed through the MAG 2.

Table 2 - Flow Binding Cache

#	Priority	Traffic selector	BID	Action
1	1	Destination is MN1 and TCP destination port 80	1	Forward
2	35	Destination is MN1 and TCP destination port 5060	2	Forward

The changes in the data structures were the first step to support the flow mobility, but additional substantial changes were necessary to enable flow mobility.

Flow Scheduler Module

The FSM is a module that receives information about a flow chooses a route for it. In this implementation this module is a scheduler that randomly chooses one path from the ones that are available. In a more sophisticated scenario this module could easily adopt another strategy for a better flow scheduling. For example this entity can communicate with an broker [39] that can provide more reliable information about the network status and the MN traffic requirements.

The solution is prepared to work with any implementation of this module. And the only requirement is that the provided module implements the interface provided for the flows schedulers.

Mobility Manager Module

The MMM is an event driven thread that waits for events related with the mobility process, like PBUs. This module is the core of the PMIPv6 implementation.

Figure 17 shows the FSM of the MMM. When the MMM receives a PBU message it will search in its BC for a BCE belonging to the node referenced in the PBU. The MMM identifies a BCE by the tuple MN ID<->Serving MAG address. This means that there exists one BCE for each MN connected interface.

In this implementation it's not possible for a client be connected to the same MAG by two or more distinct interfaces. For example, in a normal situation where a MN is connected to two MAGs, the MN will have two entries in the binding cache, one which maps the MN ID with the serving MAG 1 and another that maps the MN-ID with the serving MAG 2. If the MN connects to the same MAG, it would have two equal entries that map MN ID to the MAG where it is connected. In the architecture proposed this is not a problem since each femtocell can have multiple technologies simultaneously, and it isn't a common procedure to connect distinct wireless interfaces to the same AP.

After searching for the BCE, the LMA may perform one of four different actions, as represented in Figure 17: It can register the attachment from a MN (1), it can renew the lifetime of a BCE (2), it can perform a handover (3) or it can delete a BCE (4).

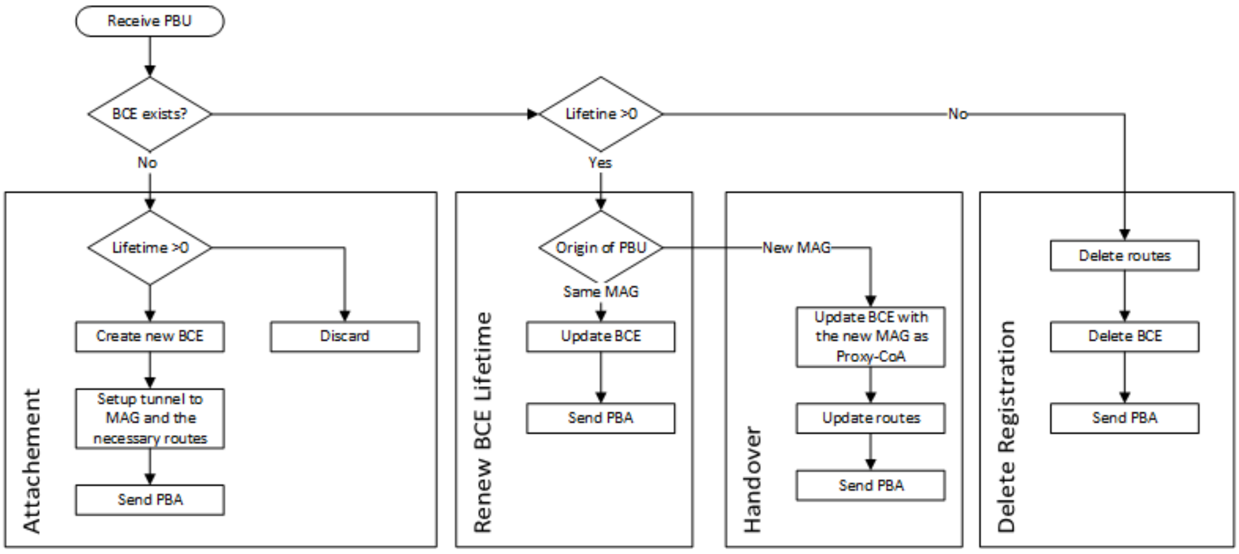


Figure 17 – Mobility Manager Module Finite State Machine

The attachment (1) corresponds to the connection of a network interface from a MN. This event is triggered by a PBU message. If the PBU is valid, the LMA will create a new BCE in the binding cache structure and setup the necessary routes/tunnels to send the traffic to the MN. This action ends by sending a PBA message to the MAG.

As said in the standard, the BCE must have a lifetime to avoid old routes that are no longer active. The renewal (2) of a BCE is triggered by receiving a PBU with a Lifetime value higher than 0 and when there already exists a BCE for that PBU.

If there is a PBU for that MN, but it's coming from a different MAG it means that the MN has done a handover. In this case (3) the LMA will update the old BCE and the routes associated with it. Both the renewal and the handover process ends with the sending of the PBA message to the MAG that sent the PBU.

Finally, the delete action (4) is triggered by receiving a PBU with a Lifetime equals to 0, this means that the MAG is requiring the deletion of BCE associated with that PBU. In this situation, the LMA will delete the BCE and the system routes.

Both the handover (3) and the delete action (4) have impact on the flows that are addressed to the MN. In these situations, the LMA will have to reschedule all the flows of the client. This is done by requesting to the flow scheduler to reroute the flows addressed to the MN.

Flow Identifier Module:

The last component is the FIM. This module is a thread that processes the data packets addressed MNs. This module is constantly looking for flows that haven't a defined routing rule. It extracts the packets that are put in the user space queue and identifies to which MN they belong.

When parsing a packet, the FIM generates a traffic selector, the traffic selector it's 6-tuple selector that it's composed by the following values, extracted from the packet: the source and destination IPv6 addresses, the source and destination transport protocol ports, the transport protocol and the flow label extracted from the IPV6 header. These values are enough to uniquely identify a flow and since they are all in the packet and datagram headers they are very fast to extract.

By querying the BC with the destination IPv6 address the FIM can identify to which MN the flow belongs. When the FIM identifies the MN it passes the traffic selector and the MN identification to the FSM so that it can create a routing rule to that flow

This module must be able to analyze all the packets that are addressed to the MNs. If the traffic to the MNs isn't routed through the machine where the LMA is installed, the FIM won't be able to identify the flows. And without the identifications of the flows the FSM cannot install routes that make the flow mobility possible. But since the LMA works as an anchor point it controls all the traffic related to its clients, so at any moment the traffic to the MNs must pass through the LMA.

4. Implementation and Testbed

An important part of this work is the implementation and test of the developed prototype. Before starting the coding of the solution an analysis was made. The analyses consisted in evaluating the already available PMIPv6 software implementations. After the evaluation one was chosen to serve as base to the flow extensions. The section 4.1 describes in detail the analyses made and the details of the proposed implementation.

Then the section 4.2 describes how the tests were conducted. This section describes in detail: what hardware and software was used to test the prototype, the test conditions, the network topology and the how the results needed were measured from the network devices.

4.1. Implementation

This section presents and compares the available software solutions that provide PMIPv6 without flow mobility. Then it describes the proposed solution by this thesis in more detail, namely the decisions taken in account for the implementation phase.

PMIPv6 implementation analysis

This work is based on the PMIPv6 protocol, so it’s important to choose an already existing open-source implementation that can be modified to support the flow mobility extension. Reusing an existing PMIPv6 implementation has of the following advantages:

- Quality: the flow mobility extensions will be implemented over software that has already been tested by the community. Normally this type of software is included in Linux distributions. In this way it’s assured that the PMIPv6 core is complaint with the existing standards.
- Acceptance: If this work is an add-on to existing software it will be easier to persuade others to use it and test it.
- Documentation and support: Having good documentation and a community that assures future support for the software decreases the development time and improves the overall software quality.

There are two PMIPv6 implementations available as open-source software: the Eurecom OAI-PMIPv6[4], and OPMIP[5]. There is also a commercial version of the PMIPv6 protocol from Sibridge Technologies [6]. The last solution hasn’t been considered for this work due to the lack of public documentation and available results.

[4] http://www.openairinterface.org/openairinterface-proxy-mobile-ipv6-oai-pmipv6
[5] http://helios.av.it.pt/projects/opmip
[6] http://www.sibridgetech.com/PMIPv6.aspx

Both the Eurecom and OPMIP implementation are good candidates to receive flow mobility extensions. Eurecom project was developed on top of a MIPv6 implementation, the UMIP Project[7]. The UMIP project is an open source implementation of MIPv6 maintained by the open source community.

The Eurecom project started as an investigation work. Nowadays it's aggregated to the UMIP project and it's maintained by the community.

On the other hand, the OPMIP project is a PMIPv6 implementation made by ATNOG group from the University of Aveiro. This group is a research group that develops solutions to solve mobility related problems.

Table 3 shows a summary of the decisions made to choose a PMIPv6 implementation to start working on.

The programming language is an important metric because choosing the correct one can simplify a lot the process of cross compilation and decrease the development time. The c language was preferred by this work due to the high adoption by the Linux community, there are a lot of tools to develop and cross compile software to multiple hardware in an easy way.

It is important to choose a solution that isn't already outdated. The last update for OAI-PMIPv6 was in 2011, and the last update for OPMIP was in 2013. Both solutions aren't being active developed. Despite being the oldest, the OAI-PMIPv6 seems to be the most adequate solution. This solution is based on UMIP. The UMIP project is the project that implements MIPv6 support in the Linux Kernel and it's already shipped by default in some Linux kernels. On the other hand, the OMIP was developed from the scratch and it isn't adopted in a large scale.

To decrease the development time, it is important that the solution adopted has available good documentation. Both have support of mailing lists and technical documentation. OPMIP is also supported by a thesis [25] that describes in detail the decisions made during the design and implementation phase.

For last, there is the requirements that the solutions have. Both require a kernel compiled with support for MIPv6. The OAI-PMIPv6 also requires a radius server to store the MNs authorization parameters.

Apart the metrics shown in Table 3 there is another more subjective, but also important to consider. What is the solutions that is more easy to distribute to the community? From this work perspective it is the Eurecom OAI-PMIPv6. The Eurecom software was based on the older UMIP project. The UMIP project is already disseminated by the most part of Linux Kernel versions. After developing the flow mobility extensions on top of UMIP a simple patch is enough to distribute the solution through the mailing lists of Linux community. It won't require external libraries that normally are difficult to push to the main Linux development tree.

[7] http://umip.org/

Table 3 - PMIPv6 available implementations comparison

	OAI-PMIPv6	OPMIP
Programming language	Standard c	C++ with Boost library
Last Update	OAI-PMIPv6 was launched in 2011 based in the UMIP 0.4 launched in 2005	2013
Further developments	Project has stopped	Project has stopped
Documentation	Technical documentation and mailing lists	Thesis and technical documentation
Requirements	Linux with kernel compiled with support for MIPv6 and radius server	Linux with kernel compiled with support for MIPv6

After analyzing the metrics shown in Table 3 and doing some code review, the Eurecom OAI-PMIPv6 was the chosen project to receive the flow mobility extensions. It is important to notice that it was a difficult choice. Technically both solutions are solid enough to have flow mobility extensions implemented. The choice was partly based in more subjective metrics.

The solution described in the rest of this chapter has been developed on top of the Eurecom OAI-PMIPv6 implementation.

Mobile node identification

The Eurecom OAI-PMIPv6 uses the MN Link layer address to identify a MN. In a multihoming scenario the MN identifier must identify the MN independently of the interface in use. This work uses the NAI [40] as the MN identifier. The NAI is a 63-byte identifier that it’s assigned by the network administrator. The NAI must uniquely identify a MN independently of its interfaces.

This is a relatively small change, and the software has been modified to use a NAI instead of a link layer address to identify a MN.

Database and HNPs assignment

The Eurecom OAI-PMIPv6 uses a Radius server as the AAA server. This work has included an extra parameter to the radius database, the MN NAI. As such, the database now stores the following parameters:

- MN interface ID in EUI-64 [41] format. This parameter identifies a MN interface;

- Network prefix to be assigned to the MN. This is the network prefix that the MAG must assign to this specific interface of the MN.
- MN NAI. It's a unique identifier to the MN.

With this configuration there is one record for each MN interface. This assumption is valid for a weak-host model that have a unique link layer address for each interface. A client using a LIF will always use the same link layer address, so in practice there is only one entry in the database.

This configuration assumes that the LMD administrator can obtain all of this information about its clients. This is a relative simple task for a network operator that has access to a client database which have the characteristics of the MNs.

As said before, this considers the clients that either follow weak host model or use a LIF. In this work each network interface receives a unique HNP. A weak-host client that has two interfaces will receive two distinct HNPs. On the other hand, a LIF client will receive the same HNP independently of the attachment point.

As said in the previous chapter, when making a handover to another MAG that doesn't know the MN HNP it's necessary that the LMA send an update message to the MAG informing it about the new MN HNP. In this work, each MAG has a static route that forwards all the traffic coming from the LMA to the interface that leads to the MNs. This is possible because in our setup, the traffic exchanged between the MAG and the LMA always travels through a tunnel. It's reasonable to assume that all the traffic coming from the tunnels that is not address to the MAG itself must be addressed to any of the MNs.

Mobility Access Gateway

The MAG responsibility in the mobility environment is to track the user movements and keep the LMA informed. As explained before, this work will use IEEE 802.21 to monitor the MNs movements.

The Eurecom implementation of PMIPv6 uses syslog messages to detect the attachment and detachment of clients. This method uses libpcap[8] to capture all syslog messages from the network so that they can be parsed by the application to extract MN information, in this case the MN MAC Address. In the Eurecom implementation, the software has a thread capturing all syslog messages from the network. When it receives a message, it parses it to extract the MAC address of the client and the type of event, either an attachment or a detachment. This implementation is very dependent of the MAG technology. Different equipment can generate different syslog messages on attachment/detachment events, or they even cannot generate messages at all.

This work decouples the client management from the PMIPv6 implementation. This work has been integrated with ODTONE[9] [42], an implementation of IEEE 802.21 protocol. The ODTONE project provides an interface defined in c programing language that defines the necessary messages to

[8] http://www.tcpdump.org/
[9] http://atnog.github.io/ODTONE/

communicate with the MIHF. ODTONE also provides several Link Saps for different link layer technologies. This work uses the IEEE 802.11 Link Sap in the tests that will be described later on.

The use of ODTONE is however optional, and the software may still use the old method for detecting clients, by just disabling a macro before compiling.

When using ODTONE the software reads some additional parameters from the configuration file. Table 4 shows the extra parameters that should be configured when using ODTONE for movement detection. The first parameter is the MAC address of the interface that the MAG wants to manage, i.e., the interface that it's acting as AP to the clients; When a MN connects to that AP the ODTONE will generate a link up event. The second and third parameter are the pair IP address and port where the MIHF is listening for requests. The fourth and fifth parameters are ids to use while exchanging messages between IEEE 802.21 entities.

Table 4 - Extra configuration parameters for ODTONE integration

MAC address of the interface that the MAG manages	16 Byte
IP address of MIHF	IPv4 address
MIHF listening port	Integer
User name of the MAG to use while registering in the MIHF	4 char
ID of the MIHF	4 char

Local Mobility Anchor

A Linux system usually forwards packets based on the destination address. This is achieved by matching the destination address with longest prefix in a routing table and retrieve the next hop. Using only this strategy is not convenient to forward packets based on policies. The standard routing mechanism can only forward packets based on the destination address, but the LMA must implement a policy based routing strategy. The policy based routing provides a mechanism to make routing decisions based on all the information present in the packet. In Linux, the policy based routing is supported by having multiple routing tables and by using forwarding marks. A forwarding mark is a 32 bit field that can be associated with a packet that is traversing the Linux network stack. This mark is not part of the packet and, it only exists as metadata in the kernel while the packet is in the system.

During the setup phase, the LMA creates one routing table for each MAG that it manages. Each routing table has only one entry that forwards all the traffic through the adequate tunnel. By default, the Linux kernel only consults multiple routing tables to route packets if it's explicitly asked to. The LMA must also create a policy based rule that instructs the kernel that a packet that has a specific mark is going to be routed with the information present in one specific routing table. This setup is made by using tools from the iproute2[10] package, namely the "ip" tool. Figure 18 shows a flow chart about how the operating system handles this kind of routing. When it's time to choose a route for a packet the operating system will check if the packet has a mark. For example, in Figure 18 if a packet has the mark "50" the operating system will search a forwarding rule in a forwarding tabled called "MAG1". The search in this table will

[10] http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2

result on one route that forwards the packet towards its destination. If the packet has no mark, the operating system will use the default table, normally called the “main table”.

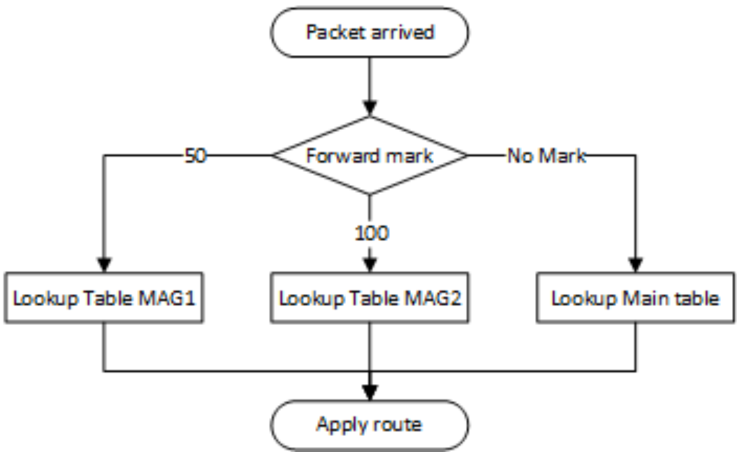


Figure 18 - Linux policy based routing example

The marking of the packets can be made with a packet filter framework. This work will consider the use of both IPTables and NFTables, from netfilter project[11]. The particular framework can be chosen before compiling the software by setting a specific compile macro. One of these packet filtering frameworks will then be used by the FSM to install routes for the flows.

These type of tools can analyse the packet headers and data to apply a forwarding mark to the packet that will work as a traffic selector. Normally these type of tools are used in static scenarios where a system administrator configures a list of rules that are installed during a setup phase. But in this work it’s necessary to dynamically change the forwarding rules. This requirement is necessary because the LMA must parse all the packets that are addressed to MNs, so that it can choose an adequate route based on the forwarding policy.

An important aspect of these tools is that the majority of the packet processing takes place in kernel space. Since the LMA is a user space application it was necessary to divert the packets from the kernel to the user space. So that the routing decision could be taken by the LMA, more specifically the FIM and the FSM as explained in the previous chapter 3.3.

Diverting all packets to user space is not efficient. To minimize the impact on the system, the LMA only diverts to the user space the first packet from the flows that it hasn’t yet analysed. To achieve this, it uses the libnetfilter_queue[12]. This library is a framework that allows a user space application to manipulates packets that are placed in a user space queue by a packet filter framework. The packets that are diverted to the user space queue are waiting for a routing decision made in user space, and they will not be forwarded to the destination until some application applies a decision to them.

As seen in Figure 19 the LMA must parse the packets from the flows that aren’t marked. For example, when the first packet of the flow A arrives to the system, the kernel will try to find a rule to mark that

[11] http://netfilter.org/projects/nftables/
[12] http://www.netfilter.org/projects/libnetfilter_queue/

packet. As this is the first packet, the LMA hasn't yet configured a rule to mark it, so this packet will match the default rule that sends all non-marked packets to the user space queue. In the user space, the LMA will parse the packet to identify the destination MN. After identifying the MN, it will apply its policy based rules to get a route to that flow. This route will be installed on the system by installing a forwarding rule in the kernel that marks all the packets of that flow with a forward mark. When the next packet of flow A arrives in the kernel, a corresponding rule is already installed, so it skips the user space and continues the normal path in the kernel.

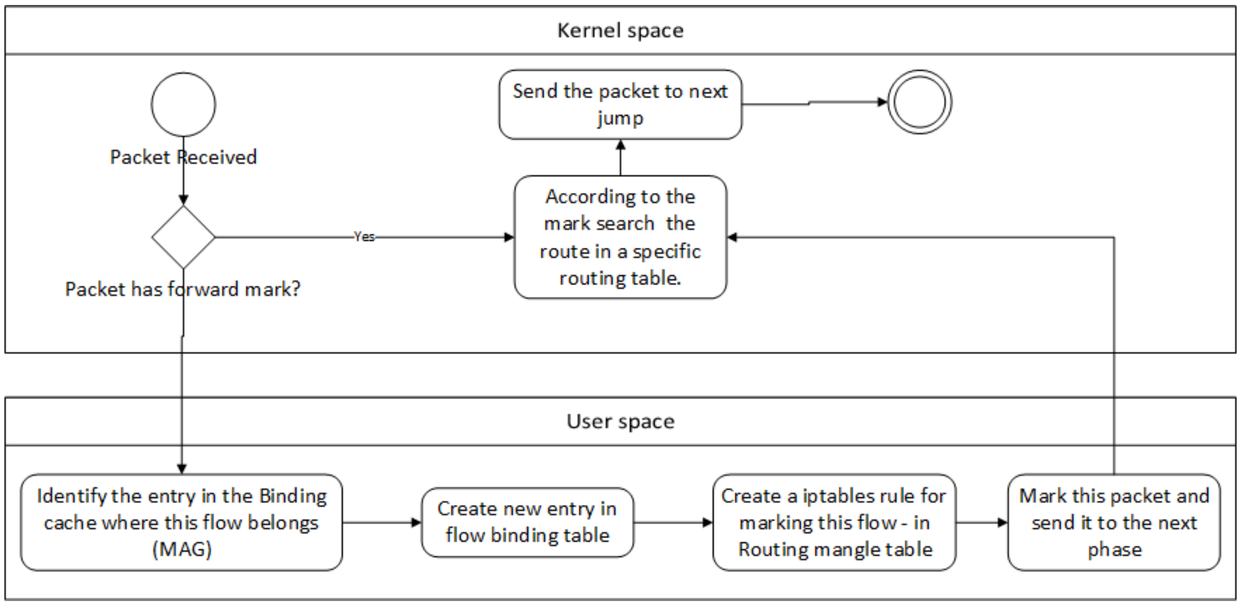

Kernel space
Send the packet to next jump
Packet Received
According to the mark search the route in a specific routing table.
Yes
Packet has forward mark?
User space
Identify the entry in the Binding cache where this flow belongs (MAG)
Create new entry in flow binding table
Create a iptables rule for marking this flow - in Routing mangle table
Mark this packet and send it to the next phase

Figure 19 – User space packet handling

4.2. Testbed Setup

This section describes in more detail: how the software was tested, used equipment, the configurations used for the software and the multiple tools used to perform the tests and collect the results.

The scenario for the tests consisted in one LMA and four MAGs, as shown in Figure 20. The four MAGs are installed in two equipments. In the testbed each equipment provides two distinct WiFi networks. In this situation each equipment provides one network in 2.4 GHz band and another in the 5 GHz band.

Since different tests require different configurations, for flexibility, the LMA was installed in a virtual machine. The client is also a virtual machine that has two WiFi interfaces available, one supporting only the 2.4GHz band and the another supporting both 2.4 GHz and 5 GHz band. Both were available via USB dongles.

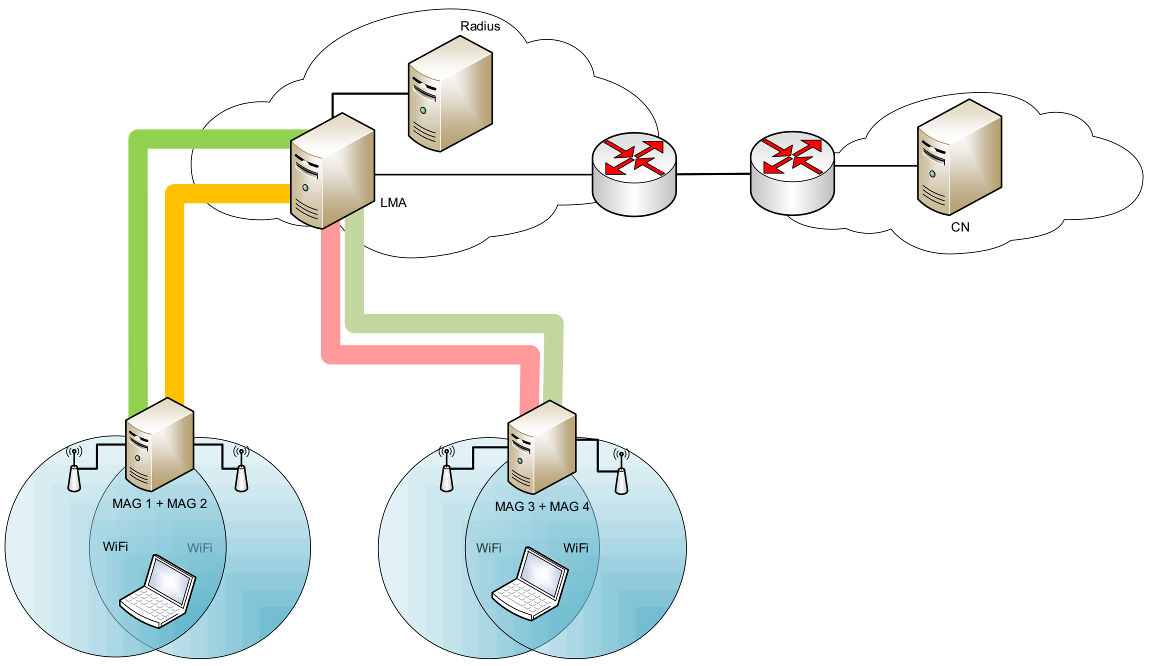


Figure 20 – Generic testbed network diagram

This testbed layout was planned taking in account one of the main objectives that this work wanted to achieve and demonstrate the feasibility of multi technology femtocell and their handover capabilities

The MAGs were deployed Small Office Home Office (SOHO) routers with the OpenWrt[13] OS. We have chosen this type of routers because of their capabilities like: low cost, extensive documentation, open source software and the support for plug and play external hardware, like USB dongles. These factors were essential to speed up prototyping and testing. Having a Linux OS installed has permitted the configuration in a single equipment of two independent APs with different technologies, simulating a multi technology femtocell.

In total, this testbed provides four independent APs using two SOHO routers, enabling us to handover the flows between two access points in the same equipment or between two different equipment. Another advantage of this type of equipment is the available support for using OpenWrt OS. The OpenWRT OS is a custom Linux distribution tailored to embedded devices. This OS has all of the advantages of a Linux system, namely the customizability and the performance. The OpenWrt project also gives us tools to cross-compile software to the different hardware it supports, and with this tools the MAG software may be developed to a generic architecture and later cross-compiled to router specific architectures.

Contrary to the MAG, the LMA must be a more capable machine. The LMA must act as a gateway in the network and process all the traffic addressed to MNs.

As above mentioned, the client is a laptop with two WiFi interfaces. As expected the only extra software that was installed in the client was the software to make the network measurements.

[13] www.openwrt.org

To isolate the entities involved in the tests from the rest of the network, they were all connected through a local switch. To complete this scenario a server that acts as a CN was added at an external network. This server acts as a simple traffic generator that can send one or more data flows to the MNs. This server could also work as a traffic sink.

Despite the fact that PMIPv6 can be used to provide mobility between heterogeneous networks, our tests consider networks from the IEEE 802.11 family. The support for 3G networks require specialized hardware and software that is not easy to obtain, but our solution is generic enough to be easily used with other wireless access technologies. To test our solution with 3G networks its required to have at least two USB 3G dongles, one connected to the MAG and another acting as the client interface. We would then create a IPv4 tunnel through the 3G network between the two nodes. In this situation the communication between the client and the MAG is done through the 3G network. The PMIPv6 and IPv6 configuration related messages are sent encapsulated in IPv4 packets through the tunnel.

Table 5 shows with more detail the hardware that was used in the testbed. All entities, except the MAG, were installed in virtual machines. Despite of the virtual machine overhead, we decided to use them to facilitate the deployment process. With the virtual machines it's very easy to re-deploy all the network in a different hardware configuration, or even replicate entities with different configuration parameters. The VMWare Workstation 10[14] was uses as the Hypervisor.

Table 5 - Testbed hardware

Hardware	CPU	RAM	Network	OS	Function
TP-Link WDR4300	Atheros AR9344 560MHz	128 MB	Gigabit Ethernet	OpenWrt 12.09	MAG
Server 1	Intel P4@2.4GHz	942 MB	Fast Ethernet	CentOs 6.5	VM Host
Server 2	Intel E5335@2GHz (x2)	4GB	Gigabit Ethernet	CentOs 7	VM Host
Laptop	Intel 3537	4GB	Gigabit Ethernet	LMDE	VM Host
Alfa AWUS036H	-	-	WiFi 2.4GHz	-	Client Interface
TP-Link TL-WDN3200	-	-	WiFi 2.4/5GHz	-	Client Interface

The TP-Link router serves as host to the MAG related software. This router is using a customized version of the OpenWrt OS[15]. The installed OS is a customized image of the official OpenWrt 12.09 release that was installed with the MAG software, the ODTONE entities, the MIHF and the Link Saps developed or customized for this research.

Each physical router has two MAGs instances, one MIHF instance and two Link SAPs instances. This set of software is able to provide two independent wireless accesses to MNs. One of the wireless access

[14] www.vmware.com
[15] https://openwrt.org/

is working on the 2.4 GHz band and the other in the 5 GHz band. Both the ODTONE entities and the MAG software had to be cross compiled to the TP-Link hardware. The necessary configuration files are available in this project source code repository[16].

As above mentioned, for convenience, on the hardware specified in Table 5, virtual machines have been installed as shown in Table 6.

Table 6 - Virtual machines used

VM Host [Table 4]	Cpu Cores	RAM	OS	Kernel	Description
Server 2	4	1 GB	Debian 7	3.15.7 x86_64	LMA with IPTables and Radius Server
Server 2	4	1 GB	Debian 7	3.15.7 x86_64	LMA with NFTables and Radius Server
Server 1	1	512 MB	Debian 7	3.2.0 x86_32	Traffic generator
Laptop	1	1 GB	LMDE	3.11-2-486	Client

The four virtualized entities were distributed by different physical machines. The table shows two LMAs but only one was used at a simultaneously.

Traffic generators and Measurement software

To emulate the data flows of the users, several servers, generating data flows to a MN connected to the LMD, have been used:

- Poisson traffic generator [17]
  This software generates a traffic flow following a Poisson distribution.
- Constant traffic generator - IPerf[18]
  The main use for this tool is to measure network performance, but for this work only the traffic generation capabilities have been used. It can generate both TCP and UDP Constant Bit Rate (CBR) streams.
- Generic traffic generator – MGEN 5[19]
  This is a toolset for generating multiple types of traffic. This work has used it to generate a burst traffic pattern.

As will be seen in chapter 5 this work requires the measure of two values, the time to forward a packet and the handover time. The handover time is measured in the MN, as the time between the last packet

[16] https://github.com/hugo-ma-alves/OAI-PMIPv6-FM
[17] http://www.spin.rice.edu/Software/poisson_gen/
[18] https://iperf.fr/
[19] http://www.nrl.navy.mil/itd/ncs/products/mgen

received before the link down and the first packet received after the link down. This measurement is made by parsing a PCAP file generated by the TCPDump[20] software running in the MN. This software captures all the traffic, including timestamps, and stores it in the PCAP file. Then this file is parsed by a python script to extract the handover value.

To measure the time to forward a packet this work developed a custom measurement toolset. The packet forward time must be measured in the most accurate way possible. Ideally, measurements should be made by external hardware/software that doesn't interfere with the machine running the PMIPv6 software. To minimize this problem, a kernel module was developed to extract the necessary information from the packets that are routed by the LMA and the MAG. With a kernel module it's possible to obtain results with an adequate precision without a significant performance impact on the machine running the module. The developed software uses the netfilter framework present in Linux Kernel. This framework allows the registration of callbacks functions at important phases of the packet processing chain.

Figure 21 shows a simplified diagram of Linux network stack with the netfilter hooking points represented with orange background. In each one of this points the module has access to a c structure called sk_buff. This structure has the packet raw data and some metadata. The metadata includes the timestamp in ns of when the packet first entered the chain. For example, a packet that was received by the hardware interface will arrive to the chain in netif_receive_skb, and the kernel will create for the first time the skb_buff structure with the timestamp at that moment.

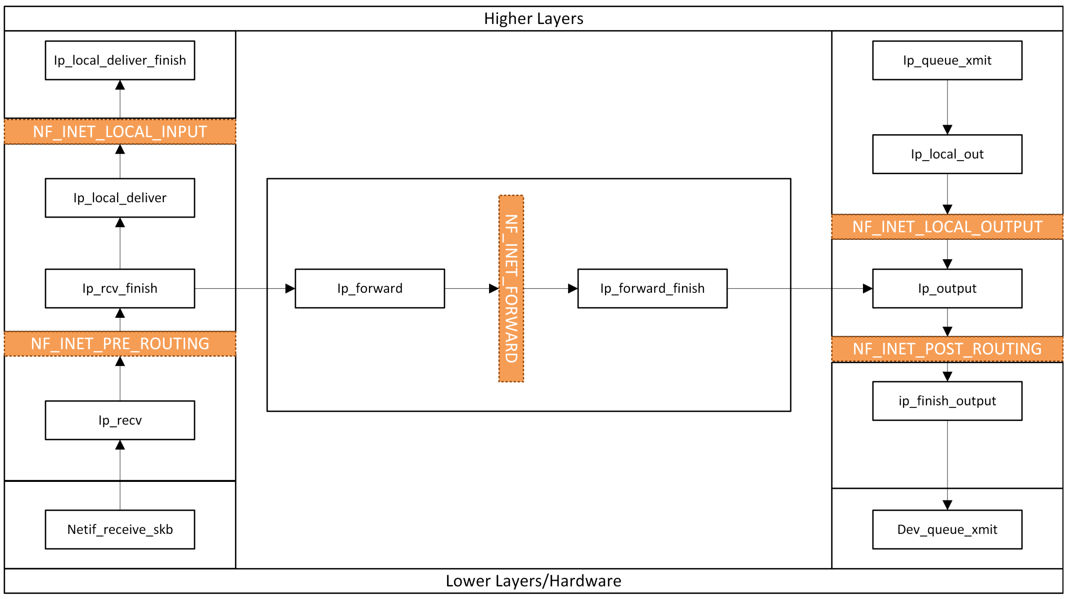


Figure 21- Netfilter Hooks

The module is executed for each packet that traverse the selected hook. The values that the module extracts from the skb_buff are stored in memory. This memory is accessed through a virtual file in the Linux virtual file system "proc".

[20] http://www.tcpdump.org

In the MAG the module registers at the NF_INET_POST_ROUTING hook. At this point the module can easily calculate the packet process time by making the difference between the current timestamp and the timestamp saved in sk_buff. Due to limitations in the MAG memory and processing capabilities, this module only stores in memory the packet process time.

The LMA is a bit more complex because it uses tunnels to encapsulate and forward traffic to the MAGs. In this type of scenario, a single packet will traverse the chain twice, in a kernel 3.x. One when it enters in the LMA coming from the CN and another when it's encapsulated and sent through the tunnel. Therefore, for the LMA it is necessary to run two different modules: one that registers itself on NF_INET_PRE_ROUTING that access the raw packet that enters in the LMA coming from the external network, i.e. from the CN; and another in NF_INET_POST_ROUTING that access the packet that is forwarded through the tunnel to the MAG.

Each module saves in memory a structure with the following values: entry timestamp, exit timestamp, transport source port, transport destination port, UDP packet checksum.

With these values, it's possible to match the data collected from the two modules to obtain a unique set of data. The packet entry timestamp is collected from the pre routing module output and the packet exit timestamp is collected from the post routing module output.

5. Evaluation

This chapter presents the evaluation results of the developed software. The objectives of the evaluations are to verify if the software works as expected and to evaluate the gains of having flow mobility.

The following sections describe in detail the tests performed and their results. Section 5.1 compares the performance of the developed solution when using IPTables or NFTables. As seen in the previous chapters the developed solution can use any one of these packet filtering framework. This test compares the packet forward time and the time to setup a new flow of both solutions.

Section 5.2 has measured the extra the packet forwarding time, at LMA and MAG, when using flow mobility. Then section 5.3 also measured the packet forwarding time in both LMA and MAG, but when there is no flow mobility. With these results, we can see in detail what is the cost, in terms of extra delay, of adding flow mobility capabilities to a PMIPv6 network.

The tests conducted in section 5.4 measure the delay introduced by the flow identifier module. As seen in previous chapters the flow identifier module must analyzes the packets of the new flows to identify a route to them. This test has measured the packet forwarding time of the first fifth packets of the flow.

Finally, section 5.5 presents the handover time results. These tests measured the time after a link has gone down until the handover of the affected data flows occur.

Figure 22 shows the testbed network diagram were the results shown in this chapter were measured. The testbed shown in Figure 22 is a simplification of the testbed already shown in section 4.2. The network created is composed by one LMA, two MAGs, one multihomed MN and one CN. The data flows are UDP streams, generated with Iperf, that have origin in the CN. For these tests the LMA FSM randomly chooses one route to each flow, in practice the load is evenly distributed by the available MAGs.

In this network the two MAGs are provided by a single equipment, a femtocell. This femtocell provides two independent WiFi networks. Each femtocell has two instances of the MAG software running. In all tests performed the MN is a weak host with two WiFi network interfaces.

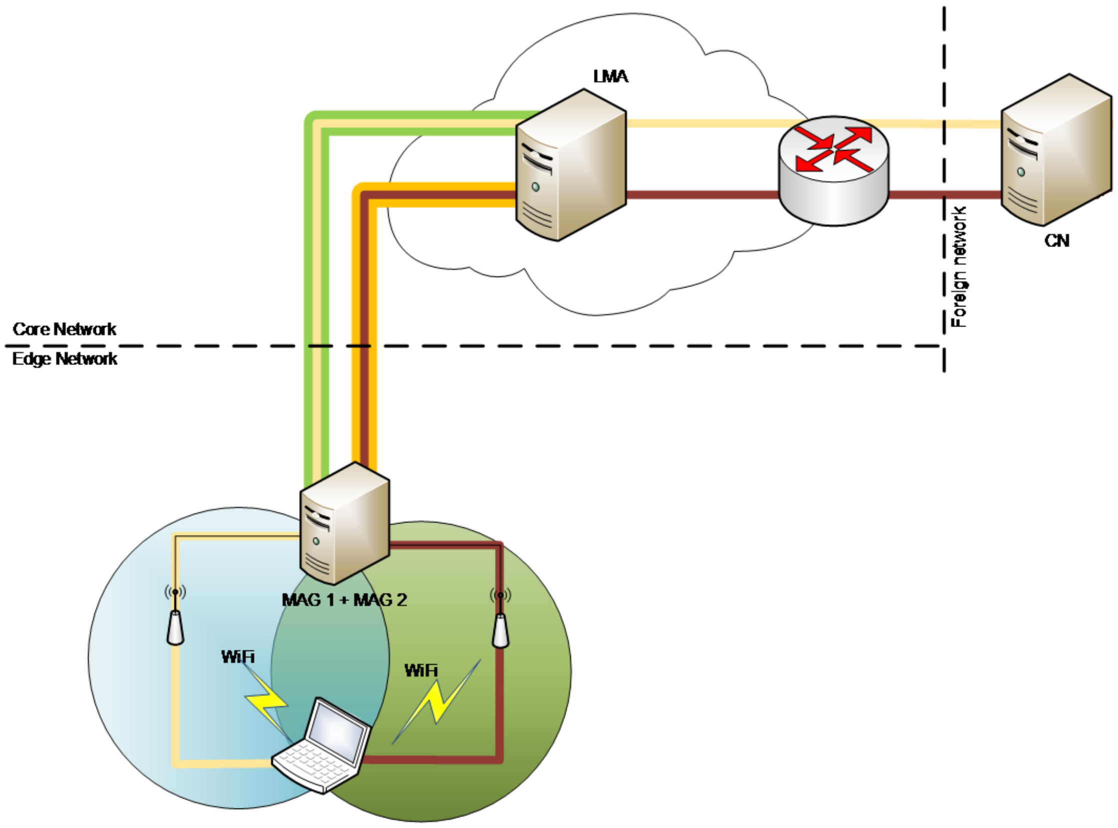


Figure 22 –Testbed used in evaluation

5.1. IPTables and NFTables performance

As described in previous chapters, the LMA may use two different packet filtering frameworks to forward the packets based on traffic selectors. This test compares the performance of IPTables and NFTables. In this test the performance is measured as the time taken to forward a packet since it enters the kernel until it leaves the device, and the time it takes to setup a new flow.

Both IPTables and NFTables frameworks consist in a list of rules to be matched against a packet. That have their performance limited by the speed of the search algorithm in use. The worst case scenario happens when the rule that matches the packet being processed is the last rule to be searched either by IPTables or NFTables.

This test consisted in having one constant traffic flow from the CN to the MN. The flow traversed the LMA and the MAG until it reached the MN. During the test new rules were being added to the IPTables or NFTables at regular intervals, see Figure 23. These rules didn't match any real packet in the system, and they were inserted to delay the IPTables and NFTables search algorithm on purpose. The increment on the number of rules emulates a scenario where the LMA is being congested by an increasing number of flows, and the only flow that has real traffic is the oldest one.

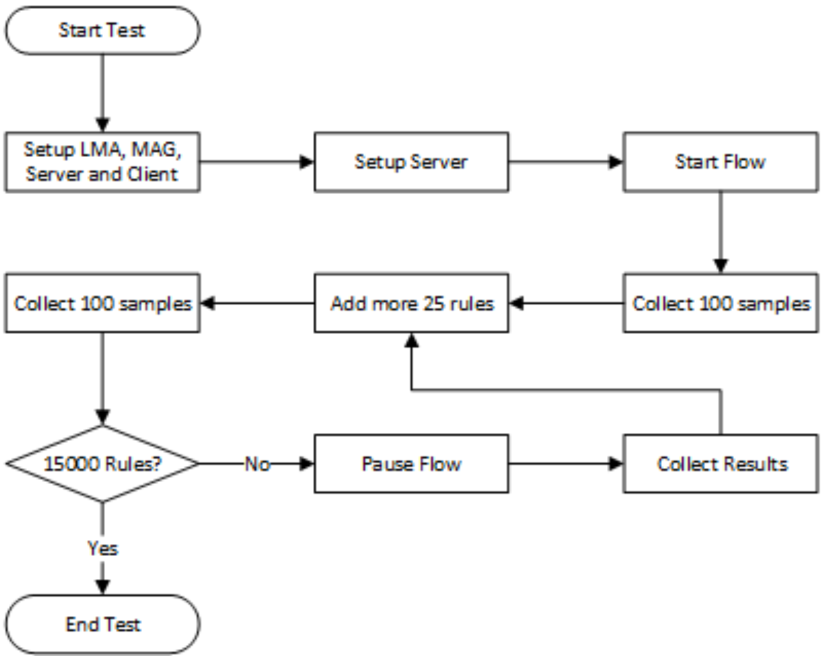


Figure 23 - IPTables vs NFTables test scenario diagram

This test has measured two parameters, the packet processing time in the LMA and the time that it takes to insert a new rule either in the IPTables or in NFTables. Both of the values were plotted in function of the number of rules in IPTables or NFTables.

Figure 24 shows the evolution of the packet processing time, for both IPTables and NFTables, as a function of the number of rules in the system. The figure shows clearly that both solutions have higher packet processing times when the system has a high number of rules installed, as it was logically expected.

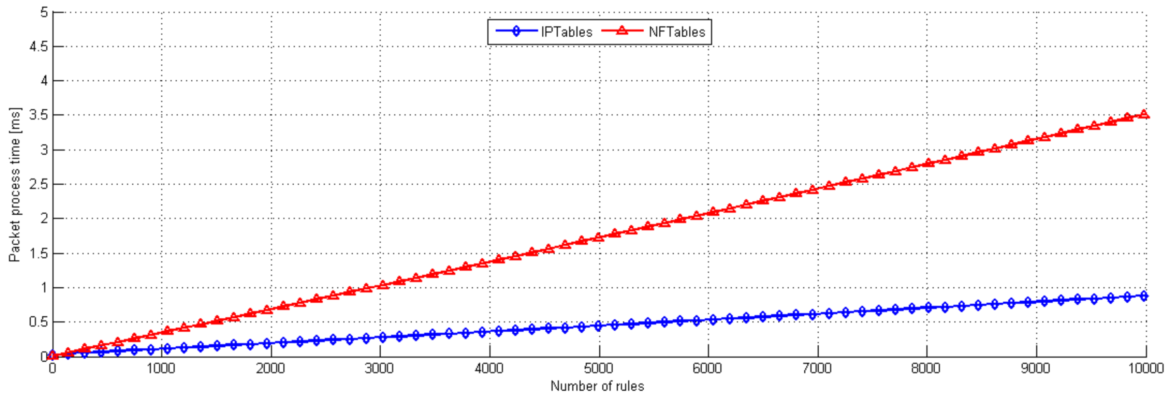


Figure 24 - IPTables vs NFTables packet process time at 10 Kbps

Even with the increase of the number of rules, the IPTables presents an almost stable performance. The packet processing time when using IPTables increases slower than the scenario using the NFTables. When using IPTables the worst case scenario has a packet processing time of 1 ms. Despite being a very high value to be considered in a real environment, it is substantial better than the 3.5 ms of the NFTables. Both values only occur on a limit case, when there are 10000 flows being processed by a single LMA.

When we analyze the same situation for a 100 Kbps flow, see Figure 25, we can see that the results are very similar.

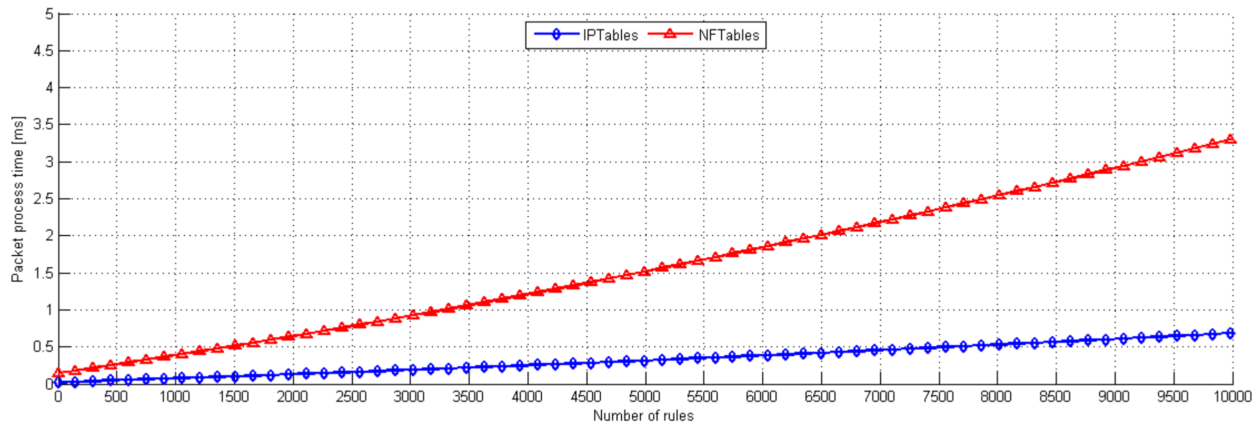


Figure 25 - IPTables vs NFTables packet process time at 100 Kbps

Even the 3.5 ms maximum delay with NFTables can be considered a reasonable value to a packet forward time in a prototype. Even for Voip conversation a latency of 3.5 ms is very tolerable, the maximum one-way latency for a high quality conversation should be at maximum 150 ms [43], [44].

We will now analyze in more detail the behavior of both NFTables and IPTables until the 500 rules mark. As seen in Figure 26, until the mark of 500 rules both solutions have a relatively small packet processing time value. However, it is obvious that the IPTables has the most stable behavior, and it's always faster than NFTables. Both the 10 Kbps and 100 Kbps curves of IPTables don't have packet process times higher than 75 μs. On the contrary, the NFTables solution has an increasing packet processing times when more rules are added. Even when there are few rules on the system the NFTables presents a bigger packet process time than the IPTables, except for the 10 Kbps test when there are less than 50 rules on the system.

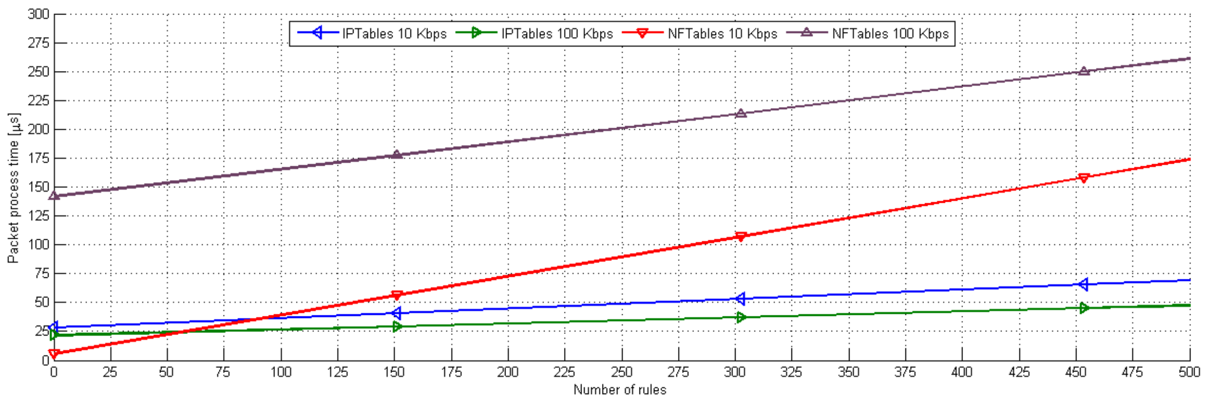


Figure 26 - IPTables vs NFTables packet process time summary

Another aspect to consider, when choosing between IPTables and NFTables is the time that it takes to add a new rule to the system. This is a very important metric to take into account. A slow insertion of the rule in the system delays the other flows, and this may deteriorate the users Quality of Experience (QoE).

When we consider the time it takes to insert a new rule in IPTables or NFTables, the results are the opposite of the packet process time. As seen in Figure 27, the NFTables presents a constant value independently of the number of rules already inserted in the system. On the other side, the IPTables time to insert a rule is very sensitive to the number of rules already there and it is proportional to the number of rules in the system. When there are 10000 rules on the system the time it takes to insert a new rule is about 0.7 s.

Until the 3000 rules mark the IPTables has a smaller packet processing time and a smaller time to insert a new rule, when compared to NFTables. If it’s expected that the LMA won’t have to manage more than 3000 flows, the IPTables is the most appropriate solution. But if it’s expected that the LMA has to handle more than 3000 flows, the NFTables framework should be considered instead. For this kind of load, the NFTables has a bigger packet processing times but it has a lower time to insert rule.

If we just analyze the results between 1 and 500 rules at 100 Kbps, see Figure 27 and Figure 28, both solutions have a apparent constant time to insert a new rule. Regarding the packet processing time in this region both frameworks have a relatively stable behavior, in this case the IPTables outperforms the NFTables by about 160 ms.

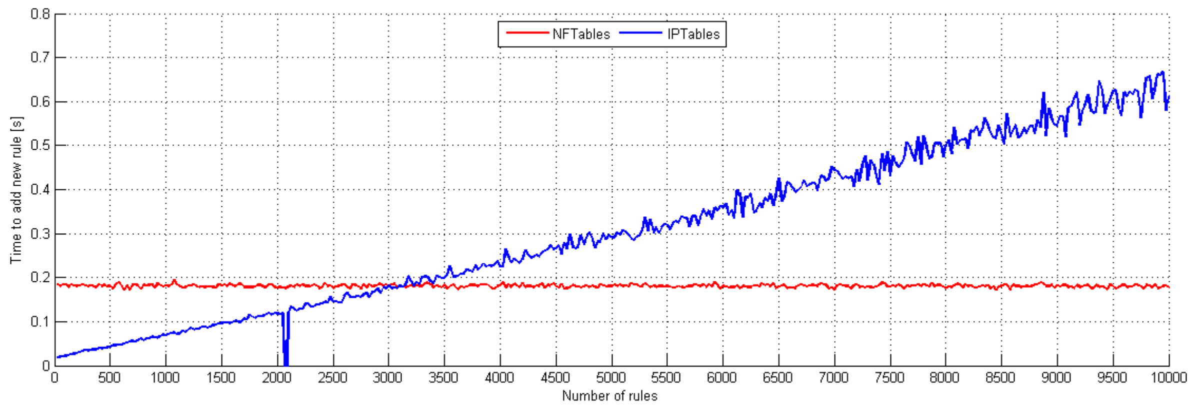


Figure 27 - IPTables vs NFTables time to insert new rule at 100 Kbps - detail

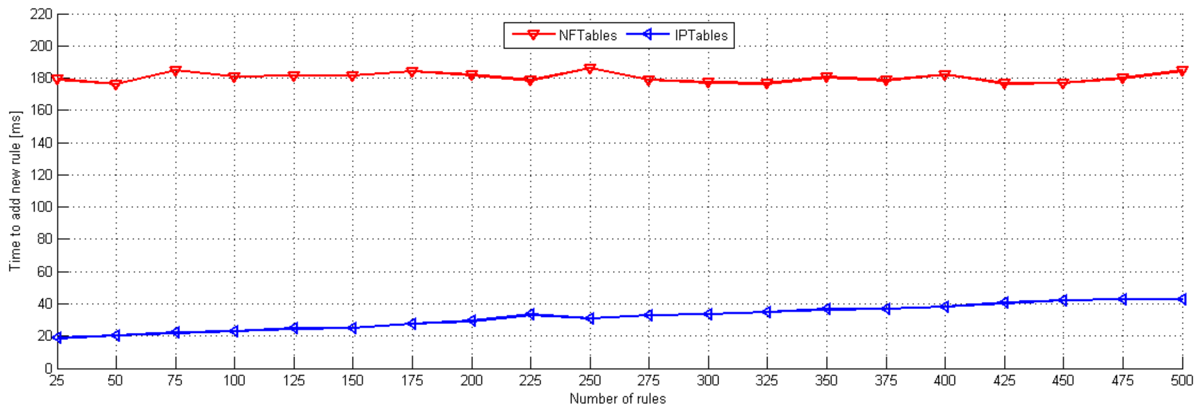


Figure 28 - IPTables vs NFTables time to insert new rule at 100 Kbps - detail

Taking in account both metrics, the packet processing time and time to insert a new rule, it's not easy to say what's the best framework. If a load higher than 3000 simultaneously flows is expected, it's more adequate to use IPTables instead of NFTables.

In this case it's also necessary to consider the volatility of the flow mobility cache. If the system has a very dynamic number of flows, being deleted and created, it starts to make more sense to use NFTables, due to the fact that it is faster when creating and deleting forwarding rules in the system.

For a small deployment, with fewer than 3000 simultaneously flows, the performance difference between IPTables and NFTables has no real impact.

5.2. Packet forward time at LMA and MAG

This test focused on measuring the delay introduced by the LMA and the MAG in the network. In see section 5.3 these results are also compared with a situation where there is no flow mobility support, i.e., a standard PMIPv6 network.

The results shown below were measured in a stable situation where all the flows were already processed by the LMA user space software. In this situation the only entities involved in the packets forwarding is the kernel and IPTables. This test was not made with NFTables because the purpose of the test is to measure solely the impact of the number of flows and the throughput in the packet processing times in LMA and MAG. And as the previous test has shown for a relative low number of flows the performance differences of IPTables and NFTables is irrelevant.

Figure 29 shows the empirical distribution function results for the packet processing times of the MAG and LMA for different throughputs. Figure 29 shows that the MAG packet forward time is not affected by the throughput until the 100 Kbps per flow. For 50 flows at 100 Kbps each the MAG can forward 90% of the packets in less than 10 μs. In the MAGs plot the results for 1 flow at different rates is always worse than the results for the higher throughputs. Once again this happens because of the optimization made in the OS by the NAPI. This was the expected behaviour for the MAG, in this scenario the MAG is just acting as a bridge between the LMA and the MNs, it doesn't make any relevant task that could delay the packets. The tests shows that the maximum combined throughput of all the flows is 5 Mbps, the MAG forwards all the packets in less than 15 μs.

The LMA on the other hand presents different results. The first thing to notice is that the forwarding times of the LMA are approximately twice the ones got on the MAG. The reason to this behaviour is the complexity of the LMA routing tables. The LMA has a set of forwarding rules slightly more complex than the MAG. As seen in the previous chapters the LMA must check all the incoming packets to verify if the flow has already a defined route.

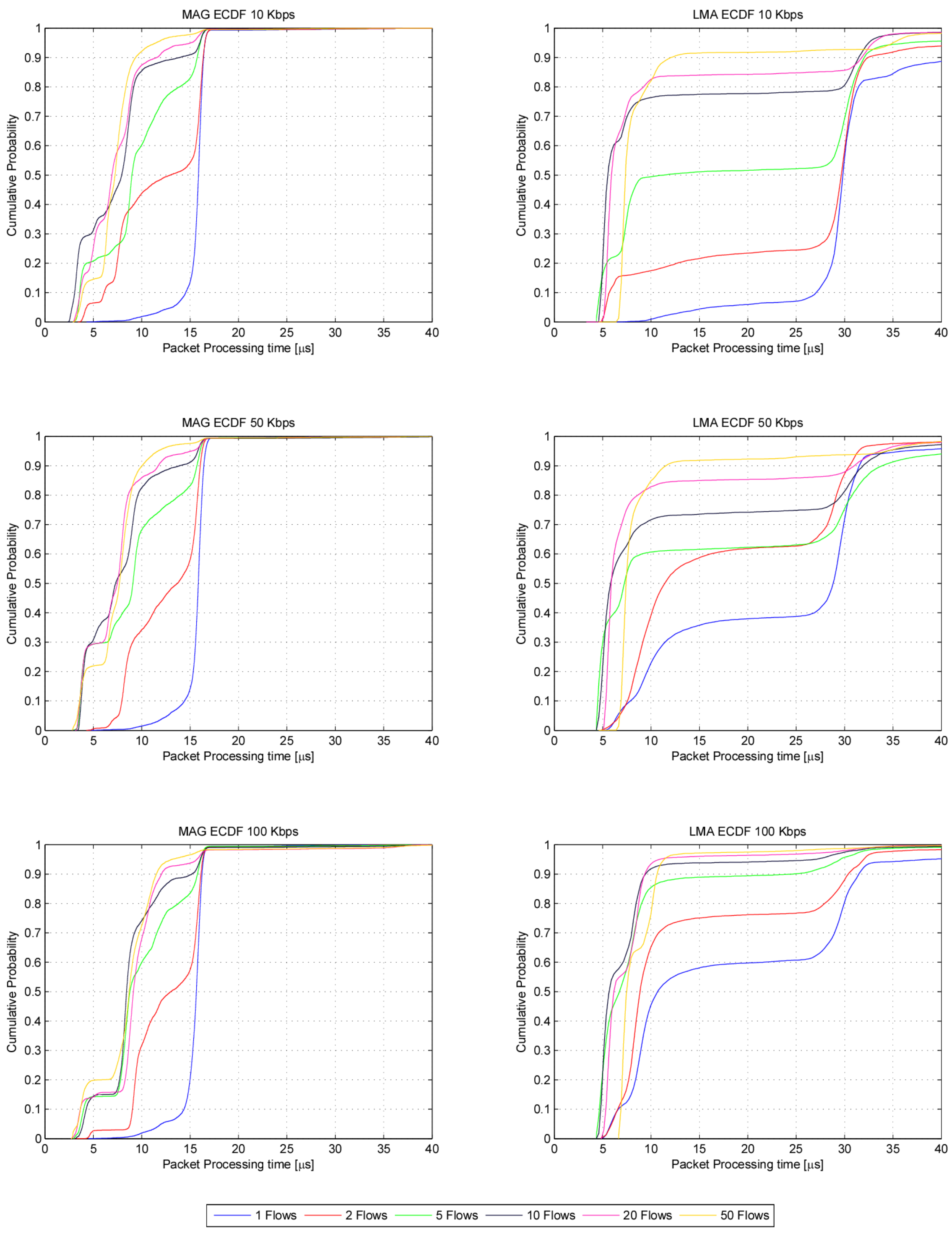


Figure 29 – Packet processing time at LMA and MAG for different throughputs

For the sake of the clarity of the results, another relevant aspect to notice in Figure 29, is that the higher throughputs have apparently lower packet process times. This is a contradiction to the expected behaviour. The behaviour that was expected is the increasing of the packet processing while the machine load starts to increase.

In reality, the phenomenon that it's shown in Figure 29 it's an optimization made by the OS to improve the performance in high throughput networks. To understand this optimization, it's important to give a brief explanation about how the network driver works.

Most of the OSs use an interrupt based mechanism to react to hardware events. When the network interface receives a frame it puts that frame in a buffer, this buffer is shared with the OS. After inserting the frame in the buffer, the network interface sends an interrupt to the CPU. The CPU detects the interrupt and immediately calls the correspondent driver that handles that specific interrupt, and the frame starts being processed.

It's rather obvious that with high throughputs the interrupt rate will be excessively high. This is a problem because the hardware interrupts have priority over all the other tasks that the OS is executing. And the CPU must spend resources saving the actual state/caches so that after the interrupt processing it can return to the exact same status that it had before the interrupt.

In networks with a high packet arrival rate this behaviour can slow down all the system [45], [46]. In a worst case scenario the system can drop its throughput to 0, this happens when the CPU is spending more time processing the interrupts than performing other tasks [47].

This is a serious problem in high throughputs networks, most commonly known as gigabit networks. To overcome this problem the Linux has introduced a new mechanism to avoid the high number of interrupts, the New API (NAPI) [48]. This mechanism removes the inconveniences of having one interrupt for each packet.

With NAPI, the driver has to provide a buffer, either in the OS or in hardware, to store the frames received from the wire, normally a ring buffer. Additionally, it has to provide a poll method that retrieves received frames from the network interface. This method can be invoked by the networking subsystem of the Kernel when it wants to get a batch of frames to start the processing in the network stack. Basically the interrupt based strategy is replaced by a poll mechanism.

When the driver receives the first hardware interrupt it disables further interrupts until: the call of a poll method, when enough packets are received, or a specific timeout is reached. Then hardware interrupts are re-enabled. The moment when the driver goes from poll mechanism to interrupt mechanism is not defined.

The poll strategy can significantly improve the performance in high throughputs networks, like shown by a test made by Linux Foundation[21] where a node receiving 890 Kpps, only 17 interrupts were generated. If the NAPI hadn't been used, this scenario would have generated 1 interrupt for each received frame. This kind of load would slow down significantly any machine.

The nodes that use this mechanism to forward traffic at low throughputs will introduce an extra latency in the network. For example, if the driver waits for 2 frames before starting the frame processing, and the system is receiving 1 frame per second. In this situation the frames that was received first will have

[21] http://www.linuxfoundation.org/collaborate/workgroups/networking/napi

an extra delay of 1 second. In this situation it is more beneficial use the traditional hardware interrupts based mechanism.

The NAPI behaviour is shown in the Figure 29 plots. The NAPI is introducing a delay for the low throughput traffic. As soon as the throughput starts to increase the packets processing time converge to the same range of values. Disabling the NAPI requires the recompilation of the network driver. This type of details must be taken in account for a production environment.

5.3. Packet forward time without flow mobility

The results shown in Figure 29 are relative to a scenario were flow mobility is enabled. Figure 30 compares the previously obtained results, for 100 Kbps, with the same scenario but without using flow mobility. This scenario uses the standard PMIPv6 implementation, from Eurecom, without flow mobility capabilities.

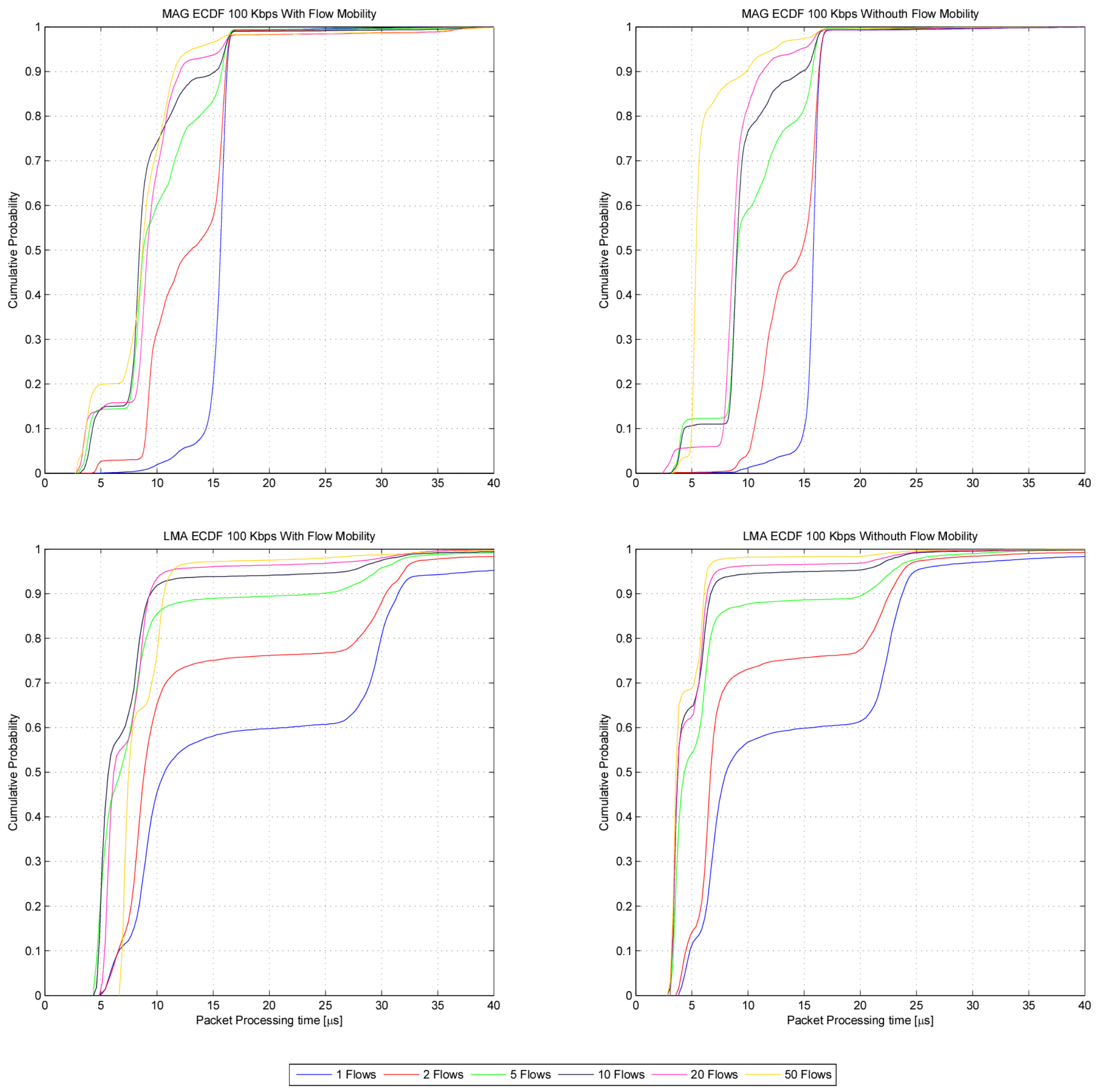


Figure 30 – Packet processing time with and without flow mobility

The MAG results of the scenario without flow mobility are identically to the scenario with flow mobility. This was the expected result, since the changes made to the MAG to support flow mobility weren't expressive.

It is expectable to have a slightly worse performance in the LMA when flow mobility is enabled, since the LMA has suffered significantly changes that can influence the normal forwarding of the packets.

A brief analysis of the LMA results shows that the performance penalty for having flow mobility is actually extremely low. The solution without flow mobility is slightly faster than the solution with flow mobility, but delay values are kept under 30 μs in both scenarios. This is a very low delay value.

For 1 and 2 flows the solution without flow mobility can forward 90% of the packets under 23 μs. But the solution with flow mobility can only guarantee that 90% of the packets are forwarded under 33 μs. For higher throughputs, 10, 20 and 50 flows the situation is identical, and the LMA can forward 90% of the packets under 7 μs in the solution without flow mobility versus the 11 μs in the solution with flow mobility.

This results have shown that our proposal introduced only a marginal delay in the network when compared with the previous solution without flow mobility. Some delay was already expected, since the proposal of this work has introduced changes in the way how the packets are routed. LMA must analyze all of the existing flows in order to apply forwarding routes to each one individually. This level of detail in the routing has to have a cost. Despite of the extra delay introduced, this test has shown that the extra cost is almost insignificant when compared to the advantages brought to the clients and to the operator. For 90% of the packets the difference between the two solutions is on the worst case 5 μs in the LMA. The MAG performance is identically in both scenarios.

5.4. Delay introduced by the flow identifier module

As explained in Chapter 3 the first packets from a user flow must be redirected to the user space so that the LMA can process them. After the packets have been identified by the LMA the forwarding is done in kernel space. It is expected that the packets that doesn't reach the user space have smaller forwarding times. It is also expected that a higher throughput increases the packets forwards time, this happens because the queue that is in the user space starts to accumulate packets before the LMA can process them.

This test consisted in an environment with one LMA and two MAGs, the MAGs were installed in the same equipment. Then a server started to incrementally sending new flows to the client, each new flow is only started after the last flow has been processed in the LMA. The objective of this test is to measure only the impact of the user space code, the LMA flow identifier module. This test only measures the forward time of the first packet from a flow, the routing decision for this packet is taken in user space. The impact of the packet filter framework on this situation is negligible.

Figure 31 shows the results of this test, in x axis it's represented the nth flow to be added and in y axis the time taken to process the first packet of that flow. For example, the flow 5 was the fifth flow to be added to the system. The plot shows that for a low number of flows in the system, the time taken to setup the route for the first packet of a flow is independent of the number of flows in the system. Setting up the 50th flow takes approximately the same time that it takes to setup the first. It's also shown that until the 50th flow the time taken to process the first packet is independent of the throughput.

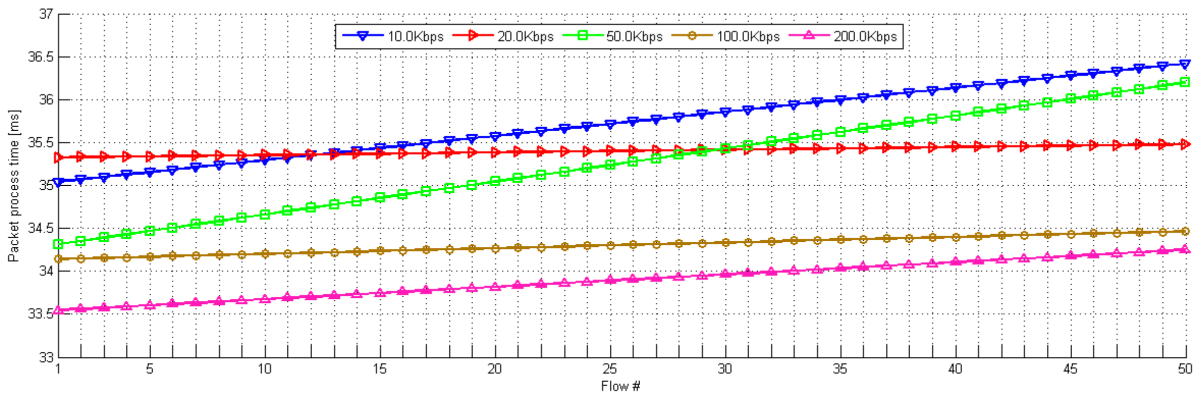


Figure 31 - Time to process the first packet at 100 Kbps per flow

For all throughput values shown in figure, the time to process the first packet is always greater than 33 ms. This can be an unacceptable value for real-time traffic. Fortunately, this only happens for the first packet. After the first packet, see Figure 32, the packet process time starts to diminish rapidly to acceptable values, as also shown in section 5.2.

Figure 32 shows what happens to the packet process time when it is also considered the packet process time of the second to 10th packet. By other words, what is the penalty for having to divert the first packet of a flow to the user space for classification? The figure shows that the time to process packet diminishes rapidly after the first one. After the fifth packet the packet process time is less than 40 μs, the same values shown in section 5.2.

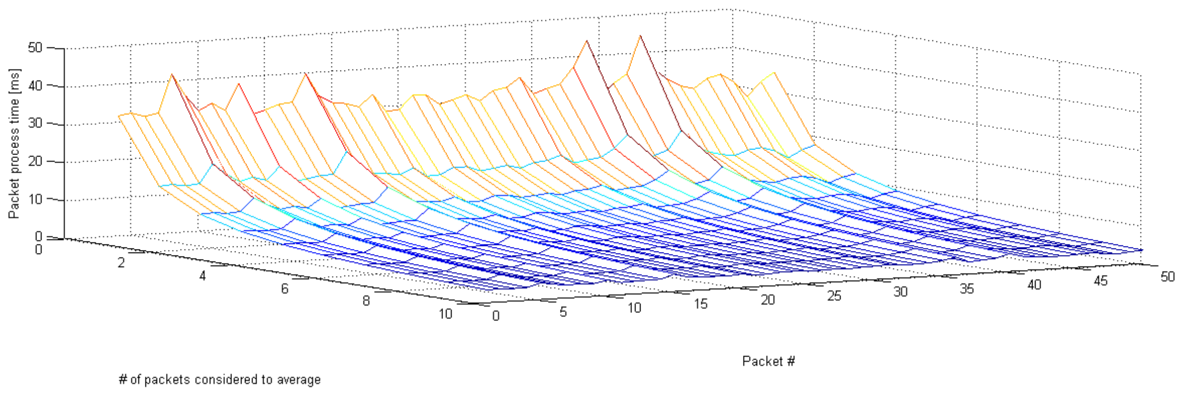


Figure 32 - Evolution of the packet processing time

This chapter shown that there is an extra cost to process the first packet, as it was expected. But that costs starts to get rapidly diluted after the 5th packet of a flow. Due to the gains that flow mobility has, this is a tolerable cost. This cost can be reduced by having more capable machines or by tuning the packet classification framework in Linux.

5.5. Handover Times

This test consists on having a client with two interfaces. Each interface is connected to a different MAG. One of the interfaces was receiving a constant stream of multiple flows, and the other interface was receiving one flow that suffered a handover. The objective of this test is to measure the time it takes to perform a handover.

Because of the limitations of the weak host model all of the flows were addressed to the IP of one interface. To do the handover the interface that wasn't being addressed was shut down. This shut down was detected by the Link SAP in the AP. The Link Sap then triggered the handover process.

If the interface that had the IP address that was being used as destination address was shut down, the MN would de-configure the IP address and all the traffic addressed to it would be dropped. This is a limitation of the weak host model.

The handover time was measured as the difference between two consecutives packets on different interfaces, both packets belong to the flow that has suffered the handover. The handover start time is the timestamp of the last packet received in the interface that has been shut down. And the handover end timestamp is the timestamp of the first packet received in active interface.

This measure is not a real handover time, it's a packet inter-arrival time. Since the data flow has a constant throughput, the period between packets will be approximately constant. If the average of the packet period is subtracted to the inter-arrival time measured during the handover, we will obtain a reasonable accurate measure of the handover time.

This methodology is only valid for high throughputs were the period between consecutives packets is very small. For low throughputs the period between packets is relatively high, there is a huge chance that the handover occurs between two consecutive packets, so in this case, our measure would be less accurate.

For this test to work correctly the traffic generated by the CN must have a constant throughput. Due to some hardware limitations on the CN machine the traffic generated didn't had a constant throughput.

Figure 33 shows the measured handover times. In the area shown, the maximum handover time is between 50 ms and 150 ms. Despite not being values that can be neglected, for some type of traffic they are perfectly tolerable.

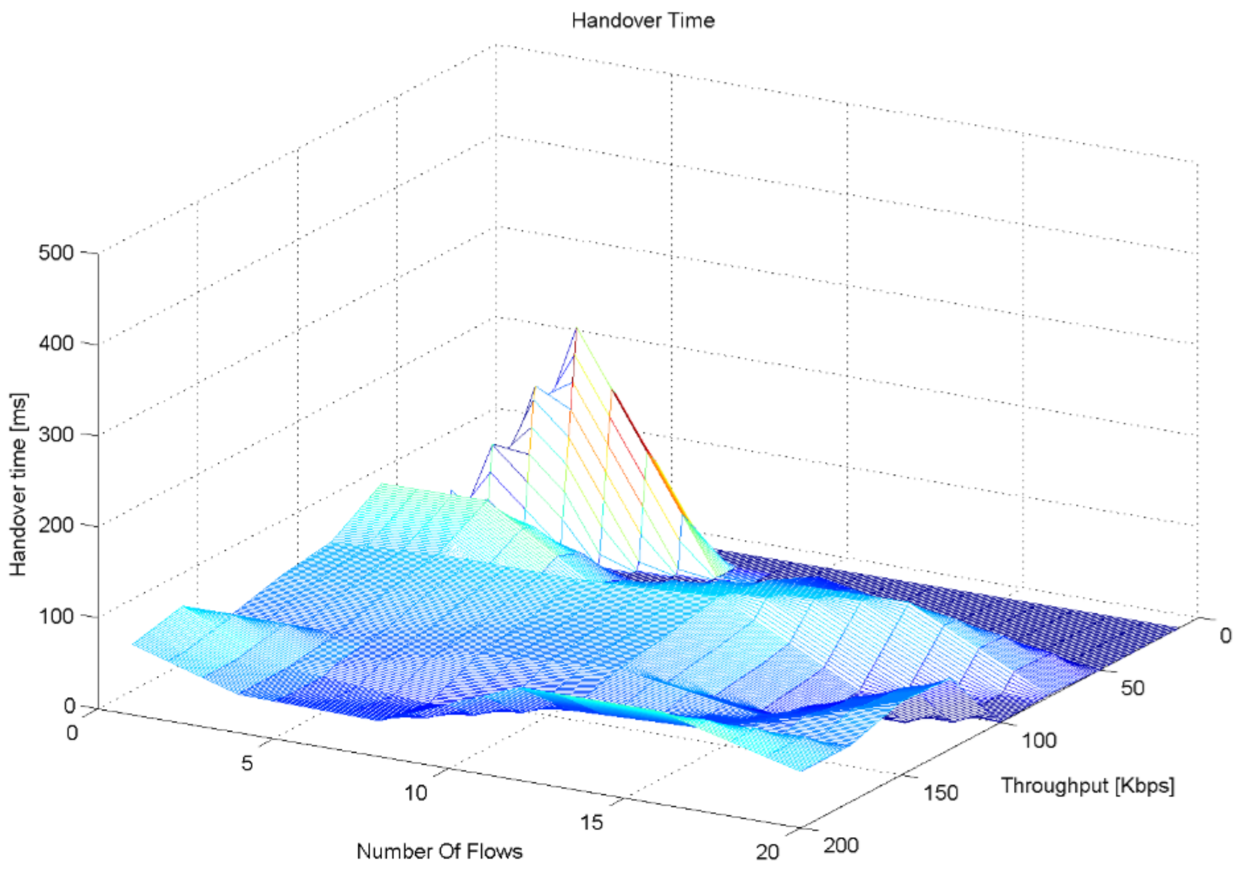


Figure 33 - Handover time

6. Conclusion

This work has presented a proposal to provide network based flow mobility to the MNs of an operator network. With this objective accomplished, the users can now benefit from theirs MNs multi access capabilities. On the other end, the operator can control the routing decisions to each specific flow. The possibility of having a centralized control gives to the operator the necessary tools to improve the QoS that it provides to its clients.

Another aspect of the developed solution that it's worth mentioning is the integration with the IEEE 802.21 framework. This integration has permitted to decouple the user tracking from the mobility protocol.

The scheduling of the flows was also decoupled from the mobility protocol. Each operator can replace the flow scheduler by its own. The proposal can support any custom implementation of a flow scheduler.

This work has also validated the solution by running tests with real equipment. It was shown that it is possible to obtain: (1) flow mobility transparent to the user, (2) implement the extensions defined by the NETEXT WG to provide flow mobility in a PMIPv6 network, (3) the penalty for having flow mobility is negligible when compared to the standard PMIPv6 without flow mobility and (4) that the operators have advantages for reusing the existing WiFi network before starting to expand the cellular network.

It's also important to note that the MAG can be as simply as a SOHO router. The MAG is a simple network node that the operator can ship to their customers houses at a low cost. This type of equipment can provide both WiFi and LTE to the clients of the operator in the neighborhood. It can be seen as a way to deploy a large number of smaller cells with a low cost for the operator.

For future work we suggest a full integration with IEEE 802.21 protocol. In this thesis solution IEEE 802.21 is only used to track the MNs movements. However, the IEEE 802.21 can provide useful information to improve the handovers. The network can start to be more proactive to deal with the variations of the links quality.

The next objective is to create a more intelligent broker in the network that communicates with the flow scheduler. The broker may improve the QoS by using all of the available information about the network status to route the QoS sensitive flows through the most reliable path.

7. References

[1] Cisco, “Cisco Visual Networking Index : Global Mobile Data Traffic Forecast Update , 2014 – 2019,” USA, 2015.

[2] Ericsson SA, “MOBILITY REPORT,” 2013.

[3] ITU, “Facts and figures,” 2011.

[4] Alcatel-Lucent, “Turning WI-FI Hotspots into a Hot Business Opportunity for Wireless Service Providers,” 2012.

[5] A. D. la Oliva, “IP flow mobility: smart traffic offload for future wireless networks,” *Commun. Mag. IEEE*, no. October, pp. 124–132, 2011.

[6] T. Melia, C. J. Bernardos, A. De La Oliva, F. Giust, and M. Calderon, “IP flow mobility in PMIPv6 based networks: Solution design and experimental evaluation,” in *Wireless Personal Communications*, 2011, vol. 61, no. 4, pp. 603–627.

[7] K.-K. Yap, “USING ALL NETWORKS AROUND US,” Ph.D. Dissertation, Stanford University, 2013.

[8] K.-K. Yap, T.-Y. Huang, M. Kobayashi, Y. Yiakoumis, N. McKeown, S. Katti, and G. Parulkar, “Making use of all the networks around us,” *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4. ACM, New York, NY, USA, p. 455, 2012.

[9] F. Giust, “IP Flow Mobility support for Proxy Mobile IPv6 based networks,” M.S. thesis, Università degli Studi di Padova, 2011.

[10] IEEE, “IEEE standard for local and metropolitan area networks: Media independent handover services - Part 21: Media Independent Handover Services,” IEEE Std 802.21, 2009.

[11] H. Alves, R. N. Marinheiro, and J. Moura, “Flow-Mobility for PMIPv6,” 2015, p. 4.

[12] P. Louin and P. Bertin, “Network and host based distributed mobility,” *Wirel. Pers. Multimed. Commun.*, 2011.

[13] M. Sánchez, C. Bernardos, A. de la Oliva, and P. Serrano, “Energy consumption savings with 3G offload,” *ict-crowd.eu*.

[14] K.-K. Yap, “Hercules- The Freedom to Use All Networks.”

[15] D. Johnson, C. Perkins, and J. Arkko, “Mobility support in IPv6,” RFC 6275 (Standard), 2011.

[16] S. Thomson, T. Narten, and T. Jinmei, “IPv6 stateless address autoconfiguration,” RFC 4862 (Standard), 2007.

[17] R. Droms, J. Bound, B. Volz, T. Lemon, and C. Perkins, “Dynamic host configuration protocol for IPv6 (DHCPv6),” RFC 3315 (Standard), 2003.

[18] S. Krishnan, J. Woodyatt, E. Kline, J. Hoagland, and M. Bhatia, “A Uniform Format for IPv6 Extension Headers,” RFC 6564 (Standard), 2012.

[19] S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” RFC 2460 (Standard), 1998.

[20] R. Wakikawa, T. Ernst, V. Devarapalli, G. Tsirtsis, and K. Nagami, “Multiple Care-of Addresses Registration,” RFC 5648 (Standard), 2009.

[21] G. Tsirtsis, H. Soliman, N. Montavont, G. Giaretta, and K. Kuladinithi, “Flow Bindings in Mobile IPv6 and Network Mobility (NEMO) Basic Support,” RFC 6089 (Standard), 2011.

[22] H. Soliman, “Mobile IPv6 support for dual stack hosts and routers,” RFC 5555 (Standard), 2009.

[23] Z. Zhang, X. Wen, and W. Zheng, “A NAT Traversal Mechanism for Peer-To-Peer Networks,” in *2009 International Symposium on Intelligent Ubiquitous Computing and Education*, 2009, pp. 129–132.

[24] S. Gundavelli, K. Leung, and V. Devarapalli, “Proxy Mobile IPv6,” RFC 5213 (Standard), 2008.

[25] Bruno Filipe and S. Santos, “Implementação de Gestão de Mobilidade Localizada usando PMIPv6,” 2011.

[26] G. Iapichino and C. Bonnet, “Experimental Evaluation of Proxy Mobile IPv6: an Implementation Perspective,” *Wirel. Commun. …*, 2010.

[27] B. Aboba, M. Beadles, J. Arkko, and P. Eronem, “The Network Access Identifier,” RFC 4282 (Standard), 2005.

[28] C. J. Bernardos, “Proxy Mobile IPv6 Extensions to Support Flow Mobility,” Internet-Draft, 2013.

[29] E. Piri and K. Pentikousis, “IEEE 802.21: media independent handover services,” *Internet Protoc. J.*, pp. 7–27, 2009.

[30] D. Corujo and C. Guimaraes, “Using an open-source IEEE 802.21 implementation for network-based localized mobility management,” *Commun. Mag. IEEE*, vol. 49, no. September, pp. 114–123, 2011.

[31] C. Bernardos, A. Oliva, J. Juniga, and S. Das, “PMIPv6 operation with IEEE 802.21,” Internet-Draft, 2009.

[32] A. Mateus and R. N. Marinheiro, “A Media Independent Information Service Integration Architecture for Media Independent Handover,” *2010 Ninth Int. Conf. Networks*, pp. 173–178, 2010.

[33] C. Bernardos, A. D. la Oliva, and F. Giust, “A PMIPv6-based solution for Distributed Mobility Management,” Internet-Draft, 2013.

[34] R. Barden, “Requirements for InterNet Hosts-Application and Support,” RFC 1123 (Standard), 1989.

[35] M. Wasserman and P. Seite, “Current practices for multiple-interface hosts,” RFC 6419 (Standard), 2011.

[36] T. Melia and S. Gundavelli, “Logical Interface Support for multi-mode IP Hosts draft-ietf-netext-logical-interface-support-10,” Internet-Draft, 2014.

[37] H. Balakrishnan and V. Padmanabhan, “TCP performance implications of network path asymmetry,” RFC 3449 (Standard), 2002.

[38] S. Krishnan, S. Gundavelli, M. Liebsch, H. Yokota, and J. Korhonen, “Update Notifications for Proxy Mobile IPv6,” RFC 7077 (Standard), 2013.

[39] J. C. Silva, J. A. Moura, and R. N. Marinheiro, “Optimizing 4G Networks with Flow Management using An Hybrid Broker,” 2010.

[40] A. Patel, K. Leung, and M. Khalil, “Mobile node identifier option for mobile IPv6 (MIPv6),” RFC 4283 (Standard), 2005.

[41] I. S. Association, “Guidelines for 64-bit global identifier (EUI-64),” 2007. [Online]. Available: http://standards.ieee.org/develop/regauth/tut/eui64.pdf. [Accessed: 28-Oct-2014].

[42] C. Guimarães, “Discovery of media independent services for heterogeneous networks,” M.S. thesis, Universidade de Aveiro, 2011.

[43] Telchemy, “Impact of Selay in Voice over IP Services,” 2006.

[44] SANS Institute, “Latency and QoS for Voice over IP This,” 2004.

[45] J. Mogul, D. E. C. Western, K. K. Ramakrishnan, T. B. Laboratories, and J. C. Mogul, “Eliminating Receive Livelock in an Interrupt-driven Kernel Eliminating Receive Livelock in an Interrupt-driven Kernel,” no. January, 1996.

[46] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, “A Study of Network Stack Latency for Game Servers,” *net.in.tum.de*.

[47] K. K. Ramakrishnan, “Performance considerations in designing network interfaces,” *IEEE J. Sel. Areas Commun.*, vol. 11, no. 2, pp. 203–219, 1993.

[48] L. Foundation, “NAPI.” [Online]. Available: http://www.linuxfoundation.org/collaborate/workgroups/networking/napi.