Final Degree Thesis

Towards Seamless Mobility

An IEEE 802.21 Practical Approach

By

RUBÉN GONZÁLEZ MUÑOZ

FINAL DEGREE THESIS 30 ECTS,

APRIL 2010, TELEMATICS ENGINEERING

UPC

telecom BCN

Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Abstract

In the recent years, mobile devices such as cell phones, notebook or ultra mobile computers and videogame consoles are experiencing an impressive evolution in terms of hardware and software possibilities. Elements such a wideband Internet connection allows a broad range of possibilities for creative developers. Many of these possibilities can include applications requiring continuity of service when the user moves form a coverage area to another.

Nowadays, mobile devices are equipped with one or more radio interfaces such as GSM, UMTS, WiMax or Wi-Fi. Many of these technologies are ready to allow transparent roaming within their own coverage areas, but they are not ready to handle a service transfer between different technologies. In order to find a solution to this issue, the IEEE has developed a standard known as Media Independent Handover (MIH) Services with the aim of easing seamless mobility between these technologies.

The present work has been centered in developing a system capable to enable a service of mobility under the terms specified in the stated standard. The development of a platform aiming to provide service continuity is mandatory, being a cross-layer solution based in elements from link and network layers supplying a transparent roaming mechanism from user's point of view.

Two applications have been implemented in C/C++ language under a Linux environment. One application is designed to work within a mobile device, and the other one in the network access point. The mobile device basically consists in a notebook equipped with two Wi-Fi interfaces, which is not a common feature in commercial devices, allowing seamless communication transfers aided by the application. Network access points are computers equipped with a Wi-Fi interface and configured to provide Internet wireless access and services of mobility.

In order to test the operation, a test-bed has been implemented. It consists on a pair of access points connected through a network and placed within partially overlapped coverage areas, and a mobile device, all of them properly set. The mobile detects the networks that are compatible and gets attached to the one that provides better conditions for the demanded service. When the service degrades up to certain level, the mobile transfers the communication to the other access point, which offers better service conditions. Finally, in order to check if the changes have been done properly, the duration of the required actions has been measured, as well as the data that can have been lost or buffered meanwhile.

The result is a MIH-alike system working in a proper way. The discovery and selection of a destination network is correct and is done before the old connection gets too degraded, providing seamless mobility. The measured latencies and packet losses are affordable in terms of MIH protocol, but require future work improvements in terms of network protocols that have not been considered under the scope of this work.

Acknowledgements

Quisiera dedicar estas líneas a plasmar una inmensa gratitud hacia todas aquellas personas que indirectamente me han ayudado durante los meses en que me he visto inmerso en la elaboración del PFC. Todos ellos han conseguido, de un modo u otro, hacerme sonreír y sentirme mejor cada vez que han brindado su apoyo incondicional y su ayuda tanto en los buenos como en los no tan buenos momentos.

En estas líneas no puedo dejar de mencionar y agradecer el soporte y la colaboración recibida por parte de la Fundació I2Cat, ya que sin ellos no hubiese sido posible que yo realizase este trabajo.

Del mismo modo, tampoco puedo dejar de mencionar a mi tutor de proyecto, Josep Paradells Aspas, que me ofreció la oportunidad de llevar a cabo este trabajo depositando plena confianza en mí, además de haberme facilitado la tarea y de haberme apoyado tanto a mí como al resto de mis compañeros del Wireless Networks Group.

Finalmente, pero no por ello menos importante, infinitas gracias a mi padre, mi madre y mi hermano no solo por haberme aguantado durante tantos años de dudoso buen humor, sino simplemente por ser los mejores del mundo.

Muchas gracias a todos y cada uno una vez más.

Rubén González Muñoz

Contents

List of Figures

List of Tables

List of Acronyms

3G	Third Generation
3GPP	3G Partnership Project
A	
ACK	Acknowledgement
AICH	Acquisition Indication Channel
AID	Action Identifier
AP	Access Point
API	Application Program Interface
AR	Access Router
ARP	Address Resolution Protocol
ARQ	Automatic Repeat Request
ASN.1	Abstract Syntax Notation 1
B	
BCH	Broadcast Channel
BER	Bit Error Rate
BGC	Boehm Garbage Collector
BMC	Broadcast Multicast Control
BOOTP	Bootstrap Protocol
BPSK	Binary PSK
BS	Base Station
BSS	Basic Service Set
BU	Binding Update
C	
CA	Collision Avoidance
CCK	Complementary Code Keying
CDMA	Code Division Multiple Access
CIR	Carrier to Interference Ratio
CLI	Command Line Interface
CN	Corresponding Node
CoA	Care of Address
CORBA	Common Object Request Broker Architecture
CPICH	Common Pilot Channel
CRC	Cyclic Redundancy Check
CRDA	Central Regulatory Domain Agent
CSMA	Carrier Sense Multiple Access
CS-MGW	Circuit Switched Media Gateway
CTS	Clear To Send
D	
dBm	decibel mili watt
DBPSK	Differential Binary Phase Shift Keying
DCD	Downlink Channel Descriptor
DCF	Distributed Coordinated Function
DCH	Dedicated Channel
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
DO	Downlink Only
DPCCH	Dedicated Physical Control Channel
DPCH	Dedicated Physical Channel
DPDCH	Dedicated Physical Data Channel
DQPSK	Differential Quadrature Phase Shift Keying
DS	Distribution System
DSSS	Direct Sequence Spread Spectrum
DSL	Digital Subscriber Line
DVB	Digital Video Broadcasting
DVB-H	DVB for Handhelds
E	
EAP	Extensible Authentication Protocol
EDGE	Enhanced Data-rates for GSM Evolution
ESS	Extended Service Set
ETSI	European Telecommunications Standards Institute
F	
FBACK	Fast Binding Ack

FBU	Fast BU
FDD	Frequency Division Duplex
FEC	Forward Error Correction
FHSS	Frequency Hope Spread Spectrum
FMIP	Fast Handovers for MIP
FNA	Fast Network Advertisement
FSL	Free Space Loss
FSF	Free Software Foundation

G

GDB	GNU Debugger
GCC	GNU Compiler Collection
GGSN	Gateway GPRS Support Node
GMSC	Gateway MSC
GNOME	GNU Network Object Model Environment
GNU	GNU is Not UNIX (recursive)
GPL	GNU General Public License
GPRS	General Packet Radio System
GSM	Global System for Mobile communications
GTK	GNU Tool Kit GUI
GUI	Graphical User Interface

H

HA	Home Agent
HACK	Handover Ack
HAL	Hardware Abstraction Layer
HI	Handover Initiated
HIP	Host Identity Protocol
HMIPv6	Hierarchal MIPv6
HLR	Home Location Register
HoA	Home of Address
HSDPA	High Speed Downlink Packet Access
HSUPA	High Speed Uplink Packet Access
HSPA	High Speed Packet Access

I

I/O	Input/Output
IDE	Integrated Development Environment
IE	Information Element
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IMT-2000	International Mobile Telecommunications 2000
IOCLT/ioclt	Input/Output Control
IP	Internet Protocol
IPC	Inter Process Communication
IPv4	IP version 4
IPv6	IP version 6
IPSec	Internet Protocol Security
ISM	Industrial Scientific and Medical

J

JDK	Java Development Kit
JVM	Java Virtual Machine

L

L1	Layer 1
L2	Layer 2
L3	Layer 3
LAN	Local Area Network
Libpcap	Library pcap
Libm	Library Mathematics
LAM	Local Mobility Agent
LAN	Local Area Network
LLC	Logical Link Control
LMDS	Local Multipoint Distribution System
LOS	Line Of Sight
LTE	Long Term Evolution

M

MAC	Media Access Control
MADWiFi	Multiband Atheros Driver for Wi-Fi
MAG	Mobile Access Gateway
MAP	Mobile Anchor Point

ME	Mobile Equipment
MIB	Management Information Base
MICS	Media Independent Command Service
MIES	Media Independent Event Service
MIH	Media Independent Handover
MIHF	MIH Function
MIIS	Media Independent Information Service
MIMO	Multiple Input Multiple Output
MIMS	Media Independent Management Service
MIP	Mobile IP
MediaFLO	Media Forward Link Only
MLME	Multi Protocol Label Switching
MN	Mobile Node
MoS	Mobility Services
MPI	Message Parsing Interface
MSC	Mobile Switching Center
MSDU	MAC Service Data Unit
MSGCF	MAC State Generic Convergence Function

N

NAV	Network Allocation Vector
nAR	new AR
nCoA	new CoA
NIC	Network Interface Card

O

oAR	old AR
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OS	Operating System
OSI	Open System Interconnection
OVSF	Orthogonal Variable Spreading Factor

P

PC	Personal Computer
Pcap	Packet Capture
PCF	Point Coordination Function
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Digital Assistant
PDCP	Packet Data Convergence Protocol
PDU	Packet Data Unit
PHY	Physical Layer
PLCP	Physical Layer Convergence Procedure
PMD	Physical Medium Dependant
PMIP	Proxy MIP
PoA	Point of Attachment
PoS	Point of Service
PPP	Point-to Point Protocol
PRACH	Physical Random Access Channel
PSK	Phase Shift Keying
PSTN	Public Switched Telephone Network

Q

QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature PSK

R

RARP	Reverse ARP
RDF	Resource Description Framework
RF	Radio Frequency
RFC	Request For Comments
RLC	Radio Link Control
RMI	Remote Method Invocation
RNC	Radio Network Controller
RP	Reference Point
RSSI	Received Signal Strength Indicator
RTS	Request To Send
RTSolPr	Router Solicitation Proxy
RTT	Round Trip Time

S

SAP	Service Access Point
SCTP	Stream Control Transmission Protocol
SF	Spreading Factor
SFD	Start Frame Delimiter
SGSN	Serving GPRS Support Nodes
SID	Service Identifier
SIP	Session Initiation Protocol
SMIv2	Structure of Management Information Version 2
SNMP	Simple Network Management Protocol
SNR	Signal to Noise Ratio
SOFDM	Scalable OFDM
SPARQL	Protocol and RDF Query Language
SS	Subscriber Station
SS7	Signaling System 7
SSDT	Site Selection Diversity Transmit
SSID	Service Set Identifier
STA	Station (in a WLAN)

T

TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDM	Time Division Multiplex
TDMA	Time Division Multiple Access
T-DMB	Terrestrial Digital Video Broadcasting
TLV	Type, Length and Value
TTI	Transmission Time Interval

U

UCP	User Control Protocol
UE	User Equipment
UIR	Unauthenticated Information Request
UMB	Ultra Mobile Broadband
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
USIM	UMTS Subscriber Identity Module
UTRAN	UMTS Terrestrial Radio Access Network

V

VoIP	Voice over IP
VPN	Virtual Private Network

W

WCDMA	Wideband CDMA
WECA	Wireless Ethernet Compatibility Alliance
WEP	Wired Equivalent Privacy
WEXT	Wireless Extensions
Wi-Fi	Wireless Fidelity
WiMAX	Wireless Interoperability for Microwave Access
WinPcap	Windows Pcap
WLAN	Wireless LAN
WPA	Wireless Protected Access

X

XML	Extensible Mark-up Language

1. Introduction

1.1. Introduction

Nowadays, one emerging feature present in several electronic devices is the increasing number of connectivity interfaces towards the world. Equipments such as cell phones, notebook or laptop computers, ultra mobile Personal Computers (PCs), Personal Digital Assistants (PDAs) or even portable video game consoles carry one wireless network interface controller (NIC) at least. This allows an enhanced user experience permitting a peer to be connected to Internet without wires.

Problems arise when mobility is required for a device. Different wireless technologies such as the cellular phone system provide mechanisms to enable roaming from a coverage area to another without connectivity losses. Nevertheless, when the coverage of a given technology is about to be lost and another technology can provide it, no global solutions are implemented yet in order to maintain the communication. Roaming across heterogeneous access networks is a feature not developed, but required.

In order to solve this problem, the Institute of Electrical and Electronic Engineers (IEEE) 802 committee defines a standard for local and metropolitan area networks called IEEE 802.21 Media Independent Handover (MIH) Services. This standard defines a media access independent entity called MIH Function (MIHF) which allows optimizing handovers between heterogeneous wireless systems. This type of handovers are commonly called inter technology handovers or vertical handovers and are intended for Wireless Local Area Networks (WLAN), Worldwide Interoperability for Microwave Access (WiMAX), Third Generation (3G) Partnership Project (3GPP), 3G

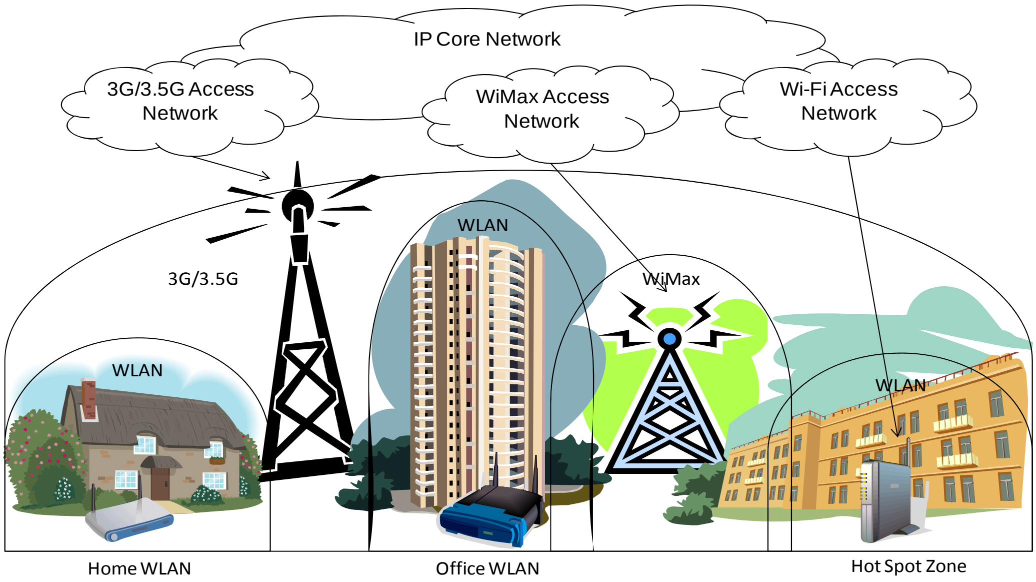


Figure 1.1: Coexisting wireless technologies in a determined area

Partnership Project 2 (3GPP2), other and other IEEE 802 technologies, even Ethernet. There, a set of technical issues related for management is defined in order to provide seamless connectivity when the connection needs to be moved from an interface to another.

Under the framework defined in IEEE 802.21 standard, the Internet Protocol (IP) becomes the common convergence layer for heterogeneous networking. The potential complexity of the communication through different access networks requires a set of services in order to maintain a single or multi path Internet connectivity. These services allow the possibility or capability to gather information related to neighboring cells as well as disposable resources in the networks nearby. The MIHF provides a common interface to abstract technology specific functionalities.

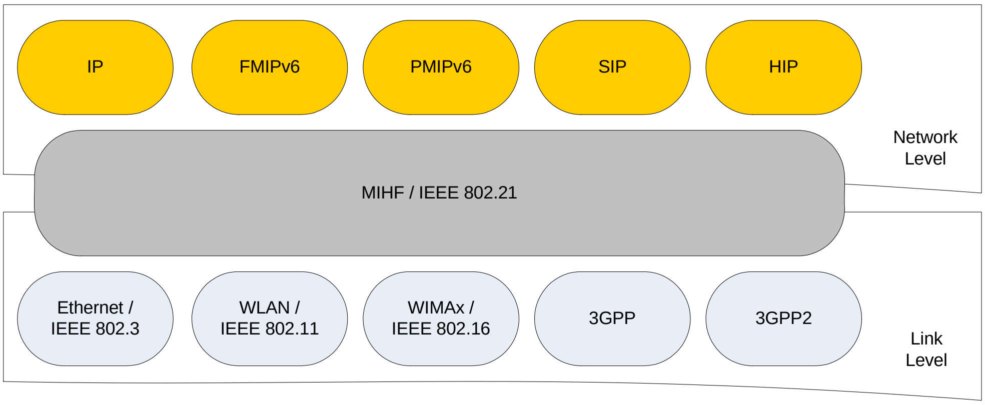


Figure 1.2: Cross-layer design: suitable network level protocols and link level technologies compatible with MIHF

1.2. Motivation

Given the increasing number of devices with one or more network interfaces and applications demanding Internet connectivity everywhere, the development of a platform aiming to provide service continuity is mandatory. A MIHF development provides a cross-layer solution based in elements from layers 2 and 3 (L2 and L3) with a transparent handover mechanism from user's point of view.

1.3. Goal

The goal of the current work is to develop a wireless communication system capable to allow bootstrap and handover decisions based on IEEE 802.21 standard premises. These decisions must rely not only on received signal strength, but also on information shared between different services in order to improve the performance of choice.

Due to the large scope of the standard, the access networks have been bounded just for different WLAN networks. The intention is to compose a scenario capable to reproduce a set of conditions defined under IEEE 802.21 standard in order to evaluate the performance of the system by roaming from an IEEE 802.11 network to another.

1.4. Work done

Different points are needed to be born in mind to reach the proposed goals. First, different applications have been developed in order to support MIHF functionalities. These applications have been implemented in the peers directly implicated in the communication, which are the surrounding access points (APs) and the user's laptop. A MIHF function has been included in both elements with their corresponding features.

These applications need to take elements from the underlying link layer which is in charge of providing network connectivity. Some entities have been developed in order to interface with this lower layer providing a way to communicate with the MIHF. The current development has just considered access to different WLAN networks, bounding the extension and complexity of the premises defined under IEEE 802.21 standard.

No upper layer applications have been developed demanding MIH services. Instead, a MIH user in the mobile peer has been implemented carrying a module capable to make handover and bootstrapping decisions. No L3 mobility protocol has been used. Instead, a classical IP association defined under WLAN standards has been introduced.

Once the scenario has been set out and developed, a set of tests is done in order to determine the advantages of the system respect to those which do not implement IEEE 802.21 functionalities. These proofs are useful to check whether the handover system has succeeded in terms of service continuity. As stated, no user applications have been developed. Instead, some simple programs not directly bounded with the implementation can be used with the same purpose.

1.5. Structure of this report

The documentation of the current memory has been structured in 5 main chapters with a different purpose for each of them.

The **Chapter 1: Introduction** shows a general overview of this memory. Here, a brief introduction to the subject matter is provided, introducing the reader to the current situation of the wireless communications and their evolution and convergence. Moreover, mandatory aspects such as motivation, goals and work done have been introduced too. Finally, there has been also commented the matters which fall out of scope have of the current work.

The **Chapter 2: Background** provides a review of the standards and technologies involved in the current development. Three basic blocks define this chapter: the first one describes the framework of mobility under the IEEE 802.21 standard premises; the second one is a revision of the concepts that involve a technology compatible with the IEEE 802.11 standard of wireless local area networks (WLAN), which is access technology applied; and the third one provides an overview of the Operating System (OS) and the utilities and tools that it supplies used to develop the applications. Moreover, a brief description of other radio access technologies (which do not take part in the current development) is also provided.

The **Chapter 3: Design and Development** is a revision of the work done in terms of implementation. First, it is described the tools used to develop the applications, as well as some reference applications which have inspired some key points of the design. Then, a description of the system operation is supplied, providing a general point of view about how the involved entities are expected to work. The particular scenario that has been implemented is also described next, as well as the issues of the development. Finally, it is explained how the code of the applications has been structured.

The **Chapter 4: Validation and Results** provides a practical overview of the system operation. One part of it is a general installation guide of the applications developed and system configuration, as well as the system requirements. The second part shows an example of operation through different screenshots taken from the running applications. The last part supplies the result of tests applied to the system in order to get some conclusions about the capacities of the development.

The **Chapter 5: Conclusions and Future Work** summarizes and concludes this memory with a review of the current work as well as some discussion about suggested changes in this development or even in current standard specifications. A view of the suitable future work related to the current design is supplied at the end to conclude this report.

1.6. Out of scope

A set of limitations need to be stated in order to define the scope of this work. The IEEE 802.21 standard is a document which does not cover all the aspects that involve the development of a complete MIH framework; nevertheless it provides information to develop handovers between several different wireless technologies which are out of the scope of this work. The only technologies dealt here are those defined for wireless LANs under the IEEE 802.11 standard. Access networks such as Ethernet, WiMAX, 3GPP or 3GPP2 have been left for further developments and are not the aim of the current development.

Although the layers or applications above the MIH Function are not defined within the standards, a user of this layer is needed in order to complete and demonstrate the functionality of this entity. Informative examples propose and illustrate upper layer protocols of mobility such as Fast Handovers for Mobile Internet Protocol version 6 (FMIPv6) or Proxy Mobile Internet Protocol version 6 (PMIPv6). Some papers even comment the compatibility with other protocols such as Session Initiation Protocol (SIP), Host Identity Protocol (HIP) or other variants of Mobile Internet Protocol (MIP). The best option was using FMIPv6, a protocol which is not defined by the time this work has been developed. So, in order to simplify the task and to gather efforts in the development of the rest of the operation, a classical Internet Protocol (IP) connection has been chosen to develop the user of this MIH function. Applications using protocols of the family of MIP are left for further developments and do not fall within the scope of this work.

Finally, the work done does not involve the development of parallel network protocols which help and complement the operation defined in IEEE 802.21 standard. This work is left for further implementations or enhancements of the resulting applications.

2. Background

2.1. IEEE 802.21 Std.: Media Independent Handover Function (MIHF) [1]

The standard IEEE 802.21 provides a set of mechanisms which allows, facilitates and optimizes handover between heterogeneous IEEE 802 networks and other cellular networks with independent media access.

The aim is to improve the user experience of mobile devices by facilitating handover between IEEE 802 networks whether or not they are of different media types, including both wired and wireless and to make it possible for mobile devices to perform seamless handover. These mechanisms are also applicable for handovers between IEEE 802 networks and non IEEE 802 networks.

2.1.1. Introduction

This standard provides link-layer intelligence and other related network information to upper layers to optimize handovers between heterogeneous networks. This includes media types specified by 3GPP, 3GPP2, and both wired and wireless media in the IEEE 802 family of standards.

The purpose of this standard is to enhance the experience of mobile users by facilitating handovers between heterogeneous networks. It is addressed for both mobile and stationary users. For mobile users, handovers can occur when wireless link conditions change due to the users' movement. For the stationary user, handovers become imminent when the surrounding network environment changes, making one network more attractive than another.

This standard supports cooperative use of information available at the Mobile Node (MN) and within the network infrastructure. Both the MN and the network make decisions about connectivity. In general, both the MN and the network points of attachment can be capable of supporting multiple radio standards and simultaneously supporting connections on more than one radio interface.

The overall network can be performed by a mixture of cells of different sizes and types, such as IEEE 802.15, IEEE 802.11, IEEE 802.16, 3GPP, and 3GPP2, with overlapping coverage. The handover process can be initiated by measurement reports and triggers supplied by the link layers on the MN. Those reports an include metrics such as signal quality and transmission error rates. The standard is composed of the following basic elements:

* A framework that enables service continuity while a MN transitions between heterogeneous link-layer technologies. The framework relies on the presence of a mobility management protocol stack within the network elements that support the handover. The framework presents MIH reference models for different link-layer technologies.

* A set of functions within the protocol stacks of the network elements which enable handovers and a new entity in those stacks called Media Independent Handover Function (MIHF).

* The definition of new link-layer Service Access Points (SAPs) and associated primitives for each link-layer technology. The new primitives help the MIHF collect link information and control link behavior during handovers.

* A Media Independent Handover Service Access Point (MIH SAP) and associated primitives are defined to provide MIH users with access to the services of the MIHF.

2.1.2. General architecture

Two basic handover methods are supported under this standard, namely make-before-break, which is also known as a soft handover, and break-before-make, also called hard handover. The first type provides enhanced data transport facilities respect the second one during a handover with packet exchange between the MN and the network.

Handover decision making involves cooperative use of both MN and network infrastructure. Algorithms or patterns to take handover decisions are not within the scope of IEEE 802.21 standard. The following clauses give an overview of certain aspects of the overall handover procedure that fall within the scope of this standard.

2.1.2.1. Service continuity

Service continuity is the continuation of the provided service before and after a handover. Aspects such as data loss and connectivity loss must be optimized. Changes in those parameters should not be noticeable for the user, and also a re-establishment of the service should not be performed by this one. Changes in service quality can appear due to transitions between different networks with different capabilities and characteristics.

2.1.2.2. Quality of Service (QoS)

The Quality of Service (QoS) experienced by an application depends on the accuracy, speed, and availability of the information transfer in the communication channel. The IEEE 802.11 standard provides support to accomplish with the QoS degree required by an application when performing a handover.

The standard considers the QoS in two manners. The first one defines QoS as the one experienced during a handover. The second one considers QoS as a parameter involved in handover decisions. The standard provides related QoS mechanisms in order to facilitate seamless handover, although this alone does not guarantee seamless mobility. Depending on the level of application requirements, seamless mobility is performed optimizing the handover duration as well as the packet loss.

2.1.2.3. Network discovery and selection

The standard defines a set of information related to network discovery and a mechanism to make it available to MIH users. This information can be about link type, link identifier, link availability, link quality, and others.

The network selection mechanism is the process by which an entity in the network chooses a target network normally between more than one available. The criteria to select one or another is not defined within the IEEE 802.21 standard, but is based in features such as QoS, costs of service, user preferences or operator policies. A set of information and mechanisms are provided in this standard to implement network selection.

2.1.2.4. Power management

The services defined in MIHF allow a user to discover the surrounding networks with link-layer technologies within a single radio interface at the MN. By this way, actions like powering up or down multiple interfaces in a MN, as well as active scans are avoided. Other aspects about power saving are not within the scope of this standard, and their implementation rely on the policies of the different link layer technologies.

2.1.2.5. General design principles

This standard is based on the following general design principles:

* MIHF is a logical entity that facilitates handover decision making. MIH users make handover decisions based on inputs from the MIHF.

* MIHF provides abstracted services to higher layers. The service primitives defined by this interface depend on the different technologies in the access networks. The MIHF communicates with the link layers of the protocol stack through media dependent interfaces.

* Higher layer mobility management protocols provide mechanisms for intra-technology handovers. Additionally, different access network technologies have defined handover signaling mechanisms to facilitate them. The signaling methods in horizontal handovers are not within the scope of this standard, except in the case of handovers across different Extended Service Sets (ESSs) in WLANs.

* Support for remote event notification is provided via MIH protocol. Events are just advisory and the actions to take respect them depend on implementation considerations outside of the scope of this standard.

2.1.3. MIHF services

The MIHF is composed of different services that facilitate handovers between heterogeneous access links:

* Media Independent Management Service (MIMS). This service handles the connections between different entities in the network providing capability discovery, registration and event subscription.

* Media Independent Event Service (MIES). It provides event classification, event filtering and event reporting corresponding to changes in link characteristics such as status or quality.

* Media Independent Command Service (MICS). This service is enabled by MIH users and sets the requested link features related to handover and mobility.

* Media Independent Information Service (MIIS). It provides relevant information of L2 and L3 for handover performance about the neighboring access networks, facilitating an effective network access or handover decision.

The MIHF provides asynchronous and synchronous services through SAPs for link layers and MIH users. These services help the MIH users in maintaining service continuity, service adaptation to varying QoS, battery life conservation, network discovery, and link selection. In a system containing heterogeneous network interfaces of IEEE 802 types and cellular, the MIHF helps the MIH users to implement effective procedures to offer services across heterogeneous network interfaces.

2.1.3.1. Service management

Prior to providing the MIH services from one MIHF to another, the MIH entities need to be configured properly. This is done through MIH capability discovery, MIH registration and MIH event subscription, which are MIH functions.

2.1.3.1.1. MIH capability discovery

A MIH capability discovery procedure is used by an MIH user to discover a local or remote MIHF, as well as its capabilities and offered services. MIH capability discovery can be performed through the MIH protocol or through technology specific mechanisms (i.e., IEEE 802.11 Beacon frames).

2.1.3.1.2. MIH registration

MIH registration is used to request access to a specific inter-technology handover framework, and can be used by an MN to declare its presence to a selected MIH Point of Attachment (PoA). MIH registration is mandatory for use with the MICS and to push information to the MIIS.

2.1.3.1.3. MIH event subscription

The MIH event subscription mechanism allows an MIH user to subscribe for a particular set of events that originates from a local or remote MIHF.

2.1.3.1.1. Service management primitives

Table 2.1 defines the set of service management primitives and provides a brief description of their functionality:

Management primitive	Function
MIH_Capability_Discover	Discover the capabilities of a local or remote MIHF.
MIH_Register	Register with a remote MIHF.
MIH_DeRegister	Deregister from a remote MIHF.
MIH_Event_Subscribe	Subscribe for one or more MIH events with a local or remote MIHF.
MIH_Event_Unsubscribe	Unsubscribe for one or more MIH events from a local or remote MIHF.

Table 2.1: Service management primitives

2.1.3.2. Media Independent Event Service (MIES)

Events indicate changes in state and transmission behavior of the physical, data link and logical link layers, or predict state changes of these layers. The recipient of the event can be located within the node that originated the event or within a remote node. The destination of an event is established with a subscription mechanism that enables an MN or network node to subscribe its interest in particular event types.

Handovers can be initiated either by the MN or by the network. Events relevant to handover originate from MAC, PHY, or MIHF at the MN, and at the network PoA. Thus, the source of these events is either local or remote entity. A transport protocol is needed for supporting remote events.

The MIES is divided into two categories, Link Events and MIH Events. Both Link and MIH Events travel from a lower to a higher layer. Link Events are defined as events originated at link layers with destination MIHF. Once in it, Link Events are propagated to MIH users that have subscribed for the specific events. MIH events are defined as events that originated at the MIHF going to MIH users, or link events originated at link layers with destination MIH users propagated through MIHF.

An event can be local or remote. All Link Events are local in nature and propagate from the local lower layer to the local MIHF. MIH Events are local or remote. A remote MIH Event traverses the medium from a remote MIHF to the local MIHF and is then dispatched to local MIH users that have subscribed to this remote event. A Link Event that is received by the MIHF can also be sent to a remote MIH entity as a remote MIH Event.

2.1.3.2.1. Use case

The event service is used to detect the need for handovers. For example, an indication that the link will cease to carry Media Access Control (MAC) Service Data Units (MSDUs) at some point in the near future is used by MIH users to prepare a new PoA ahead of the current PoA ceasing to carry frames. This has the capability to reduce the time needed to proceed with the handover between attachment points.

2.1.3.2.2. Event subscription

Event subscription is a mechanism by which upper layers can receive selected events. It can be divided into link events subscription and MIH events subscription. Link events subscription is performed by the MIHF with the abstracted source link entities to determine which events of interest can produce. MIH events subscription is done between MIH user and MIHF entities to determine which events are useful in MIH users. Table 2.2 provides a brief description of each feasible event and MIH event.

Event /MIH event	Function
Link_Detected /MIH_Link_Detected	Link of a new access network has been detected.
Link_Up /MIH_Link_Up	L2 connection is established and link is available for use.
Link_Down /MIH_Link_Down	L2 connection is broken and link is not available for use.
Link_Parameters_Report/ MIH_Link_Parameters_Report	Link parameters have crossed a specified threshold (often related to signal strength) and need to be reported.
Link_Going_Down /MIH_Link_Going_Down	Link conditions are degrading and link connection loss is imminent.
Link_Handover_Inminent /MIH_Link_Handover_Inminent	L2 handover is imminent based on either the changes in link conditions of additional information available in the network.
Link_Handover_Complete /MIH_Link_Handover_COmplete	L2 handover to a new PoA has been completed.
Link_PDU_Transmit_Status /MIH_Link_PDU_Transmit_Status	Indicate transmission status of a Packet Data Unit (PDU).

Table 2.2: Events and MIH events

2.1.3.3. Media Independent Command Service (MICS)

The MICS enables higher layers to control lower layers. MIH user handles the configuration or selection of an appropriate link through a set of handover commands. Command services are used to determine the status of one or many links in a multi-technology device and to enable MIH users to facilitate optimal handover policies.

The link status varies with time and MN mobility. Information provided by MICS is dynamic information composed of link parameters such as signal strength and link speed; whereas, information provided by MIIS is less dynamic or static in nature and is composed of parameters such as network operators and higher layer service information. MICS and MIIS information could be used in combination by the MN and network to facilitate the handover.

When an MIHF receives a command, it is always expected to execute the command. Commands are invoked by MIH users (MIH Commands), as well as by the MIHF itself (Link Commands). The destination of a command is the MIHF or any lower layer. The recipient of a command is located within the protocol stack that originated the command, or within a remote protocol stack. Table 2.3 provides a summary of the link and MIH commands described under the standard as well as a brief description of its functionalities:

Link Command	Function
Link_Capability_Discover	Discover the list of supported link-layer events and link layer commands.
Link_Event_Subscribe	Subscribe to one or more events from a link.
Link_Event_Unsubscribe	Unsubscribe from a set of link-layer events.
Link_Get_Parameters	Get parameters measured by the active link, such as signal-to-noise ratio (SNR), bit error rate (BER), received signal strength indication (RSSI).
Link_Configure_Thresholds	Configure thresholds for Link Parameters Report event.
Link_Action	Request an action on a link-layer connection.
MIH Command	Function
MIH_Link_Get_Parameters	Get the status of a link.
MIH_Link_Configure_Thresholds	Configure link parameter thresholds.
MIH_Link_Action	Control the behavior of a set of links.
MIH_Net_HO_Candidate_Query	Network initiates handover and sends a list of suggested networks and associated points of attachment.
MIH_MN_HO_Candidate_Query	Command used by MN to query and obtain handover related information about possible candidate networks.
MIH_N2N_HO_Query_Resources	This command is sent by the serving MIHF entity to the target MIHF entity to allow for resource query
MIH_MN_HO_Commit	Command used by MN to notify the serving network of the decided target network information.
MIH_Net_HO_Commit	Command used by the network to notify the MN of the decided target network information.
MIH_N2N_HO_Commit	Command used by a serving network to inform a target network that a MN is about to move towards that network.
MIH_MN_HO_Complete	Notification from MIHF of the MN to the target or source MIHF indicating the status of handover completion.
MIH_N2N_HO_Complete	Notification from either source or target MIHF to the other MIHF indicating the status of the handover completion.

Table 2.3: Link commands and MIH commands

2.1.3.3.1. Use case

The commands generally carry the MIH user decisions to the lower layers on the local or remote entity. The standard provides tools to develop both mobile initiated and network initiated handovers. Handovers are due to changes in the wireless environment leading to the selection of a network that can support an access technology different to the serving network.

During network selection, the MN and the network need to exchange information about available candidate networks and select the best network. The network selection mechanism can select a different network than the attached one, which can require an inter-technology handover. Once a new network has been selected and handover has been initiated, mobility management protocols handle packet routing aspects such as address update and transfer of packet delivery to the new network.

2.1.3.3.2. Handover initiation

This standard supports a set of media independent commands that help with network selection under different conditions. These commands allow both the MN and the network to initiate handovers and exchange information about available networks and negotiate the best available network under different conditions. These commands do not affect packet routing aspects and can be used in conjunction with other mobility management protocols such as MIP and SIP to perform inter-technology handovers.

2.1.3.3.2.1. Mobile initiated handovers

In this case, the MN initiates the handovers. The network selection entity resides on the MN. The MN can use MIH commands to query the list of available candidate networks, reserve any required resources at the candidate target network, and indicate the status of handover operation to the MIHF in the network.

2.1.3.3.2.2. Network initiated handovers

In this case, the network initiates the handovers. The network selection entity resides on the network. The network can use MIH commands to query the list of resources currently being used by the MN, the serving network can reserve any required resources at the candidate target network, and the network can command the MN to perform a handover to a specific network.

2.1.3.4. Media Independent Information Service (MIIS)

The MIIS provides the corresponding mechanisms by which an MIHF entity can discover and obtain network information existing within a geographical area to facilitate the handovers. The neighboring network information discovered and obtained by this framework and mechanisms can also be used in conjunction with user and network operator policies for optimum initial network selection and access.

In certain scenarios information cannot be accessed at L2, or the information available at L2 is not sufficient to make an intelligent handover decision. In such cases information can be accessed via higher layers. Hence this standard enables both L2 and L3 transport options for information access. MIIS typically provides static link-layer parameters such as channel information, the MAC address and security information of a PoA. MIIS also provides the capability for obtaining information about lower layers such as neighbor maps and other link-layer parameters, as well as information about available higher layer services such as Internet connectivity. Information about available higher layer services in a network can also help in more effective handover decision making before the MN actually attaches to any particular network. The objective is to help the higher layer mobility protocol to acquire a global view of the heterogeneous networks to perform seamless handover across these networks.

A MN is freed from the burden of powering up each of its individual radios and establishing network connectivity for the purpose of retrieving heterogeneous network information. MIIS enables this functionality across all available access networks by providing a uniform way to retrieve heterogeneous network information in any geographical area.

2.1.3.4.1. Key motivations for MIIS

The main goal behind the Information Service is to allow MN and network entities to discover information that influences the selection of appropriate networks during handovers. This information is intended to be primarily used by a policy engine entity that can make effective handover decisions based on it. This Information Service provides mostly static information, although network configuration changes are also considered. Other dynamic information about different access networks, such as current available resource levels, state parameters, and dynamic statistics should be obtained directly from the respective access networks. Some of the key motivations behind the Information Service are as follows:

* Provide information about the availability of access networks in a geographical area. This information can be retrieved using any wireless network.

* Provide static link layer information parameters that help the MNs in selecting the appropriate access network.

* Provide information about capabilities of different PoAs in neighbor reports to aid in configuring the radios optimally for connecting to available or selected access networks.

* Provide an indication of higher layer services supported by different access networks and core networks that can aid in making handover decisions. Such information is not available directly from the lower layers of specific access networks, but can be provided as part of the MIIS.

2.1.3.4.2. Information elements

MIIS primarily provides a set of Information Elements (IEs) that are the information structure and its representation, and a query/response mechanism for information exchange. IEs provide information that is essential for a network selector to make intelligent handover decisions. MIIS specifies a common way of representing this information across different technologies by using a standardized format such as Extensible Mark-up Language (XML) or binary Type-Length-Value (TLV) encoding. A structure of information is defined as a schema. MIIS elements are classified into the following three groups:

* General Information and Access Network Specific Information: These IEs give a general overview of the different networks providing coverage within an area.

* PoA Specific Information: These IEs provide information about different PoAs for each of the available access networks. They include PoA addressing information, PoA location, data rates supported, the type of PHY and MAC

layers and any channel parameters to optimize link-layer connectivity. This also includes higher layer services and individual capabilities of different PoAs.

* Other information that is access network specific, service specific, or vendor specific.

For each network supported by an operator there is a list of supported PoAs. For each PoA the PoA IEs are specified.

2.1.3.4.2.1. IE representation and query methods

MIIS defines two methods for representing Information Elements: binary representation and Resource Description Framework representation (RDF). MIIS also defines two query methods. For requests using the binary representation, the TLV query method is used. For requests using the RDF representation, the SPARQL query method is used.

2.1.3.4.2.1.1. Binary TLV query representation

In the binary representation method, Information Elements are represented and encoded in Type-Length-Value form.

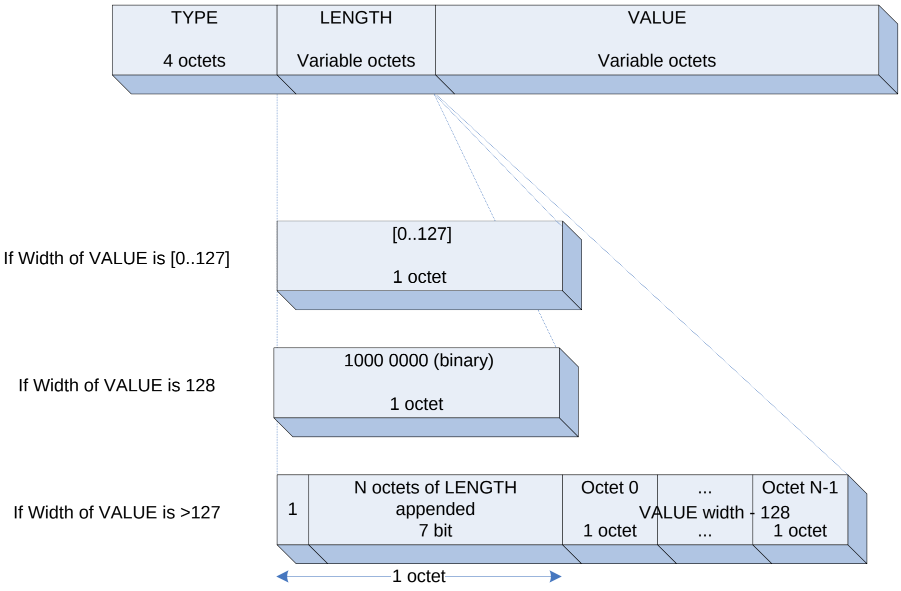


Figure 2.1: TLV encoding for Information Elements

2.1.3.4.2.2. IE containers

In the binary representation method, three IE containers are defined, namely the IE_CONTAINER_POA, the IE_CONTAINER_NETWORK, and the IE_CONTAINER_LIST_OF_NETWORKS:

* IE_CONTAINER_LIST_OF_NETWORK. Contains a list of heterogeneous neighboring access networks for a given geographical location

* IE_CONTAINER_NETWORK—contains all the information depicting an access network

* IE_CONTAINER_POA—contains all the information depicting a PoA and optionally one or more Vendor Specific PoA IEs.

2.1.4. MIHF reference model

The MIHF provides asynchronous and synchronous services through well-defined service access points for MIH users.

2.1.4.1. IEEE 802 architectural considerations

The MIH reference models for different IEEE 802 technologies and the general MIH framework is designed to be consistent with the IEEE 802 architecture for different link layer technologies. The MIH Function is a management entity that obtains link layer information from lower layers of different protocol stacks and also from other remote nodes. The MIH Function coordinates handover decision making with other peer MIH Functions in the network.

The MIH protocol provides the capability for transferring MIH messages between peer MIH Function entities at L2 or at L3. These messages transfer information about different available networks and also provide network switching and handover capability across different networks.

2.1.4.2. General MIHF reference model and Service Access Points (SAPs)

All exchanges between the MIHF and other functional entities occur through service primitives, grouped in Service Access Points (SAPs). The media agnostic general MIH reference model includes the following SAPs:

* MIH_SAP: Is a media independent interface of MIHF with the upper layers of the protocol stack. It always maintains the same name and same set of primitives for every media specific reference model.

* MIH_LINK_SAP: Is an abstract media dependent interface of MIHF with the lower layers of the media specific protocol stacks. It assumes media-specific names and sets of primitives resulting in amendments to media-specific SAPs due to additional functionality being defined for interfacing with the MIHF. All communications of the MIHF with the lower layers of media specific protocol stacks take place through media-specific instantiations of MIH_LINK_SAP.

* MIH_NET_SAP: Is an abstract media dependent interface of MIHF that provides transport services over the data plane on the local node, supporting the exchange of MIH information and messages with the remote MIHF. For all transport services over L2, the MIH_NET_SAP uses the primitives specified by the MIH_LINK_SAP.

The message exchanges between peer MIHF instances, in particular the type of transport they use, are sensitive to several factors, such as the nature of the network nodes that contain the peer MIHF instances (whether or not one of the two is a MN or a PoA), the nature of the access network (whether IEEE 802 or 3G cellular), and the availability of MIH capabilities at the PoA.

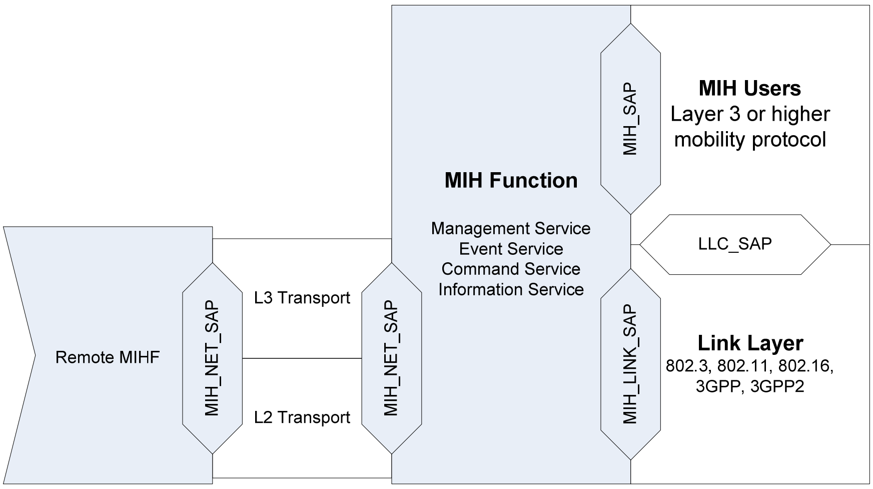


Figure 2.2: IEEE 802.21 stack

When connected to an IEEE 802 network, an MN directly uses L2 for exchanging MIH signaling, as the peer MIHF can be embedded in a PoA. The MN does this for certain IEEE 802 networks even before being authenticated with the network. However, the MN can also use L3 for exchanging MIH signaling, for example in cases where the peer MIHF is not located in the PoA, but deeper in the network. When connected to a 3GPP or 3GPP2 network, an MN uses L3 transport to conduct MIH signaling.

2.1.4.1.3. MIHF reference model for IEEE 802.11

The payload of MIHF services over IEEE 802.11 is carried either in the data frames by using existing primitives defined by the LSAP or by using primitives defined by the MAC State Generic Convergence Function (MSGCF) SAP (MSGCF_SAP). The MSGCF has access to all management primitives and provides services to higher layers.

It should be noted that sending MIHF payload over the Link SAP (LSAP) is allowed only after successful authentication and association of the station to the AP. Moreover, before the station has authenticated and associated with the AP, only MIIS and MIH capability discovery messages can be transported over the MSGCF_SAP. The MIH_SAP specifies the interface of the MIHF with MIH users.

2.1.5. Service Access Points (SAPs)

The MIHF interfaces with other layers using SAPs. Each SAP consists of a set of service primitives that specify the interactions between the service user and provider. These primitives taken together provide a set of services.

2.1.5.1. Media dependent SAPs

Each link-layer technology specifies its own technology-dependent SAPs. For each link-layer technology, the MIH_LINK_SAP maps to the technology-specific SAPs. The MIH Function uses the following media dependent SAPs for interfacing with other entities: MIH_LINK_SAP and MIH_NET_SAP.

2.1.5.1.1. MIH_LINK_SAP

The MIH_LINK_SAP specifies an abstract media dependent interface between the MIHF and lower layers media specific protocol stacks of technologies such as IEEE 802.3, IEEE 802.11, IEEE 802.16, 3GPP, and 3GPP2. For different link-layer technologies, media-specific SAPs provide the functionality of MIH_LINK_SAP. Amendments are suggested to the respective media-specific SAPs to provide all the functionality as described by MIH_LINK_SAP.

2.1.5.1.2. MIH_NET_SAP

MIH_NET_SAP defines the abstract media dependent interface of the MIHF that provides transport services over the data plane on the local node, supporting the exchange of MIH information and messages with remote MIHFs. For L2, this SAP uses the primitives provided by MIH_LINK_SAP.

2.1.5.2. Media independent SAP: MIH_SAP

The MIH_SAP specifies a media independent interface between the MIHF and upper layers of the mobility management protocol stack such as an upper layer mobility protocol or a handover function that might reside at higher layers or a higher layer transport entity as well. The upper layers need to subscribe with the MIHF as users to receive MIHF-generated events and also for link-layer events that originate at layers below the MIHF but are passed on to MIHF users through the MIHF. MIHF users directly send commands to the local MIHF using the service primitives of the MIH_SAP.

2.1.6. Media Independent Handover protocol

The MIH Function entities in MN and network entities communicate with each other using the MIH protocol messages. The MIH protocol defines message formats for exchanging these messages between peer MIH Function entities. These messages are based on the primitives that are part of the MIH Services.

2.1.6.1. Key concepts

2.1.6.1.1. MIH protocol transaction

The media independent handover protocol defines a message exchange between two MIHF entities to support remote MIHF services. An MIH transaction is identified by a sequence of messages with the same Transaction Identifier (Transaction ID) submitted to, or received from, one specific remote MIHF Identifier (MIHF ID).

2.1.6.1.2. Reliability

MIH protocol messages are delivered via media dependent transport. To ensure proper operation, a reliable message delivery service is required. If the media dependent transport is unreliable, then the Acknowledgement Service must be enabled. If the media dependent transport is reliable, there is no need to use the acknowledgement service. A reliable media dependent transport is one that exhibits a message loss rate of less than 0.01%.

2.1.6.1.2.1. MIH protocol acknowledgement service

The acknowledgement service must be used when the MIH transport used for remote communication does not provide reliable services. When the MIH transport is reliable, the use of the acknowledgement service is not needed. The acknowledgement service is particularly useful when the underlying transport used for remote communication does not provide reliable services. When the MIH transport is reliable, the acknowledgement service is optional.

2.1.6.2. Transaction state diagram: state machines

A node that has a new available message to send related to a new transaction is called transaction source and starts a transaction source state machine. In the same manner, a node that receives a message related to a new transaction is called transaction destination node and starts a transaction destination state machine. If the acknowledgement (ACK) feature is being used by the source and/or destination transaction node, the ACK-Requestor and/or ACK-Responder state machine is started. The ACK related state machine is run in parallel to the transaction source/destination state machines.

Each transaction is represented in an MIHF by an instance of the transaction source or destination state machine. Optionally, each transaction can also have one instance of ACK-Requestor or one instance of ACK-Responder state machine, or both.

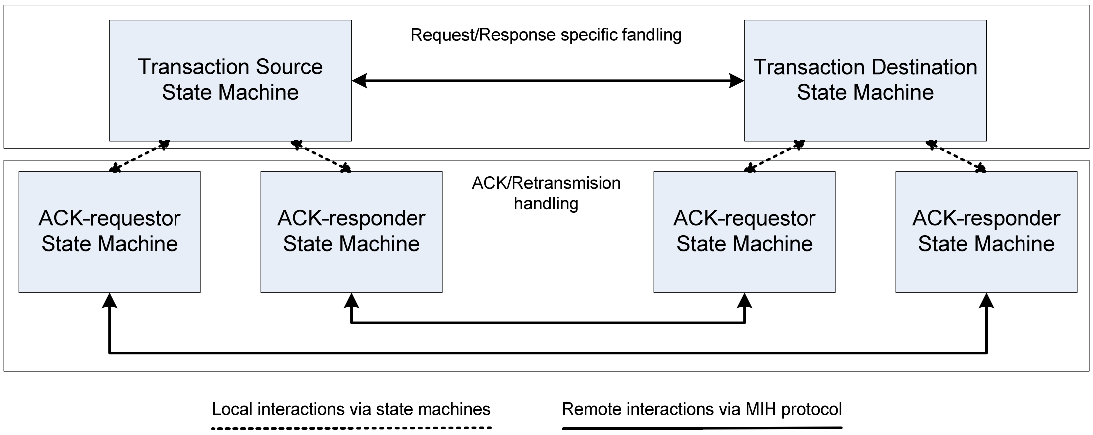

Request/Response specific fandling
Transaction Source State Machine
Transaction Destination State Machine
ACK/Retransmision handling
ACK-requestor State Machine
ACK-responder State Machine
ACK-requestor State Machine
ACK-responder State Machine
Local interactions via state machines
Remote interactions via MIH protocol

Figure 2.3: State machine interactions

2.1.6.2.1. Transaction Timers State Machine (TTSM)

The Transaction Timers State Machine (TTSM) for a given transaction is responsible for decrementing the timer variables for this transaction each second, in response to an external system clock function. The timer variables are used, and set to their initial values, by the operation of the individual state machines for the transaction.

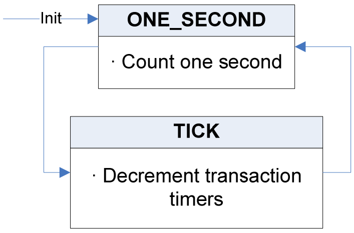


Figure 2.4: Transaction Timer State Machine (TTSM)

2.1.6.2.2. Transaction Source and Destination State Machines (TSSM and TDSM)

A Transaction Source or Destination State Machine (TSSM or TDSM) is started, and related transaction initiated when a message related to a new transaction is available to be sent (source) or is received (destination). The transaction terminates when it has succeeded and any ACK related state machine if started were terminated; or if it fails the transmission in case of a transaction source state machine. An instance of transaction source or destination state machine can cease to exist once the transaction has succeeded or failed.

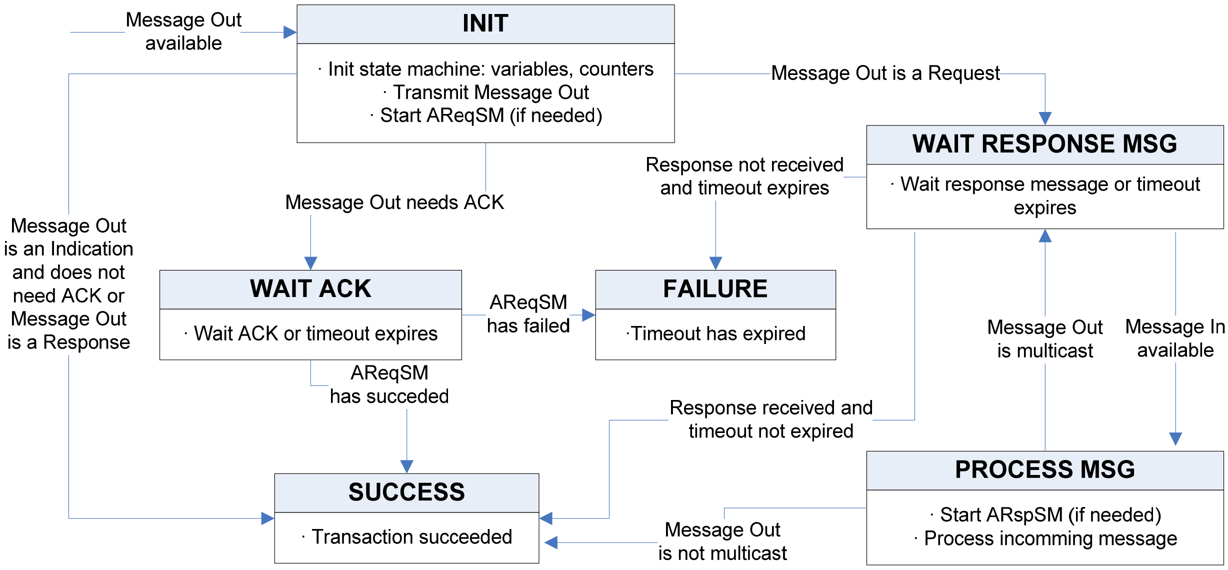

Message Out available
INIT
· Init state machine: variables, counters
· Transmit Message Out
· Start AReqSM (if needed)
Message Out is a Request
WAIT RESPONSE MSG
· Wait response message or timeout expires
Response not received and timeout expires
Message Out needs ACK
Message Out is an Indication and does not need ACK or Message Out is a Response
WAIT ACK
· Wait ACK or timeout expires
AReqSM has failed
FAILURE
·Timeout has expired
Message Out is multicast
Message In available
AReqSM has succeded
Response received and timeout not expired
PROCESS MSG
· Start ARspSM (if needed)
· Process incomming message
SUCCESS
· Transaction succeeded
Message Out is not multicast

Figure 2.5: Transaction Source State Machine (TSSM)

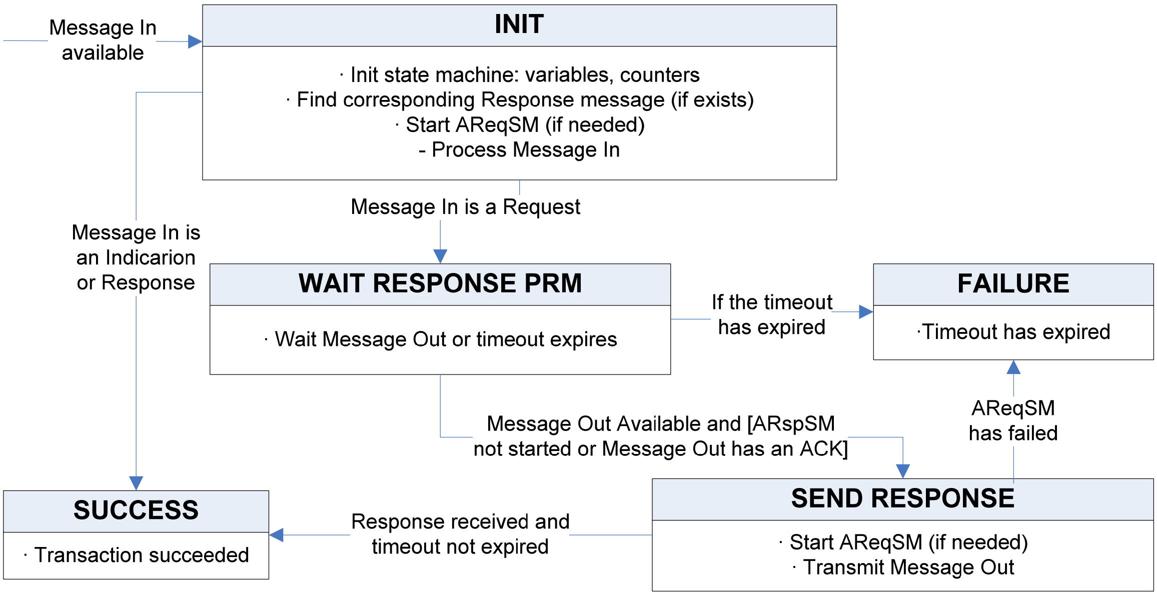

Message In available
INIT
· Init state machine: variables, counters
· Find corresponding Response message (if exists)
· Start AReqSM (if needed)
- Process Message In
Message In is a Request
Message In is an Indicarion or Response
WAIT RESPONSE PRM
· Wait Message Out or timeout expires
If the timeout has expired
FAILURE
·Timeout has expired
AReqSM has failed
Message Out Available and [ARspSM not started or Message Out has an ACK]
SEND RESPONSE
· Start AReqSM (if needed)
· Transmit Message Out
Response received and timeout not expired
SUCCESS
· Transaction succeeded

Figure 2.6: Transaction Destination State Machine (TDSM)

2.1.6.2.3. ACK Requestor State Machine (AReqSM)

The ACK Requestor State Machine (AReqSM) is started when the ACK requestor service is requested in a TSSM or a TDSM. An instance of AReqSM can be brought to the end once the transmission has succeeded or failed or its associated TSSM or TDSM is shut down.

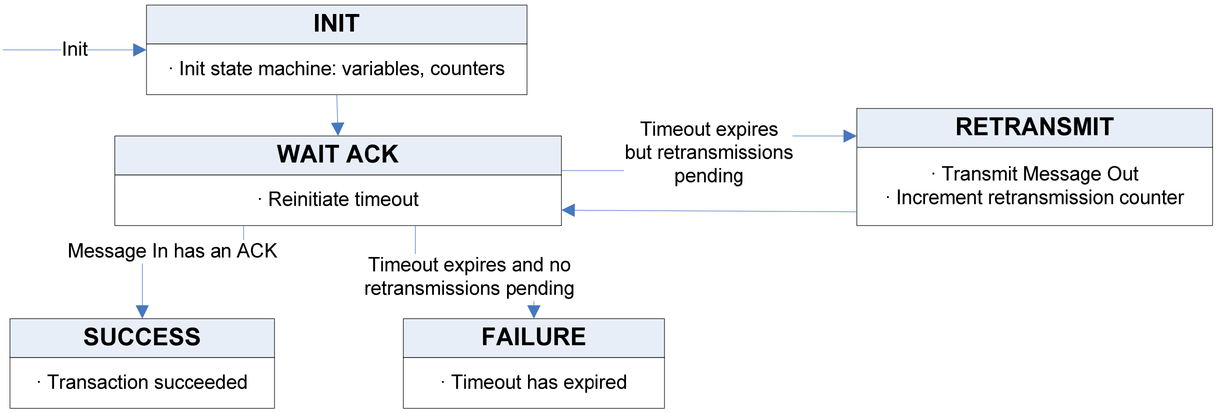

Init
INIT
· Init state machine: variables, counters
WAIT ACK
· Reinitiate timeout
Timeout expires but retransmissions pending
RETRANSMIT
· Transmit Message Out
· Increment retransmission counter
Message In has an ACK
Timeout expires and no retransmissions pending
SUCCESS
· Transaction succeeded
FAILURE
· Timeout has expired

Figure 2.7: ACK Requestor State Machine (AReqSM)

2.1.6.2.4. ACK Responder State Machine (ARspSM)

The ACK Responder State Machine (ARspSM) is started when the ACK responder service is requested in a source or destination transaction state machine. An instance of ARspSM can be brought to the end once its associated TSSM or TDSM is shut down.

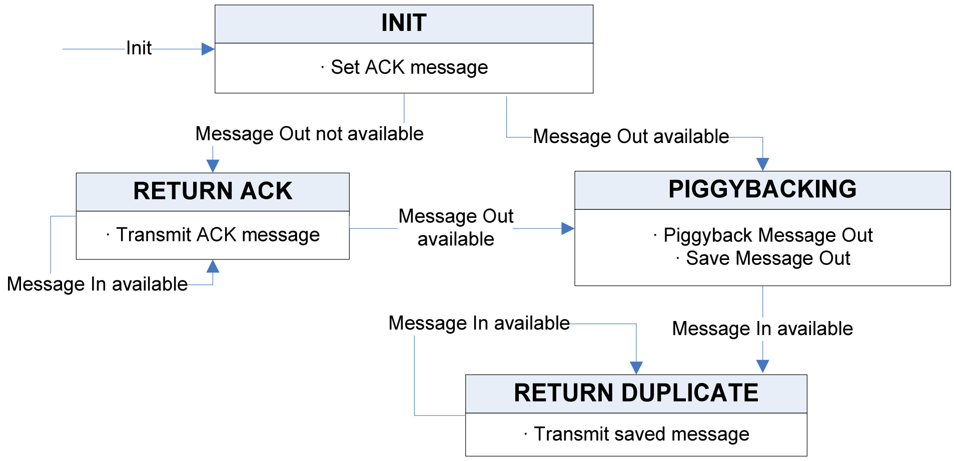


Figure 2.8: ACK Responder State Machine (ARspSM)

2.1.6.3. MIHF discovery

The MIHF discovery refers to the procedure that allows one MIHF to discover its peer MIHFs. MIHF discovery can be done either at L2 or L3. MIHF discovery at L2 is performed either in media-specific manner (e.g., using IEEE 802.11 Beacon frames) or using multicast data frames. MIHF discovery mechanisms at L3 are defined in several Internet Engineering Task Force (IETF) drafts.

2.1.6.3.1. Discovery and capability discovery over data plane

Combined MIH function discovery and capability discovery is performed to discover the MIHF identities, the peer MIHF transport address, and MIHF capabilities at the same time. MIHF Discovery can be implicitly performed using the MIH capability discovery when both MIH nodes are residing in the same multicast domain, or what is the same, where a MIH node's multicast data frame can be delivered using a group MAC addresses.

If MIHF ID and transport address are known MIHF uses MIH_Capability_Discover messages to discover MIHF capabilities only. The following clauses refer to the MIH capability discovery both as a means to discover the MIHF and its capabilities.

2.1.6.3.1.1. Unsolicited MIH capability discovery

An MIHF discovers peer MIHF entities and their capabilities by listening to media-specific broadcast control messages. For example, by listening to a media-specific broadcast message such as a Beacon frame in IEEE 802.11 standard, link layers on an MN can then forward the detected MIH capabilities to its MIHF.

2.1.6.3.1.2. Solicited MIH capability discovery

A MIHF entity discovers its peer MIH functions and capabilities by polling with a MIH_Capability_Discover request message to either its multicast domain or a known MIHF identity, respectively. Only MIH network entities respond to a multicast MIH_Capability_Discover request. When a peer MIH function receives the MIH_Capability_Discover request message, it sends MIH_Capability_Discover response message back to the requestor. The response is sent by using the same transport type over which the request message was received. When the requestor receives the unicast MIH_Capability_Discover response message, it learns the responder's MIHF identifier by checking the source identifier of MIH_Capability_Discover response.

2.1.6.4. MIH protocol frame format

In MIH protocol messages, all TLV definitions are always aligned on an octet boundary and so no padding is needed. An MIH protocol payload carries a Source MIHF Identifier TLV and a Destination MIHF Identifier TLV followed by MIH Service Specific TLVs. The TLV codification is the same as for information elements, but using a one byte length type field.

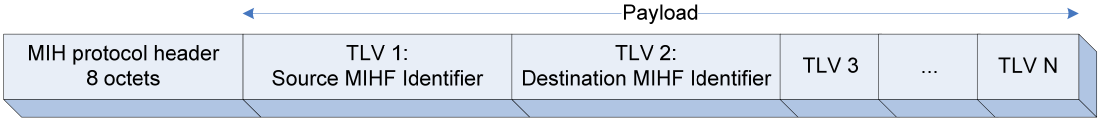


Figure 2.9: MIH Message frame

The MIH protocol header carries the essential information that is present in every frame and is used for parsing and analyzing the MIH protocol frame.

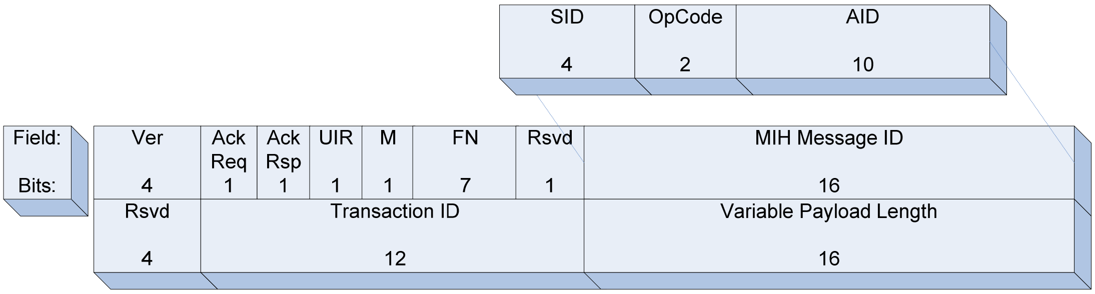


Figure 2.10: MIH Message header

* Version: This field is used to specify the version of MIH protocol used.

* Ack Req: This field is used for requesting an acknowledgement for the message.

* Ack Rsp: This field is used for responding to the request for an acknowledgement for the message.

* Unauthenticated Information Request (UIR): This field is used by the MIIS to indicate if the protocol message is sent in pre-authentication or pre-association state.

* More Fragments (M): This field is used for indicating that the message is a fragment to be followed by another fragment.

* Fragment Number (FN): This field is used for representing the sequence number of a fragment.

* Service Identifier (SID): Identifies the different MIH services.

* Operation Code (OpCode): Type of operation to be performed with respect to the SID.

* Action Identifier (AID): This indicates the action to be taken with regard to the SID.

* Transaction ID: This field is used for matching Request and Response, as well as matching Request, Response and Indication to an ACK.

* Variable payload length: Indicates the total length of the variable payload embedded in this MIH protocol frame.

2.2. IEEE 802.11: Wireless LAN (WLAN) [2]

The term 802.11 refers to a family of protocols which include 802.11, 802.11a, 802.11b, 802.11g, 802.11n and others. IEEE 802.11 is a wireless standard which specifies connectivity for fixed stations, portable and mobile stations into a local area network. The aim of the standard is to provide wireless connectivity to ease implementation and give a quicker implantation of the machines which require this.

The standard is officially designed for specifications MAC and physical (PHY) of a WLAN. It defines the protocols needed to support a wireless networking within a LAN. The main service of this standard is to deliver MSDUs between peer devices on the Logical Link Control (LLC) layer. In general, a radio NIC and one or many APs provide the IEEE 802.11 standard functionalities.

The MAC and PHY features for WLANs are specified in 802.11, 802.11a, 802.11b, 802.11g, 802.11n amongst other standards. The MAC layer of this standard is designed to support additional physical layers depending on the spectrum capacities and the new techniques of modulation.

The purpose of the following section is to provide an overview of the IEEE 802.11 Standard, in order to understand the basic concepts, the principle of operations, and the reasons behind some of the features and components of the standard. A description of the PHY layer and the functional MAC aspects which are defined under the IEEE 802.11 protocol stack, as well as a basic introduction to the Wi-Fi certification is also provided in these clauses.

2.2.1. WLAN media

In a WLAN, the radio waves are used to transmit information between one or more transponders without using a guided physical medium. The transmitted data is introduced into a radio carrier to be extracted identical to the original at the receiver. But reproducing the original signal is not an easy task. The radio channel introduces a distortion that must be fought with proper techniques. The basics about the nature of the radio channel and its modeling will be given in the present sub-section.

2.2.1.1. Propagation of Radio Frequency (RF) waves [3]

In space, RF microwaves travel at light speed forever and in the same direction they were originated unless they interact with some kind of material. Into the atmosphere the microwaves have a very similar speed as in space.

Some RF waves are modified when they travel through transparent materials like the air. By this way, the speed of microwaves at 2,4Ghz and 5GHz is also modified.

The surface of an object is considered smooth in case the size of its irregularities is small compared to the wave length. The waves getting in contact with interposed objects become diffracted around them. If the object is small, the wave will pass around it without being perturbed. However, if the obstacle has a big size it will produce a considerable shadow between itself and a part of the energy of the wave will be reflected. Other part of the energy can be refracted**,** which means it can travel through the material.

2.2.1.1.2 Reflection

The reflection phenomenon is originated when the wave bounces in the general direction it came to an obstacle. A great part of the energy of the waves that get in contact with the object surface is bounded. When a wave travels from a medium to another, a part of its energy is reflected.

The reflexive properties of a place where a WLAN needs to be placed are extremely important and can determine if a WLAN works properly of it fails.

2.2.1.1.3 Diffraction

The dispersion of a wave around an obstacle is called diffraction. The radio waves can get diffracted in small and large scale. They are diffracted in small scale when they are scattered into an indoor environment. An example of diffraction in large scale is the waves getting scattered around a mountain towards an inaccessible place.

2.2.1.1.4 Dispersion

A different effect takes place when the waves collide with small particles. Depending on the frequency of the wave and the particles' composition, it is possible the phenomenon of dispersion. Generally, the dispersion results in redirected wave energy to not desired directions.

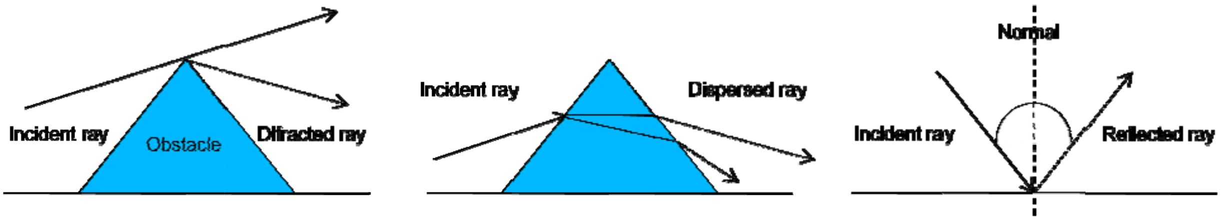


Figure 2.11: Phenomenon: diffraction, dispersion and reflection

2.2.1.2. Phenomena causing channel attenuation

In the radio channel, the mentioned phenomena are combined causing significant modifications on the attenuation of the received signal at the side of the receiver. These alterations are difficult to predict because many environmental factors take part in them. Basically, a radio propagated signal can be affected by path loss and fading.

]

2.2.1.2.1. Path loss

A decisive factor that determines if a communication system is successful is the amount of energy that arrives to the receiver. The electromagnetic waves can be affected in many different ways, including reflection, diffraction and dispersion. Those effects can be combined and treated through path loss calculations. These calculations determine the part of energy lost along a communication path.

A simple model of propagation which does not consider the effects of reflection, diffraction and dispersion is called Free Space Loss (FSL). In this model, every time the distance between transmitter and receptor is doubled, the signal level decreases in 6 dB.

2.2.1.2.2. Slow fading: Shadowing

Slow fading arises when the coherence time of the channel is large relative to the delay constraint of the channel. In this regime, the amplitude and phase change imposed by the channel can be considered roughly constant over the period of use. Slow fading can be caused by events such as shadowing, where a large obstruction such as a hill or large building obscures the main signal path between the transmitter and the receiver. The amplitude change caused by shadowing is often modeled using a log-normal distribution with a standard deviation according to the log-distance path loss model.

2.2.1.2.3. Fast fading: Multipath

When the geometry of the reflected propagation path varies rapidly, as for a mobile radio traveling in an urban area with many highly reflective buildings, a phenomenon called fast fading results. In this regime, the amplitude and phase change imposed by the channel varies considerably among the period of use.

The received signals affected by multipath are characterized by rapid change in signal strength due to phase cancellation and a frequency modulation caused by movements of the transmitter, receiver or the environment (also called Doppler shift). The reception of different instances of the transmitted signal is caused by the multipath propagation delay. The receiver is very sensitive to its environmental obstacles.

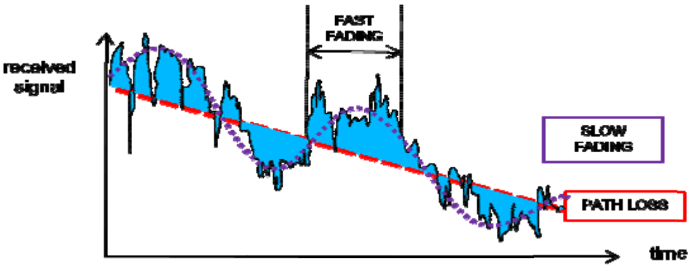


Figure 2.12: Propagation phenomena

2.2.1.3. Propagation models

A radio propagation model is an empirical mathematical formulation for the characterization of radio wave propagation as a function of frequency, distance and other conditions. A single model is usually developed to predict the behavior of propagation for all similar links under similar constraints.

2.2.1.3.1. General propagation models: Rayleigh and Rice

Rayleigh fading model assumes that the magnitude of a signal that has passed through a transmission medium will vary randomly, or fade, according to a Rayleigh distribution. The resulting process is the radial component of the sum of two uncorrelated Gaussian random variables.

Rayleigh fading is a reasonable model for tropospheric and ionospheric signal propagation as well as the effect of heavily built-up urban environments on radio signals. Rayleigh fading is most applicable when there is no dominant propagation along a Line Of Sight (LOS) between the transmitter and receiver.

Rician fading model consists on an anomaly propagation caused by partial cancellation of a radio signal by itself. The signal arrives to the receiver by two different paths (hence exhibiting multi path interference), and at least one of the paths is changing (lengthening or shortening). Rician fading occurs when one of the paths, typically a LOS signal, is much stronger than the others.

2.2.1.3.2. Large scale propagation models.

Large scale models predict a signal behavior averaged over distances much larger than a wave length. The models are function of distance and significant environmental features, and approximately frequency independent. There are two branches which embrace these models: outdoor and indoor environments.

2.2.1.3.2.1 Outdoor environment models: 2-ray ground reflection and diffraction for hilly terrain

The 2-ray ground reflection model consists on an approximation of the signal by two rays. The first one is placed in the LOS and the second one arrives to the receiver reflected by the ground. This reflection causes a phase shift of 180º in the secondary ray which can result in a destructive interference combined with the LOS received signal.

In a hilly terrain, the propagation can be result of the ray diffraction over one or more ridges. If there is no LOS, propagation can be done via diffraction.

2.2.1.3.2.2 Indoor environment models [4]

Indoor environments are more random in terms of how the signal is affected by propagation phenomena. The environment is more dynamic than outdoors and significant features are physically smaller. There is more clutter and scattering, and LOS is not a common feature. It is ordinarily used log-normal shadowing model for no LOS available.

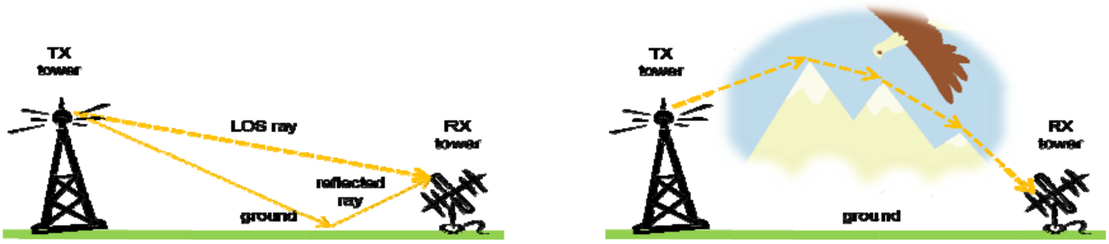


Figure 2.13: Left: 2-ray ground reflection model; Right: diffraction for hilly terrain model

2.2.1.3.3. Small scale propagation models

Small scale fading models describe signal variability on a scale of a wave length. Multipath effects dominate, which can produce phase cancellation. Path attenuation is considered constant whereas frequency and bandwidth are considered dependent.

The difficulty relies on modeling rapid fading, which produces changes in signal over a short distance or length of time. The factors that influence fading are the following: the motion of the receiver, transmission bandwidth of signal, and the multipath propagation.

2.2.2. IEEE 802.11 architecture

An IEEE 802.11 WLAN is a cellular network where each cell is determined by the coverage area of its correspondent base station. In this context, a WLAN cell is called Basic Service Set (BSS) and its base station is known as Access Point (AP). A user whose service is provided by the AP is commonly known as station (STA).

Commonly, installations can be performed by one or several cells. The first case corresponds with a typical home infrastructure, whereas the second distribution is more common in wide spaces such as campus, museums or other type of places where an AP is not enough to provide service to all the coverage area. A set of BSSs interconnected by some kind of backbone is known as Extended Service Set (ESS), and the referred backbone is called Distribution System (DS).

Other configurations such peer to peer allow creating communication boundaries without using AP elements.

2.2.2.1. Wi-Fi certification

In 1999 Nokia and Symbol Technologies created the Wireless Ethernet Compatibility Alliance (WECA), which four years later changed its name by Wi-Fi Alliance. The aim of this association was creating a brand that allowed an easier encouragement of the wireless technologies under the IEEE 802.11 standard and interoperability between different equipments.

In April of 2000 the WECA certifies interoperability between equipments which worked under the IEEE 802.11b under the brand WI-FI. This means that all the equipments under the Wi-Fi certification are compatible, and they can work together without problems independently of the manufacturer.

The IEEE 802.11 Standard was designed to be the wireless equivalent of the PHY and MAC layers of the IEEE 802.3 Standard (Ethernet). This means that the only difference between a Wi-Fi and an Ethernet is the way the data frames or packets are transported over the network. Therefore, these two kinds of network are totally compatible in terms of services.

It is interesting to note that the term Wi-Fi comes from the short of Wireless Fidelity, as well as Hi-FI comes from High Fidelity. Actually, Wi-Fi does not stand for anything. The original name had a tag line which tried to be explanatory, but only drove to more confusion: “The standard for wireless fidelity”. The mistake came because Wi-Fi Alliance does not try to invent Wireless standards, not even to take part in IEEE 802.11. The Wi-Fi Alliance focuses on interoperability certification and branding. It does not invent standards. It does not compete with IEEE. It complements their efforts. Wi-Fi and the Wi-Fi logo were just the inventions of a firm that the WECA hired in order to give a name to their standard that was easy to identify and to remember.

Figure 2.14: Wi-Fi alliance logo

2.2.2.2. Components and topology

2.2.2.2.1. Stations: laptops, PDA's, mobile phones

Laptops and notebook computers, as well as palm tops, PDAs and other small computing devices are becoming more popular every day. The main difference between a desk computer and a laptop is that the components are smaller. There are Personal Computer Memory Card International Association (PCMCIA) slots instead of expansion slots. There is where NICs, wireless NICs and modems (as well as other useful devices) can be inserted. Using this type of wireless NICs suppresses the need for adapters, connectors and cables.

Some of these devices use a wireless integrated NIC whereas others use a NIC based in a PCMCIA card or CompactFlash. They also work in all aver the seven layers defined under the Open System Interconnection (OSI) model.

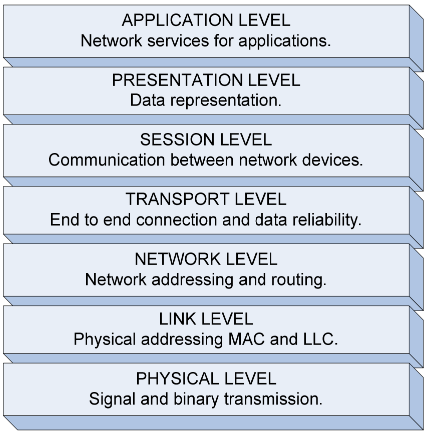

APPLICATION LEVEL
Network services for applications.
PRESENTATION LEVEL
Data representation.
SESSION LEVEL
Communication between network devices.
TRANSPORT LEVEL
End to end connection and data reliability.
NETWORK LEVEL
Network addressing and routing.
LINK LEVEL
Physical addressing MAC and LLC.
PHYSICAL LEVEL
Signal and binary transmission.

Figure 2.15: Open System Interconnection (OSI) stack model

2.2.2.2.2. Adapters

The adapters of a WLAN, also called client adapters or NICs, are basically radio modules. The main function of this wireless NICs is providing a transparent data communication between other devices, wireless or wired. They cover the functionalities of the layers 1 (L1) and 2 of the OSI reference model. The adapters work in a similar way as a standard network adapter. The exception is that the cable has been replaced by a radio connection. No extra wireless networking functionalities are required. The existing applications work properly as well as they worked over a wired network using wireless adapters.

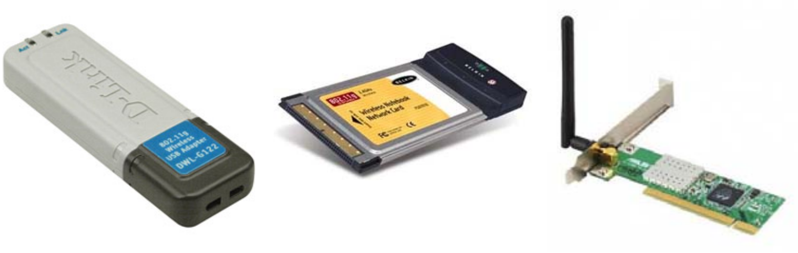


Figure 2.16: Various types of wireless NIC

2.2.2.2.3. Access points and bridges

Access points and a wireless bridges work at L1 and L2 of the OSI reference model. An AP is a WLAN device which can interact as the central point of a wireless network. It can also be used as a connection between wired and wireless networks. On vast installations, the roaming functionality provided by many APs allows wireless users to roam through the placement while keeping a seamless access to the network.

A wireless bridge is a device designed to interconnect two or more networks generally at different buildings. It reaches high data speed and a superior throughput for applications at LOS.

2.2.2.2.4. Antennas

The antennas work on the L1 of the OSI reference model. This layer defines the electrical specifications, as well as mechanicals, procedures, and functionalities to activate, keep and disable the physical link between end systems. Features such as voltage levels, timings, physical data rates, physical connectors and other attributes alike are defined by physical layer specifications.

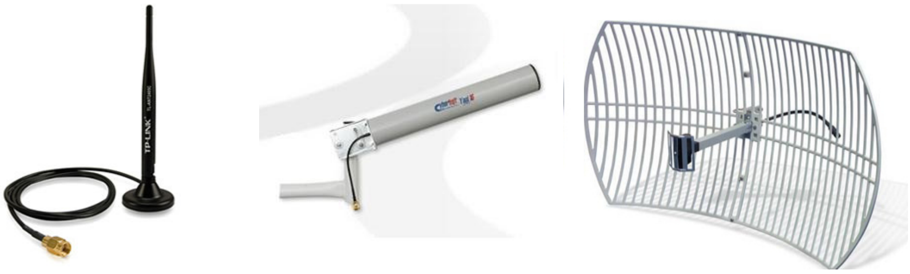


Figure 2.17: Various types of antennas

Some kind of antennas typically used in wireless devices are dipoles, pillar mounts, ground planes, patch mounts, mast mounts, Yagi masts and solid dishes.

2.2.2.2.5 Topology

The wired LANs require the users stay fixed in a placement. WLANs are an extension of the wired LANs and can also act as substitute of a whole system of typical wired networks.

The BSS, also known as micro-cell, is the coverage RF area provided by a single access point. A BSS can be extended by adding another AP to the existing network. When more than a BSS gets engaged to another BSS, it is called ESS. Adding another access point is a way to add wireless devices and to extend the coverage area of an existing wired system.

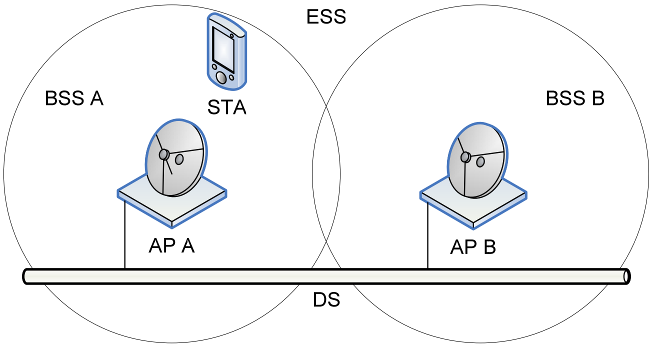


Figure 2.18: BSS, ESS, DS and STA

The AP is connected to the Ethernet backbone, also called DS, and communicates all the wireless devices of the cell area. The AP acts as master on its area, and controls the traffic flow from and to the network. The remote devices are interconnected through the access point.

If a remote cell does not provide enough coverage, it is possible to add any number of cells in order to expand its range. It is needed a certain overlap between adjacent BSS cells to allow remote users do roaming without loss of RF connectivity.

There are basically two types of basic topologies common in WLANs: ad-hoc or peer to peer infrastructure and BSS or ESS infrastructure.

The peer to peer topology is a set of wireless services that can be formed by a pair of MNs (laptops or whatever device with a WLAN NIC). This configuration does not include an AP, so it is called independent BSS (IBSS). This topology is useful to allow many users sharing files. However, the coverage area is an important limitation in this kind of infrastructure because everybody should be able to listen and to be listened.

The BSS/ESS infrastructure has at least one AP and one or more MNs. Each station communicates with the AP. If a station has to communicate with another one that is attached to the same AP, the communication is done through the access point. When the communication has to be done with another station outside the current BSS, but in the same ESS, the station communicates with its attached AP and this one transmits the information to the AP which has the destination station via DS.

Moreover, other topologies such as mesh networks can also be considered. The different nodes of a mesh topology can act as stations as well as gateways or, what is the same, they can be the source, the destination or a routing node in the same network. Many protocols are implemented in order to optimize parameters such as power consumption or the number of hopes in mesh networks.

2.2.2.3. IEEE 802.11 stack

As any 802.x protocol, the 802.11 protocol covers the MAC and PHY layers. The standard defines a single MAC which interacts with three PHYs, all of them running at 1 and 2 Mbps: Frequency Hope Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS) and Infrared.

Beyond the standard functionality usually performed by MAC Layers, the 802.11 MAC performs other functions that are typically related to upper layer protocols, such as fragmentation, packet retransmissions, and acknowledges.

2.2.2.3.1. Logical Link Control (LLC) IEEE 802.2 Layer

Logical Link Control (LLC) is the highest layer on the IEEE 802 reference model. The basic purpose of the LLC layer is to exchange data between ending users through a LAN which provides MAC protocols based on 802. LLC provides a higher layer protocol identification, data link control functions and connection services. It is independent of the technology, the transmission medium and the access to the shared media techniques used in MAC and PHY layers. The higher layers, as well as network layer, supply the user data to the LLC expecting free error data through the network.

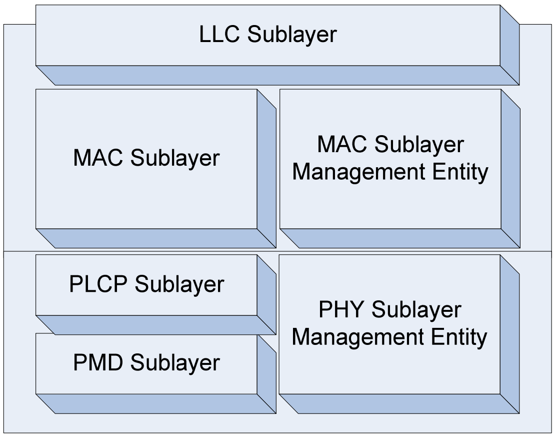

LLC Sublayer
MAC Sublayer
MAC Sublayer Management Entity
PLCP Sublayer
PMD Sublayer
PHY Sublayer Management Entity

Figure 2.19: IEEE 802.11 stack

The LLC layer provides the following connection services: unconfirmed and not connection oriented services, confirmed and connection oriented services, and confirmed and not connection oriented services. These services are applied for the communication between LLC layers.

2.2.2.3.2. Media Access Control (MAC) 802.11 layer

A feature of the standards definition for an inter-operable wireless network is to provide standards of services at MAC and PHY layers. Three services are provided by the MAC IEEE 802.11 layer:

* Asynchronous data services. This service provides the LLC entities the capacity to exchange MSDUs. To allow this service, the local MAC uses the underlying services from the PHY layer. There does not exist any guarantee for a MSDU to be delivered successfully. The broadcast and multicast service is also a part of the data asynchronous service. All stations support this service.

* Security services. The IEEE 802.11 security services are provided by the authentication service and the Wired Equivalent Privacy (WEP) service. The scope of the provided security services is bounded to a message exchange between stations. For the standard porpoises, WEP is seen as a transparent service for the LLC and other layers over MAC layer. The security services were designed to accomplish the following security targets: confidentiality, data integrity and access control.

* MSDUs ordering. The provided services by the MAC also allow the reordering of the received MSDUs. These units will be reordered by the MAC layer just causing a change in the delivery order of broadcast and multicast MSDUs.

2.2.2.3.2.1. MAC 802.11 layer architecture access methods

The MAC Layer defines two different access methods: the Distributed Coordination Function (DCF) and the Point Coordination Function (PCF).

Distributed Coordinated Function (DCF)

This access method is a quarrel mechanism based in Carrier Sense Multiple Access (CSMA) Collision Avoidance (CA). The CSMA protocol works as follows: a station desiring to transmit senses the medium, if the medium is busy then the station will wait for a later time. If the medium is sensed free then the station is allowed to transmit. This mechanism is very effective in mediums where there is not a heavy load of traffic.

Although it is a collision avoidance protocol, some packets could collide because two or more stations decided to transmit at the same time. The MAC layer recognizes these situations and retransmits the collided packets using an exponential random back-off algorithm to implement retransmission timings.

In case many stations could collide when transmitting because they cannot hear each other, the collision avoidance protocol is enhanced in order to book the media before retransmissions. A Network Allocation Vector (NAV) is received at the stations into the BSS to tell them the time the media will be used by a station during a period of time. Short packets called Request To Send (RTS) and Clear to Send (CTS) are used for this issue. After that, collisions can still happen, but the collision probability has been reduced because the RTS packet has shorter length than a data packet. No data packets cause collisions, just signaling packets.

Point Coordinated Function (PCF)

This technique is based in a point coordinator (the AP) which coordinates the communications within the network. The AP has always the priority to access the channel. The access of the stations to the media is centralized and controlled by the point coordinator which will poll every station and permit them transmit any frame. If a station has nothing to transmit, it will transmit null frames. This mechanism has no collisions, but could cause in a low efficiency in case the traffic load was low.

2.2.2.3.3. Physical Layer (PHY)

The MAC layer is just half the total operation of 802.11. The Physical Layer (PHY) standard is the other half. Most of the PHY definitions have three functional entities. Different PHYs are defined as a part of the standard.

2.2.2.3.3.1. Physical Layer Convergence Procedure (PLCP)

The convergence function of the PHY adapts the media dependent system capabilities for the MAC service. PLCP defines a method to map the protocol data units from the MAC layer into a framing format apt to be sent and received between two or more stations.

2.2.2.3.3.2. Physical Medium Dependant system (PMD)

This system defines the features and transmission and reception methods through a wireless medium between two stations, each of them using the same PHY system. It also defines sets of primitives to describe the interface between PLCP and Physical Medium Dependant (PMD) system called SAP_PMD.

2.2.2.3.3.3. Basic PHY standard specifications

PHY DSSS IEEE 802.11b

The IEEE 802.11b is a revision of the original standard introduced in 1999. It improved the transmission rate by introducing a Complementary Key Coding. This revision is compatible with the original IEEE 802.11 standard. The following are some basic features:

Spectrum system: Direct Sequence Spread Spectrum (DSSS).
Band: 2.4 GHz Industrial Scientific and Medical (ISM).
Channels: 11 (three channels not overlapped).
Coding: Barker Coding or Complementary Code Keying (CCK).
Modulation: Differential Binary Phase Shift Keying (DBPSK) or Differential Quadrature PSK (DQPSK).
Max data rate: 11 Mbps.

Table 2.4: IEEE 802.11b PHY features

PHY IEEE 802.11a

Although this physical revision was approved in 1999, the first products did not appear in the market until 2001. The most important achievement of this version was introducing the use of the 5 GHz band which allowed a wider use of this technology, avoiding interferences with other communication systems on the 2.4 GHz band such as Bluetooth devices or even with microwave ovens. Another important feature is the use of Orthogonal Frequency Division Multiplex (OFDM) modulation which increased the data rate until 54 Mbps.

Two important issues are against this technology. The first one is that the attenuation suffered at the 5 GHz band is much higher and the use of this type of devices is restricted to areas with LOS. The other contra is that working in this band makes this physical technology incompatible with the others transmitting at 2.4 GHz.

Band: 5 GHz ISM.
Channels: 12; 8 for the wireless network and 2 for point to point connections.
Modulation: BPSK, QPSK, 16QAM or 64QAM.
Max data rate: 54 Mbps.

Table 2.5: IEEE 802.11a PHY features

PHY IEEE 802.11g

This standard was introduced in June of 2003 and supposed the evolution of the IEEE 802.11b standard. It reached the theoretical max data rate of 54 Mbps (as well as 802.11a) by using the same techniques as 802.11a of coding and spectrum occupation. Although two communications b and g use the same frequency channels, they are both perfectly compatible and can coexist pacifically with a media reservation mechanism.

PHY IEEE 802.11n

This PHY is a standard proposal modification with the aim to improve significantly the maximum transmission rate to reach 600 Mbps in practice (which means that theoretically this rate could be much higher). Nowadays the standard supports physical rates near to 300 Mbps, what is seen as a user data rate of about 100 Mbps.

The basic improvements introduced in this PHY are a Multiple Input Multiple Output (MIMO) antenna system and a wider channel of 40 MHz also known as Channel Bonding. The first one takes profit of the non LOS signals which arrive to the receivers to improve signal recognition via space diversity and the second one permits increasing the amount of data that can be transmitted. The system can also operate at both bands of 2.4 and 5 GHz.

2.2.3. IEEE 802.11 protocol [5]

2.2.3.1. Type of frames

There are three main types of frames:

* Data Frames: used for data transmission.

* Control Frames: used to control access to the medium RTS, CTS, and ACK.

* Management Frames: frames that are transmitted the same way as data frames to exchange management information, but are not forwarded to upper layers.

Each of these types is subdivided into different subtypes as well, according to their specific function.

2.2.3.2. Format of frame

All 802.11 frames are composed by the following components:

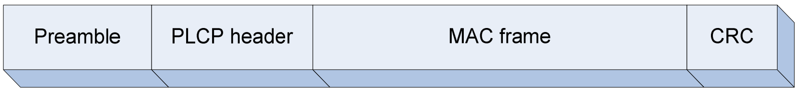


Figure 2.20: PHY 802.11 frame

2.2.3.2.1. Preamble

This is PHY dependent, and includes:

* Synch: A sequence of alternating zeros and ones, which is used by the PHY circuitry to select the appropriate antenna if diversity is used, and to reach steady-state frequency offset correction and synchronization with the received packet timing. The length of this field can be 128 (long preamble) or 56 octets (short preamble) and is generally transmitted at 1 Mbps in both cases.

* SFD: A Start Frame delimiter which consists of the 16-bit binary pattern which is used to define the frame timing.

2.2.3.2.2. PLCP header

The PLCP header is commonly transmitted at 1 (with long preamble) or 2 Mbps (with short preamble). It contains logical information that will be used by the PHY layer to decode the frame. Consists of:

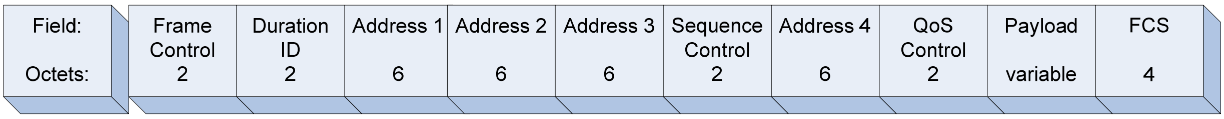
* PLCP_PDU length word: which represents the number of bytes contained in the packet; this is useful for the PHY to correctly detect the end of packet.

* PLCP signaling field: currently contains only the rate information, encoded in 0.5 Mbps increments from 1 Mbit/s to 4.5 Mbit/s.

* Header Error Check Field: a 16 Bit CRC error detection field.

2.2.3.2.3. MAC header

Figure 2.21 shows the general MAC Frame Format, part of the fields are only present on part of the frames as described later.

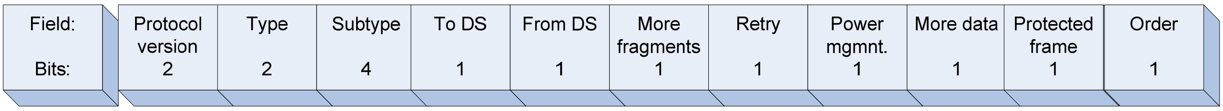


Field:	Frame Control	Duration ID	Address 1	Address 2	Address 3	Sequence Control	Address 4	QoS Control	Payload	FCS
Octets:	2	2	6	6	6	2	6	2	variable	4

Figure 2.21: MAC 802.11 frame.

2.2.3.2.3.1. Frame Control Field

The Frame Control field contains control information used for defining the type of 802.11 MAC frame and providing information necessary for the following fields to understand how to process the MAC frame. A description of each Frame Control subfield is as follows:



Field:	Protocol version	Type	Subtype	To DS	From DS	More fragments	Retry	Power mgmnt.	More data	Protected frame	Order
Bits:	2	2	4	1	1	1	1	1	1	1	1

Figure 2.22: MAC 802.11 header control field

* Protocol Version provides the current version of the 802.11 protocol used. Receiving STAs use this value to determine if the version of the protocol of the received frame is supported.

* Type and Subtype determines the function of the frame. There are three different frame type fields: control, data, and management. There are multiple subtype fields for each frame type. Each subtype determines the specific function to perform for its associated frame type.

* To DS and From DS indicates whether the frame is going to or exiting from the DS (distributed system), and is only used in data type frames of STAs associated with an AP.

* More Fragments indicates whether more fragments of the frame, either data or management type, are to follow.

* Retry indicates whether or not the frame, for either data or management frame types, is being retransmitted.

* Power Management indicates whether the sending STA is in active mode or power-save mode.

* More Data indicates to a STA in power-save mode that the AP has more frames to send. It is also used for APs to indicate that additional broadcast/multicast frames are to follow.

* WEP indicates whether or not encryption and authentication are used in the frame. It can be set for all data frames and management frames, which have the subtype set to authentication.

* Order indicates that all received data frames must be processed in order.

2.2.3.2.3.2. Duration/ID

This field has two meanings depending on the frame type. In Power-Save Poll messages this is the Station ID, and in all other frames this is the duration value used for the NAV Calculation.

2.2.3.2.3.3. Address fields

A frame may contain up to 4 Addresses depending on the ToDS and FromDS bits defined in the Control Field, as follows:

* Address-1 is always the Recipient Address (i.e. the station on the BSS who is the immediate recipient of the packet), if ToDS is set this is the Address of the AP, if ToDS is not set then this is the address of the end-station.

* Address-2 is always the Transmitter Address (i.e. the station who is physically transmitting the packet), if FromDS is set this is the address of the AP, if it is not set then it is the address of the STA.

* Address-3 is in most cases the remaining, missing address, on a frame with FromDS set to 1, then the Address-3 is the original Source Address, if the frame has the ToDS set then Address-3 is the destination address.

* Address-4 is used on the special case where a Wireless Distribution System is used, and the frame is being transmitted from one AP to another, in this case both the ToDS and FromDS bits are set, so both the original destination and the original source addresses are missing.

2.2.3.3. Joining an existing AP

When a station wants to access an existing BSS the station needs to get synchronization information from the AP or from the other stations when in ad-hoc mode. The station can get this information by one of two means: passive scanning or active scanning. In passive scanning the station just waits to receive one or two beacon frames from the AP. In active scanning the station tries to find an AP by transmitting probe request frames, and waiting for probe response from the AP. The two methods are valid, and either one can be chosen according to the power consumption or performance requirements.

2.2.3.3.1. Authentication

A basic security principle is to keep the intruders away. In most of the cases, this means building stronger walls and to establish little well protected gates to provide access to a selected group of people. The security solutions must be integrated seamlessly and must be transparent, flexible and administrable. These strategies work better for wired networks than in WLANs.

Security means assuring the users can do the tasks that they are authorized to do and to get the information they are authorized to obtain. Security also means that the users can not damage the data, the applications or the environment of a system. Protection against malicious attacks is also a concept included into security, as well as the error control and equipment failures.

Originally, security was not a fundamental feature into a WLAN system. The equipments were expensive and difficult to get. The first authentication methods were very simple and practically useless. Many WLANs used the Service Set Identifier (SSID) which was a shared identifier between the AP and an allowed station. This is not considered a valid method and should not be used for authentication. One method not defined under IEEE 802.11 standards used a mapping table into the AP to determine which physical addresses were allowed to be authenticated. Lately, the WEP encryption seemed to be a valid method to protect the communication against malicious attacks using a shared key protocol. This authentication is based on a challenge method. The AP sends a plain text message to the client who has to cipher it and return it to the AP. The AP will compare its ciphered message with the one received back, and if it is the same, the station will be authenticated. However, it is proved that the shared key is not a secure method because it is easy to break. To solve this problem, second generation security methods were developed such as Wireless Protected Access (WPA) also called Simple Security Network.

WPA uses IEEE 802.11i standard to assure WLAN security in long term. WPA allows user identification through a 802.1X standard. So, 802.1X provides mutual identification for both AP and stations. 802.1X can use many variants of its authentication protocol called Extensible Authentication Protocol (EAP) such as LEAP, EAP-TLS, PEAP, PEAP-MD5, EAP-OTP, EAP-SIM, EAP-TTLS and Kerberos.

2.2.3.3.2. Association

When the station is authenticated, then it will start the association process, which is the exchange of information about the stations and BSS capabilities, and which allows a set of APs to know about the current position of the station. Only after the association process is completed, a station is capable of transmitting and receiving data frames.

When a client turns to be on-line, it transmits a broadcasted polling query. An AP which listened this query will response with information about itself such as traffic load hops to the backbone and others. If more than an AP responds, the client will need to choose the AP to associate with it. The access points periodically transmit beacon frames. A beacon has similar information details as the included into probe response frames. The STA listens to every AP and creates a table list with the information of the surrounding listened APs.

Unlike the authentication method, the association has a set of simple and well defined steps:

* The client sends a probe request frame to query information.

* An AP or many APs reply with a probe response frame with the basics of this network and AP.

* The client evaluates received probe responses and selects the best AP. Then it sends an authentication request frame.

* The target AP registers the client and sends an authentication confirmation back to the client.

* The client sends an association request to the selected AP.

* Finally, the selected AP confirms association and registers the client.

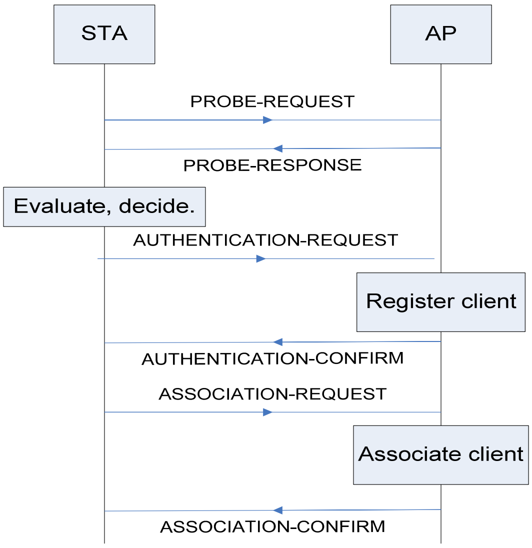


Figure 2.23: Association protocol

2.2.3.3.3. Roaming

Roaming is the process of moving from one BSS to another with or without a disruption in the connection. This feature is similar to the handover function implemented in cell phones when they roam between cells but considering some differences.

First, a WLAN is a system based in packet transactions. The transition from cell to cell may be based in packet transmissions as opposed to a telephony system, where the transition is done while a continuous flow of data from a phone conversation. This could make the roam process from WLAN to WLAN easier. Nevertheless, the voice system is ready to support disruptions on its service that would not affect the conversation whereas a temporary disconnection will reduce significantly the performance on a WLAN because the retransmission of data would be performed by upper protocols.

The IEEE 802.11 standard does not define how should the roaming be performed, but defines the basic tools for that, including scanning, association and re-association where a station which is roaming from one AP to another will become associated with the new one.

2.2.3.3.3.1 Re-association

The received signal strength decreases when a station moves in the opposite direction of the coverage area of the AP. At the same time, the received signal strength coming from an adjacent AP starts to increase. The re-association process is the one given in figure 2.24. The same kind of transference can be done if the load of an AP has been exceeded and another AP has enough resources available.

* The station listens for beacons coming from the APs, evaluates its information and selects the AP destination of the communication.

* The station sends the request for association to the selected AP.

* The destination AP confirms back the association and registers the station.

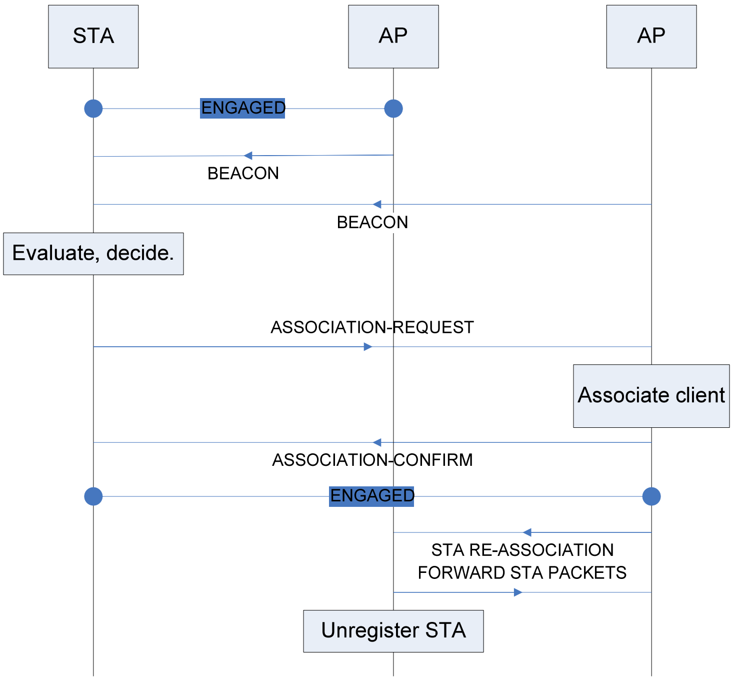


Figure 2.24: Re-association protocol

* The target AP informs to the old AP of a re-association.

* Finally, the old AP forwards the stored packets to the target AP and unregisters the station.

2.3. Technologies included in IEEE 802.21

2.3.1. IEEE 802.16: Worldwide Interoperability for Microwave Access (WiMAX)

2.3.1.1. Basics

WiMAX is the acronym for Worldwide Interoperability for Microwave Access. It is a radio wave transmission technology for last mile local buckles. The protocol used under this technology is IEEE 802.16 and one of its main purposes is supplying broadband services in locations where fiber or cable deployments are costly due to the low density of population (mainly in rural areas). The services and applications that are supported are basically the following:

* Connects IEEE 802.11 APs to the internet

* Provides broadband access as an alternative for xDSL (Digital Subscriber Line) technologies

* Support for open protocols for ending services

* Video, web downloads and streaming

* IP multicast, Voice over IP (VoIP)

* Terminal location

* Connections of Virtual Private Networks (VPN)

* Provides portable connectivity

The IEEE standards, which WiMAX is based in, have their corresponding purpose in European Telecommunications Standards Institute (ETSI) standards. In order to provide interoperability between devices, the organization known as WiMAX Forum defines a set of standards to be accomplished. The resulting rules appear from the basis of a set of IEEE standards designed of type “a”. Table 2.6 summarizes these “a” standards:

IEEE 802.16 Standards of type “a”	Band	ETSI related designation	Band
IEEE 802.16-Single Carrier (SC)	2-66 GHz	ETSI HyperAccess	11-42 GHz
IEEE 802.16a-SC	2-11 GHz		
IEEE 802.16a- OFDM	2-11 GHz	ETSI HyperMAN	2-11GHz
IEEE 802.16a- OFDMA	2-11 GHz		

Table 2.6: IEEE 802.16 standards of type “a”. Bandwidth from 1,25 MHz to 28 MHz

2.3.1.2. Standards of type “a”: IEEE 802.16-Single Carrier (SC)

This standard specifies a technology for wideband wireless Metropolitan Area Networks (MAN). Its intention is to bound hotspots (IEEE 802.11) to Internet, what is known as *last mile xDSL*. The following are some of the most relevant features regarding to the PHY and MAC layers of the type “a” family stack:

* The maximum distance of transmission in LOS is 50 km

* The maximum theoretical transmission rate at BW=28MHz is 134Mbps

* The transmission band is allocated in 2-66 GHz of the radio spectrum

* Uses adaptive modulations: MAC layer supports several PHY layers depending on the frequency, QoS requirements and security

* Considers services based in Time Division Multiplex (TDM) and IP connection oriented. The medium access mechanism is TDM/Time Division Multiple Access (TDMA) with high speed burst modems

* Supports half duplex, Frequency Division Duplex (FDD), Time Division Duplex (TDD)

For fixed stations, the communication is done from point to multipoint. The corresponding network architecture is sketched in figure 2.25 where two new elements can be seen: a Base Station (BS), which distributes the communication coming from the wired network; and the Subscriber Station (SS), which is the receiver of the transmitted information routing it to the corresponding element of the network infrastructure (commonly, an AP).

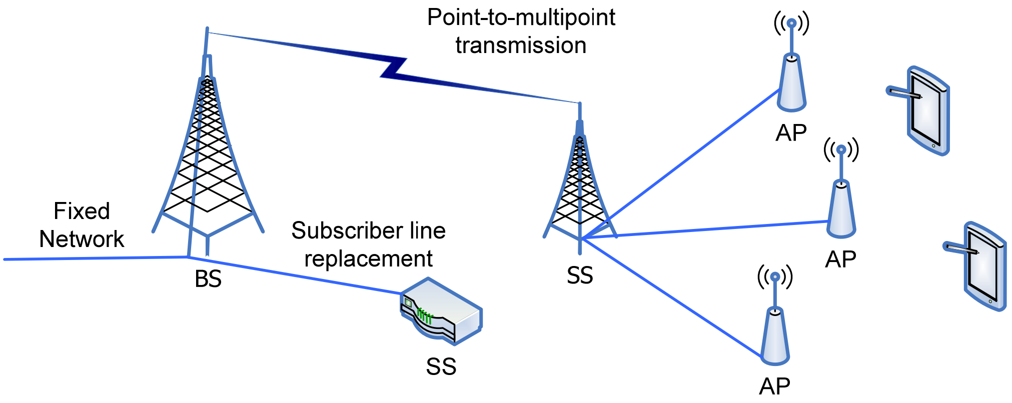


Figure 2.25: Network architecture of WiMAX

2.3.1.3. Standardization for ETSI and IEEE standards: WiMAX Forum

The organism capable to certify the accomplishment of the standards and interoperability between equipments of different manufacturers is the WiMAX Forum. This is a nonprofit organization formed to promote the adoption of WiMAX compatible products and services. The WiMAX Forum also promotes the spread of knowledge about WiMAX. Products which fit with the conformance and interoperability features designed by WiMAX Forum achieve the *WiMAX Forum Certified*. Vendors whose products are not officially certified by WiMAX Forum design them as *WiMAX-ready*, *WiMAX-compilant* or *pre-WiMAX*.

The WiMAX Forum takes up again the IEEE and ETSI standards to propose very accepted versions. They certify products that must fit the conditions imposed by this organization. Two solutions are proposed: for fixed networks, the rule IEEE 802.16-2004 which embraces IEEE 802.16a, IEEE 802.16a and IEEE 802.16d standards; and for mobile or portable networks, the IEEE 802.16e-2005.

2.3.1.3.1. IEEE 802.16-2004: Fixed WiMAX

This standard is based in the last one from IEEE and the one defined by ETSI HiperMAN, both appeared in 2004. A similar technology in the market which would compete with fixed WiMAX is Local Multipoint Distribution System (LMDS), also conceived as a broadband, fixes wireless, point-to-multipoint technology for use in the last mile. The following is a list of some of the most relevant features depicted at the PHY layer:

* Uses OFDM. Can change its location, but cannot transmit correctly in movement. Uses BPSK, QPSK, 16QAM and 64QAM modulations from least signal strength to most, respectively

* Works in the band of 3,5 GHz with a maximum theoretical transmission rate of 70 Mbps LOS (scope of 50 km); and in the band of 5,8 GHz at 20 Mbps in downlink and 8Mbps in uplink NLOS (scope of 4-9 km)

* The bandwidth ranges from 1,5MHz to 28MHz

* The downlink uses TDM method, whereas the uplink uses TDMA method. Both are adaptive depending on the number of stations connected to the same BS.

The fixed WiMAX has a more efficient MAC sub-layer compared to Wi-Fi. The following is a list of some of the most relevant features depicted at the MAC sub-layer:

* Is a connection oriented system, using an Automatic Repeat Request (ARQ) mechanism to perform error control in data transfers

* Implements security through certificates

* Uses power control to minimize interferences and to save energy

* In most cases, applies channel codification (Reed-Solomon or turbo-codes)

2.3.1.3.2. IEEE 802.16e-2005: Mobile WiMAX

There exist plans to develop certification and interoperability profiles for equipments accomplishing the IEEE 802.16e standard, which ease mobility, as well as a complete solution for the network structure which integrates the fixed and mobile accesses. The development of profiles for mobile environments are designed to work in the bands of 2,3 GHz (licensed in EEUU and Europe) and 2,5 GHz. The following is a list which sums up the most relevant features of the mobile WiMAX specification:

* Uses Scalable OFDM Access (SOFDMA), which eases the multiple access

* The maximum theoretical transmission rate is 70Mbps within a scope of 50Km

* Supports QoS for VoIP

* The mobile device can be moving at a speed up to 120 Km/h

Moreover, the mobile WiMAX presents Improvements for NLOS following a similar pattern to the used in mobile telephony including methods such as the use of a Forward Error Correction (FEC) with concatenated adaptive codification and adaptive modulations. Moreover, the RF module is also improved including smart antennas which use diversity in transmission and reception (MIMO).

2.3.1.4. IEEE 802.21 reference model for IEEE 802.16/WiMAX

Figure 2.26 shows the MIHF for IEEE 802.16 based systems.

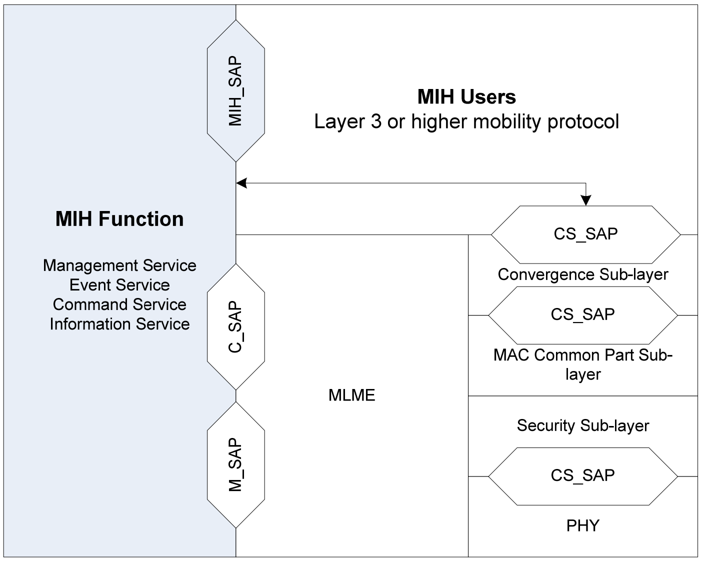


Figure 2.26: IEEE 802.16 reference model

The primitives specified by the Management SAP (M_SAP) are used by an MN to transfer packets to a BS, both before and after it has completed the network entry procedures. M_SAP provides the interface between the MIHF and the IEEE 802.16 management plane functions.

The Control SAP (C_SAP), defined in IEEE 802.16 standard, specifies the interface between the MIHF and control plane. It is used for MIH exchanges between the MIHF and the lower layers of the management plane as part of the IEEE 802.16 instantiation of the MIH_LINK_SAP. M_SAP and C_SAP also transport MIH messages to peer MIHF entities.

The Convergence Sub-layer SAP (CS_SAP) is used to transfer packets from higher layer protocol entities after appropriate connections have been established with the network. The CS_SAP provides the interface between the MIHF and the service-specific Convergence Sub-layer in IEEE 802.16 networks. This SAP is used for the L2 transport of MIH messages across IEEE 802.16 access links.

The MIH_SAP specifies the interface of the MIHF with other higher layer entities such as transport layer, handover policy engine, and L3 mobility protocol. In this model, C_SAP and M_SAP provide link services defined by MIH_LINK_SAP, C_SAP provides services before network entry, while CS_SAP provides services over the data plane after network entry.

2.3.2. Third Group Partnership Project (3GPP and 3GPP2)

The Third Group Partnership Project (3GPP) and 3GPP2 are worldwide collaboration agreements in mobile technology. The aim of 3GPP is to develop the 3G global applications based in the evolution of Global System for Mobile communications (GSM) systems such as International Mobile Telecommunications 2000 (IMT-2000), currently known as Universal Mobile Telecommunications System (UMTS). The 3GPP2 is in charge of developing the specifications based in Interim Standard 95 (IS-95) technologies, commonly known as Code Division Multiple Access 2000 (CDMA2000). The successor of CDMA2000 is Ultra Mobile Broadband (UMB). However, the development of this technology ended in November 2008 favoring Long Term Evolution (LTE) instead (which is the basis of the fourth-generation).

2.3.2.1. Universal Mobile Telecommunications System (UMTS)

UMTS is one of the most developed third-generation mobile technologies. The first development of UMTS is called *Release 99,* specified by 3GPP. The most common form of UMTS is based in Wideband CDMA (WCDMA), although the system also covers other type of access also CDMA based. Table 2.7 depicts some of the most common features of this type of access.

UMTS R99 Common features:
* Chip rate: 3,84 Mbps * Carrier bandwidth: 4,4 MHz to 5 MHz
FDD. Wideband CDMA (WCDMA)
* Medium access method: CDMA * Modulations: DL QPSK - downlink, Dual Code BPSK – uplink * Variable transmission: through spreading factor (SF) and code sequence
TDD. Time Division CDMA (TD-CDMA)
* Medium access method: CDMA with TDMA * Modulations: QPSK – in both downlink and uplink * Variable transmission: through spreading factor (SF), code sequence and multi slot

Table 2.7: Radio interface for UMTS

Although UMTS reuses a part of the infrastructure existing in GSM/GPRS, it requires new base stations and frequency allocations. Almost all existing UMTS devices also support GSM allowing dual mode operation.

2.3.2.1.1. UMTS Network Architecture

The architecture of a UMTS network can be divided in three different blocks: first, there is the User Equipment (UE), which is composed of a Mobile Equipment (ME) and the UMTS Subscriber Identity Module (USIM) card; then, there is the UMTS Terrestrial Radio Access Network (UTRAN), composed by sets of Node B and Radio Network Controllers (RNC); and finally the GSM/UMTS core network, whose elements depend on the state of the *Release*. A brief description of the elements of the UTRAN architecture is provided next, as well as some details on the elements of the core network present in each *Release*.

2.3.2.1.1.1. UTRAN [6]

UTRAN is the designation of a collective of Node B's and RNCs which compose the radio access network of UMTS. This network can carry traffic types such as real-time circuit switched and IP based packet switch. The UTRAN allows connectivity between the UE and the core network.

The Node B supplies the radio interface for UMTS. Provides physical features such as spreading, modulation, bit-rate adaptation and makes the quality measurements that are sent to the RNC. It also performs the combination for soft handover between cells.

The RNC is the entity in charge of resource management, controlling the load in each Node B. It also performs admission control and soft handover through macro-diversity.

2.3.2.1.1.2. GSM/UMTS core network

The 3GPP standards are structured as *Releases*. The following is a brief description of each of the elements which compose the evolution of the UMTS core network in every different *Release*:

Release 99 (R99). This core structure is based in GPRS infrastructure. It is composed of the following elements: Mobile Switching Centers (MSC), which sets up and releases the end-to-end connection, handles mobility and handover requirements during the call and takes care of charging; the Gateway MSC (GMSC) that determines which visited MSC the subscriber who is being called is currently located and interfaces with the Public Switched Telephony Network (PSTN); Serving GPRS Support Nodes (SGSN), which delivers data packets from and to the mobile stations within its geographical service area; a Gateway GSN (GGSN), which is responsible of interworking between the current core network and external packet switched networks such as internet; and finally a Home Location Register (HLR), which is a database that contains details of each authorized subscriber to use this core network.

Release 4 (R4). This release improves the previous one by adding voice gateways and a packet based core via Circuit Switched Media Gateways (CS-MGW) incorporating IP facilities into the core network.

Release 5 (R5). This release introduces IP Multimedia Subsystem (IMS) with SIP and QoS. IMS is a framework intended to deliver services over IP. It provides support for networks such as GPRS, WLANs and CDMA2000 based. Moreover, the system introduces High Speed Downlink Packet Access (HSDPA) enhancement, which will be briefly reviewed in later sub-sections.

Release 6 (R6). The system introduces further WLAN integration, as well as improvements in IMS. Multimedia broadcast and multicast services are included. Moreover, High Speed Uplink Packet Access (HSUPA) enhancements are also introduced. HSUPA will be briefly overviewed in later sub-sections.

Release 7 (R7). This release focuses on decreasing latency, improvements to QoS and real-time applications such as VoIP. This specification also focuses on High Speed Packet Access Evolution (HSPA+) and Enhanced Data-rates for GSM Evolution (EDGE) Evolution.

Releases 8 (R8) and further (R9 and R10). These releases make effort in refactoring of UMTS as an entirely IP based fourth-generation network with LTE.

2.3.2.1.2. Protocol Architecture

The UMTS protocol architecture describes the interfaces and protocols used between them within UTRAN and the core network. The current sub-section provides a brief description of the protocols exiting in the stack of the UMTS protocol architecture.

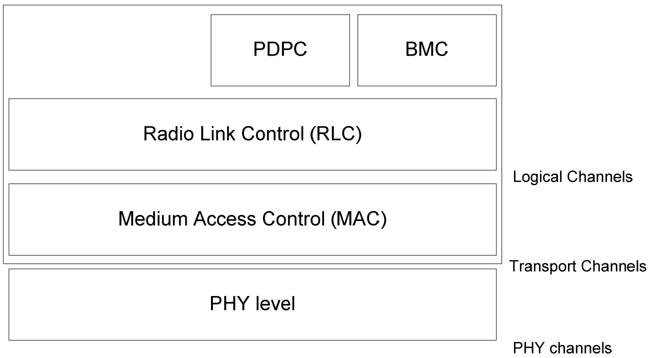


Figure 2.28: UMTS protocol architecture

Broadcast Multicast Control (BMC). It is a protocol for the diffusion of the information in a given cell.

Packet Data Convergence Protocol (PDCP). This protocol encapsulates IP (version 4 or 6) and Point-to Point Protocol (PPP) data to the protocol of lower layer, allowing header compression of transport and network layer protocols.

Radio Link Control (RLC). It offers information transference between the RNC and the UE. Three modes can be used: transparent, which just segments and re-assembles in the arriving order; without confirmation, which improves the transparent mode by introducing an encrypted sequence number; and with confirmation, which improves the previous one by introducing ARQ.

Media Access Control (MAC). This layer has the aim of mapping the logical channels into the transport channels choosing the format of these ones. It also offers QoS mechanisms through a system of priorities and queue ordering. Moreover, it ciphers the transparent data from the RLC layer and performs the multiplex of data. The MAC is divided into three sub-layers: MAC-d, which handles the dedicated channels (DCH); MAC-c/sh, which is in charge of common and shared channels; and MAC-b, which manages the broadcast channel (BCH).

2.3.2.1.3. PHY procedures

A user is allowed to deal with several transport channels, each of them with its corresponding QoS (e.g. a voice call with signaling and more added traffic). Protection mechanisms are enabled such as CRC, interleaving and codification. The CRC is used to detect errors. Data with errors is not discarded because its information can be used in some methods. The interleaving redistributes the errors in reception avoiding bursts of errors. The codification uses convolutional codes or turbo-codes to offer FEC mechanisms.

The segmentation is also another PHY procedure which allows dividing the transport block in several minor blocks to fit it into the physical block.

Finally, the rate adjustment allows fitting the transport bits to the physical channel. If the number of bits has to be increased, some of the information is repeated providing robustness. If the number of bits has to be reduced, the transport block suffers a perforation.

2.3.2.1.3.1. PHY channels

The physical frames last 10 ms and carry 38400 chips per frame, with 15 slots per frame. The signal is spread by direct sequence through a variable SF which ranges from 4 to 256 in the uplink, and from 4 to 512 in the downlink. The channel access is CDMA-DS (Direct Sequence).

The downlink uses Orthogonal Variable Spreading Factor (OVSF) for spreading. They separate users at downlink and can be reused in each cell. A dedicated physical channel (DPCH) carrying data and control is defined in downlink.

The uplink uses Gold sequences for the scrambler in order to separate users. Nevertheless, these sequences cannot be reused in adjacent cells. A dedicated physical control channel (DPCCH) and a dedicated physical data channel (DPDCH) are defined in the uplink. Up to 6 DPDCH with the same SF can be controlled by a DPCCH.

2.3.2.1.3.2. Power control

Power control is the key feature for a good behavior of the CDMA system. WCDMA uses three types of power control: open loop, inner loop and outer loop. The aims are assuring enough quality in the UE and optimizing the power consumption as well as minimizing interferences and battery consumption.

Open loop. This power control monitors the common pilot channel (CPICH) of the cell and sends through the broadcast channel the power which the CPICH is transmitted, the level of interference with the adjacent channel and a constant. From these values and the measure of the received power in the CPICH, the transmitted power is calculated. This system can produce errors with the difference of frequencies due to the FDD.

Inner loop. This mechanism is based in the relationship between useful received signal and the defined interference, also called Carrier to Interference Rate (CIR). The transmitted power is varied to reach a target CIR. This procedure transmits 1500 instructions per second.

Outer loop. It provides the mechanism of CIR target adjustment. This adjustment depends on the Bit Error Rate (BER). This method has the advantage that it does not depend on the frequency.

2.3.2.1.3.3. Macro-diversity

In the uplink, different Nodes B receive the signal from a UE. The Radio Network Controller receives information from these three sources and decides which the best one is. The RNC uses selection combination to choose the best signal.

In the downlink, just a Node B transmits to the UE. Selection in transmission is used to avoid interferences. The UE is the device which chooses the cell which will transmit the data signaling it through a Site Selection Diversity Transmit (SSDT).

2.3.2.1.3.4. Random Access

The access to the medium is done through quarrel method using slotted Aloha with 16 different codes and power adjustment (power ramping). The attempts are done through a Physical Random Access Channel (PRACH) and the confirmations are received through the Acquisition Indication Channel (AICH).

2.3.2.2. High Speed Packet Access (HSPA)

HSPA is the collection of HSDPA and HSUPA protocols which improve the performance in downlinks and uplinks of a WCDMA based system (UMTS). HSPA provides a better performance by using improved modulation schemes and protocols by which ME and UTRAN communicate, increasing bitrates by improving the utilization of the assigned radio bandwidth. The following is a summary of some of the features which HSPA introduces respect to UMTS:

* Physical scheduling (Node B level), which prioritizes users with better conditions. This improves the efficiency in the channel use and so the MEs do not need to receive the same power level.

* Physical retransmissions (Node B level). These ones are faster due to the fact that they do not have to go to the RNC. This would involve crossing a part of the access network.

* Fixed SF equal to 16, suppressing the associated signaling

* Adaptive modulation and codification

* Modulation 16QAM is introduced, providing higher bit-rates

* Shared channel transmission, leading to a more efficient use of WCDMA resources

* Shorter Transmission Time Interval (TTI), reducing the Round Trip Time (RTT) and optimizing tracking on channel variations

2.3.2.2.1. Adaptive modulation and codification

HSDPA introduces an additional modulation which is 16QAM, capable to transmit 4 bits per symbol.

The DCH of UMTS used convolutional coding of its frames and turbo-codes. The protection level is established though the error rate and the degree of perforation. In HSDPA this protection is handled via modulation, codification and perforation.

The changes on the adaptive codification and modulations are done with the TTI cadency. In UMTS, this period is variable (10, 20, 40 and 80 ms) whereas in HSDPA this is a fixed interval of 2ms. The times used for UMTS (R99) are larger than enough.

2.3.2.2.2. Physical level retransmissions

HSDPA increases its performance by retransmitting data between ME and Node B. The system gains in rapidity due to the proximity of the Node B.

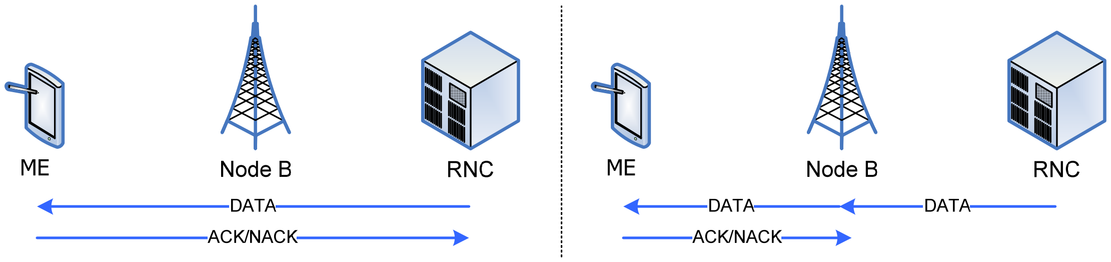


Figure 2.29: Left: UMTS RNC retransmissions; Right: HSDPA Node B retransmissions

HSDPA uses a hybrid system of ARQ. When a frame is received with errors, the information included in it is not discarded. When the same retransmitted frame is received, the data is pondered by the quality of the link. Through the mechanism of combination between different retransmissions, it is feasible to obtain a correct transmission reducing the number of retransmissions.

2.3.2.2.3. Physical level scheduling

In HSDPA, the scheduling is performed by the Node B. If the target is to maximize the use of the shared channel using the received information in the Node B, this one would assign the shared channel to the ME which had a better quality. HSDPA requests performing a quick response to assign the resources depending on the instantaneous information. The following are the scheduling schemes followed:

* Maximize CIR. The channel is assigned to the ME which has a better CIR. Provides maximum bit-rate, but is an unfair procedure. This method is indicated for burst transmissions.

* Round Robin. The time of channel use is distributed equally for each ME. Although it is fair regarding to the distribution of time, it leads to inefficiencies when MEs that do not need great transmission requirements are wasting resources whereas other MEs require more of them. This method is indicated when different continuous traffic needs to be transmitted.

* Proportional and fair. This is a solution half way of the two previous. It assigns the channel to the ME which has the maximum coefficient of dividing the instantaneous transmission rate by the average transmission rate.

The system also supports QoS, distinguishing between services and features, so it can grant different treatments depending on the priority.

2.3.2.2.4. Power control

The serving cell sends the absolute changes: it has the control of the cell in macro-diversity, but the power control on the mobile also uses information from the surrounding cells.

2.3.2.2.5. Resulting transmission rates

The enhanced downlink provides up to 14Mbps with significantly reduced latency. The channel reduces the cost per bit and enhances support for high performance packet data applications. Majority of deployments provide up to 7.2Mbps in the down-link.

The enhanced uplink adds a new transport channel to WCDMA. An enhanced uplink creates opportunities for a number of new applications including VoIP, uploading pictures and sending large e-mails increasing the capacity and the data rate to 5.8 Mbit/s, and also reducing the latency.

2.3.2.3. IEEE 802.21 reference model for 3GPP/3GPP2 technologies

Figure 2.30 illustrates the interaction between the MIHF and a 3GPP/3GPP2-based system. The MIHF services are specified by the MIH_3GLINK_SAP. However, no new primitives or protocols need to be defined in the 3GPP/3GPP2 specification for accessing these services. The MIHF services are mapped to existing 3GPP signaling. For 3GPP2, a mapping between IEEE 802.21 link-layer primitives and 3GPP2 primitives as defined in IETF RFC 1661 and 3GPP2 C.S0004-D is already established.

It is noteworthy that there will be no direct communication between the 3GPP2 PHY and MAC layers with the MIHF. The architectural placement of any MIHF is left to 3GPP/3GPP2 standards.

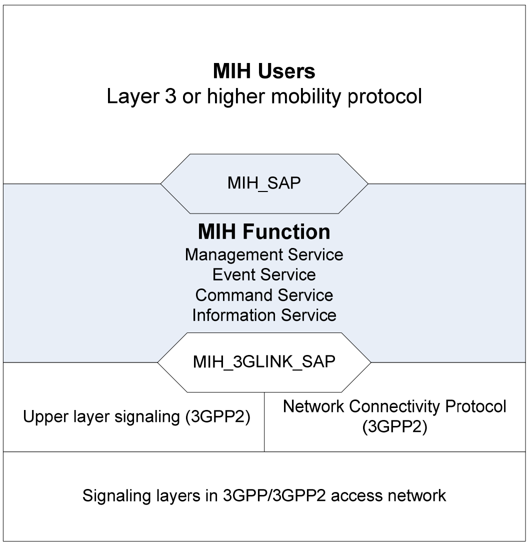

MIH Users
Layer 3 or higher mobility protocol
MIH_SAP
MIH Function
Management Service
Event Service
Command Service
Information Service
MIH_3GLINK_SAP
Upper layer signaling (3GPP2)
Network Connectivity Protocol (3GPP2)
Signaling layers in 3GPP/3GPP2 access network

Figure 2.30: 3GPP/3GPP2 reference model

Primitive information available from Upper Layer Signaling and PPP can be directly used by mapping LAC SAP and PPP SAP primitives to IEEE 802.21 service primitives in order to generate an event. For example, events received from the Upper Layer Signaling through the LAC layer SAP or at the PPP SAP can be mapped and generated through the MIH_3GLINK_SAP.

2.3.2.3.1. Particular SAP: MIH_3GLINK_SAP

This SAP works as an umbrella that defines the interface between the MIHF and the different protocol elements of the cellular systems. The existing service primitives or media-specific SAPs as defined in 3GPP and 3GPP2 specifications are directly mapped to MIHF services, and hence no new primitives need to be defined in these specifications.

2.4. Operating System (OS) background

An Operating System (OS) is a set of computation programs responsible of the resource management of a host such as a personal computer or a mobile phone. The OS works as an interface between the different applications running on a host. The main task in charge is to handle the hardware operation details leaving freed from this charge the applications. The programming process becomes easier by this way. The majority of the hosts, from personal computers to mobile phones, have some type of OS.

The OS offers a set of services to application programs and users. The applications can get access to those services through different Application Program Interfaces (API) or through system calls. Moreover, in mobile and desktop systems a Graphical User Interface (GUI) uses to be part of the operating system, whereas in larger systems it is implemented apart.

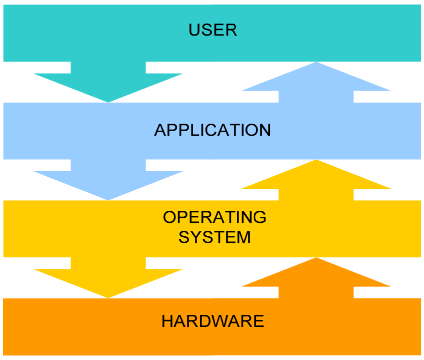


Figure 2.31: Abstraction user-host

Some of the most nowadays common operating systems are Microsoft Windows, MAC OS X and GNU/Linux.

2.4.1. UNIX/Linux OS environment

2.4.1.1. UNIX

UNIX is a computer OS originally developed in 1969 by a group of AT&T employees at Bell Labs. Today the term UNIX is used to describe any OS that conforms to UNIX standards, meaning the core OS operates the same as the original UNIX OS. Today's UNIX systems are split into various branches, developed over time by AT&T as well as various commercial vendors and non-profit organizations.

Only systems fully compliant with and certified according to the Single UNIX Specification are qualified to use the trademark; others are called *UNIX system-like* or *Unix-like*. Today Unix-like OSs such as Linux and Berkley Software Distribution (BSD) are commonly encountered. The term *traditional UNIX* may be used to describe a UNIX or an OS that has the characteristics of either Version 7 Unix or UNIX System V.

In the 1970s was created the project *Unics*, term that stood for UNiplexed Information and Computing Service, a play on Multics (Multiplexed Information and Computing Service). The Uni and Multi prefixes refer to the number of simultaneous users the OS can support, so when Unics could at last support many simultaneous users, it was renamed UNIX.

2.4.1.1.1. Unix Basics

UNIX was designed to be portable, multi-tasking and multi-user in a time-sharing configuration. UNIX systems are characterized by various concepts: the use of plain text for storing data; a hierarchical file system; treating devices and certain types of inter-process communication (IPC) as files; and the use of a large number of software tools, small programs that can be coordinated through a Command Line Interface (CLI) using pipes, as opposed to using a single monolithic program that includes all of the same functionality. These concepts are known as the UNIX philosophy.

The UNIX OS consists of many utilities along with the master control program, the kernel. The kernel provides services to start and stop programs, handles the file system and other common low level tasks that most programs share and schedules access to hardware to avoid conflicts if two programs try to access the same resource or device simultaneously. To mediate such access, the kernel was given special rights on the system, leading to the division between user-space and kernel-space.

In 1973, UNIX was rewritten in the C programming language, contrary to the general notion that something as complex as an operating system, which must deal with time-critical events, had to be written exclusively in assembly language. The migration from assembly language to the higher-level language C resulted in much more portable software, requiring only a relatively small amount of machine-dependent code to be replaced when porting UNIX to other computing platforms.

Both UNIX and the C programming language were developed by AT&T and distributed to government and academic institutions, which led to both being ported to a wider variety of machine families than any other operating system. As a result, UNIX became an open system.

2.4.1.1.2. Free Unix-like operating systems

In 1983 started the GNU project, an ambitious effort to create a free software Unix-like system; *free* in that everyone who received a copy would be free to use, study, modify, and redistribute it. The GNU project's own kernel development project, GNU Hurd, had not produced a working kernel, but in 1992 Linus Torvalds released the Linux kernel as free software under the GNU General Public License (GPL). In addition to their use in the GNU/Linux operating system, many GNU packages have gone on to play central roles in other free UNIX systems as well.

Linux and BSD are now rapidly occupying much of the market traditionally occupied by proprietary UNIX operating systems, as well as expanding into new markets such as the consumer desktop and mobile and embedded devices. Due to the modularity of the UNIX design, sharing pieces is relatively common; consequently, most or all UNIX and Unix-like systems include at least some BSD code, and modern systems also usually include some GNU utilities in their distributions.

2.4.1.2. Linux

Linux is a generic term referring to Unix-like computer OSs based on the Linux kernel. Their development is one of the most prominent examples of free and open source software collaboration typically all the underlying source code can be used, freely modified, and redistributed, both commercially and non-commercially, by anyone under licenses such as the GNU GPL.

A Linux-based system is a modular Unix-like operating system. It derives much of its basic design from principles established in UNIX during the 1970s and 1980s. Such a system uses a monolithic kernel, the Linux kernel, which handles process control, networking, and peripheral and file system access. Device drivers are integrated directly with the kernel.

Separate projects that interface with the kernel provide much of the system's higher-level functionality. The GNU user space is an important part of most Linux-based systems, providing the most common implementation of the C library, a popular shell, and many of the common UNIX tools which carry out many basic OS tasks. The GUI used by most Linux systems is based on the X Window System.

The primary difference between Linux and many other operating systems is that the Linux kernel and other components are free and open source software. Linux is not the only such operating system, although it is by far the most widely used.

Free software projects are often produced independently of each other. The fact that the software licenses explicitly permit redistribution provides a basis for larger scale projects that collect the software produced by independent projects and make it available all at once in the form of a Linux distribution.

2.4.1.3. System calls in UNIX/Linux

A system call is the mechanism used by an application program to request service from the OS based on the kernel or to system servers on operating systems based on the kernel structure. It is a request made by any program to the OS for performing tasks which the program does not have required permissions to execute in its own flow of execution.

System calls provide the interface between a process and the operating system. Most operations interacting with the system require permissions not available to a user level process, e.g. input/output (I/O) performed with a device present on the system or any form of communication with other processes. These operations require the use of system calls.

The OS executes at the highest level of privilege and allows the applications to request services via system calls, which are often implemented through interrupts. If allowed, the system enters a higher privilege level, executes a specific set of instructions which the interrupting program has no direct control over, then returns control to the former flow of execution.

2.4.1.3.1. Implementations over AMD and Intel processors

Implementing system calls requires a control transfer which involves some sort of architecture-specific feature. A typical way to implement this is to use a software interrupt or trap. Interrupts transfer control to the OS so software simply needs to set up some register with the system call number they want and execute the software interrupt.

One example is SYSCALL/SYSENTER (AMD), SYSRET/SYSEXIT (Intel). These are *fast* control transfer instructions that are designed to quickly transfer control to the OS for a system call without the overhead of an interrupt. Linux 2.5 began using this on the x86, where available; formerly it used the INT instruction, where the system call number was placed in the EAX register before interrupt 0x80 was executed.

2.4.1.3.2. Common system calls on UNIX/Linux

Generally, systems provide a library that sits between normal programs and the operating system, usually an implementation of the C library (libc), such as glibc. This library exists between the OS and the application, and increases portability.

On UNIX, Unix-like and other POSIX-compatible OSs, popular system calls are *open*, *read*, *write*, *close*, *wait*, *exec*, *fork*, *exit*, *pipe* and *kill*. Many of today's operating systems have hundreds of system calls. For example, Linux has 319 different system calls. The following are the basic system calls in a Linux OS:

* System calls for process management.

fork: creates a new process which will be the son of the process who has made the calling. It establishes a father-son relationship in a process hierarchy.

exit: ends the execution of the process who has invoked it. It frees the associated memory of this one: code, data and stack data.

wait: a parent process waits for one or more of its son process to end.

exec: executes another program.

* System calls for file and virtual devices management.

open: the OS opens or creates a file or device.

close: the OS takes the required actions to close a file or device indicating that the task with it has ended.

read: read from a file or a virtual device.

write: write to a file or to a virtual device.

* System calls for communications between processes.

pipe: creates an ordinary pipe to implement to allow a communication between many processes.

kill: generates a signal to the selected process, which will handle it.

* System calls for user-space <-> kernel-space communications:

ioctl: short for Input Output Control, allows an application to control or communicate with a device driver.

2.4.2. Execution threads

Technically, an execution thread is defined as a flow of instructions which can be set to be executed into an OS. From a programmer's point of view, it is a process which works independently of its main program.

When there are many processes being executed simultaneously and independently through an OS, it is said that there is a multi-threaded program. These threads exist into the processes of UNIX because they can be launched by the OS and to be executed as independent entities. These threads just duplicate the essential resources needed to exist as executable code.

In a UNIX environment, a thread:

* Exists into a process and uses its resources.

* Replicates just the essential resources to be executed independently.

* Can share resources of the own process with other independent execution threads.

The threads have a low overhead. This means that they need a low user space because the corresponding overhead has been previously created within the parental process.

Changes done by different threads into the same process will be seen by the rest of the threads because they share the same resources. It is possible to read and write over a shared memory area, but it requires a certain explicit synchronization to avoid unexpected results.

2.4.2.1. Execution threads vs. processes

The execution threads differ from the classical multi task system processes in that the processes are independent, contain a considerable state information, have separated memory spaces and can only interact through inter process communication mechanisms.

Typically, the threads share state information within the same process. They also share memory and other data directly. A change of context between different threads within the same process is generally quicker than the one done between different processes. The threads also require atomic operations, generally implemented through semaphores, to avoid modifying common data simultaneously, or to be read at the same time they are modified. An incorrect use of the shared memory can lead to a situation of deadlock.

The communication between processes or IPC is a set of techniques used to exchange data between 2 or more execution threads within one o more processes. These techniques are divided in methods to pass messages, synchronism, shared memory, and calls to remote procedures. The IPC method varies depending on the bandwidth and the latency of the communication between threads, and also the type of data they are communicating.

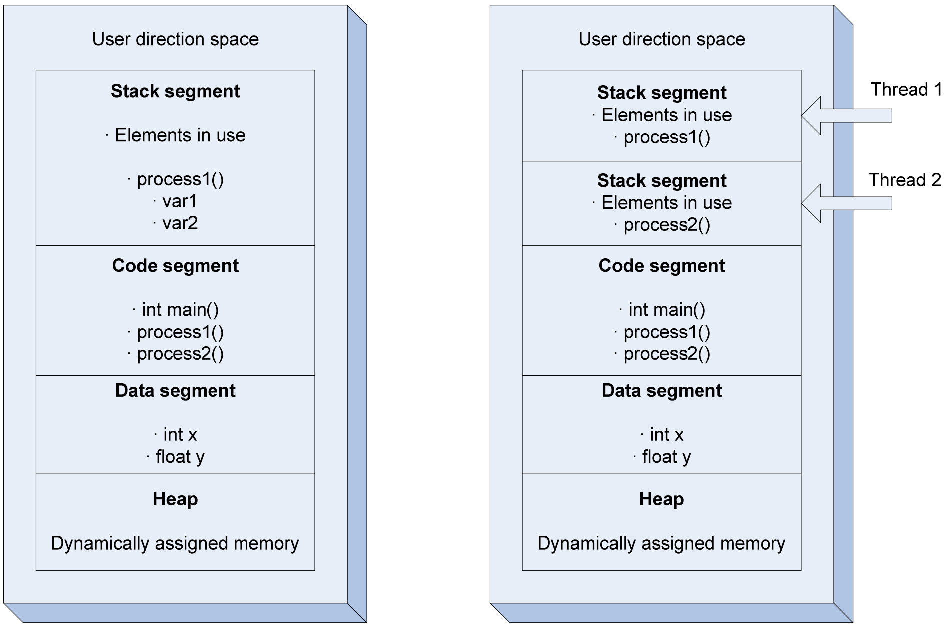


Figure 2.32: Single threaded application vs. a multiple threaded application

2.4.2.2. Communication between processes

The communication can be as simply as letting another process know that an event has happened, or can involve the data transmission between processes. The signals are the standard mechanism to notify a process that an event has occurred. Linux also implements a semaphore mechanism in which a process can wait as easily as it waits for a signal.

2.4.2.2.1. Communication methods between processes

UNIX and other Unix-like OSs provide mechanisms for facilitating communications and data sharing between applications. Collectively, the activities enabled by these mechanisms are IPCs. Some forms of IPC facilitate the division of labor among several specialized processes. Other IPC forms ease sharing the task among computers on a network. The following points summarize the main methods commonly used to communicate processes into a Unix-like framework:

* *Signal*. A *signal* is essentially an asynchronous notification sent to a process to notify that an event has occurred. When a *signal* is generated, the OS modifies its normal operation scheme. In case a procedure has been

established with priority to handle this signal, this procedure is the one that is executed. Otherwise, the default action which handles the signal is executed.

* *Socket*. A *socket* is defined by an IP address, a protocol and a port number. That Is a connection method between two programs located into different machines, in the way they can exchange information in a ordered and reliable way.

* *Pipe*. A set of processes are linked by their standard flows in a UNIX *pipe*. Every standard output from the previous process is fed by the standard input of the following process. Each of these connections is implemented by a mechanism called *pipe*.

* *Semaphore*. It is a protected variable or a set of abstract protected data that performs a classic method to restrict the access to shared resources, as well as shared data into an environment of multiple programs under execution. The semaphores exist as different variables, as the common semaphore or the binary semaphore. The first variable is a counter of set of user available resources, and the second one is more likely a flag to get access rights over a unique resource. The semaphores are the most classical solution to prevent race conditions.

* *Messages*. They are a way of communication where the *messages* are sent to a receptacle. The different *messages* involve function invocations, signal, data and packets. The prominent computation models based in pass of *messages* include the models of actor and calculating processes. Message Parsing Interface (MPI), Java Remote Method Invocation (RMI) and Common Object Request Broker Architecture (CORBA) are some examples.

* *Ports*. A network port is a single interface to establish a communication with a program through the network. The ports use to be numbered being some network level protocol the one which dispatches some port number to the sent information. The implementation of the protocol at the destination side will make use of this number to decide to which program deliver the received data.

2.4.2.2.2. Data exchange between processes

The *pipe* mechanism allows a son process to inherit a communication channel with the parental process that has created it. By this way, written data in a side of the *pipe* is read at the other side.

Other method is the use of *shared memory*. This provides an extremely quick way to communicate large or small quantities of data. Any data written by a process at a shared memory region can be read by any other process that has mapped in his direction space the same memory region.

2.4.2.3. UNIX Threads: library POSIX Threads (Pthread)

POSIX Threads or *Pthread* is a POSIX standard library which provides an interface to design multi-threaded applications. *Pthread* is commonly used in UNIX systems, but also implementations exist for Ms Windows systems.

Pthread is defined as a set of programming types and procedure calls in C language implemented through a library or a header file called *pthread.h,* which is also a part of *libc* library.

The threads into a process use the same direction space. The communication between them is much more efficient, in most cases, than in a inter process communication. The programs which fit the following features are candidates to be designed with multiple threads:

* Programs whose tasks can or must be executed simultaneously.

* Programs that need to be blocked for long waits in incoming or outgoing actions.

* Programs that must respond to asynchronous events.

* Programs in which some part of the work has a higher priority.

The programming multi thread models are the following:

* *Manager/worker*: a thread assigns work to the rest of the working threads. The manager handles all the incoming information and divides it between the rest of tasks.

* *Pipeline*: a task is divided in a set of sub operations, each of them handled in serial but concurrently through different threads.

* *Peer*: is a *manager/worker* alike method, but is slightly different because once the tasks are assigned, the manager thread also takes part in the work.

2.4.2.3.1. Shared memory model

Is a model in which different processes work together concurrently over shared memory areas. Reading or writing over some shared memory areas can produce not desired results. This is the reason why the use of synchronization mechanisms is mandatory in order to protect global shared data. By the other side, each thread can use of their own private data, which will not require the use of synchronization methods.

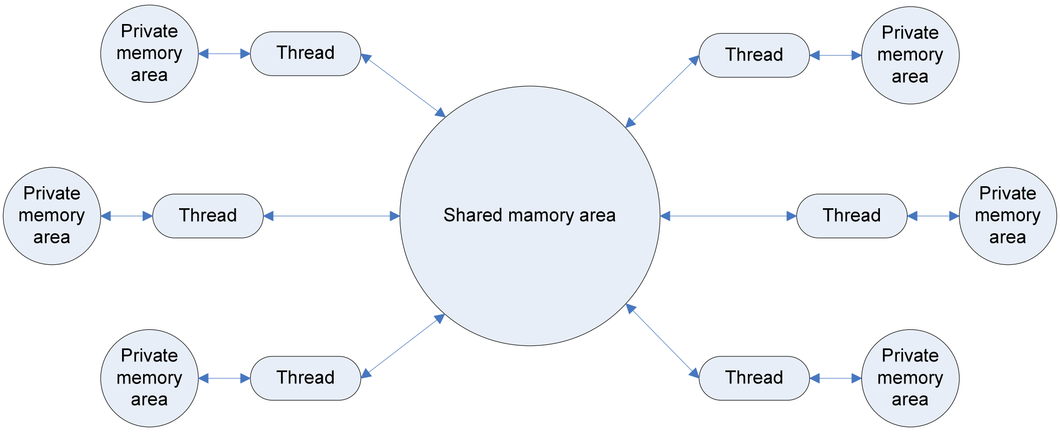


Figure 2.33: Thread access to shared memory

2.4.2.3.2. The Pthread API [7]

The *Pthread API* is defined at ANSI/IEEE POSIX 1003.1 – 1995 standard. The elements composing it can be divided into three categories:

* Thread management: creating, join, and fork of execution threads. Some functions are included to set the thread attributes.

* *Mutexes*: they deal with the synchronization and the access to shared memory. *Mutex* is the acronym of *mutual exclusion*. Some functions are provided in order to ease the creation, destruction, locking and unlocking of *mutex* variables.

* Condition variables or waiting queues: these functions are used to address communications between threads that share a *mutex*. Functions for creation, destroy condition variables and to awake processes that use this type of variables are included. There are also included some functions to set and to interrogate these variables.

2.4.2.3.2.1. Thread management

Thread creation

The `pthread_create` routine creates a new execution thread. Once created, the threads are new processes, and can create new threads as well. There does not exist a hierarchy or implicit dependency between threads. The maximum number of threads that can be started only depends on the application.

Thread exiting

The threads can end their execution by different ways. One way is ending their code and coming back to their starting routine, as well as invoking `pthread_exit`. This routine is used to exit explicitly from the execution of a thread. It is commonly called when a thread has finished its task and its existence is not required anymore.

Thread joining

Thread joining is a way to accomplish with the synchronization of many of them. The `pthread_join` routine locks the thread which invokes that once the specified thread ends its execution. It is possible to get the ending state of the target thread if it is specified through a call to `pthread_exit`.

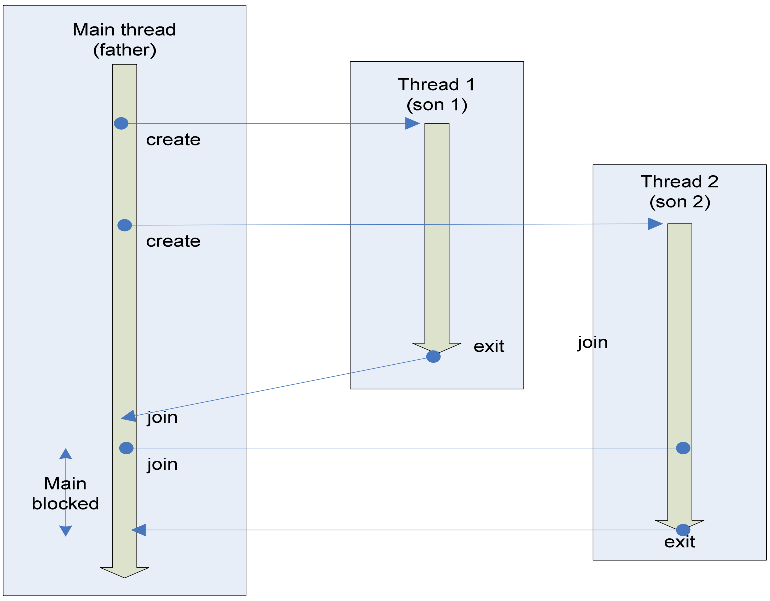


Figure 2.34: thread creation and joining

2.4.2.3.2.2. Mutex variables

As mentioned before, *mutex* is the acronym of mutual exclusion. The *mutex* variables are the simplest mechanism to implement the save access methods to variables which require a shared use in reading and writing actions.

A *mutex* variable acts as a lock, restricting the access to a shared resource. Only one process can take the control a *mutex* variable for a determined period of time. By this way, many processes can attempt to take the control of it without succeeding because previously some process must have locked it. No thread can access to the variable until the current user unlocks it.

Generally, actions done by a thread that has the control of a *mutex* variable use to update global variables. This is a secure way to assure that the final value of a variable is the same as if this action where done by just a thread when many threads want to access it. The portion of code where a shared variable is read or written is called *critical zone*.

Mutex locking

This routine used by a thread to get the control of the specified *mutex* variable is called `pthread_mutex_lock`. If the variable has been locked previously and remains locked by a thread, this call will block the calling thread until the thread that has its control unlocks it.

Mutex unlocking

The routine that unlocks a *mutex* variable is called `pthread_mutex_unlock`. Calling this routine is required after a thread has finished doing its task within the *critical zone*. This routine must not be called if the variable was previously unlocked or in case the specified *mutex* is not under the control of the thread that invokes it. In this case, it would cause an error.

2.4.2.3.2.3. Condition variables or waiting queues

The condition variables allow threads to synchronize themselves depending on the current value of certain data. Without those variables, the programs should be permanently polling (frequently over critical sections) to prove if certain condition has been met. This method is called *active waiting*. This causes in a waste of CPU resources because the processor is continuously busy with this activity. The condition variables solve this problem because they are not generating a continuous polling. This method is called *passive waiting*.

Condition variables are always used with *mutex* variables together.

Waiting over condition variables

The `pthread_cond_wait` procedure blocks the calling thread until the specified condition variable is signaled. This routine is invoked while the *mutex* variable rests locked, and the calling will unlock it while the process remains waiting. When the thread awakes, it will take the control of the *mutex* variable again locking it. Finally, the mentioned variable should be unlocked by the time the process does not need it going out from the critical zone.

Signaling over condition variables

The `pthread_cond_signal` routine is used to awake a thread that is waiting on a condition variable. Must be called after the *mutex* has been locked, unlocking the *mutex* passed as a parameter into `pthread_cond_wait` and letting it being completed.

2.4.2.3.3. Access pattern to shared memory with pthread

At the end, the use of Pthreads is reduced to a simple schema which is reproduced with slight variations. Table 2.8 describes an example of the access pattern to a shared variable within two working threads:

<table>
<tr><th colspan="2">Main thread</th></tr>
<tr><td colspan="2">* Declaration and initiation of global data that require synchronization.
* Declaration and initiation of condition variables.
* Declaration and initiation of the associated <i>mutex</i> variable.
* Creation of the working threads.</td></tr>
<tr><th>Thread B</th><th>Thread B</th></tr>
<tr><td>* Does its task until it finds a shared variable.
* Locks the associated <i>mutex</i> and checks the value of this variable.
* A pthread_cond_wait call can be used to realize a blocking wait by other thread. This call unlocks the <i>mutex</i> variable passed as a parameter and so Thread B can take its control.
* When the Thread B signaling is received, the current thread is awaken and gets the <i>mutex</i> control again.
* Unlocks the <i>mutex</i>.
* Goes on with its task.</td><td>* Does its task.
* Locks the associated <i>mutex</i> variable.
* Changes the value of the shared global variable which Thread A is waiting for.
* Checks the global variable value that waits for the Thread A. If certain condition is met, signals the Thread A (awakes it).
* Unlocks the <i>mutex</i>.
* Goes on with its task.</td></tr>
<tr><th colspan="2">Main thread</th></tr>
<tr><td colspan="2">* Joins the working threads
* Clears <i>mutex</i> and condition variables.</td></tr>
</table>

Table 2.8: multi threaded access pattern to a shared variable

2.4.2. Linux wireless management subsystem

In Unix-alike systems, the OS consists of two parts: a wide set of utility programs that lead the majority of operations, and a kernel part which runs the programs. The kernel runs in supervisor mode and acts as a program loader and supervisor for the rest of utility programs performing the whole system. It also provides locking and I/O services for these programs. The following clauses provide an overview of the elements that take part in the configuration of the kernel and user space of the OS wireless part.

2.4.2.1. Kernel space entities [8] [9]

The kernel is the central component of most OSs. It means a nexus between applications and the actual data processing done at the hardware level. The kernel responsibilities include managing the system resources, what means the communication between hardware and software components. Usually as a basic component of an OS, a kernel provides the lowest abstraction for the resources that software must control to perform its functionalities.

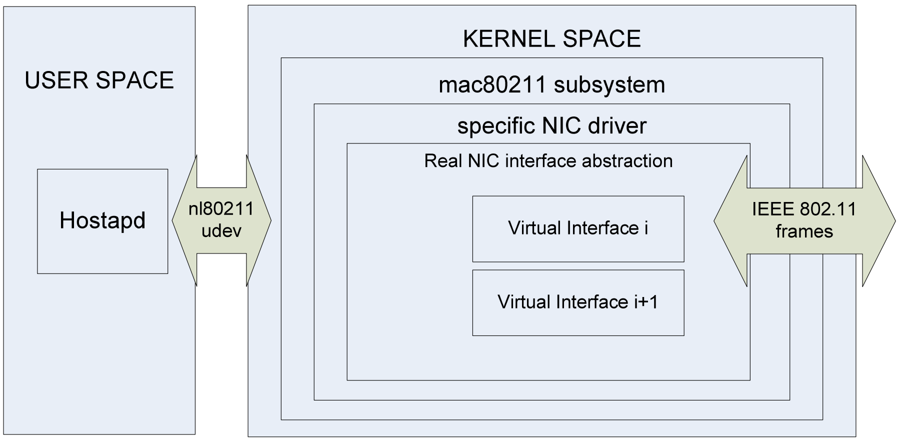


Figure 2.35: User space and kernel wireless abstraction

2.4.2.1.1. mac80211

The Linux kernel API *mac80211* is used to develop *SoftMAC* drivers for Wi-Fi devices. This means that the Media Layer Management Entity (MLME) code is included, apart from the common code shared between the *SoftMAC* devices. One of the advantages of this stack is that the consistency of the different drivers for each card can be improved.

Some of the features supported by *mac80211* subsystem are its compatibility with IEEE 802.11abgn standards, as well as roaming between *wpa_supplicant*, as well as IEEE 802.11r, management of various types of virtual interfaces and quality of service, amongst many other things.

The *mac80211* subsystem is introduced into the Linux kernel in May of 2007, merging in the 2.6.22 version. In October of 2007. John Linville becomes the maintainer responsible of *mac80211* stack.

2.4.2.1.2. cfg80211

The wireless device configuration API *cfg80211* is indented to replace Wireless Extensions (WEXT) in long term. This is a thin layer between user space and *mac80211* stack, but thicker than WEXT. Its main function is protocol translation, keeping the coherence between them. The API *cfg80211* does not pretend to add new features to the currently existing in WEXT, just looks for solving its problems. The main features of *cfg80211* are the following:

* Device registry. The drivers are registered into *cfg80211* introducing capabilities such as bands, channels, bitrates per band, and interface supported modes.

* Regulatory support. The Central Regulatory Domain Agent (CRDA) has this responsibility. This entity supplies support at user space and provides information about the applied restrictions updating the registered channel list and notifying it to the driver. The communication between kernel and user space is done through *uevents* when a notification is passed to the user space and through *nl80211* when *udev* uses CRDA to upload information to the kernel.

* AP management. It allows adding, deleting, and modifying stations. It also stores a list of STAs.

* Key management. It is applied with AP mode.

* Virtual interface management. It allows creating and deleting virtual interfaces, as well as changing its mode and its flags in case they had been set to monitor mode. More than one virtual interface can be set just in case they are associated to the same physical device sharing their physical features such as the operating channel. The driver is in charge of rejecting the impossible configurations.

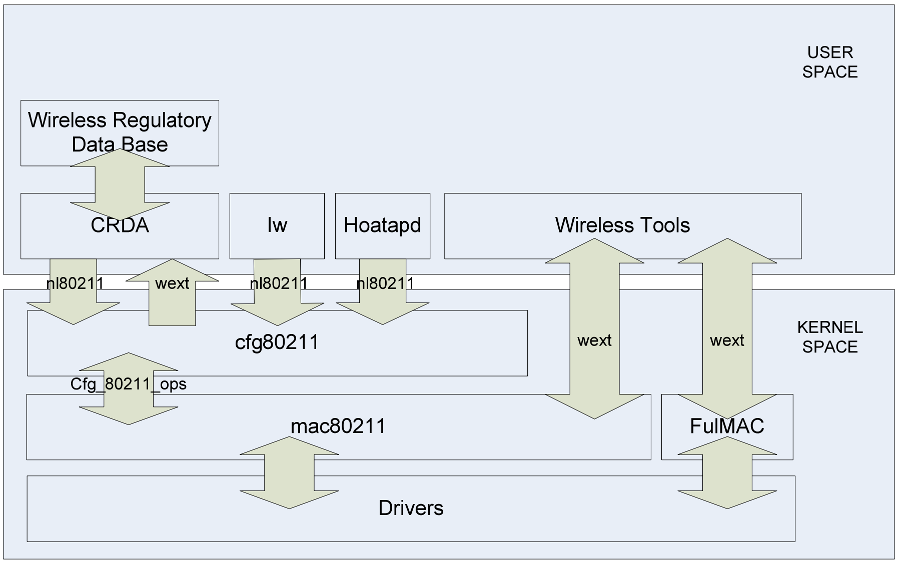


Figure 2.36: Linux wireless kernel and user space architecture

* Scanning features. Allows many more options than WEXT, like scanning multiple SSIDs, specifying a channel and insertion of IEs. The scanning flow is developed as follows: a query is generated at user space and transmitted via *nl80211* (or WEXT) to the kernel, being handled there. The *mac80211* stack (or the driver) scans according to the query. A list of BSSs is filled with the information got from the beacons (passive scanning) or the probe-response frames (active scanning). This information is handled in *cfg80211*. Once scanning has finished, the user space is notified via *nl80211* (or WEXT). Then, the user space queries the list of BSSs.

2.4.2.1.3 libnl / nl80211

The transport layer *libnl* is used to communicate the user space with the kernel through the *netlinks*. It works with a TLV based protocol of object exchange. It implements different families, being the generic *netlink* one amongst them.

The 802.11 *netlink* interface public header *nl80211* is defined in */include/Linux/nl80211.h*. It is used to configure devices providing user space access to *cfg80211* functionality. Current entities using it at user space are *iw*, CRDA, WPA supplicant and Hostapd.

2.4.2.1.4. udev

The device manager *udev* is used at kernel space (2.6 versions and later) which is executed through the *udev* daemon. Its mission is controlling the device files allocated in /dev folder. The daemon detects when a system device has been plugged or unplugged, updating the files from the mentioned folder and giving them a fix name independently from other connected devices or from the order in which they were added to the system. *udev* allows the management of device names from user space, making possible to assign their names without modifying the kernel. In this context, *udev* is a tool the kernel uses to pass events or calls to CRDA (see page 66).

2.4.2.1.5. Wireless Extensions (WEXT)

WEXT is an API which allows the user to manipulate any wireless networking device in standard and uniform way. This is a flexible and extensible interface providing device configuration, wireless statistics and wireless events aware. This interface evolved with the apparition of new devices and with specific needs, but nowadays is not being further developed. Its functionality is currently being implemented into *cfg80211* and *nl80211* entities.

Wireless Extensions make use of *ioctls*. Each *ioctl* command is usually an undocumented system call, and there is no way to monitor these calls in any sort of comprehensive manner. It is also difficult to make the unstructured *ioctl* arguments work identically on all systems. This is the main reason why the development of WEXT was abandoned.

2.4.1.2. User space entities

The kernel is a space strictly reserved for running itself, its extensions, and some device drivers. In contrast, user space is the memory area where all user applications are run and this memory can be swapped out when necessary. Each user space application runs in its own memory space and cannot access the memory of other applications. This is the basis for memory protection in today's OSs. The following clauses provide a brief description of the user space entities involved in the wireless part of the Linux kernel.

2.4.2.2.1. *iw*

The tool *iw* is based in command line and *nl80211* to configure wireless devices. It is possible to configure drivers included into mac80211 stack using this tool. It is still under development and will replace and enhance the set of tools provided by Wireless Tools in mid or long term.

The capacities of a device can be obtained using *iw*, as well as scanning, listen to events, get characteristics of a station, create or destroy virtual interfaces and set them to different modes (monitor, ad-hoc, managed or mesh), set the frequency or the operation channel or even update the regulatory domain.

2.4.2.2.2. Central Regulatory Domain Agent (CRDA)

CRDA is a communication support between kernel and user space for compatibility between regulatory domains. It lays over *nl80211* to establish the communication between both entities. It is implemented in the manner that it can only be executed through an *udev* communication from the kernel.

Entities as *iw*, *wpa_supplicant* or the visual application *NetworkManager* can be used for managing and updating the regulatory domain through queries to the kernel.

2.4.2.2.3. Hostap Daemon (Hostapd)

Hostapd is a daemon capable to configure Wi-Fi cards as access points under the IEEE 802.11 standard. Moreover, Hostapd has authentication capabilities such as IEEE 802.1X, WPA, WPA2, EAP, and RADIUS. Before *mac80211* stack appeared into the Linux kernel, Hostapd worked as a support for APs based in drivers HostAP, MADWiFi and Prism54. Since the apparition on this stack, many drivers have been added allowing access points capabilities through the *nl80211* that Hostapd uses.

The *mac80211* layer transfers all the features of the access point mode to the user space. By this way, Hostapd establishes each of the aspects of a Wi-Fi infrastructure using a configuration file. Likewise, the classical method to create APs through the CLI Wireless Tools invocation is no longer functional, as well as Madwifi Tools for cards using the *ath_pci* driver from MADWiFi.

Hostapd daemon configures the virtual interface indicated in the configuration file (commonly, interface wlan0) for the IEEE 802.11 packet transit and through the mentioned *nl80211*. Moreover, a new interface in monitor mode is created (commonly, interface mon.wlan0) where management frames are sent and received through.

2.4.2.2.4. Other: Wireless Tools, Madwifi Tools

The Wireless Tools is a set of tools allowing handling the WEXT. They are implemented as command line instructions with the aim of supporting the full WEXT. The most common used tools are the following:

* *iwconfig*: tool to manipulate the basic wireless parameters.

* *iwlist*: tool that allows to initiate scanning and list frequencies, bit-rates, encryption keys and others.

Wireless Tools supports fully IEEE 802.11 parameters and devices, support older style of devices and most proprietary protocols, and are prepared to handle *HiperLan* as well.

The Madwifi Tools are similar to Wireless Tools, but implemented just for Atheros devices. These devices can be configured with both set of tools, but Madwifi Tools enhances the possibilities provided by Wireless Tools on Atheros devices for Linux OSs.

2.4.3. Chronological wireless Linux improvements [10]

The kernel of Linux is continuously evolving. Security patch updates appear every day in order to improve the performance of these based systems. New kernel versions are released with a period of one or two month between them. Many developers around the world are involved in different branches of the development of this kernel, as well as enhancing the user-space experience.

In some cases, the development of an application for a UNIX or Unix-like OS can drive to difficulties. The continuous evolution of these OSs not only provides a better performance, but also deprecates some designs. Even sometimes, the required features for a particular design are not available yet. The current design had to evolve since the beginning because the existing resources did not fit with the requirements at all. In particular, it was a lack of resources supplied by the wireless subsystem. A brief description of the Linux wireless kernel evolution is provided next, embracing the kernels used since the beginning of the work until the last working version.

2.4.3.1. Linux Kernel 2.6.24. (February 2008)

This was the kernel version included in the Debian 4.0 release, amongst other distributions. A new branch of the kernel was announced by this time. Its name was *Wireless Testing*, and represented the focus the developers' effort into a development aspect just for the wireless part. The common development branch was abandoned from this moment on to gather efforts on the *Wireless Testing* branch.

2.4.3.2. Linux Kernel 2.6.25. (April 2008)

An important evolution of the wireless drivers' cards hierarchal organization came with this version. Some drivers started to be ported to the emerging *mac80211* stack. Different drivers join together to be handled into this API. In particular, the free version of Atheros driver named *ath5k* and the *b43* driver for Broadcom chipsets were included into this stack.

Relevant features added to *mac80211* stack were key management, beacon stuffing, and other elements designed to manage stations via *cfg80211*.

2.4.3.3. Linux Kernel 2.6.26. (July 2008)

This kernel improved the *mac80211* stack adding new features such as a better support to the proper configuration of monitor interfaces (*cooked* monitor). By other side, *cfg80211* added a new API which implemented the channel and bit rate conversion intensifying the *netlink* use instead of the classical socket. It resulted in an improved communication between user space and kernel. The driver *ath5k* and *b43* were improved adding new features and solving some bugs present in the previous versions.

During the development of this kernel version the access point creation support received a strong effort by the developers' part. Those features were not present at this kernel version anyway.

2.4.3.4. Linux Kernel 2.6.27. (October 2008)

This version did not represent a relevant improvement in its wireless part. Some bugs were solved and the code was improved to make it more efficient. The *mac80211* stack added some features for spectrum management, and some algorithms were introduced into this module.

The *b43* driver added firmware support for authors.

During this time between versions, Atheros announced the upcoming open code of its driver Hardware Abstraction Layer (HAL). From this moment on, all the Atheros code became open source. This meant that the improvements of *ath5k* drivers would be easier to implement, and the developers would not have to do reverse engineering to program the driver.

2.4.3.5. Linux Kernel 2.6.28. (December 2008)

The most relevant feature of this schedule was the inclusion of support for countries regularization. It consists on a CRDA data base available in user space that acts as a support for *udev* for the communication between kernel and user space for the regulatory compatibility.

The API *cfg80211* added a new infrastructure for wireless domains' regularity. Options for BSSs were added for AP mode configuration.

2.4.3.6. Linux Kernel 2.6.29. (March 2009)

The most representative feature in this schedule was the inclusion of access point functionality for the drivers which allowed this. *mac80211* layer is ready to operate in master mode but only through the Hostapd daemon and a configuration through *cfg80211*. Tools such as *iwconfig* are not enough to set and handle APs.

2.4.3.7. Linux Kernel 2.6.30. (June 2009)

The driver for Realtek Universal Serial Bus (USB) devices got support for many features by this time such as power save and mesh configuration, but did not take part of *mac80211* stack still. For their part, the tuple *cfg80211/nl80211* enhanced its set of features providing support for scanning, settable via *iw* tool. Moreover, the *ath5k* module received a new improvement adding support for transmitted power calibration.

2.4.3.8. Linux Kernel 2.6.31. (September 2009)

At the end, the driver for Realtek USB devices *rt2800usb* got support into the *mac80211* stack, but still experimental. The driver *ath5k* enabled support for AP mode, among other achievements.

By this time, the wireless devices working under the *mac80211* stack, such as those handled by *ath5k* and *rt2800usb* controllers, are seen as homogeneous devices under the OS point of view. They can be handled through the *iw* tool, are displayed as *wlanX* and *phyX* virtual and physical devices respectively, and allow STA features as well as monitor modes, amongst many other features.

3. Design and Development

3.1 General scenario

The following clause describes the key points regarding communication between different MIHF entities in the Mobile Node (MN) and the network. A brief introduction to the elements and interfaces composing a general scenario is provided in order to clarify these points.

Two parts can be distinguished into an IEEE 802.21 communication framework: the client side and the network side. While a client side is composed just of a mobile node with one or many radio interfaces, the network side can be constituted by many entities. Reference points interfacing different elements between sides and into the network side are declared just for illustration purposes, and although they are referenced in standards, they do not define any specific deployed network system architecture. Each communication between entities can involve different types of interfaces, transport mechanisms and even different service contents.

A Point of Service (PoS) is a part of the network which can provide direct access to a MN at Layer 2 (L2. A MN exchanges information directly with its PoS. If a network entity cannot be connected directly to this MN, it does not act as PoS for that particular MN. So, a MN with multiple radio interfaces can use a L2 transport link for exchanging information with a PoS residing in the local network, and at the same time can be using a Layer 3 (L3) transport protocol to exchange information with other PoA not residing in the same local network.

Figure 3.1 provides an overview of a general MIH framework. The following are the elements taking part in it, as well as the reference points, all of them seen under the MIH level point of view:

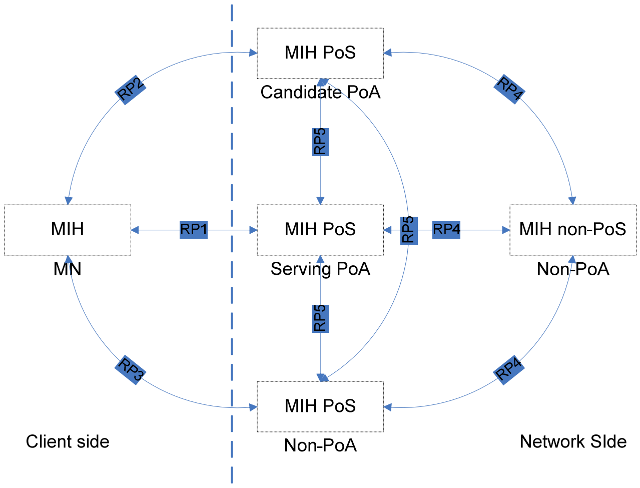


Figure 3.1: MIHF communication model

* A MIHF in the mobile node

* A MIH which is PoS, and acts as serving PoA for the MN.

* A MIH which is PoS, and acts as candidate PoA for the MN.

* A MIH which is PoS, but has no functionalities of PoA for the MN.

* A MIH which is not a PoS, and so cannot include PoA functionalities for the MN.

The reference points (RP) between entities can be summarized as follows:

* RP1: is a link between MN and PoS providing serving PoA facilities. Communication at L2 and above is allowed.

* RP2: is a link between MN and PoS providing candidate PoA facilities. Communication at L2 and above is allowed.

* RP3: is a link between MN and PoS that does not provide PoA facilities. A communication can be done at L3 and above, and with transport protocols at L2 such as Ethernet or Multi Protocol Label Switching (MPLS).

* RP4: is a connection between local PoS and foreign non-PoS entities. Communication at L3 and above is allowed.

* RP5: is a connection between local PoS and foreign PoS entities. Communication at L3 and above is allowed.

3.2. Tools and platforms used

An application design and implementation involves considering several components that can take part in the end or not, ranging from those related to the programming language to the specific tools chosen for building the application. The following sections provide an overview of those which have appeared during the implementation of the current software and somehow have contributed to its development.

3.2.1. Programming language

A programming language is an artificial language designed to control the physical or logical behavior of a machine which in most cases is a computer. It is composed of a set of symbols and semantic rules that define its structure and the meaning of its expressions. These elements, when properly combined, perform the code of a program. Due to the fact that a machine needs to understand this code, an application called *compiler* translates it to computer language. The upcoming sub-sections supply a brief description of the programming languages as well as some relevant features about the compiler that have been used in the present work.

3.2.1.1. C programming language [11]

C is a procedural programming language created in 1972 by Bell Laboratories. It supposed the evolution of its predecessor, the B programming language. Its authors developed it with the purpose of implementing the UNIX operating system. Although its target was just implementing software for systems, it has also been efficient for developing applications.

C is considered a mid-level programming language, but it has many features of low and high level languages. For example, it can deal with common data structures typical in high level languages. It also provides constructions allowing a very low level control of the data. Compilers typically offer language extensions that allow mixing C code with Assembly, or to get direct access to memory or peripheral devices.

3.2.1.1.1. Pointers

A pointer is a variable that records the address of another variable, or in general, an object, which is stored somewhere in memory. A pointer can be de-referenced to access data stored at the direction it is pointing to, or to invoke a pointed function. It can also be modified by using pointer arithmetic. Pointers are commonly defined as the data type they are pointing to, so its arithmetic is automatically scaled to their type of data. Pointers can be used for many different purposes such as dynamic memory allocation, to implement callbacks for function handlers or simply to manipulate the values of an array.

A NULL pointer is a pointer that does not point to any valid memory address. When declaring a pointer, it is automatically pointing to NULL, so it is not possible to assign a value to the memory it is pointing before assigning it a valid memory direction.

Void is a reserved word typically used in programming to design a variable that has no type. Void pointers point to objects of unknown type, and can therefore be used as generic data pointers. Void pointers cannot be de-referenced, and no arithmetic is valid, but they can easily be converted to and from any other pointer type.

3.2.1.1.2. Memory management

There are two basic ways in which memory can be allocated in a C application:

* *Static allocation*. The memory occupied by an object is defined in compile time at the source code.

* *Dynamic allocation*. The memory occupied by an object is defined in run time and can be variable. C language provides tools such as `malloc` and `free` to define the memory the object will occupy and to free it, respectively, from a region of memory called *heap*. A pointer is the type of variable defined to point a portion of memory to be assigned dynamically.

Moreover, the memory can also be allocated temporally when a block uses objects that temporary stores at stack. Once the said block ceases to exist, the stack related objects is automatically freed.

3.2.1.1.2. Extensions to C language: libraries

In C, a library is a set of functions contained within a file. Each library typically has a header file, which contains the prototypes of the functions contained within the library that may be used by a program, and declarations of particular data types and macros used with these functions. In order for a program to use a library, it must include the library header file as a pre-compiler directive.

The most common C library is the C standard library, which comes with every C implementation. This library supports stream input and output, memory allocation, mathematics, character strings, time values, etc. In UNIX and Unix-like systems also exists a set of libraries into POSIX standards providing functions which allow interfacing with the kernel, amongst others.

3.2.1.2. C++ programming language [12]

C++ is a programming language designed in 1983. Its purpose was enhancing the C programming language with features that allowed object handling, making C perfectly compatible with C++. Afterwards, generic programming features were added to complete a multi-paradigm language: a structured language, allowing object oriented programming and generic programming.

C++ also allows redefining operators functionalities within the scope of a class (also called operator overloading) and the possibility of creating new types as derived from the standard ones.

3.2.1.2.1. Classes and objects

A class is a container of one or more data together with functions that can handle these data. In C++, the objects are abstracted through classes. An object is composed of methods or member functions, and attributes or member variables. So, an object becomes an instance of a class.

3.2.1.2.2. Constructors and destructors

A constructor is a special method within a class which is executed when a class is instanced. In C++, a constructor has the same name as the class it belongs, and does not return any value. Its general purpose is starting the attributes of the created object and allocating the required memory for it. Its implementation is mandatory into a C++ class.

Destructors are special member functions that are called automatically when an object ceases to exist. Its target is releasing no more required memory resources from the associated object in execution time. Its implementation is not mandatory, but recommended.

C++ does not have itself any mechanism to automatically release unreferenced objects. An entity doing this task is called a garbage collector. It is important to bear it in mind since the use of the garbage collector can help to obtain an efficient program implementation as well as to avoid memory exhaustion. See further sections (IDEs, compilers) for affordable solutions.

3.2.1.2.3. Templates

Templates are the resource provided in C++ to implement generic programming. They provide a way to work with abstracted data types into classes and functions, defining the data type they require later. For instance, it is possible building a generic vector allowing any data type within. Afterwards, it can be instanced to perform a vector containing variables of `int`, `float`, `char`, `struct` or whatever data type.

3.2.1.2.4. Inheritance

In C++, inheritance is an abstraction mechanism designed to ease and improve the design of a class. It is possible to create new classes from the existing ones via inheritance.

The derived classes inherit methods and attributes from the base class, allowing a redefinition of its behavior and adding new member functions and variables. To accomplish with the encapsulation principle, it is possible to define the way the methods and attributes are inherited. If a member function or variable is defined as `protected`, its visibility will be public if it is seen from a class of the same hierarchy or private otherwise.

3.2.1.3. GNU Compiler Collection (GCC) [13]

GCC is a set of free compilers created by the GNU Project under the General Public License (GPL). These compilers are considered standard for open source operating systems UNIX or Unix-alike, as well as owner operating systems such as Mac OS X.

Originally, GCC meant GNU C Compiler because it only compiled C programming language. Afterwards it was enhanced to compile other languages such as C++. The development of GCC uses an open development environment and supports many other platforms with the goal of promoting the use of a global compiler and optimizer. It would result attractive for many development teams, allowing GCC and GNU systems running in several architectures and environments.

The latest version of GCC includes front ends (part of the software that interacts with the user) for the languages *Ada*, *C* (*ANSI C*), *C++*, *Fortran*, *Java*, *Objective-C* and many other languages which are not included in the principal distribution.

3.2.1.3.1. Compiler g++ [14]

The g++ tool is an UNIX based C++ compiler frequently operated through the command line interface. It is released by the Free Software Foundation (FSF) and often comes into a UNIX or Unix-alike distribution.

The tool g++ builds object code directly from C++ program sources. It does not generate intermediate C code, resulting in a better object code and better debugging information. The GNU Debugger (GDB) works with this information in the object code to provide comprehensive C++ source level editing capabilities.

3.2.1.3.2. Boehm Garbage Collector (BGC) [15] [16]

A garbage collector is an implicit mechanism of memory management developed in programming languages that deal with the concept of memory. The developer does not need to look after where or when to release memory resources because the garbage collector does it automatically. When an application is compiled, a subroutine corresponding to the garbage collector is included in it. The garbage collector is informed about the number references a resource of memory has. When this counter equals zero, it means this memory is not longer being used, and so the resources can be released.

The Boehm GC can be used as a garbage collector for `malloc` and `new` of C and C++ programming languages respectively. It allows allocating memory by using them, but an explicit mechanism to release memory is no longer needed. Several languages using C as an intermediate language prior to compilation can also use Boehm GC to implement garbage collection. In practice, Boehm GC replaces the C functions `malloc` and `realloc` by its own in C code, and also removes `free` calls.

3.2.2. Integrated Development Environments (IDEs)

An Integrated Development Environment (IDE) is an application composed of a set of programming tools in one or many programming languages. It consists on a code editor, a compiler, a debugger and a Graphical User Interface (GUI). The following sections provide a description of the main frameworks used to develop the current project and a slight description of its components.

3.2.2.1. Anjuta [17]

Anjuta is a versatile IDE for C and C++ on GNU/Linux licensed under the GNU GPL. It features a number of advanced programming facilities including project management, application wizards, an interactive debugger and a powerful source editor with source browsing and syntax highlighting.

Anjuta is an effort to marry the flexibility and power of text-based command-line tools with the ease of use of the GNU Network Object Model Environment (GNOME) graphical user interface. That is why it has been made as user-friendly as possible.

3.2.2.2. Geany [18]

Geany is a small and lightweight IDE. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. Another goal was to be as independent as possible from a special desktop environment like Kool Desktop Environment (KDE) or GNOME - *Geany* only requires the GNU Tool Kit 2 (GTK2) runtime libraries.

Some basic features of *Geany* are the following:

* Many supported types of file including *C, Java, PHP, HTML, Python, Perl, Pascal,*

* System built to compile and execute your own code,

* Simple project management.

Geany is known to run under *Linux*, *FreeBSD*, *NetBSD*, *OpenBSD*, *MacOS X*, *AIX v5.3*, *Solaris Express* and *Windows*. More generally, it should run on every platform, which is supported by the GTK libraries. Only the Windows port of *Geany* is missing some features. The code is licensed under the terms of GPL.

3.2.2.3. Code::Blocks [19] [20]

Code::Blocks is an open-source, cross-platform IDE. Using a plug-in architecture, its capabilities and features are defined by the provided plug-ins, making it able to use almost any compiler. By default, *Code::Blocks* is oriented towards C/C++. The plug-in provided with the default *Code::Blocks* installation supports *GNU GCC*, *Microsoft Visual C++ Free Toolkit 2003*, *Borland's C++ Compiler 5.5*, *DigitalMars Free Compiler*, *OpenWatcom* and *Small Device C Compiler (SDCC).* The following Is a list of its main features:

* *Code::Blocks* is an open source project, licensed by GPL.

* Many features can be added via plug-in such as multiple language or compiler support.

* The interface GNU Debugger is also incorporated to allow debugging.

* The code editor provides text complementation, auto-indentation, syntax coloration, code folding and a class browser.

* *Code::Blocks* supports managing workspaces, with several different projects within.

* Makefiles can be created to compile the project indirectly out of the IDE.

The IDE *Code::Blocks* gathers all the required features to allow a complete and pleasant development environment. Most of the work done has been developed under this IDE.

3.2.2.4. Other considered IDEs: Eclipse [21]

Originally developed by IBM, Eclipse is a multi-platform open source IDE typically intended to develop IDEs such as the Java Development Kit (JDK). Moreover, it is also currently used to create general purpose applications or even client applications based in web browser. Through an extensible plug-in system, it can support programming in several languages such as C, C++, COBOL, Python, Perl, PHP, and others.

3.2.2.5. Other considered IDEs: NetBeans [22]

NetBeans is a platform used for developing with Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala and Clojure, amongst others. It is written in Java and runs everywhere where a Java Virtual Machine (JVM) is installed, including Windows, Mac OS, Linux, and Solaris. A JDK is required for Java development functionality, but is not required for development in other programming languages. NetBeans allows composing applications through a set of modular components called modules. These ones can be created, used and extended by third parties.

3.2.3. Network wireless packet handling

The design of the current application required some knowledge about the mechanisms that the system could facilitate to handle wireless devices. Different existing applications that deal with WLAN devices have been taken as reference to research the way they get access to: management information, the data transmitted and a variety of different features for their configuration. The current section and the following will describe the basics and furthers of these applications and provide a synthesis of the tools they use that have been useful to develop the proposed applications.

3.2.3.1. pcap [23]

The pcap (packet capture) library consists of an API designed for capturing network traffic. It also implements tools to set interfaces, get their information and even to transmit packets on a network at link layer. Linux and other Unix-like systems implement pcap through the Libpcap (Library pcap) library. Windows uses a port of Libpcap known as WinPcap (Windows pcap). Several applications such as *Wireshark* (previously known as Ethereal) or *TCPdump* are designed for monitoring packet transactions and analyzing packet captures (live or not) via Libpcap.

The developers of the pcap API wrote it in C/C++ language, so other languages such as *Java* or *.NET* based generally use a wrapper to use it.

3.2.3.2. Libpcap [24]

Libpcap was originally developed by the *TCPdump* developers in the Network Research Group at Lawrence Berkeley Laboratory. The low-level packet capture, capture file reading, and capture file writing code of *TCPdump* was extracted and made into a library, with which *TCPdump* was linked. It is now developed by the same *tcpdump.org* group that develops *TCPdump*. *TCPdump* is a command line interface based network traffic sniffer.

3.2.3.2.1. Basics libpcap API overview [25]

Figure 3.2 depicts a sketch of the operation flow of the entities invoking *Libpcap* routines to proceed a wired or wireless message exchange:

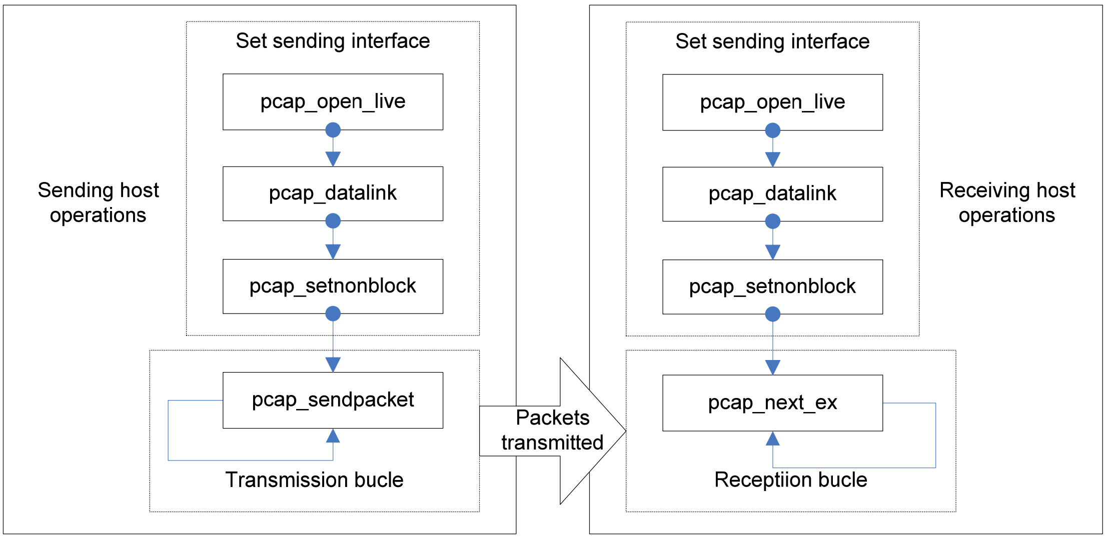


Figure 3.2: Pcap transmit and receive operation flow

3.2.3.2.1.1. Interface preparation

Before entering the capture buckle, it is needed to get a file descriptor of the associated interface. The routine `pcap_open_live` does this job. It is possible to set the capture to promiscuous mode, which means that all the traffic transmitted through the sniffed link will be captured.

Programs using the API defined by *Libpcap* are executed into user space. However, the packet captures are done into kernel space. A change of context is necessary every time a capture is done, but those changes are difficult and it is necessary to minimize them in order to optimize the throughput. `pcap_open_live` also allows specifying the time the kernel is required to gather bytes before transmitting them to user space.

The routine `pcap_datalink` is used to get the link type associated to a network interface. The return value can take many values, but `DLT_IEEE802_11_RADIO` is the one that fits best in a WLAN environment.

The `pcap_setnonblock` routine puts the capture descriptor given by `pcap_open_live` into blocking or non-blocking mode. In non-blocking mode an attempt to read from the capture descriptor will return 0 immediately rather than blocking to wait for new arriving packets, if no packets are currently available to be read.

3.2.3.2.1.2. Transmission

The function `pcap_sendpacket` is used to send a raw packet through a network interface. Note that even having opened successfully the network interface, two different circumstances may occur: not having permission to send packets on the network interface or that it does not support sending packets. The function `pcap_open_live` does not provide a flag to indicate whether to open for receiving, transmitting, or both, hence it is not possible to request an open descriptor, which supports sending, being notified at open time whether sending will be possible. Moreover, some devices might even not support sending packets.

3.2.3.2.1.3. Reception

The routine `pcap_next_ex` is used to read the next packet and returns a success or failure indication. If the packet is read without problems, a pointer to a specific structure called `pcap_pkthdr` and another to `pkt_data` are set pointing to the header and the data of the received physical packet, respectively.

3.2.3.3. Network wireless packet sniffing

Network sniffers are powerful tools used for several purposes such as monitoring networks in order to detect failures, reverse engineering network protocols, or even for helping in protocol creation. The following sections provide a basic overview of a particular network analyzer: *Wireshark*. It is also commented how it helps to deal with endianism in byte ordering.

3.2.3.3.1. Wireshark [26]

*Wireshark***,** previously known as *Ethereal*, is a graphical protocol analyzer used to analyze and solve problems in live communication networks. It is also used for software and protocol development, and for educational purposes. It runs over most of the current operating systems. Its packet capture functionalities are based in pcap library.

Wireshark implements its graphical front-end using GTK+ toolkit, and uses the pcap utilities to capture packets. It is designed to run over the most common operating systems such as Unix-like systems and Ms Windows. *Wireshark* is free software released under the terms of GPL.

Wireshark allows the user to see all traffic being passed over the network by setting the network interface to promiscuous mode. The following are some examples that *Wireshark* is used for:

* Developers use it to debug protocol implementations.

* Network technicians use it to identify networking problems.

Wireshark also provides facilities for many features such as displaying packets with very detailed protocol information and even to filter, search and sort them on many criteria.

In the current project *Wireshark* has been useful to detect whether a packet has been sent/received or not, solving problems in wireless interfaces configuration. It has also provided an accurate debugger to help in checking the packet formation under the IEEE 802.21 standards. One of the most annoying features to debug is endianism.

3.2.3.3.2. Endianism basic overview [27]

Endianism is the name used to define the order in which the information is stored into a computer or is sent through a network. In particular, the endianism of a computer is defined in terms of byte endianism.

Two basic orders of storage are defined: little endian and big endian. A computer using the first type of endianism is the one that stores the most relevant byte at the lowest memory direction, and least relevant byte at the highest. A big endian computer stores the most relevant bytes first, and the least relevant last. For instance, an Intel processor (the brand of processors provided in the current development machines) has a little endian architecture. This idea is clearly sketched in figure 3.3.

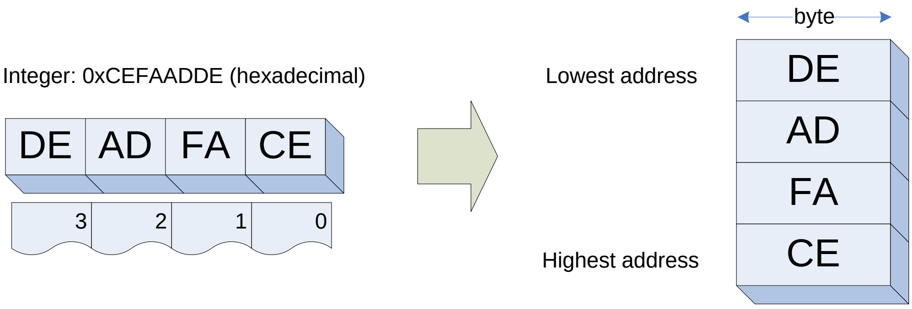


Figure 3.3: Data storage order into a little endian architecture

It is important to keep the coherence between the type of endianess between computers and the network, which is standard implemented in big endian. *Wireshark* eases this task by monitoring the received information in a peer.

3.2.4. Wireless devices handling

3.2.4.1. Starring subroutines in Libiw API provided by Wireless-tools

The API Libiw provides a set of common subroutines to all the wireless tools. The communication with the kernel is done through *ioctl* calls. The following subsections provide a quick overview of the most useful subroutines in the context of the current work extracted from this API.

3.2.4.1.1. Subroutine for statistics

The subroutine `iw_get_stats` is used to get the statistics of a determined wireless network interface. It reads the contents in */proc/net/wireless* system folder from operating system file system and extracts a set of quantitative results such as the received signal strength, amongst others.

3.2.4.1.2. Subroutine for channel handling

The subroutine iw_float2freq is designed to set the channel of the specified working interface. Note that, in a physical interface providing more than one virtual interface, changing the channel of one of the virtual interfaces will cause in a change in all of those depending on the same physical interface.

The subroutine converts a floating point to an internal custom format of frequency representation provided that the kernel has some problems handling floating point formats. Libm (library mathematics) is required in order to avoid troubles in some embedded platforms.

The subroutine iw_freq_to_channel is a support subroutine that allows converting a frequency to a channel.

3.2.4.1.3. Subroutines for wireless information

The following pair of subroutines extracts some information out from */proc/net/wireless* in order to get basic features of the wireless extensions such as version number:

* `iw_get_basic_config` gets the essential wireless configuration from the device driver. All the classical wireless *ioctl* are called on the driver through a socket in order to know what is supported and to get the settings.

* `iw_get_range_info` is the subroutine used to get the range information out of the driver.

3.2.4.1.4. Subroutines for sockets

Two routines are used to handle communications between kernel and user space:

* `iw_sockets_open` opens a new socket. Depending on the designed protocol, opens the corresponding socket. The socket allows establishing a communication with the driver.

* `iw_sockets_close` closes the socket used for the *ioctl* and releases its dedicated resources.

3.2.6.1.6. Inline functions [32]

Inline functions are those that are so simple that it is more efficient inlining them. When a function is defined as inline, the compiler will insert the body of the function everywhere that function is used in, instead of recalling it every time it is used.

The function `iw_set_ext` is the one used to act as a wrapper, pushing some wireless parameter to the driver.

3.2.4.2. Other starring tools and applications for device handling

3.3.4.2.1. ifconfig [28]

The program *ifconfig* is available in Unix-like systems and allows configuring several parameters of network interfaces as well as getting their information. It allows setting many parameters of a network layer interface such as the IP address or the network mask, as well as bringing up or down a determined virtual interface. *ifconfig* can also provide link layer details such as the MAC address or received/sent packet statistics.

3.2.4.2.2. iwconfig [29]

The program *iwconfig* is one of the utilities provided with the Wireless-tools package that allows manipulating the basic wireless parameters. It is composed by many calls to API subroutines in *Libiw*. Amongst them, there is `set_essid_info`, which facilitates setting the name of the network (the Extended Service Set, ESS) which it is desired to be connected to.

3.2.4.2.3. dhclient [30]

The application dhclient, provides a means for configuring one or more network interfaces using the Dynamic Host Configuration Protocol (DHCP), Bootstrap Protocol (BOOTP), or if these protocols fail, by statically assigning an address.

3.2.4.2.4. wlanconfig [31]

The classical Multiband Atheros Driver for Wi-Fi (MADWiFi) supports multiple APs and concurrent AP/STA mode operation on the same device. The devices are restricted to use the same underlying hardware, and are limited to coexist on the same channel and using the same physical layer features. Each instance of an AP or station is called a Virtual AP (or VAP). Each VAP can be in either AP mode, Station Mode, "special" station mode, or Monitor mode. Every VAP has an associated underlying base device which is created when the driver is loaded. The tool wlanconfig is included in the MADWiFi package, and allows creating and destroying VAPs over Atheros' NICs.

The use of this tool is currently deprecated, and *iw* is used instead.

3.3. Applied specifications and technologies

3.3.1. Functional description of system behavior

In a low charge traffic system, the Mobile Node element is the most indicated to decide where and when to be attached. In this context, it is assumed that the surrounding PoAs nearby the MN have enough resources to provide connectivity to it, and so the handover decision once attached relies in the received signal strength.

A definition of the behavior of a mobile-initiated handover system is provided in the following clauses with regards to the actions taken into the process of PoA discovery, attachment and handover inside the IEEE 802.21 framework. The text will deal with standard aspects of the stated concepts, and although its aim is being as general as possible, some particular features will be introduced.

3.3.1.1. MIH discovery and MIH capability discovery

The MIH discovery procedure is used by a MN to discover one or several Points of Attachment in a given area. The IEEE 802.21 standard provides different tools to discover them, and discovery methods for PoAs residing in local or remote networks are also proposed by many standards organizations. One method suggested in IEEE 802.21 std. is recognizing MIH compatible PoAs via information introduced into beacon frames (for WLAN technologies) or Downlink Channel Descriptor (DCD) frames (for WIMAx technologies), for instance.

By this way, the information received through this passive scanning triggers MIH_Link_Detected indication for each discovered PoA. This primitive provides MIH users a set of basic information ranging from services supported (MICS, MIES, MIIS) to degree of QoS, passing through other relevant features such as internet connection availability.

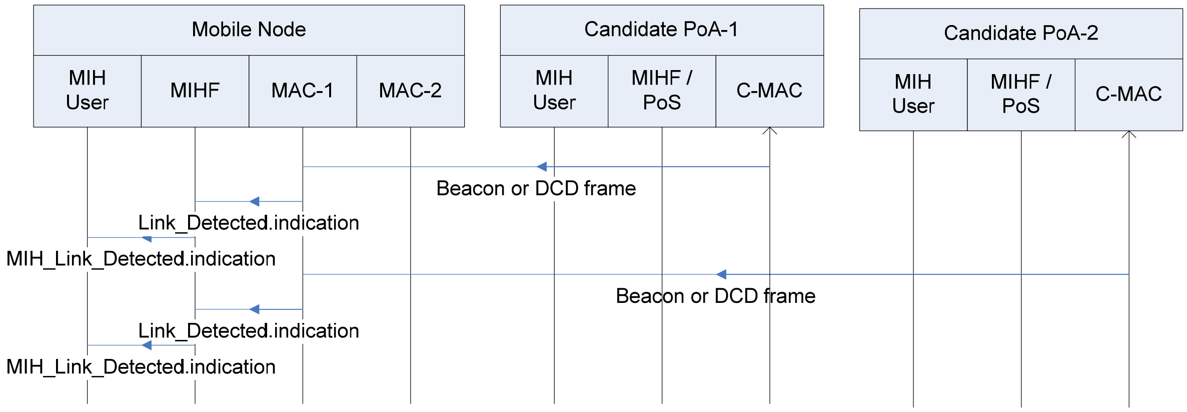


Figure 3.4: MIH PoA discovery through passive scanning

The MIH capability discovery procedure provides further information about the surrounding PoAs and networks that are visible for the MN. These PoAs must act as Point of Services because the MN has to poll them directly through the wireless media, with no intermediary. This procedure is not mandatory or, more precisely, is not defined under a general procedure schema anywhere in IEEE 802.21 std. Nevertheless, its use is recommended to provide more details about the candidate peers in a bootstrap phase (when there is not a single connection established yet). In figure 3.5 the message exchange between a MN and two candidate PoAs for MIH Capability Discovery is sketched.

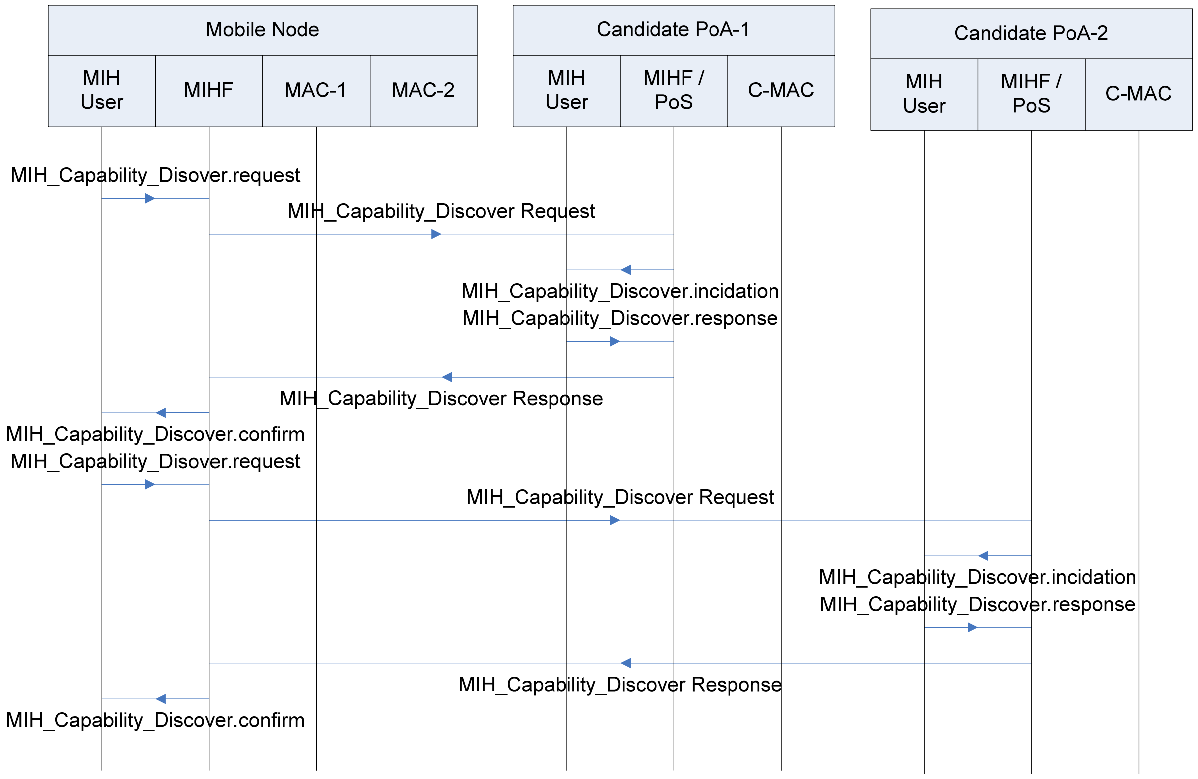

Mobile Node
MIH User
MIHF
MAC-1
MAC-2
Candidate PoA-1
MIH User
MIHF / PoS
C-MAC
Candidate PoA-2
MIH User
MIHF / PoS
C-MAC
MIH_Capability_Disover.request
MIH_Capability_Discover Request
MIH_Capability_Discover.incidation
MIH_Capability_Discover.response
MIH_Capability_Discover Response
MIH_Capability_Discover.confirm
MIH_Capability_Disover.request
MIH_Capability_Discover Request
MIH_Capability_Discover.incidation
MIH_Capability_Discover.response
MIH_Capability_Discover Response
MIH_Capability_Discover.confirm

Figure 3.5: MIH capability discovery

As stated in previous sections, a combined MIH discovery and MIH capability discovery can be performed. An option to do this is by active scanning and setting multicast MIH addresses into MIH_Capability_Discover frames. This is not the chosen option in the current project design.

3.3.1.2. Bootstrapping decision and PoA attachment

The set of features and metrics provided by the candidate PoAs, which include signal metrics amongst others, are passed to the MIH users thanks to MIH function. There, the upper layers decide where to get attached. This is commonly known as “bootstrapping decision”, and there it is determined which will be the first PoA to provide MIH services and data connectivity before turning on the system or in case an abrupt disruption of the connection occurs.

After that, the registry process takes place. Again, there does not exist any specification determining whether or not this procedure is mandatory or must/should be performed in this place or order. MIH registration is defined as a means of requesting access to specific MIH services. It is used by MN to declare its presence to a selected PoA. MIH Registration is needed to get access to the services provided by an MIHF PoS. Figure 3.6 shows the MIH registration procedure in terms of message exchange.

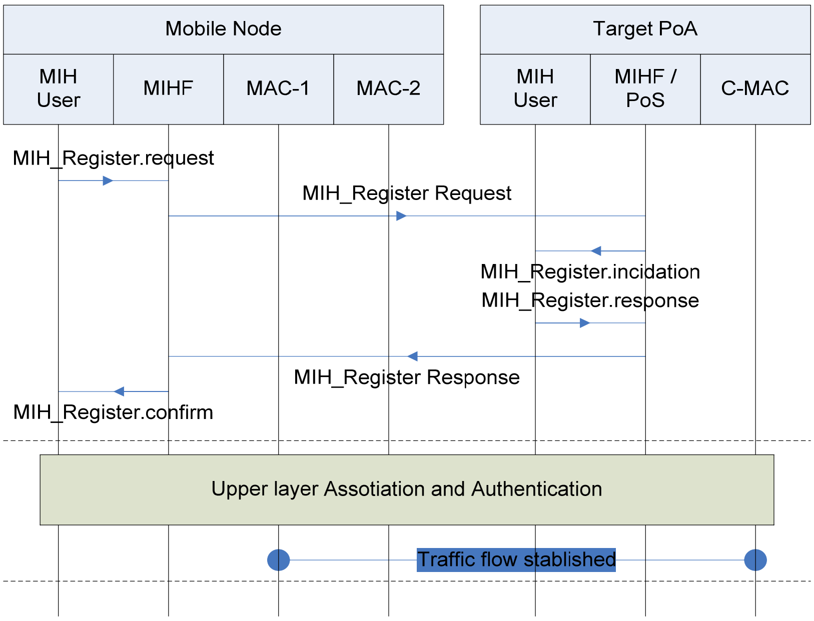


Figure 3.6: MIH Registry and upper layer registry

Last, the authentication and association procedures at upper layers are done, establishing a new traffic flow of data between the involved entities. Now, the MN is connected to the serving network via the current PoS and it has access to the MIIS, in case it exists.

3.3.1.3. Service degradation detected. Information service query and target PoA decision

Once connected to a PoA, the service can experience degradation with time. Once this degradation has crossed determined thresholds (they are not under the scope of IEEE 802.21 std.), the MIH users activate the corresponding mechanisms to restore an acceptable service level. The measurement of this degradation can be done in terms of several metrics, although the principal factor involved is the loss of signal strength.

The received signal strength can be determined in many different ways. One idea is getting it from the information included in the received packets at physical level. For instance, an IEEE 802.11 frame includes a field in its PLCP frame filled by the NIC of the receiver, which indicates the metrics of the received signal strength, noise and so its SNR. This is the way it has been implemented in the current work.

When a threshold has been crossed, the triggered indication called is MIH_Link_Parameters_Report, and it can be set for many different thresholds indicating which actions must be carried out in each different case.

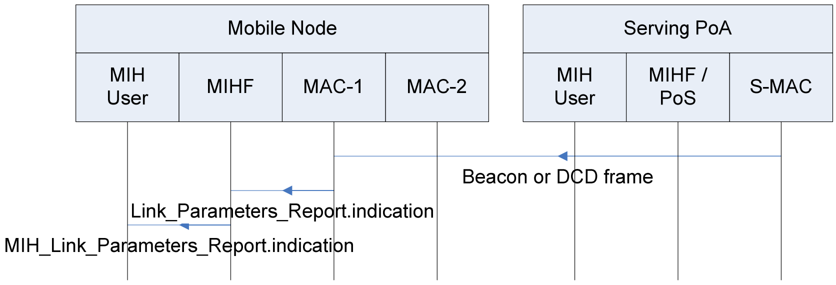


Figure 3.7: Service degrading detected

Once the first threshold has been crossed, the quality of the received signal can still be considered acceptable, but some actions must be taken before losing connection. The MN queries information about neighboring networks by sending a MIH_Get_Information request message to the MIIS. It responds with a MIH_Get_Information response message. This information query can be attempted as soon as the MN attaches to a new serving network or periodically for refreshing the information. The MIIS is commonly acceded through the current serving PoA, and is allocated somewhere in the network, not necessarily in the PoS or the local network.

Once the MIIS is queried, the MN listens to the media to detect the presence of the candidate PoAs given in the response message. A MIH_Link_Detected indication is triggered for each of them, carrying basic information about those PoAs and the networks they depend on, as commented previously. The next step is preparing handover, which will be explained in the following subsection.

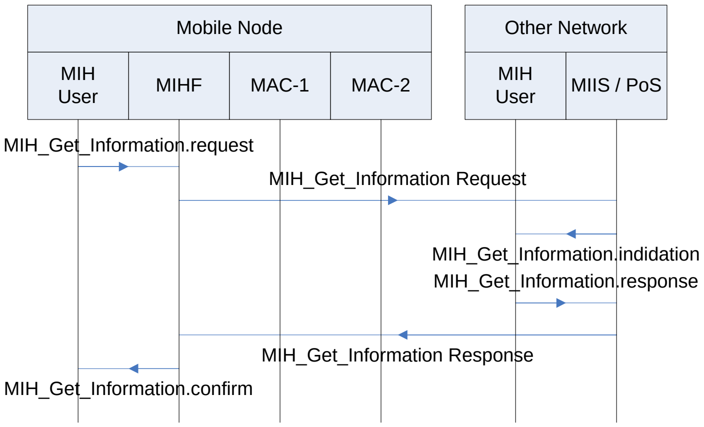

Mobile Node
MIH User
MIHF
MAC-1
MAC-2
Other Network
MIH User
MIIS / PoS
MIH_Get_Information.request
MIH_Get_Information Request
MIH_Get_Information.indidation
MIH_Get_Information.response
MIH_Get_Information Response
MIH_Get_Information.confirm

Figure 3.8: Information Service (MIIS) Query

3.3.1.4. Handover candidate query and query resources.

The MN triggers a mobile-initiated handover by sending a MIH_MN_HO_Candidate_Query request message to the Serving PoA. This request contains the information of potential candidate networks.

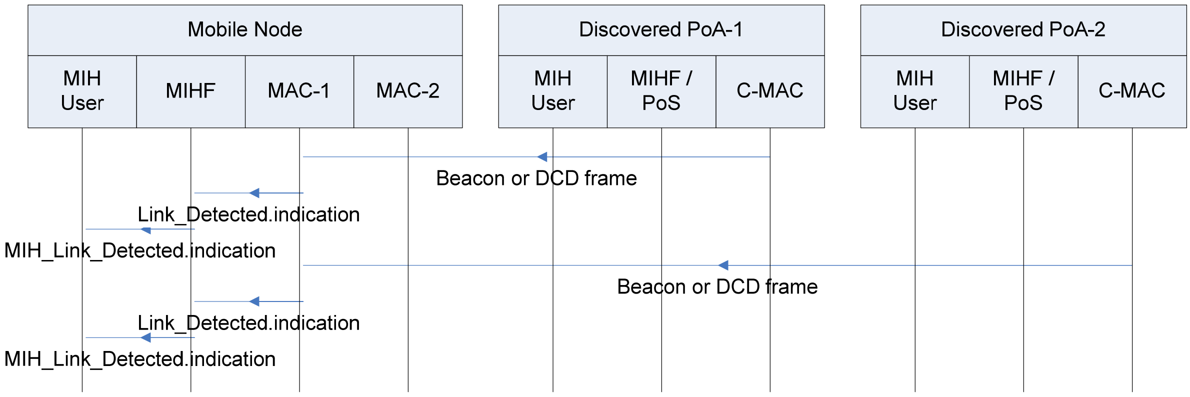


Figure 3.9: MIH PoA discovery after information service query

The Serving PoS sends the MIH_N2N_HO_Query_Resource request messages to the informed Candidate PoSs (can be more than one) in order to query the availability of the resources at the candidate networks. The Candidate PoS responds by sending the MIH_N2N_HO_Query_Resource response message to the Serving PoS. The Serving PoS in turn sends MIH_MN_HO_Candidate_Query response message to the MN. Finally, the MN decides the handover target based on the result of query about resource availability at the candidate networks.

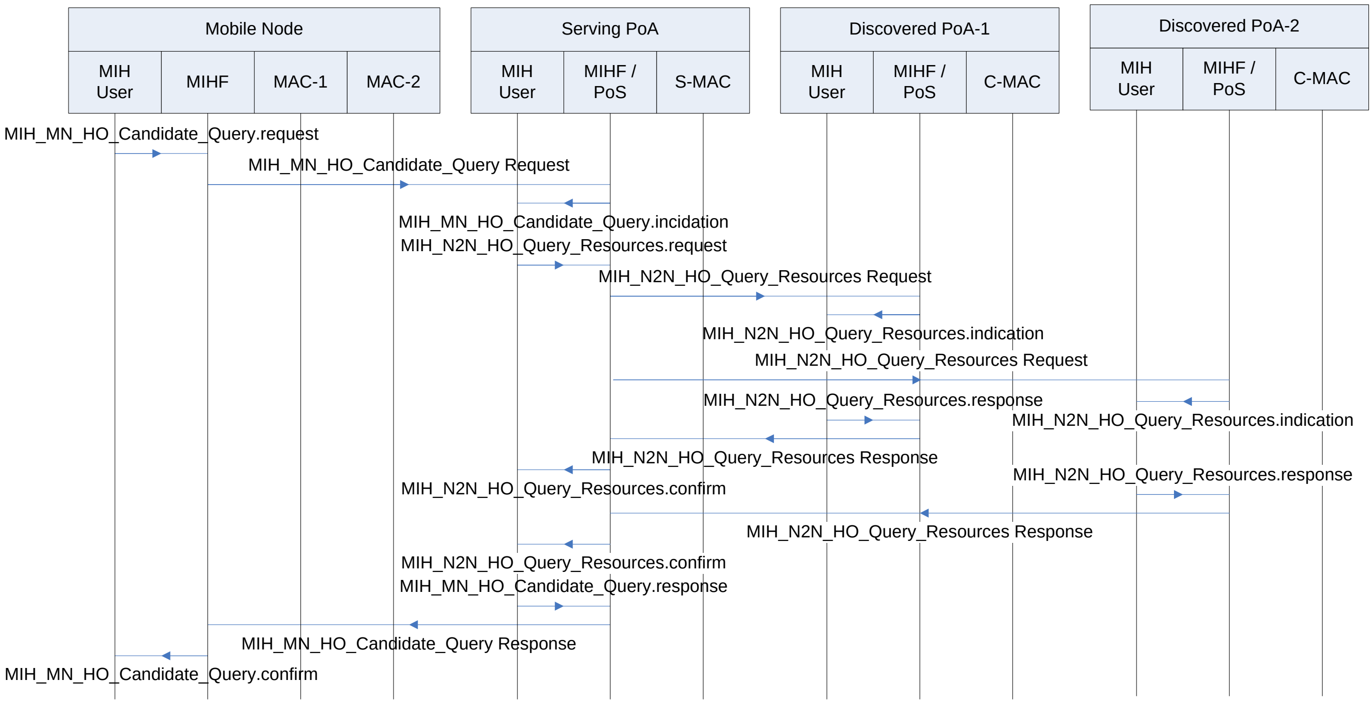

Mobile Node
MIH User
MIHF
MAC-1
MAC-2
Serving PoA
MIH User
MIHF / PoS
S-MAC
Discovered PoA-1
MIH User
MIHF / PoS
C-MAC
Discovered PoA-2
MIH User
MIHF / PoS
C-MAC
MIH_MN_HO_Candidate_Query.request
MIH_MN_HO_Candidate_Query Request
MIH_MN_HO_Candidate_Query.incidation
MIH_N2N_HO_Query_Resources.request
MIH_N2N_HO_Query_Resources Request
MIH_N2N_HO_Query_Resources.indication
MIH_N2N_HO_Query_Resources Request
MIH_N2N_HO_Query_Resources.response
MIH_N2N_HO_Query_Resources.indication
MIH_N2N_HO_Query_Resources Response
MIH_N2N_HO_Query_Resources.confirm
MIH_N2N_HO_Query_Resources.response
MIH_N2N_HO_Query_Resources Response
MIH_N2N_HO_Query_Resources.confirm
MIH_MN_HO_Candidate_Query.response
MIH_MN_HO_Candidate_Query Response
MIH_MN_HO_Candidate_Query.confirm

Figure 3.10: Handover candidate query and query resources

3.3.1.5. Target decision. Handover resource reservation

The MN decides on the target of the handover and notifies the Serving PoS of the decided target network information by sending the MIH_MN_HO_Commit request message. This message is propagated through the network from the serving PoA to the target PoA via MIH_N2N_HO_Commit request message. The response is sent back to the MN using the response messages of the previous ones.

The Serving PoS sends the MIH_N2N_HO_Commit request message to the Target PoS to request resource preparation at the target network. The Target PoS replies to the Serving PoS with the result of the resource preparation by sending MIH_N2N_HO_Commit response message. Figure 3.11 provides a clear overview of this message exchange.

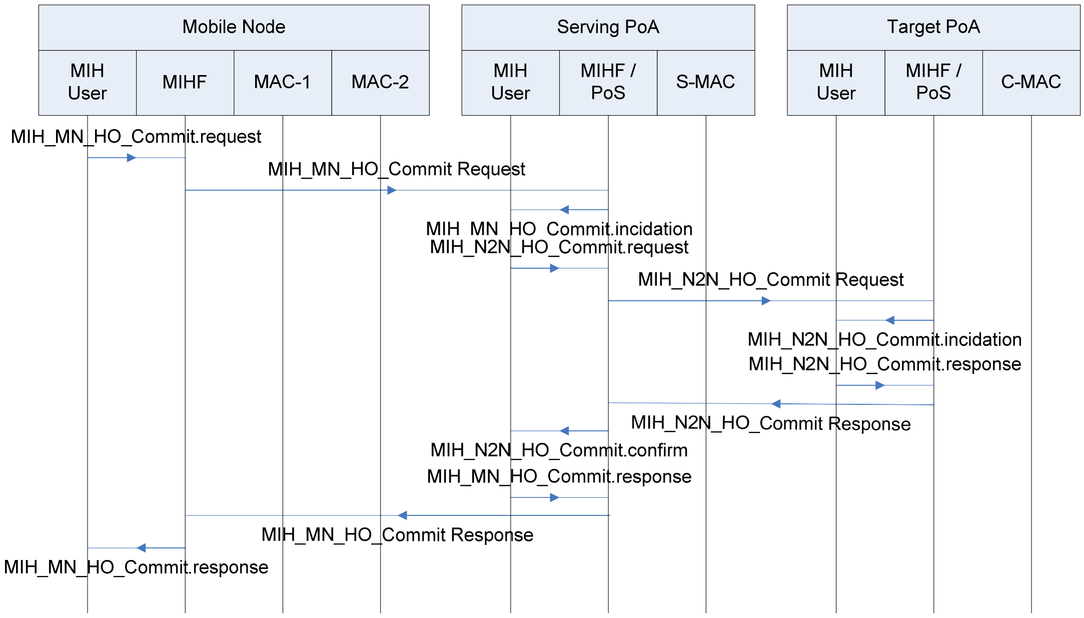

Mobile Node
MIH User
MIHF
MAC-1
MAC-2
Serving PoA
MIH User
MIHF / PoS
S-MAC
Target PoA
MIH User
MIHF / PoS
C-MAC
MIH_MN_HO_Commit.request
MIH_MN_HO_Commit Request
MIH_MN_HO_Commit.incidation
MIH_N2N_HO_Commit.request
MIH_N2N_HO_Commit Request
MIH_N2N_HO_Commit.incidation
MIH_N2N_HO_Commit.response
MIH_N2N_HO_Commit Response
MIH_N2N_HO_Commit.confirm
MIH_MN_HO_Commit.response
MIH_MN_HO_Commit Response
MIH_MN_HO_Commit.response

Figure 3.11: PoA resource reservation

3.3.1.6. Wait for RSSI changes and take proper actions

After resource reservation, the MN waits for changes in signal strength prior to take more actions. If the received quality falls under the next threshold, it means that although the coverage provided by the PoA is still acceptable, a significant loss of quality is imminent. Actions to complete handover procedure must be taken.

Along these lines, the Mobile Node commits a link switch to the target network interface by invoking the MIH_Link_Actions request primitive in order to bring up its alternative radio interface. The MN performs handover to the specified network type and PoA by the MIH_Link_Actions request primitive.

The new L2 connection is established and the required mobility management protocol procedures are carried out at upper layers between the MN and the target network.

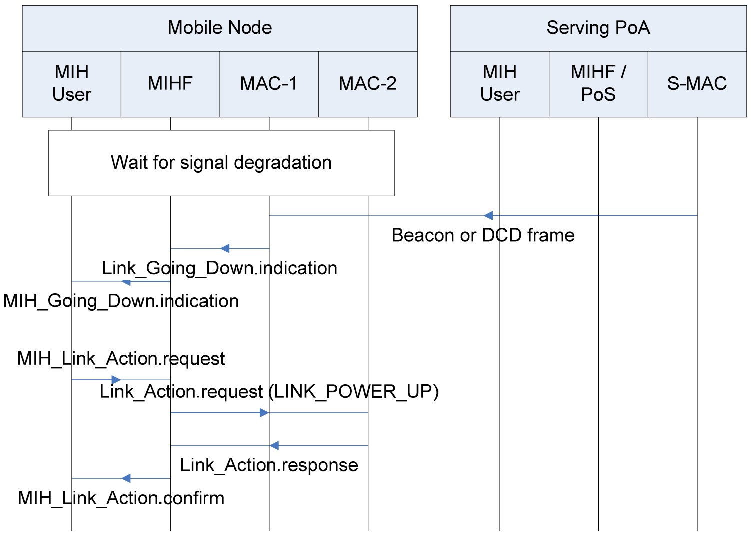


Figure 3.12: Service degradation and alternative MAC power up

3.3.1.7. New link layer connection and PoA attachment

Once the alternative link in the MN has been brought up and it is received enough signal strength from the target PoA, a MIH_Link_Up indication is triggered in the MN. From this moment on, the MIH Registry procedure between MN and target PoA can take place. Again, whether this procedure takes place at this moment or not is not specified anywhere in IEEE 802.21 standard pages, although it seems proper to place it just here. Authentication

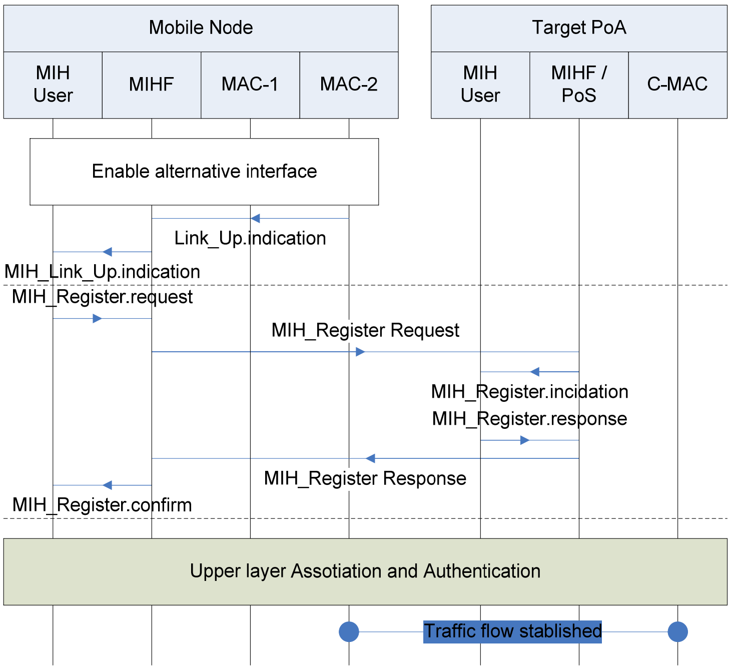


Figure 3.13: MIH Registry and upper layer registry before resource reservation

and association at higher protocol layers is done after MIH Registry.

After upper layer association procedures, a new data flow has been established between target PoA and the alternative interface of the MN. The old interface providing data transport service can be disabled because it has ceased to transmit and receive information. The MN proceeds triggering a MIH_Link_Action primitive to its own interface, indicating it has to be switched off.

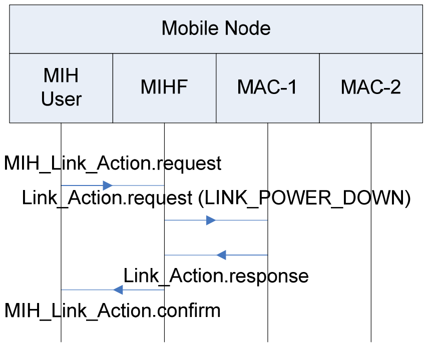


Figure 3.14: Link power down to old MAC interface

3.3.1.8. Handover completion and detachment from old PoA

The MN sends a MIH_MN_HO_Complete request message to the Target PoA. The Target PoA sends a MIH_N2N_HO_Complete request message to the previous Serving PoA to release resources allocated for the MN. After identifying that the resource is successfully released, the Target PoS sends a MIH_MN_HO_Complete response message to the MN. As it is showed in figure 3.15, the message exchange flow is the same as when

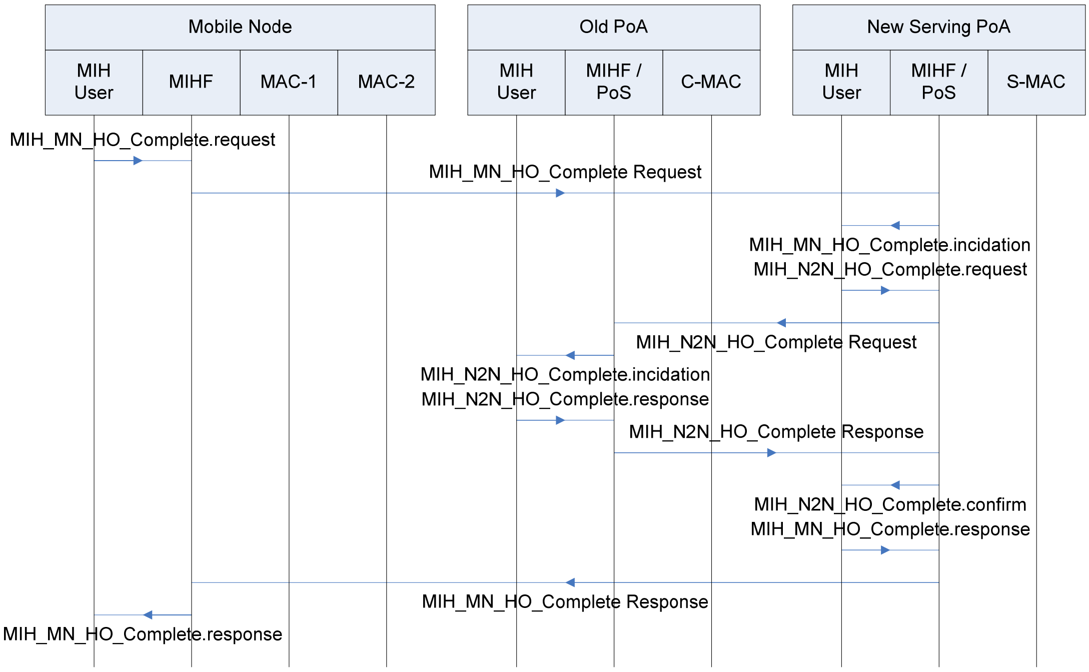

Mobile Node
MIH User
MIHF
MAC-1
MAC-2
Old PoA
MIH User
MIHF / PoS
C-MAC
New Serving PoA
MIH User
MIHF / PoS
S-MAC
MIH_MN_HO_Complete.request
MIH_MN_HO_Complete Request
MIH_MN_HO_Complete.incidation
MIH_N2N_HO_Complete.request
MIH_N2N_HO_Complete Request
MIH_N2N_HO_Complete.incidation
MIH_N2N_HO_Complete.response
MIH_N2N_HO_Complete Response
MIH_N2N_HO_Complete.confirm
MIH_MN_HO_Complete.response
MIH_MN_HO_Complete Response
MIH_MN_HO_Complete.response

Figure 3.15: MIH handover completion

performing resource reservation, with the difference that the old and new PoA have exchanged their roles.

Finally, a MIH UnRegistry process should take place between MN and the old PoA that provided services. The definition of when, where or how to proceed this step is out of the scope of IEEE 802.21 std.

3.3.2. Test bed scenario

Arising from the commented premises, a scenario has been implemented in order to develop the capabilities of an IEEE 802.21 based system. Due to the broad scope of this framework, a set of particular features have been chosen in order to establish an initial development.

First, the development has been centered in handover across different WLAN networks. Vertical or horizontal handovers between other wired or wireless technologies such as WiMAX or 3GPP/3GPP2 are not considered. In this scenario, handovers across different BSSs and even across ESSs are possible under a link layer design or upper. The infrastructure required in a WLAN deployment is simpler than in other technologies and does not depend on a part of the public infrastructure.

Next, some restrictions have been applied in order to ease the implementation of a suitable test-bed. One is the allocation of the MIIS. Its regarding data has been pushed to the PoAs providing duality in their functions: on one hand, they will act as a permanent link to the wired networks providing the basic services to keep a connection; on the other hand, they will also provide a direct access to the Information Service without the need of establishing a dialog with another entity in the network to gather information. This design condition drives to an easier implementation of the model but has the disadvantage of being less realistic. MIIS needs being replicated in each PoA and updating its information becomes more difficult.

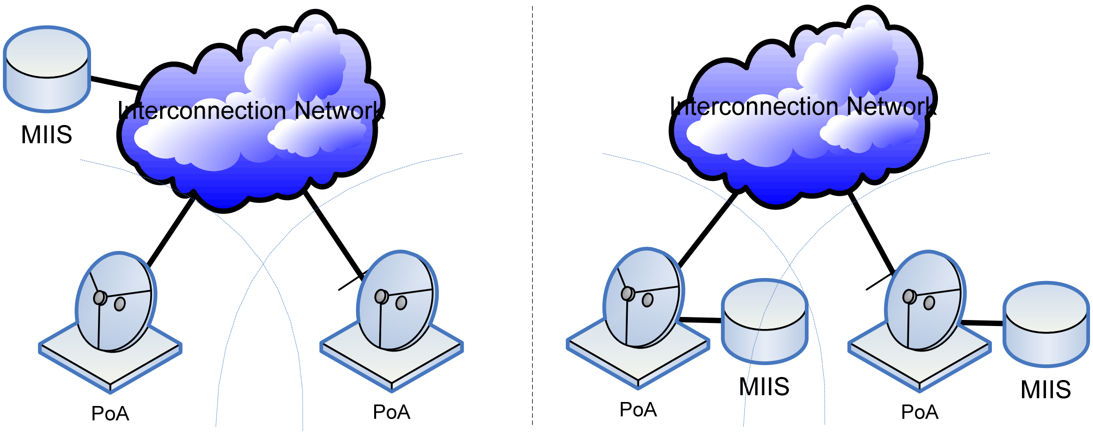


Figure 3.16: Left: general allocation of MIIS. Right: dual PoA/MIIS allocation

Moreover, some procedures for MIH discovery are not defined under the IEEE 802.21 standard. Prior to the publication of this final document and after that, many papers and documents have merged form different organisms with the aim of proposing different methods and protocols to discover MIH entities. In the current test-bed about to be defined, the MN has the need of hearing the surrounding candidate hotspots and determining whether or not they are MIH entities. Again, in order to ease the development, no extra discovery protocols have been added. Instead, it has been decided to introduce the needed information into some WLAN management frames, particularly into beacon frames. Those frames are periodically broadcasted by hotspots and can be customized allowing MIH presence discovery.

Note that, as shown in the previous clauses, even the PoSs in a determined area need to exchange information themselves. So, they also need to be discovered. It has been seen that, two different PoAs, namely Serving PoA and Target PoA, need to begin a communication in order to proceed handover. They can be allocated in different placements that do not necessarily belong to the same network. Again, a protocol for this purpose can be useful, but does not fall within the scope of the current project. In order to solve this problem, an additional data base has been added to the PoAs providing them the required information to route packets to the needed source and destination PoAs. As well as it happened with MIIS, it has the inconvenient of being static information and keeping them updated can cause some problem.

The IEEE 802.21 standard designs two protocols to carry data frames over the network: TCP and UDP. When the different PoSs need to establish a communication, they will use one of them. In the current scenario, two PoSs can exchange handover information using a TCP protocol. Choosing this protocol is due to terms of reliability. A TCP protocol guarantees the reception of the information end to end, whereas an UDP protocol does not guarantee its reception, although it does not need establishing a connection. Extra methods providing reliability should be implemented using an UDP connection. The private packet network provided by the laboratory has been chosen to act as the core wired network interconnecting PoSs.

The implementation of the MIIS also needs to be commented. The IEEE 802.21 standard defines two methods for representing Information Elements: binary representation, and RDF representation. Two query methods are also defined: TLV method for binary representation and SPARQL for RDF representation. A binary representation has been chosen in order to take profit of tools previously designed. Remember that, messages exchanged between MIHF entities are composed of data coded into TLV format, so some mechanisms can be re-used.

Note also that the final standard publication includes support for Management Information Base (MIB) implementation. This MIB provides the required and structured information for MIIS implementation. Although it is standard, the work in the current project started before the final standard was published and so this MIB was not provided. A particular data base for MIIS has been implemented instead.

Finally, it is mandatory to highlight that the resources used to implement this test-bed are basically multimedia computers, equipped with the required software and hardware elements which will be overviewed in the following subsections.

3.3.2.1. Mobile node

In the current scenario, the MN is a completely functional notebook computer equipped with two WLAN interfaces. Both of them can be viewed from MIHF layers as homogeneous interfaces thanks to media dependent SAPs. The intention is providing a MIHF layer as neutral as possible in order to ease future enhancements of this one providing compatibility with other link layer technologies via new media dependent SAPs.

The following subsections provide an overview of the elements composing the MN. Aspects regarding to the chosen technologies are discussed in order to justify the final configuration. These lines will also provide a slight description of some candidate technologies that were discarded at the end.

3.3.2.1.1. Operating System

In a framework were aspects related to hardware handling are essential, it is very important to work with tools providing a full (or almost full) access to the features of the devices. Working with open systems such as Linux or Unix-alike does not guarantee access to a full equipped documentation but, in almost all of the cases, the code is open to everybody. Moreover, the GPL license allows the developer to get the needed code, modifying it and even to redistribute it. This is the main reason why a Unix-alike OS has been chosen.

Although the aim is designing a prototype, the chosen OS had to be stable in order to provide the basis of a complex design. The distribution Debian GNU/Linux (a.k.a. Debian) is updated every two years with the most stable software by which the kernel is composed. Besides, the included applications in this distribution's user space are also updated in this period of time. This makes Debian one of the most stable OSs available.

In the particular case of the Mobile Node, the functionalities needed to develop directly depend on the drivers controlling the devices behavior. In order to ease the implementation, a homogeneous handling of these wireless interfaces is convenient. Nevertheless, when the implementation phase started the mac80211 stack described in the previous chapter (running on Linux kernel 2.6.24) did not include the particular wireless cards and drivers used. So, the tools given to handle these interfaces provided a different set of features, and had to be treated individually. The need to be homogeneous drove to the update of the kernel to its latest versions. Using a stable kernel became secondary, so even testing versions were used in order to accomplish this target. Finally, the release of the kernel 2.6.31 provided the needed enhancements to deal equally with the used wireless.

Given these conditions, the Linux distribution changed to be Ubuntu, which is Debian based and provides a more user friendly environment. The version of this distribution released in October of 2009 includes the kernel version 2.6.31, so it fullfils the requirements in the MN's side. In order to test some system capabilities, the OS used is Debian 5.0 (also with an update of the commented kernel version). It solves some conflicts Ubuntu that occur when the computer notebook tries to manage more than one wireless NIC at the same time.

3.3.2.1.2. Wireless Network Interface Controller

The election of a wireless NIC, as well as its controller, is tightly bounded to the requirements requested and the operating system allowing its presence. In this context, these NICs are supposed to be capable of promiscuously capturing wireless traffic and to inject packets to the network in parallel to an IEEE 802.11 development.

Under a Linux framework, two options were considered to be acceptable. One option was working with Intersil/Conexant devices, which allowed using HostAP drivers. The other suitable option was using an Atheros NIC, a device known for having a better sensibility than most of the commercial NICs. The final decision was using an Atheros device because it provided the required tools to perform the needed actions. The model used is equipped with a chipset of the AR5000 family, which is compatible with IEEE 802.11 b/g physical specifications.

The commented options had both mini PCI connection, and could not be connected to the laptop at the same time. So, an alternative had to be found in order to have another interface to enable seamless handovers. Using USB devices was an option, but drivers fitting these requirements were not developed yet for the existing technologies. The option was using a not expensive card providing compatibility with IEEE 802.11 protocol at least. This interface would act as secondary and would not exchange management frames other than IEEE 802.11 ones. A device of Realtek called *D-Link System DWA-140 802.11n Adapter* fulfills those requirements and became the secondary interface. Later, the mac80211 stack of Linux kernel 2.6.31 provided almost full support (although experimental) to this device, so it was provided with the same functionalities than the Atheros device. From this moment on, both devices can act under the same conditions allowing both IEEE 802.11 protocol and alternative parallel protocols, in this case a L2 implementation of IEEE 802.21 protocol. The Realtek device operates in IEEE 802.11g/n physical modes, but the regarding features are not under the scope of this implementation and so they are not considered.

3.3.2.1.3. Drivers and tools.

The election of a device driver is parallel to the election of the network card, but some considerations need to be taken in account. The first hardware selected was the Atheros NIC. The driver working under Unix-alike distributions was called MADWiFi and had some particular features. One of them was that it did not provide a completely open source driver. A part regarding to the hardware called Hardware Abstraction Layer (HAL), was still unreleased as free source, and so it forced programmers to do reverse engineering in order to develop this driver. Developments were costly and slow, until Atheros freed this part of the code. Since then, two new branches of the driver were developed: *ath5k* and *ath9k*. The first one is a completely open source driver used for chipsets of AR5000 family, which are compatible with IEEE 802.11b/g physical standards. The second one is designed for devices equipped with chipsets of AR9000 family, compatible with IEEE 802.11n physical standards and also full open source code. Since the Linux kernel 2.6.29, *ath5k* is included into the list of drivers supported under mac80211 stack and can be handled as a homogeneous interface.

The classical MADWiFi driver is currently deprecated, and so are its related tools. The MADWiFi tools provided a broad set of functionalities implemented via *ioctls*. Those were open source, and were useful to develop the required functionalities for this system but had the inconvenient of being useful only for a particular set of network cards. The apparition of *ath5k* driver and its almost immediate inclusion into mac80211 stack eased the development, together with the use of the commented *iw* user space tool and even with Wireless-tools. Both options, a MADWiFi driver or an *ath5k* driver, are considered in the development.

The remaining wireless interface was originally introduced just to provide IEEE 802.11 capabilities and so to allow developing seamless handovers. The Linux version of the driver could be obtained from the official Realtek web page, and the corresponding kernel module received the name of *rt2870sta*. When the driver was introduced into mac80211 stack, it included features to allow configuring multiple virtual interfaces in monitor mode under the capabilities provided by *iw* tool. Both options are considered in the current implementation.

3.3.2.2. Point Of Attachment

The PoA is the network element which bounds the Mobile Node to the wired network providing a set of MIH services. In a conventional WLAN development, this role can be adapted by an AP or hotspot bounded to an element capable to route packets to the wired core network.

In an IEEE 802.21 test-bed scenario, a PoA can be built from a personal laptop with both a wired and wireless NIC. The first one is required to provide connectivity to the wired packet network, whereas the second one has to be chosen in order to provide complete AP capabilities. In a Linux framework, this is done using Hostapd, a user space daemon which allows setting a wireless NIC to act as a master element in a WLAN BSS. The election of the basic elements of the PoA will be discussed in the following subsections.

3.3.2.2.1. Operating System

Again, building a base framework as open as possible was one of the most important design conditions for the development of the PoA. The basis is a Unix-alike system, capable to support a wireless NIC and allowing it to be configured as an AP. Moreover, some features of this AP should be modifiable in order to enhance it becoming a PoA. With these premises, a research is done to find a proper OS.

The first stage of the search is focused in founding a NIC that could act as an AP. Unfortunately, when the work started mac80211 stack was still underdeveloped and devices providing this feature were just a few, difficult to find and to configure, and did not follow standard configuration patterns (were not under the influence of this stack). This leads the development to use the latest version of the Linux kernel's development branch. The first kernel that allowed the required functionalities was the 2.6.30-rc4 version from the Wireless-testing branch. It included drivers for the serving NIC under the interface of mac80211, and allowed configuring the card as an AP using the user-space daemon Hostapd.

The election of the distribution is done in terms of providing a user friendly environment. Debian or Ubuntu are both proper candidates, and choosing one or other does not make a relevant difference. Finally, Debian is chosen for reasons of popularity and so it is installed in the laptops composing the test-bed, with almost an updated 2.6.30-rc4 kernel version. The bigger difficulties are encountered when electing the wireless NIC.

3.3.2.2.2. Wireless Network Interface Controller

The search for the proper NIC in the role of the AP has been a complex task. It has involved testing many different cards, compiling and running different kernels for the OS, and what is more, modifying existing software. The first cards tested did not seem they could work even before being connected to the laptop. They were some kind of hybrid between PCI and PCMCIA card. In fact, they were a PCMCIA card mounted over a PCI adapter. The design is known as a PLX device, and is due to the need of moving the oldest NICs to laptop computers. The particular devices were provided by Broadcom, and worked under the IEEE 802.11b physical standard.

Soon it has been seen that these cards give several problems with the current OSs because they are difficult to install and to maintain configured even as stations. So, configuring them as APs was a question not even considered. Unfortunately, although the next NIC used seemed to be a proper evolution of the previous ones, it has not provided the desired functionalities. The card under test is a PCI device with an Intersil Prism/Conexant 2/2.5/3 chipset and a monopole antenna. It works well as a station, but does not allow configuring it like an AP. Using Wireless-tools do not succeed, not even Hostapd gives a chance to this card for becoming AP.

From this moment on, the decision of using NICs that run under mac80211 stack is taken. A model that seemed functional and quite cheap is chosen this time. Its developer is D-Link, and the device is a DWL g122 model with USB connection. The laptop is equipped with all the needed elements: the last testing kernel, a wireless NIC that should to work as an AP thanks to Hostapd, and the rest of needed user space entities such as CRDA and *iw*. Once again, the AP cannot be mounted. Running Hostapd does not allow this configuration. The reason why resides in the version of the included chipset. An AP is configurable under those conditions, but not with the first revision of this chipset (the second version is required). So, another device has to be found.

At the end, the manufacturer Asus provides the solution through a Broadcom chipset. Its NIC is fully functional under the testing kernel 2.6.30-rc4 and can develop the role of an AP with Hostapd. The model used is *WL-138G V2* and it is installed in every laptop of the test-bed.

3.3.2.2.3. Drivers and tools

Once the oldest NICs were tested and discarded, the efforts in finding a way to create an AP whose capabilities could be enhanced had to be increased. The solution has to be related to the new emerging mac80211 stack.

The first candidate technology, an Asus *DWL g122 d-link*, run over the mac80211 API using a *p54* driver, the new evolved branch of the classical drivers used for Prism cards (often HostAP drivers). The solution was fully functional for a device running in STA mode, but the features of the NIC did not allow configuring Hostapd to make it work as an AP. The error was using a deprecated revision of the chipset, which needed to be Rev. 2 to permit being set as an AP.

Instead of using a proper revision of Asus card, the solution adopted was using another reliable wireless NICs brand: Broadcom had a device fulfilling the requirements. The card 3Com was also interfaced by mac80211 stack and used the nl80211 netlinks to communicate with the kernel. The particular driver used is *b43*, and is worth for a broad variety of Broadcom NICs.

This time, the hardware allows setting the wireless device as an AP, and permits modifying and introducing new elements to convert it in a fully functional PoA. Further details of this performance through Hostapd are given in the next sections.

3.3.2.3. Schema of the resulting scenario

To put in practice the stated premises, a test-bed scenario had to be defined. It is known how a MN and a PoA have to be performed. It also has been decided that the allocation of the MIIS would be in each PoA, instead of being an external entity somewhere in the network. Moreover, it is known that to test if the handover capabilities run properly and even if they are worth, almost two PoAs are needed.

With these conditions, a framework is proposed in order to develop a system MIH alike. The figure 3.17 shows the chosen topology and the elements taking part on it.

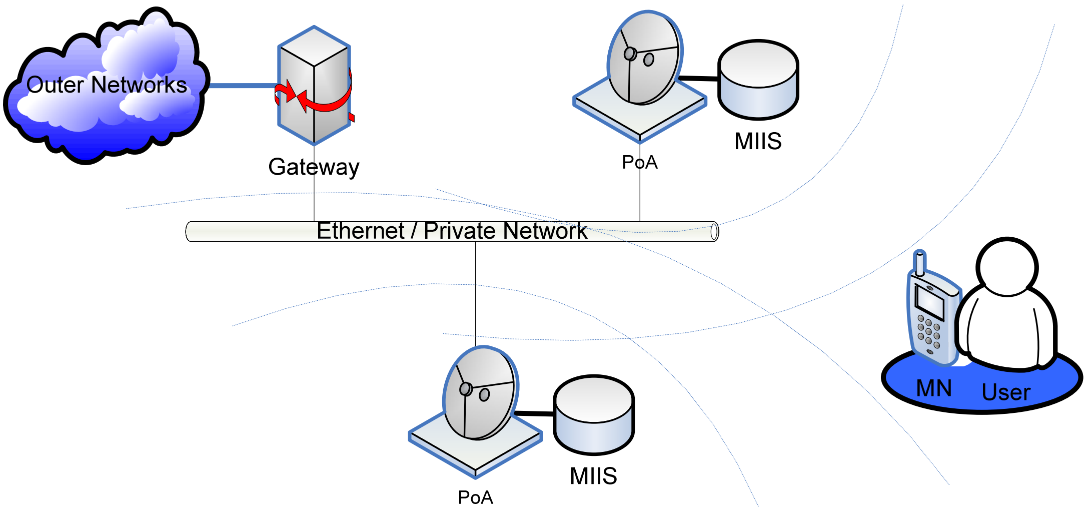


Figure 3.17: Schema of the test-bed topology

First, a MN running a daemon application allows a transparent MIH handovers in terms of link layer and network layer connectivity. Just a MN element is indicated in a MN initiated handover framework. Several MNs with needs of handover could lead to network overcharges at certain placements. Some of these features will be discussed later.

Then, two PoAs with similar MIH capabilities are placed on the test-bed. Each associated MIIS has the same static information about the environment. Both PoAs are capable to handle a MIH protocol of communications in order to provide MIH services to the MN.

Last, an interconnection for these PoAs is needed. In this context, a simple switch or hub linking them is enough, even a single network cable would be sufficient. Instead, the wired connection provided by the development laboratory, basically an Ethernet network with internet access, has been used. This eases allocating new elements in future test-bed enhancements.

3.4. Application definition

Once the framework has been defined, the applications handling the management system must be developed. These applications are indented to be working in background, transparent to the user. This kind of applications receives the designation of daemon [33]. Generally, daemons do not use a direct interface with the user. This time, the prototype will be provided of a console that will monitor key features of the MIH protocol.

Two daemons are needed: one running at the MN, and another one being executed at the PoA. Both applications will allow a MIH link layer communication in parallel to an IEEE 802.11 protocol. The following sections give key points of the decisions taken in the development of each network element, as well as some points on their implementation.

3.4.1. Common features

3.4.1.1. A L2 message exchange

The MIH standard defines a protocol of layer 2.5. It means that suitable implementations of it can take elements from both layers 2 and 3, generally running in parallel to them. The idea is to provide an upper layer transparent from the point of view of the different link layer technologies below, facilitating standard handover procedures. In practice, it is not easy to achieve because the developer has to deal with low level features.

Some standardization bodies, such as IETF (Internet Engineering Task Force), provide application level solutions in order to give an alternative functional view of the MIH Function. Many papers contribute to develop the needed network elements and procedures to manage application level entities that perform the infrastructure of a MIH compatible network.

Both ways to focus the development (parallel to layers 2 and 3 or application layer) have their own pros and cons. On one hand, the application layer option has an easier performance due to the encapsulation of the information. A simple internet socket, for instance provided by standard UNIX libraries, can be used to inject to the desired end point application the information needed to be sent. The procedures in layers below are irrelevant for the developer. A L2 development requires several efforts to provide a reliable communication, needing to implement transmission and retransmission procedures, state machines, associated counters, tools to compose and extract information from the received binary packets, and many more features derived from physical and link layers.

On the other hand, a MIH Function implemented at higher layers cannot develop an optimal performance of the features proposed in IEEE 802.21 standard. For example, the message exchange cannot take place if the MN is not previously authenticated and associated at network level. So, in these terms, to start scanning the network for suitable MIH compatible services the MN needs to have a complete network connection, which does not seem to make much sense. The services provided for network discovery are not functional in this context, and the responsibility for bootstrapping relies on the IEEE 802.11 protocol, which does not guarantee getting attached to a MIH compatible point of service. A L2 development can perform a MIH Discovery without being authenticated and associated with an AP.

In order to follow the standard document guidelines, the option of a link layer development has been chosen. Most of the efforts are put to provide a reliable and error free communication within the wireless part.

3.4.1.1.1. Packetspammer reference application [34]

Packetspammer is a free open source application designed to demonstrate mac80211 monitor mode packet injection. This application is written in C code and uses the *pcap* library to develop communication capabilities. It is also capable to handle a flexible radio-tap header making it suitable for many different technologies.

Its operation is simple: the program is installed at two different peers capable to be set in monitor mode. Once one of them is executed, it starts transmitting numbered packets that can be seen in a local console. Once the other peer runs the application, it also starts transmitting packets and listens to the media to hear the transmitted packets from the other peer. Once the peers listen each others' packets, they show by console each received packet.

The sources composing this application provide the basis of the radio communication of the designed program. It has provided the required set of instructions from the *pcap* library commented in previous sections, and also the format of the radio-tap header, amongst other relevant things.

3.4.1.1.2. Monitor mode and pcap for unauthenticated custom message exchange

A problem designing a parallel protocol to IEEE 802.11 appears when trying to inject or even to sniff a set of frames. Fortunately, the latest implementations of the *pcap* library provide facilities to inject packets through interfaces working in monitor mode, as it has been stated before. A station running in monitor mode is able to capture all kind of packets from the shared media, and even to send custom frames under a determined format of physical header. Moreover, the mac80211 subsystem provides a set of flags which allow filtering different type of MAC frames in monitor mode. So, to allow a basic wireless packet exchange, MIH custom frames must be carried into IEEE 802.11 MAC frames. It can be done by properly setting some fields into this header.

There exist our kinds of frames defined under the IEEE 802.11 standard, each of them represented with two bits into its correspondent header: management, control, data and reserved frames. The standard also defines the following states for a station into a wireless network:

* State 1: DEAUTHENTICATED and DISASSOCIATED: just receive frames of class 1.

* State 2: AUTHENTICATED and DISASSOCIATED: receive frames from classes 1 and 2.

* State 3: AUTHENTICATED and ASSOCIATED: receive frames from classes 1, 2 and 3.

In a 2.5 layer design, it is needed a link layer as permissible as possible. Admission and control mechanisms are implemented over this layer for a parallel design to the IEEE 802.11 stack. The simplest state in this architecture is state 1, so the data frames composing it must be reviewed (class 1):

* Control frames: RTS, CTS, ACK, CF-End+ACK y CF,

* Management frames: Probe request/response, beacon, authentication, deauthentication and ATIM,

* Data frames: data with ToDS and FromDs control fields into its MAC header set to false (both of them).

It is required to maintain a MAC header and a padding field filled with the IEEE 802.21 frame. So, the frame type used is the one defined at the third point: a data frame type. The codification of the frame type field to IEEE 802.11 header is set as follows:

* Version: 00 - 2 bits, version 0 of IEEE 802.11 protocol.

* Type: 10 - 2 bits, frame whose type is data frame, or in other words, which carries data within.

* Subtype : 0000 - 4 bits, just data.

The resulting byte coded into hexadecimal format is a 0x08. By this way, a peer set to monitor mode can sniff and even send wireless custom frames. The tool *Wireshark* has been intensively used to determine the correct combination of candidate bytes that could perform this type and subtype field in order to provide a wireless communication engine.

3.4.1.2. Received signal strength based handover

3.4.1.2.1. RSSI concept [35]

RSSI is a short for Received Signal Strength Indicator. It is a measure pattern for the level of signal received in a wireless peer. Generally, this value depends on the scale used by the device developer. For example, Atheros defines its own range of RSSI as follows: RSSI = [0..60], were maximum RSSI is RSSImax = 60. This range is directly related to the real measurement of the signal received. In this case, for Atheros devices this range of sensibility is: Rx Signal Power = [-95..-35] [dBm]. So, a direct relationship can be established: Rx Signal Power = RSSI – 95[dBm]

Other relevant developers such as Cisco also establish their own type of RSSI metrics, which in this case ranges between 0 at least and 100 at most. The conversion to real level of received signal is not as direct as for Atheros devices. Luckily, the current *iw* tool and the Wireless-tools provide a standard user representation of the signal level, giving this strength in [dBm] for each suitable wireless device.

3.4.1.2.2. Signal strength handover thresholds

Once attached to a PoA, it is assumed that the services and contents provided by the network are the desired by the user. The requirement for a change of serving PoA is due to a need for a better quality of the received signal strength. In this context, this handover is triggered by crossing different thresholds of the received signal level. For determining which the proper levels are, the specifications provided by Atheros have been borne in mind.

Atheros defines [36] the “routing threshold” as the 20% of RSSImax, so -83 [dBm] is the value that will trigger the beginning of the actions for a handover to take place. It is also defined the “clear channel threshold”, which is taken as the 10% of RSSImax. This value will trigger the beginning of the final actions for a handover procedure:

* Routing threshold (ROUTING_TH):	-73 [dBm]
* Clear channel threshold (CLEAR_CH_TH):	-89 [dBm]

The first threshold would trigger a query to the MIIS and would start with the resource reservation process for handover. The second threshold establishes a critical signal level received from the old PoA and triggers the actions to complete the association with the new PoA, besides of being disattached from the old PoA.

A third threshold is needed to determine whether a MN does not receive enough signal strength to be attached properly, and so the connection is lost. Atheros considers that this happens when the received signal strength falls under the 1%. To round off, this level has been established to be -94[dBm], which is still over the noise level.

For SNR calculations, a constant noise floor level can be considered if changes in the environmental temperature are irrelevant:

Noise is: N = kTB

B = 20MHz, is the bandwidth of the OFDM modulation;

k = 1,38x10e-23, is the universal Boltzmann constant;

T = 290k, is the environmental temperature, considered to be constant.

→So, N = -101,7 [dBm]

Considering that 5 [dBm] are added from the noise introduced at the amplifier chain:

→ N = -96 [dBm]

Other thresholds which Atheros considers with respect to the RSSI levels are the following:

* RSSI <= 10 (-85 [dBm]): weak level. A chip can decode low bit rate.

* RSSI >= 20 (-75 [dBm]): decent level.

* RSSI >= 40 (-55 [dBm]): very strong level. A chip can decode up to 108 Mbps.

3.4.1.2.3. Path for handover.

Some research groups have been evaluating the different possibilities for the actions to take regarding to the crossed thresholds. In particular, a group of researchers for IEEE published in 2007 a study [37] simulation proposing two operation models and providing different test results about successful handovers at layers 2 and 3.

Both operation models were triggered by two threshold levels. The differences between them were the actions taken when they were crossed. One threshold, the less critical, is defined as an association threshold, whereas the other one is set to be the handover threshold. The Operation Model A (OMA) works as follows: once approaching to the association threshold, the MN triggers a Link_Detected indication. It does not proceed with more actions until it reaches the handover threshold, where Link_Parameters_Report is triggered. After that, all the MN proceeds with the rest of the actions related to handover: handover initiation, L2 connection and L3 handover. The Operation Model B (OMB) advances most of the actions taken when crossing the first threshold: all the stated actions are taken until the establishment of the layer 2 connection (included), and crossing the handover threshold produces the L3 handover.

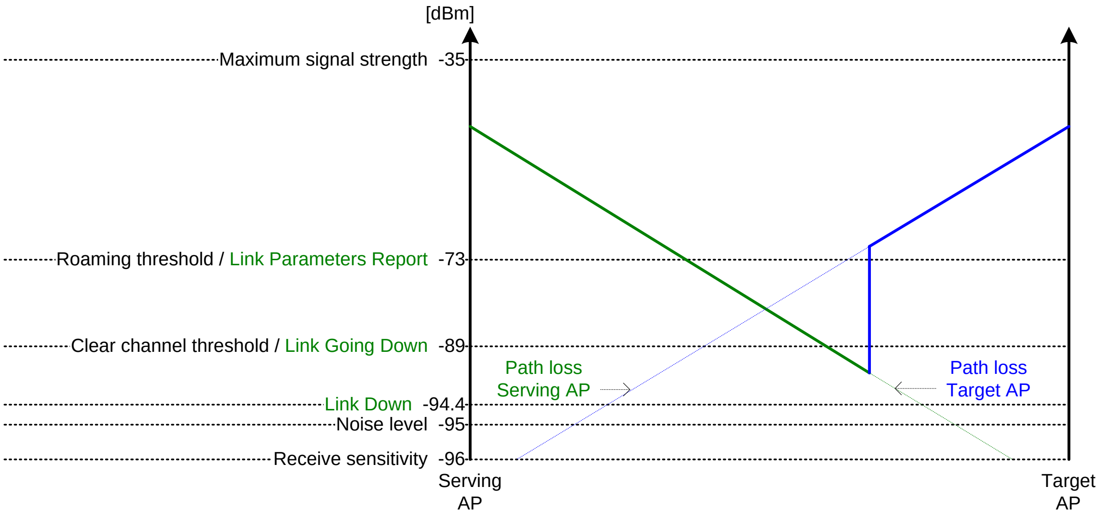


Figure 3.18: Received signal strength or path loss during a change of serving AP procedure

The proposed model for the current prototype has an OMB behavior alike. It also has two thresholds which trigger the actions to take, namely *Roaming threshold* and *Clear Channel threshold*, respectively. The main difference resides in the signal of reference taken: the OMA and OMB define their thresholds with respect to the target AP, and the proposed model for this work takes the attached AP as the reference. The first one is the less restrictive and triggers a Link_Parameters_Report indication. The actions taken are a query for information to the MIIS and handover initiation. The second threshold triggers Link_Going_Down indication and so establishes the new L2 connection and L3 authentication and association. The OMB has inspired this implementation guideline because it seems to provide a better performance in terms of successful handovers and its behavior is more in concordance with the informative flows defined in IEEE 802.21 standard.

The following table sums up the relationship between thresholds and its quantification levels:

Atheros thresholds	IEEE 802.21 thresholds	Atheros RSSI	Signal level	% Max signal level
Max signal level	-	60	-35	100
Roaming threshold	Link Parameters Report	12	-73	20
Clear channel threshold	Link Going Down	6	-89	10
-	Link Down	1	-94,4	1
Noise level	-	0	-95	0
Receive Sensitivity	-	-	-96	-

Table 3.1: signal level threshold look-up table

Note that there is defined a third threshold, namely *Link Down threshold*. A Link_Down indication is triggered into a MN when the peer ceases to receive the proper signal quality to keep the connection with its serving PoA. The handover services and procedures defined under the IEEE 802.21 standard try to avoid reaching this situation. Nevertheless, it can happen when the MN arrives to the end of a MIH coverage area. In this situation, a PoA/AP discovery process must be performed and so a hard handover, if possible, takes place. The choice of this level is done under hostile conditions of reception and results to be close to -94 or -95 [dBm], which is 1% of the maximum received signal strength.

3.4.1.2.4. Average received signal strength

The measure of the signal strength is taken from the information provided by the beacon frames from the attached PoA. The hardware in reception measures the signal level and completes a radio-tap header where this information is included. This information is afterwards extracted via *pcap* tools and used to determine average signal strength.

In an indoor environment, where the fading effects in the propagated signal are almost unpredictable, the level of received signal can change from sample to sample, leading the system to a wrong trigger activation of handover procedures. The principal idea is to develop a filter which suppresses the quick component of fading, and to establish the correspondent triggers respect the smoothed resulting wave form. Although dealing with this issue is out of the scope of this work, a little subsystem for averaging received signal strength has been implemented. The procedure is very simple: the application stores a set of samples of signal level from the received beacons and calculates its arithmetic average. Commonly the beacon interval is 100 ms, so the maximum number of samples per second is 10. The results with this averaging method could still be improved if a more plentiful flow of packets were used. Nevertheless, for a first approximation it will cover our necessities.

The implementation problems appear when trying to calculate this average. Converting –XX [dBm] signal levels to linear levels arises to a loss of precision for the data types handled by the math library in C/C++ language. This library provides logarithmic and exponential functions with double precision at most, which results in 64 bits of floating comma precision. This cipher is not enough; hence some tricky process has to be considered.

The idea is the following: working with [dBn] and nano Watts. Converting [dBm] to [dBn] or vice versa is the same as adding or subtracting 60 dB, respectively. So, when converting to the resulting linear [dBn] measure it gains a precision of 6 digitus, which is enough to calculate the linear average of the samples. To provide the result in [dBm], the process has to be reverted. The whole process is illustrated in figure 3.19.

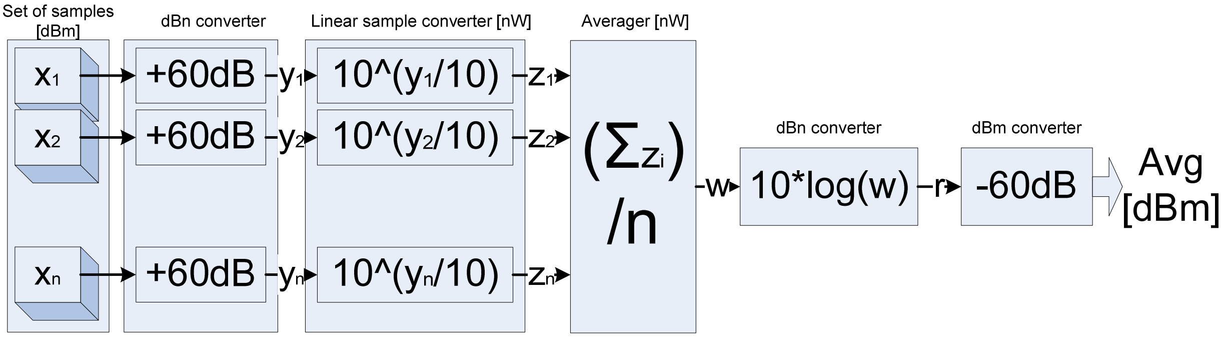


Figure 3.19: Received signal strength average procedure

3.4.1.3. Evolved IEEE 802.21 protocol stack. Resulting stack

The development of a particular scenario leads the design of protocol stacks to be as much alike to the ones proposed under the standards as they can. In fact, the philosophy of the developed stack is the same as the standard, but with slight differences. These stacks are nothing more than abstractions of the code composing the project-space for both MN and PoA.

The stack regarding to the MN is the most similar to the standard. It provides the existing physical layer from a WLAN and its MAC/LLC mechanism parallel to the developed MIH Function. Above, there is the MIH User layer. Moreover, the MIH SAP and the MIH Link SAP are implemented to communicate MIHF with the layers above and below, respectively. Everything until this point is as stated in IEEE 802.21 standard. Nevertheless, some features are added in order to complete the stack and its functionality. First, an abstraction of the physical and link layer are inserted between the physical layer and the MIH Link SAP. This provides an interface to interact with the driver setting it and gathering the required information. Second, a Transaction State Machine is introduced in parallel to MIH Function, with the purpose of dealing with the transmissions and retransmissions of packets related to a single transaction. Last, the MIH User is defined as an entity capable to take decisions about connectivity and to set properly the interfaces via MIHF. In this context, it has not been developed as a separated application: it is a part of the daemon prototype.

The resulting stack for the PoA is also implemented as much alike as possible to the defined in the standard. In this context, the PoA is a device which only has an interface to deal with connections. No MIH actions with lower layers are needed, so the MIH Link SAP is not required. Instead, as well as in MN stack, an abstraction of the lowest layer is placed between these one and the MIH Function. No MIES is needed because it does not have to report local events, so MIES has not been included. At the highest layer, a MIH User is placed in parallel to a MIIS User. The first one is designed to take decisions, as well as this layer does in the MN stack. The second one has the mission of providing responses to the information queries coming from remote entities. The MIHF has an additional entity which acts as a registry of the MN attached to the current PoA, and provides tools for handling these subscriptions. Moreover, a Transaction State Machine very similar to the one implemented for the MN is placed in parallel to the MIHF entity as well. All these elements are placed in the same they were in the MN stack. A representation of all the stated in these lines can be seen in figure 3.20.

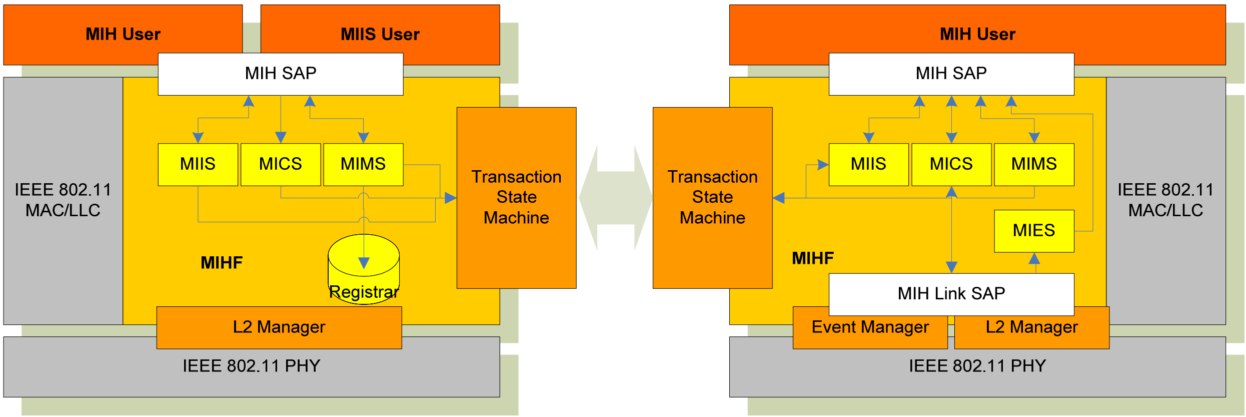


Figure 3.20: Left: PoA designed stack. Right: MN designed stack

3.4.1.4. Handover management modes: mobile initiated and network initiated

In a mobile initiated handover, the MN initiates the handover process by indicating to the network that a handover is needed. The network selection policy function resides at the upper layers of the MN. It uses the set of MIH_MN_HO_*** commands and indirectly causes MIH_N2N_HO_*** commands between PoAs when initiating and proceeding handovers. The MN can use these commands to query the list of all available candidate networks, reserve resources at the target network, and to indicate the status of an operation to the network MIHF.

In a network initiated handover, the network initiates the handover process by indicating to the mobile that a handover is needed. The network selection policy resides somewhere in the network, not necessarily in the attached PoA. The network uses the set of MIH_Net_HO_*** in conjunction with MIH_N2N_HO_*** commands when initiating and proceeding handovers. The network can use these commands to query the list of resources being currently used by the MN, and the serving network can use these patterns to reserve these resources at the destination network. Moreover, the network can command the MN to commit a handover to the target network.

A mobile controlled handover is suitable to be a third option. In this case, the mobile node has the primary control over the handover process. This procedure has not been further specified in IEEE 802.21 standards.

The current framework is a low charge system in terms of traffic. Just a MN can be generating or demanding a flow of traffic, which means that the required resources in each PoA will be enough every time the MN is attached to one of them. In this case, there does not exist a problem of overcharging a determined network by demanding an excess of its resources. So, PoAs can afford these changes without problems. A different situation could be done in case several MNs were roaming from PoA to PoA. In this case, some kind of intelligence should be placed into the network in order to manage the charge of the overall system.

So, it seems logical to allocate the handover decisions into the MN. It can measure the received signal level and determine whether it is required a better quality or not, independently from the charge in the network. In long term, the development of the MIH compatible networks can lead to the design of a network initiated system, with MN assistance, as well as it works in the current cellular telephony system.

3.4.1.5. IEEE 802.21 implemented message exchange

3.4.1.5.1. Significant differences regarding to the original specification

The annexes of the IEEE 802.21 standard document define many different situations in which a handover is needed. Many of them refer to higher layer protocols to complete the higher layer management procedures. The procedures selected to serve as example are derived from Mobile IP (MIP). In particular, they are Fast Handovers over Mobile IP (FMIP) and Proxy Mobile IP (PMIP). Although they are the key to provide service continuity through storing and forwarding packets, dealing with these protocols is out of the scope of this work. Instead, a classical IP association and authentication protocol has been imposed in order to provide upper layer connectivity.

Moreover, some changes have been introduced into the standard message exchange proposed in the quoted standard annexes. The main change refers to the suppression of some part of the actions taken related to the handover, because they seem to be redundant. The following lines will explain this feature in depth.

3.4.1.5.1.1. Standard handover message exchange flow

The common mobile node initiated handover procedure describes an exchange message flow which involves the following request and response sets of messages:

1. Information query: MIH_Get_Information

2. Resource availability check: MIH_MN_HO_Candidate_Query / MIH_N2N_HO_Query_Resources

3. Resource preparation: MIH_MN_HO_Commit / MIH_N2N_HO_Commit

4. Resource release: MIH_MN_HO_Complete / MIH_N2N_HO_Complete

The MN needs to choose which elements are going to be queried to the MIIS server by using the information query. It allows getting information elements, proceeding them and then to decide which will be the target PoA. It is interesting to study the contents into these sets of messages and see if the information included is worth at all, paying attention to the second resource availability check messages and the information within.

The resource availability check set carries the following information:

* *Source Link Identifier:* this identifies the source link for handover.

* *Candidate Link List*: is a list of PoAs, identifying candidate networks to which handover needs to be initiated. The list is sorted from most preferred first to least preferred last.

* *QoS Resource Requirements:* these are the minimal QoS resources required at the candidate network.

* *IP Configuration Methods*: is an optional element which identifies the current IP configuration methods.

* *DHCP Server Address*: is the IP address of the current DHCP Server, only included when MN is using dynamic address configuration.

* *FAAddress*: is the IP address of current Foreign Agent, only included when MN is using Mobile IPv4.

* *Access Router Address*: is the IP address of the current Access Router, only included when the MN is using IPv6.

This set of messages needs to be sent to each of the candidate PoAs like polling. When a response is received it includes a list of candidate networks which would fulfill the configuration specified and described into the request message. Each response can be different, and so the complexity of the target decision increases. Once the MN has determined which will be the target PoA for handover, the rest of the queried PoAs are discarded and the message exchange continues for both serving and target PoAs.

In addition, the information query messages provide a very flexible way to allow the MN to query a large number of IEs. Some of these elements are the ones that can be queried through the set of messages defined as resource availability check messages. Moreover, through MIH_Get_Information, which is the request message for information queries, the MN can query all the surrounding PoAs and networks in a placement within just a message. This message is addressed to an information server which is provided with all the static information about the surrounding networks within a location.

So, suppressing the set of messages composing the resource availability check eases the decisions taken at higher layers of the protocol stack and provides a more efficient message exchange protocol. Nevertheless, some modifications are needed to keep a correct and coherent message flow.

The list of mapped elements results as follows:

MIH_N2N_HO_Commit Request ↔ MIH_MN_HO_Commit Request.

* *Destination Identifier* : a table in the PoA maps the MIHF Id. ↔ Link Address/Network Id. relationship.

* *MN MIHF Identifier*: User received Source Identifier.

* *Target MN Link Identifier*: Alternative Link Id. in registry + User received Link Address into Target Network Info.

* *Target PoA Link Address*: Link Address into Target Network Info.

* *Requested Resource Set*: additional, Requested Resource Set from MN.

The MIH_MN_HO_Commit Request message is modified by adding the last element and the header of the associated primitive. The figure 3.21 shows how the message flow finally rests.

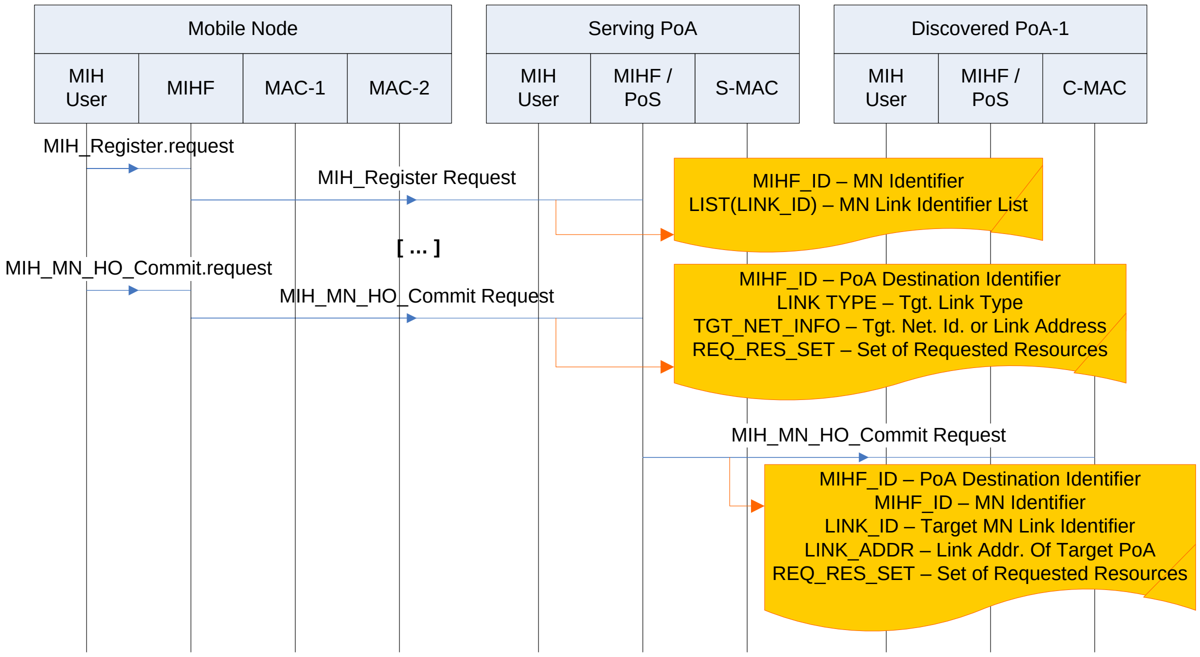

Mobile Node
MIH User
MIHF
MAC-1
MAC-2
Serving PoA
MIH User
MIHF / PoS
S-MAC
Discovered PoA-1
MIH User
MIHF / PoS
C-MAC
MIH_Register.request
MIH_Register Request
MIHF_ID – MN Identifier
LIST(LINK_ID) – MN Link Identifier List
[...]
MIH_MN_HO_Commit.request
MIH_MN_HO_Commit Request
MIHF_ID – PoA Destination Identifier
LINK TYPE – Tgt. Link Type
TGT_NET_INFO – Tgt. Net. Id. or Link Address
REQ_RES_SET – Set of Requested Resources
MIH_MN_HO_Commit Request
MIHF_ID – PoA Destination Identifier
MIHF_ID – MN Identifier
LINK_ID – Target MN Link Identifier
LINK_ADDR – Link Addr. Of Target PoA
REQ_RES_SET – Set of Requested Resources

Figure 3.21: Particularity in MIH MN handover message flow

3.4.1.5.2. Abstract message exchange flow and implemented actions

The message exchange flow has suffered some variations from its standard specification to the implementation in the application. The result is a MIH message exchange alike, which provides all the advantages of the standard definition under the premises stated in previous sections. The following subsection provides an overview of the basic actions taken during the implemented message exchange.

1. ON. The application is started.

2. The MN receives IEEE 802.11 beacon frames from APs to know the available PoAs on the near area.

3. The MN proceeds MIHF Discovery and MIH Capability Discovery procedures in order to know if the APs are MIH compatible and which services are supported in each one.

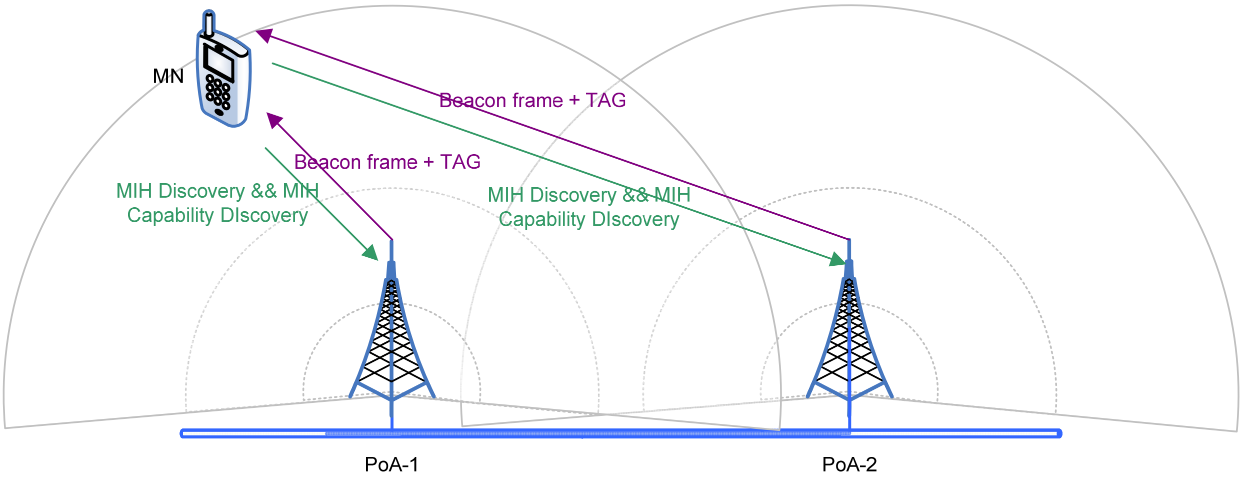

MN
Beacon frame + TAG
Beacon frame + TAG
MIH Discovery && MIH Capability DIscovery
MIH Discovery && MIH Capability DIscovery
PoA-1
PoA-2

Figure 3.22: MIH Discovery and MIH Capability Discovery

4. Based on the averaged received signal strength coming from the radio beacons and the list of services supported, the MN proceeds with BOOTSTRAPPING PoA DECISION. In this case, both PoA entities only differ on the received signal strength.

5. The MN proceeds MIH Registration with the target PoA. This alllows the MN to select a MIH PoS which will provide the information requested by the layers above.

6. The MN proceeds network layer authentication and association. This permits the MN to establish a communication between uper layer applications over the IEEE 802.11 protocol stack.

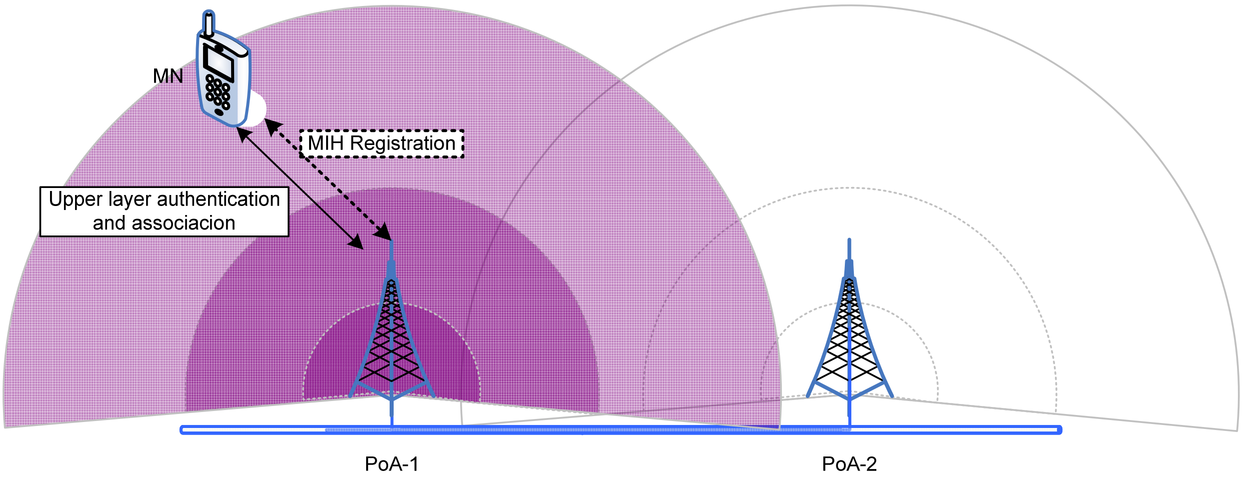


Figure 3.23: MIH Registration and upper layer authentication & association after Bootstrapping Decision

7. Now the data path is established. Application information can be exchanged between peers or a flow of data can be received from a device placed somewhere in the network. The MN can now roam across different PoAs.

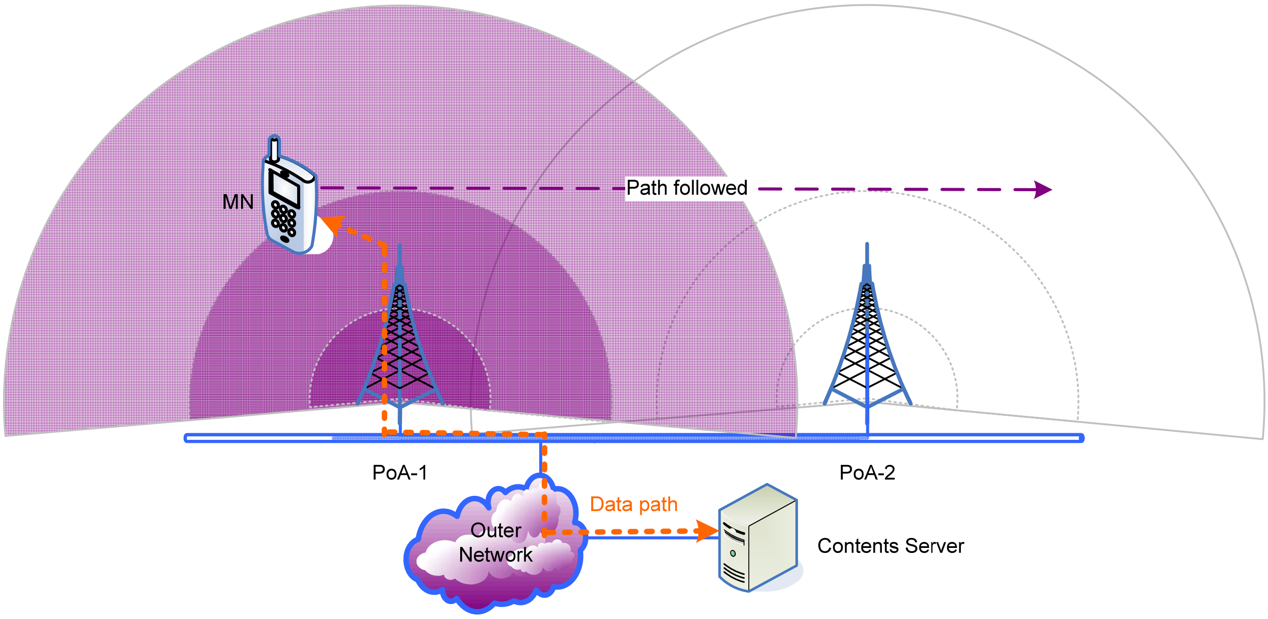


Figure 3.24: Data and roaming paths

8. Received signal thereshold Crossed: Link Parameters Report. The recieved signal strength becomes degraded. The first critical threshold has been crossed. An inminent handover is required to keep the communication. The indications are based on the averaged received signal strength from the beacon frames, but could also be taken from the application data.

9. Information query. The MN needs to send a query to the Information Server to find out data such as the kind of links supported by each PoA, type of IEs supported or the maximum number of IEs in a response frame, among many other features of the required services.

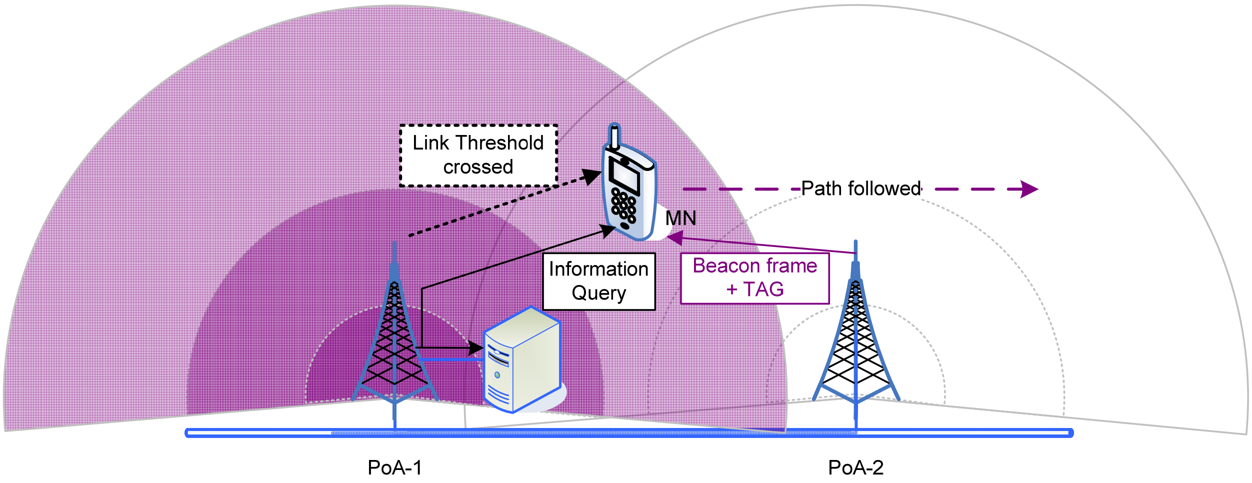
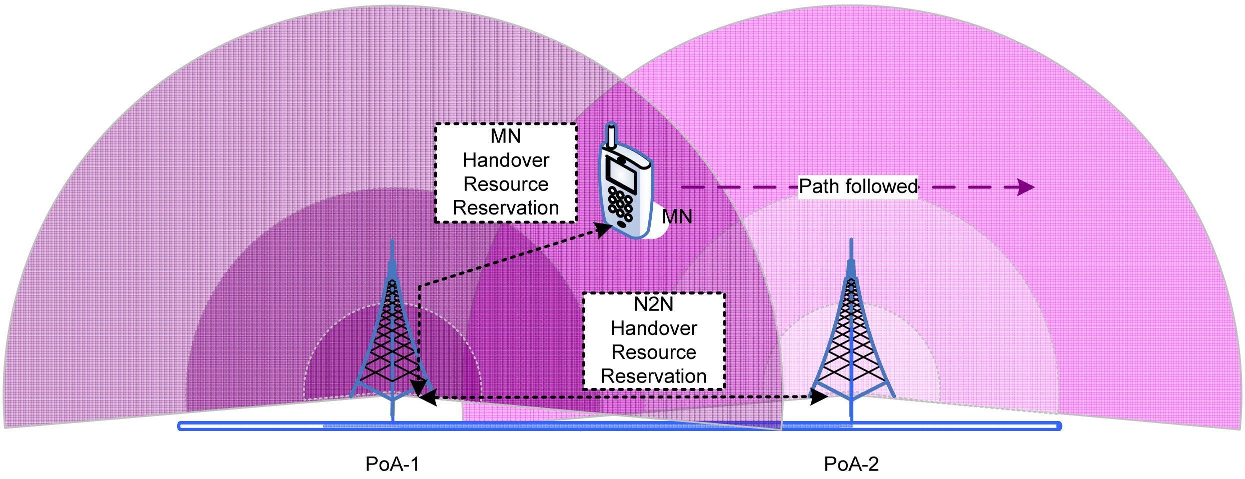


Figure 3.25: Data path established across the network and Information Query

10. The discovered PoAs are now listened and trigger Link Detected Indications. The MN listens to the media again and recognizes the beacon(s) of a (or many) candidate AP(s) that can be acting as PoAs.

11. The received information from MIIS and from the indications triggered provide the required elements to perform a TARGET PoA DECISION.

12. Handover Request, Resource Reservation and Handover Response. The MN indicates the decision of handover to the currently attached PoA. The resource reservation is proceeded in a dialog between the involved networks. The confirmation of this reservation is sent back to the MN from the candidate network, passing through the serving network.

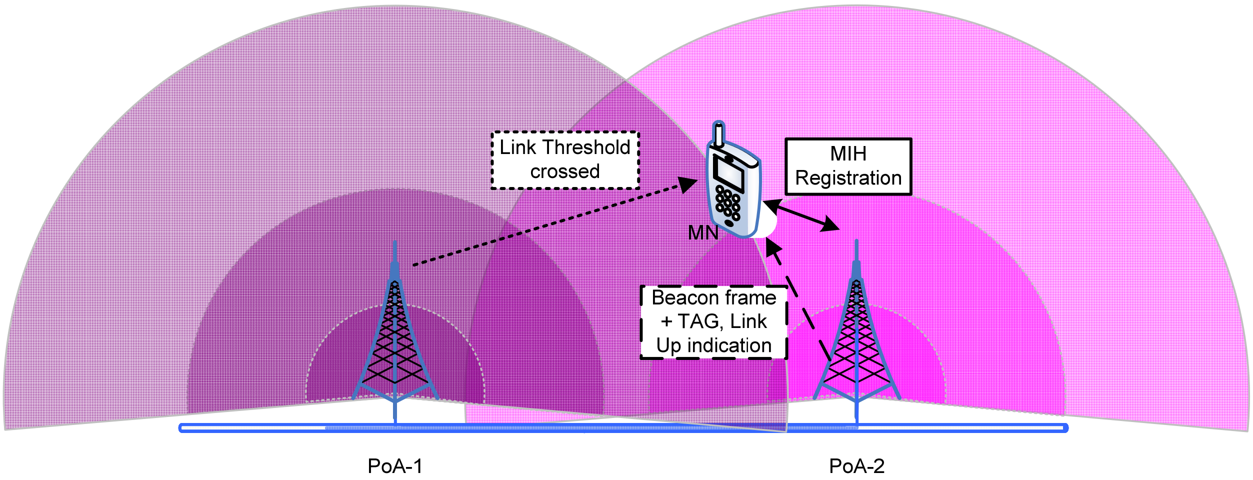


13. The MN waits for changes of signal strength from the currently attached PoA waiting for a degradation of the signal quality.

14. If the signal level is restored over the first critical threshold, the handover is aborted and the configuration is restored as originally. But, if the quality gets even more degraded and it crosses the Link Going Down threshold, it activates the remaining WLAN interface with Link Power Up and then starts to establish the new L2 connection.

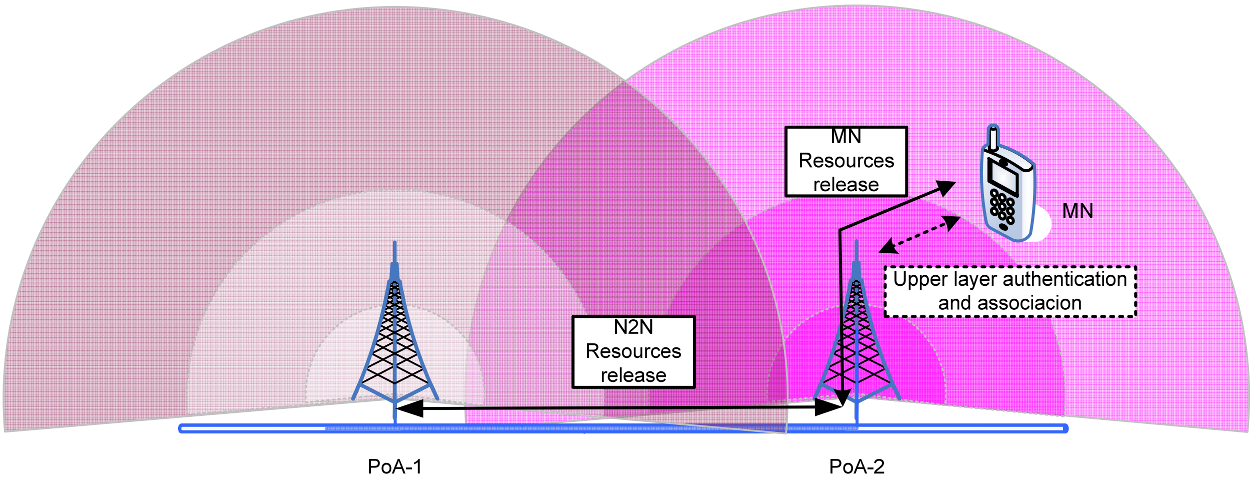
15. The activation of the alternative WLAN interface triggers Link Up indication in the MN from the candidate PoA, in case the MN has fallen on the coverage area of this one.

16. Now, the MN can proceed MIH registration with the discovered PoA.



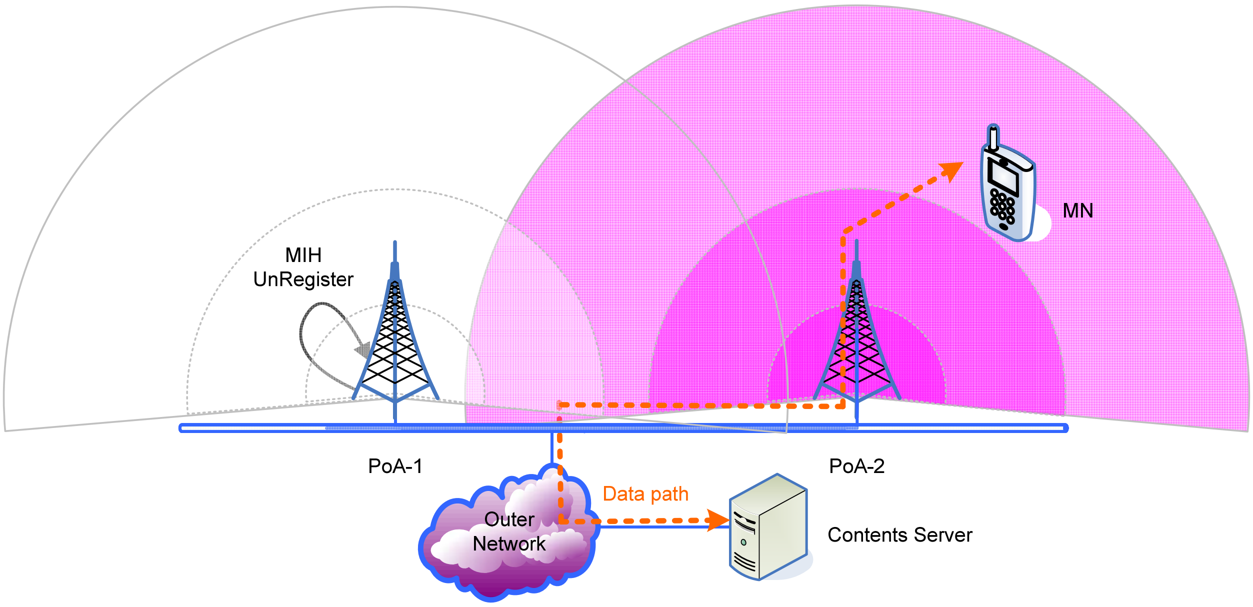
17. The MN can now be authenticated and associated with the PoA at higher layers via IEEE 802.11 and IP protocols. A Link Power Down is transmitted from user layer to the old interface, and so it gets switched off.

18. Resource Release. The MN indicates via the new serving PoA that the handover has been completed. Now the old PoA can release the resources the MN was using and so it can be MIH unregistered.



19. The new data path can be established between the MN and another entity in the network, namely a content server, another user peer or whatever.

20. Finally, the old PoA detaches the MN.



3.4.1.6. Transaction state machines

A transaction state machine is, as stated in Chapter 2, a function in charge of the management of the transactions of MIH messages between different MIH entities. The basis of the behavior of its mechanism resides in the contents of the incoming or outgoing message for a given transaction and in the reliability of the communication.

In the current design, the task of the states machine is essential. A 2.5 layer protocol needs the support of a system which implements mechanisms to assure the delivery of the messages besides of a correct handling of them. The IEEE 802.21 standard talks about instantiating state machines, so the idea of an object oriented implementation comes up. The C++ programming language provides the required tools to do it.

3.4.1.6.1. Cases

As stated in IEEE 802.21 standard, at any moment a MIH node will not have more than one pending transaction with another peer. This means that just an instance of the transaction state machines will be present in the mobile node application, and just another one in the correspondent PoA. The situations given in the present framework are the following:

A) The mobile node needs to send a message to its corresponding PoA, so it starts its own transaction source state machine. The communication is done at L2, and the wireless media does not provide reliability to the communication. For this reason, an instance of an Ack requestor state machine is also started initiating the acknowledgement service. This has been indicated in the outgoing message by setting *AckReq* bit to one. So, for a MN outgoing message, the peer creates an instance of the transaction source and Ack requestor state machines. The correct reception of the corresponding response message will cause in the termination of both state machines with success state. Otherwise, if the reception of this response has been delayed or lost, the Ack requestor state machine takes the proper actions to perform a new query. Finally, if these mechanisms fail, both instances of the state machines die with failure status. The state flow and actions taken in this prototype is described in IEEE 802.21 standard pages.

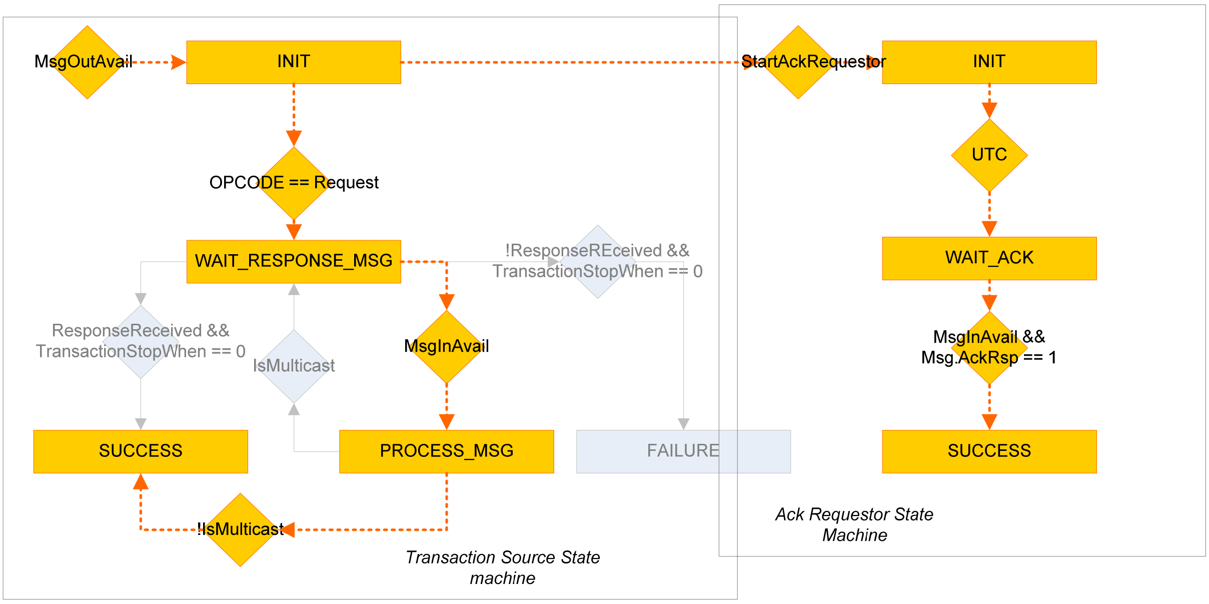

MsgOutAvail
INIT
OPCODE == Request
WAIT_RESPONSE_MSG
!ResponseREceived && TransactionStopWhen == 0
ResponseReceived && TransactionStopWhen == 0
IsMulticast
MsgInAvail
SUCCESS
PROCESS_MSG
FAILURE
!IsMulticast
Transaction Source State machine
StartAckRequestor
INIT
UTC
WAIT_ACK
MsgInAvail && Msg.AckRsp == 1
SUCCESS
Ack Requestor State Machine

Figure 3.30: Source state machines desired flow in a wireless transaction

The figure 3.30 shows the state flow diagram if no problems occur. The state order in this case is the following:

1. INIT – Transaction Source State Machine

2. INIT – Ack Requestor State Machine

3. WAIT_RESPONSE_MSG and WAIT_ACK

4. PROCESS_MSG

5. SUCCESS – Ack Requestor State Machine

6. SUCCESS – Transaction Source State Machine

B) The PoA receives a message coming from the MN which requires a response, so the transaction destination state machine is started. Due to the fact that the wireless media is unreliable, the Ack responder state machine is also instantiated. This was previously set in *AckReq* bit of the MIH header of the received message. In this context, th Ack responder state machine just piggybacks the outgoing response message by setting to one the *AckRsp* bit of the MIH header. Once the message has been sent, the Ack responder ceases to exist and the transaction destination state machine terminates with success state. The state flow and actions taken in this prototype is described in IEEE 802.21 standard pages.

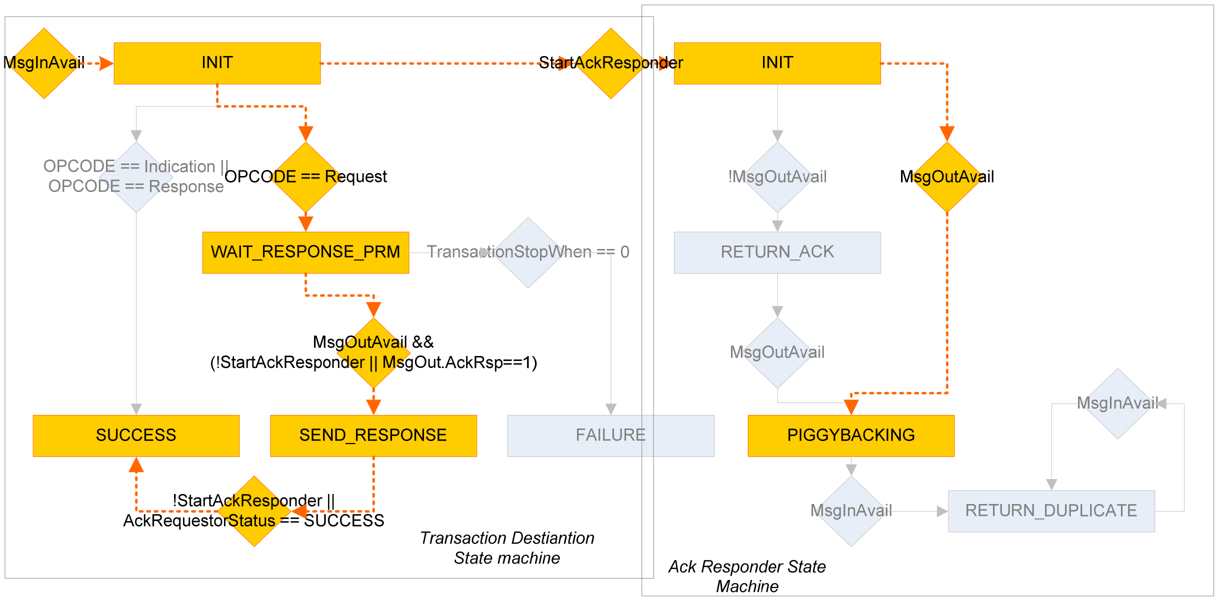

MsgInAvail
INIT
StartAckResponder
INIT
OPCODE == Indication \|\| OPCODE == Response
OPCODE == Request
!MsgOutAvail
MsgOutAvail
WAIT_RESPONSE_PRM
TransactionStopWhen == 0
RETURN_ACK
MsgOutAvail && (!StartAckResponder \|\| MsgOut.AckRsp==1)
MsgOutAvail
MsgInAvail
SUCCESS
SEND_RESPONSE
FAILURE
PIGGYBACKING
!StartAckResponder \|\| AckRequestorStatus == SUCCESS
MsgInAvail
RETURN_DUPLICATE
Transaction Destiantion State machine
Ack Responder State Machine

Figure 3.31: Destination state machines desired flow in a wireless transaction

The figure 3.31 shows the state flow diagram if no problems occur. The state order in this case is the following:

1. INIT – Transaction Source State Machine

2. INIT – Ack Requestor State Machine

3. WAIT_RESPONSE_RPM

4. PIGGYBACKING

5. SEND_RESPONSE

6. SUCCESS and termination of Ack Responder State Machine

C) A PoA needs to send a message to another PoA (the target) via the wired network. In this case, the transaction is done through a network IP/TCP socket pair, and the reliability mechanisms are implicitly implemented. So, the source host just instantiates a transaction source state machine. An Ack requestor state machine is not needed because the transport level of the protocol stack is supposed to be free of link errors (they are solved at lower layers). In this case, the transmitter is implemented as a client of a TCP service which queries the receiver and waits for a response message from another peer designed as a TCP server. The correct reception of the corresponding response message will cause the termination of the transaction source state machine with success state. Otherwise, if the reception of this response has been delayed or lost, the instance of this state machine will die with failure status. The state flow and actions taken in this prototype is described in IEEE 802.21 standard pages. It is just the same as the indicated in the point A), but without the acknowledgement service.

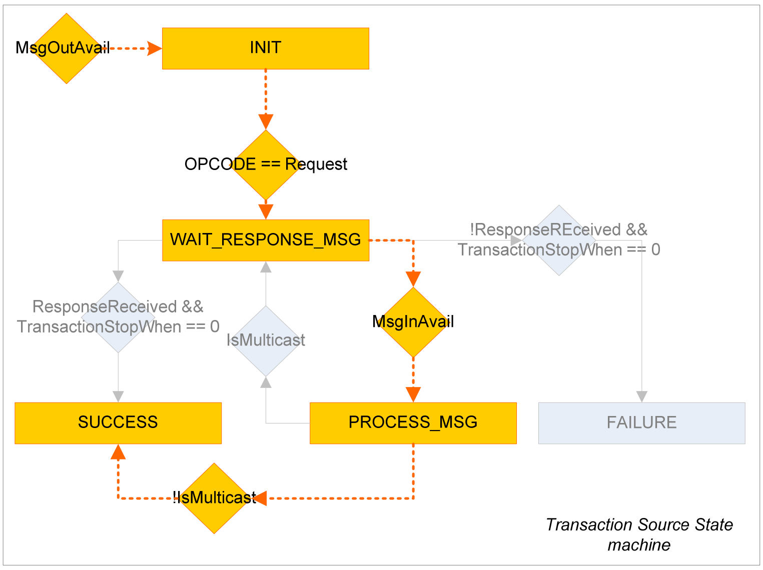


Figure 3.32: Source state machine desired flow in a wired TCP transaction

The figure 3.32 shows the state flow diagram if there are no problems. The state order in this case is the following:

1. INIT

2. WAIT_RESPONSE_MSG

3. PROCESS_MSG

4. SUCCESS

D) A PoA receives a message from another PoA (the source) via the wired network. In this case, the transaction is done through a network IP/TCP socket pair, and the reliability mechanisms are implicitly implemented. So, the destination host just instantiates a transaction destination state machine. An Ack responder state machine is not needed because the transport level of the protocol stack is supposed to be free of link errors (they are solved at lower layers). In this case, the receiver is implemented as a server of a TCP service which is queried by the requestor and replies with the correspondent response message to the requester peer designed as a TCP client. Sending the corresponding response message will cause the termination of the transaction destination state machine with success state. The state flow and actions taken in this prototype is described in IEEE 802.21 standard pages. It is just the same as the indicated in the point B), but without the acknowledgement service.

The figure 3.33 shows the state flow diagram in case of success. The state order in this case is the following:

1. INIT

2. WAIT_RESPONSE_RPM

3. SEND_RESPONSE

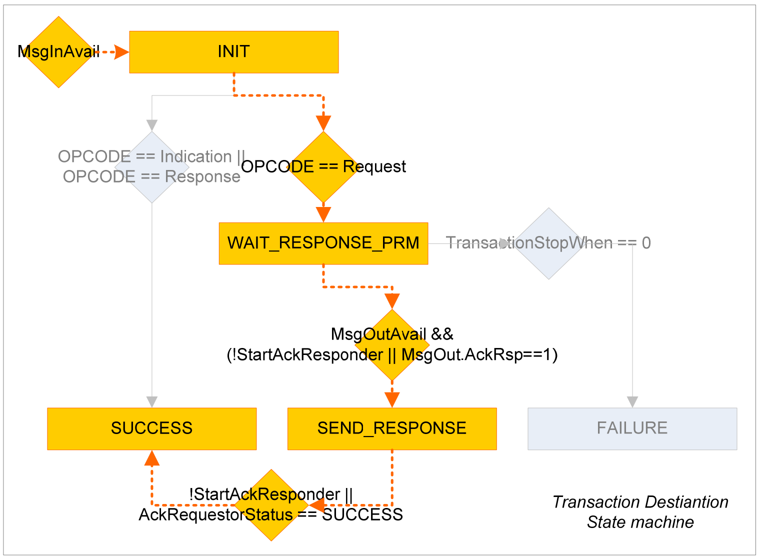


Figure 3.33: Destination state machine desired flow in a wired TCP transaction

4. SUCCESS

The exchange of messages that comprise both communications over the wireless and the wired network are considered to be composed by two different transactions. The first one is a transaction between the MN and the PoA. Here the MN throws an instance of the transaction source state machine and another one of the Ack Requestor state machine, while the PoA throws an instance of the transaction destination state machine and another one for the Ack responder state machine. The second one is a transaction between PoAs. One of them throws an instance of a transaction source state machine and the other an instance of the transaction destination state machine. The first transaction activates the wired transaction and rests paused until it finishes. Then, the wireless transaction is taken up again and is provided with the elements of the wired PoA to PoA transaction. Both transactions are independent and so they have their own transaction identifiers, inter and intra state machine variables.

3.4.1.6.2. Retransmission timeouts

As stated, the acknowledgement service shall be used when the MIH transport used for remote communication does not provide reliable services. When the MIH transport is reliable, the use of the acknowledgement service is optional.

When seeking acknowledgement service, the source MIH entity starts a retransmission timer after sending a MIH message with the *AckReq* bit set to one and saves a copy of this message while the timer is active. The IEEE 802.21 standard proposes the algorithm defined in IETF RFC 2988 [38] to calculate the value of this timer. Nevertheless, the related parameters used into the state machines refer to this value as a constant in the standard pages. The final decision is to set the timers retransmission timeout to one second, in order to ease the development and considering that in the current framework the transaction is almost instantaneous.

If the acknowledgement message is not received before the expiration of the timer, the source MIH entity immediately retransmits the saved message with the header settings kept untouched. If the source MIH entity receives the acknowledgement before the expiration of the timer on the first or any subsequent retransmitted attempt, therefore the source MIH entity has ensured the receipt of the MIH packet and hence releases the resources used for this transaction.

The procedure to calculate the retransmission timeout defined in RFC 2988 is the one required for senders to compute and manage their retransmission timer, and could be used in further improvements of this application. A summary is provided as follows:

M = measured RTT; R = smoothed RTT estimator; a = smoothed factor = 0.9 (recommended)

$R < -aR + (1-a)M$ ← Updated every time a measure is made.

b = delay variance factor = 2 (recommended)

$RTO = Rb$

A = smoothed RTT (average estimator); g = gain for average = 0.125

$Err = M-A$

$A < -A+gErr$

$RTO = A + 4D$

Initial values: $A_0 = 0s$; $D_0 = 3s$.

3.4.2. Mobile node features

3.4.2.1. MN User level actuation schema

The User level of the MN application implements a mechanism to decide which messages need to be sent or which procedures need to be executed regarding to the state of the connection. The messages sent are originated by the invocation of MIH SAP primitives that are caused by link events or even that are generated as the next corresponding set of instructions to establish or maintain the communication. The figure 3.34 shows the schema followed by the User layer at the MN part.

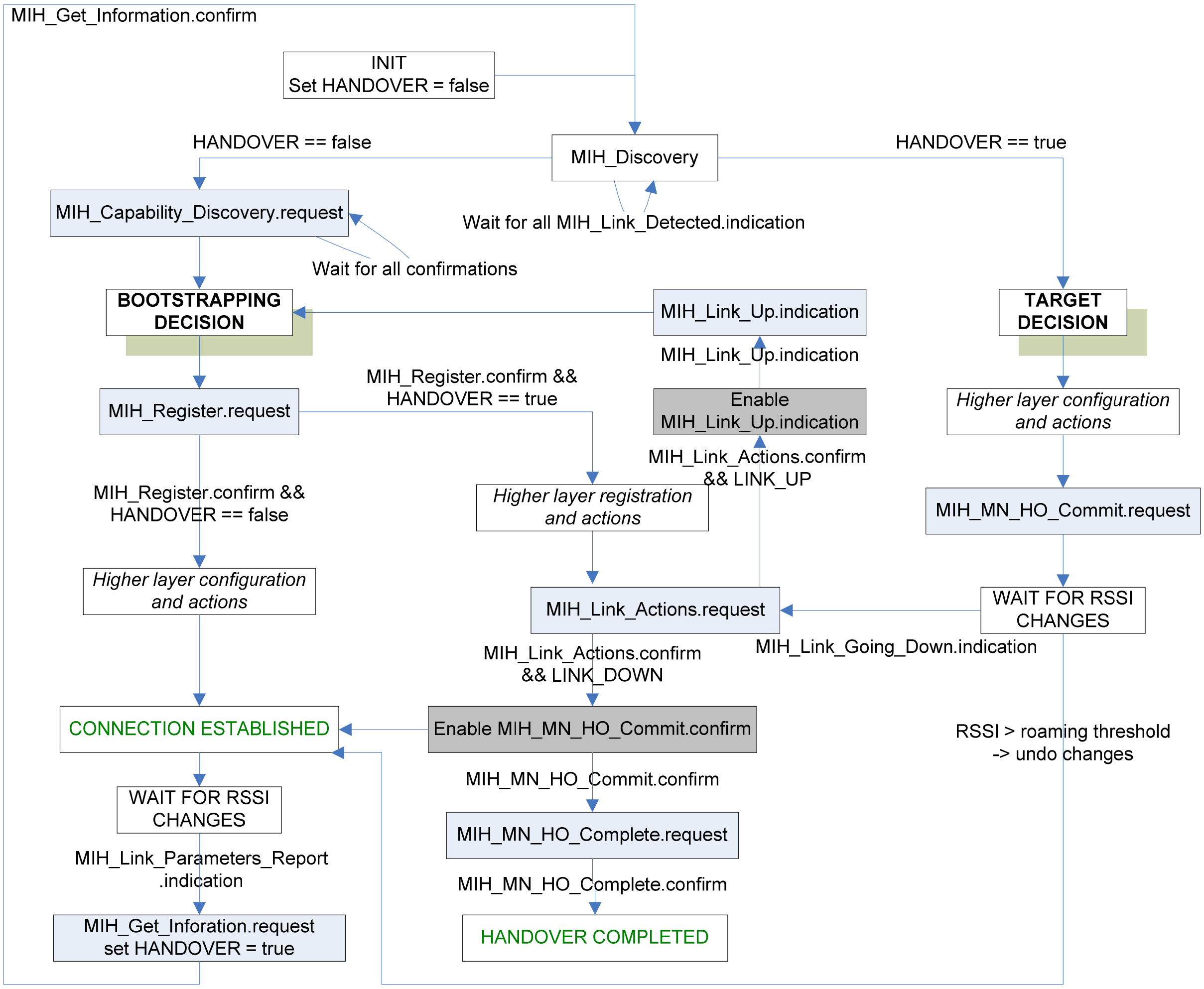

MIH_Get_Information.confirm
INIT
Set HANDOVER = false
HANDOVER == false
MIH_Discovery
HANDOVER == true
MIH_Capability_Discovery.request
Wait for all MIH_Link_Detected.indication
Wait for all confirmations
BOOTSTRAPPING DECISION
MIH_Link_Up.indication
TARGET DECISION
MIH_Link_Up.indication
MIH_Register.request
MIH_Register.confirm && HANDOVER == true
Enable MIH_Link_Up.indication
Higher layer configuration and actions
MIH_Link_Actions.confirm && LINK_UP
MIH_Register.confirm && HANDOVER == false
Higher layer registration and actions
MIH_MN_HO_Commit.request
Higher layer configuration and actions
MIH_Link_Actions.request
WAIT FOR RSSI CHANGES
MIH_Link_Actions.confirm && LINK_DOWN
MIH_Link_Going_Down.indication
CONNECTION ESTABLISHED
Enable MIH_MN_HO_Commit.confirm
RSSI > roaming threshold -> undo changes
MIH_MN_HO_Commit.confirm
WAIT FOR RSSI CHANGES
MIH_MN_HO_Complete.request
MIH_Link_Parameters_Report .indication
MIH_MN_HO_Complete.confirm
MIH_Get_Inforation.request set HANDOVER = true
HANDOVER COMPLETED

Figure 3.34: MN message exchange decision flow

User procedures start at INIT box, placed at the top of the schema. The arrows indicate the direction of the state changes in MN's User part. These arrows carry the MIH SAP primitives that trigger the state change, the value of the HANDOVER variable (false if the MN is not attached or otherwise true), or nothing if the transition between states is unconditional. In clear blue colored boxes there are represented the MIH SAP primitives called from User level and triggered by the condition indicated in the previous arrow pointing to the current box or even unconditionally triggered (when there is not an indication). In white and shadowed boxes with capital bold lyrics, there are represented the main PoA selection procedures, namely bootstrapping and target decisions. The boxes in white with capital letters represent the states where the MN is waiting for changes in signal levels once a connection has been established and remaining in a stable state, or when a handover procedure has been started and the system is monitoring the received signal strength in order to find whether it has been restored or if the handover needs to be completed. The grey colored boxes represent knobs activated by the system that allow certain indications whenever a set of requirements is met. Boxes in green colored capital letters represent the final states in a split handover completion procedure: the part that establishes the connection and the part that releases the old resources. Finally, white boxes with letters in italics represent other level operations such as IEEE

802.11 authentication and association. Although it is not indicated in the schema, note that in case the MIH connection gets lost due to a sudden drop in the link or a loss of MIH coverage (a MIH_Link_Down is triggered), the User MN procedures are restarted driving the system to INIT state and setting the HANDOVER variable to false.

The following subsections provide a complete description of the decision procedures, as well as a quick overview of other simpler mechanisms such as the sub-system which waits for changes in signal level. The higher level attachment procedures are the described in Chapter 2. Furthermore, a particular implementation solution is proposed in order to solve boundary problems between the MIHF identifier and the MAC address in the current framework.

3.4.2.2. MN MIH User decisions for target PoA

As mentioned before, the handover mechanisms are initiated by the MN, the element in the network with intelligence to take bootstrapping and handover decisions. Both mechanisms need to weight up the relevant information provided by the elements of the network and decide which PoA to establish or continue a communication best fits the conditions of the MN. All those decision proceedings are out of the scope of the standards, and are provided into the implementation of the higher MIH user layer as example. The following subsections provide a detailed description of the procedures taken at MIH user levels of the current implementation, regarding to decisions of bootstrapping attachment and roaming from PoA to PoA whenever it is needed.

3.4.2.2.1. Bootstrapping decisions

The bootstrapping procedure is given when the MN device application has been just switched on or when the MIH connectivity has been lost suddenly or by reasons of end of MIH coverage area. It is divided into two different proceedings, namely Candidate Link Decision and Candidate Decision. The first one uses the elements of information provided into the beacon frames of the MIH compatible PoAs, and the second one takes the information within the MIH_Capability_Discover response messages. Both sub-mechanisms return a parameter which is the result of pondering the information received. Finally, these parameters are considered to establish the bootstrapping decision. The figure 3.35 below shows a schema of the bootstrapping system.

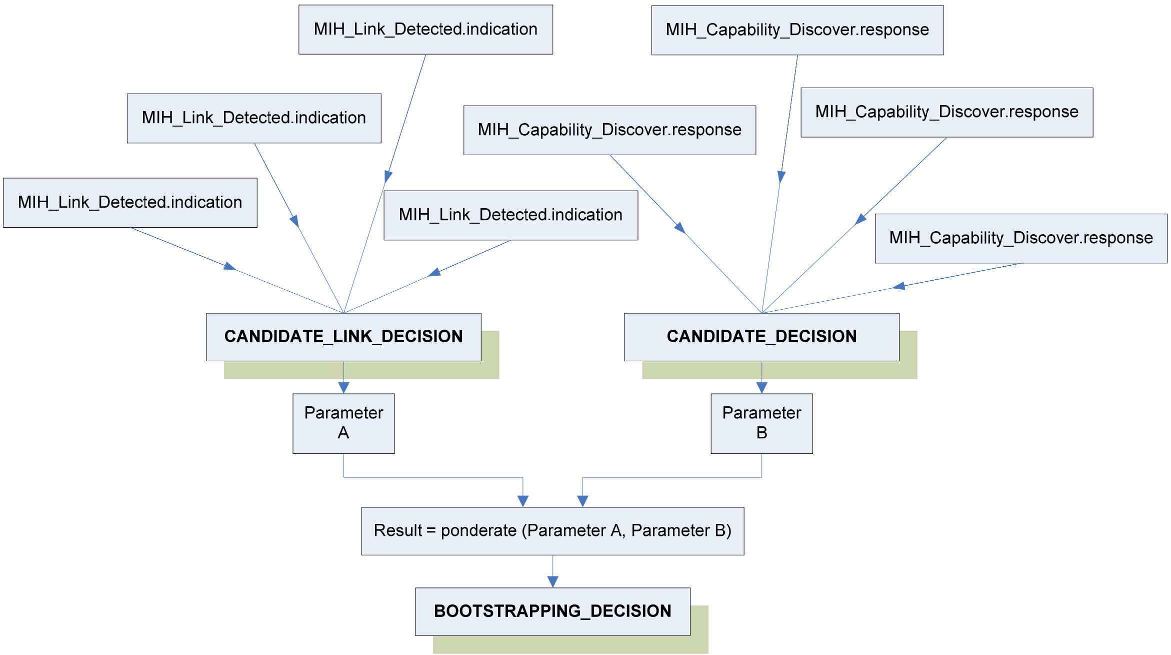

MIH_Link_Detected.indication
MIH_Capability_Discover.response
MIH_Link_Detected.indication
MIH_Capability_Discover.response
MIH_Capability_Discover.response
MIH_Link_Detected.indication
MIH_Link_Detected.indication
MIH_Capability_Discover.response
CANDIDATE_LINK_DECISION
CANDIDATE_DECISION
Parameter A
Parameter B
Result = ponderate (Parameter A, Parameter B)
BOOTSTRAPPING_DECISION

Figure 3.35: Bootstrapping decision schema

The Candidate Link Decision schema is a system of punctuation which increases a parameter in function of the capabilities available in each discovered PoA. Two conditions are stated as pre-requisites in the current development: it is mandatory for the PoA to provide command and information services (MICS and MIIS, respectively). If those requirements are not met, the return value of this procedure will be 0, and the PoA will be discarded as possible target. Features such as received signal strength or link data rates, which are ranges, provide the relevant part of the punctuation. Elements such as security, emergency and QoS services provide less relevant punctuations. Refer the figure 3.36 to see the detailed Candidate Link Decision procedure.

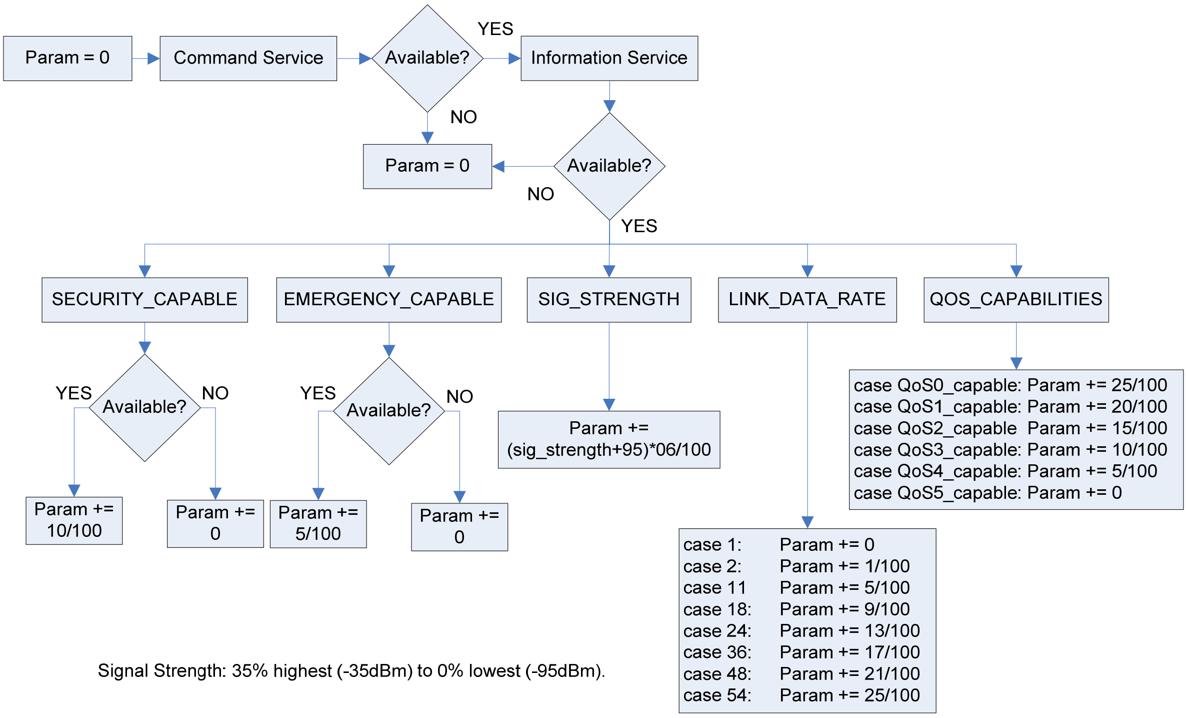


Figure 3.36: Candidate Link Decision schema

The Candidate Decision subsystem emphasizes the decision task on the available capabilities of the services provided by the evaluated PoA. It first reviews the capabilities provided by the command service in the remote PoA, and decides whether the PoA can establish a dialog with the command service of the MN. In the current implementation, MIH_N2N_HO_Query_Resources, MIH_N2N_HO_Commit, MIH_N2N_HO_Complete, MIH_MN_HO_Query_Resources, MIH_MN_HO_Commit, and MIH_MN_HO_Complete messages in the evaluated PoA are mandatory. Afterwards, the MN reviews if the elements from the information service are available in binary type. If no binary coding is available, the PoA is discarded as target of the communication. Finally, the capabilities for transport layer and the ability of providing make-before-break handovers are evaluated. The figure 3.37 shows a detailed schema of the whole Candidate Decision procedure.

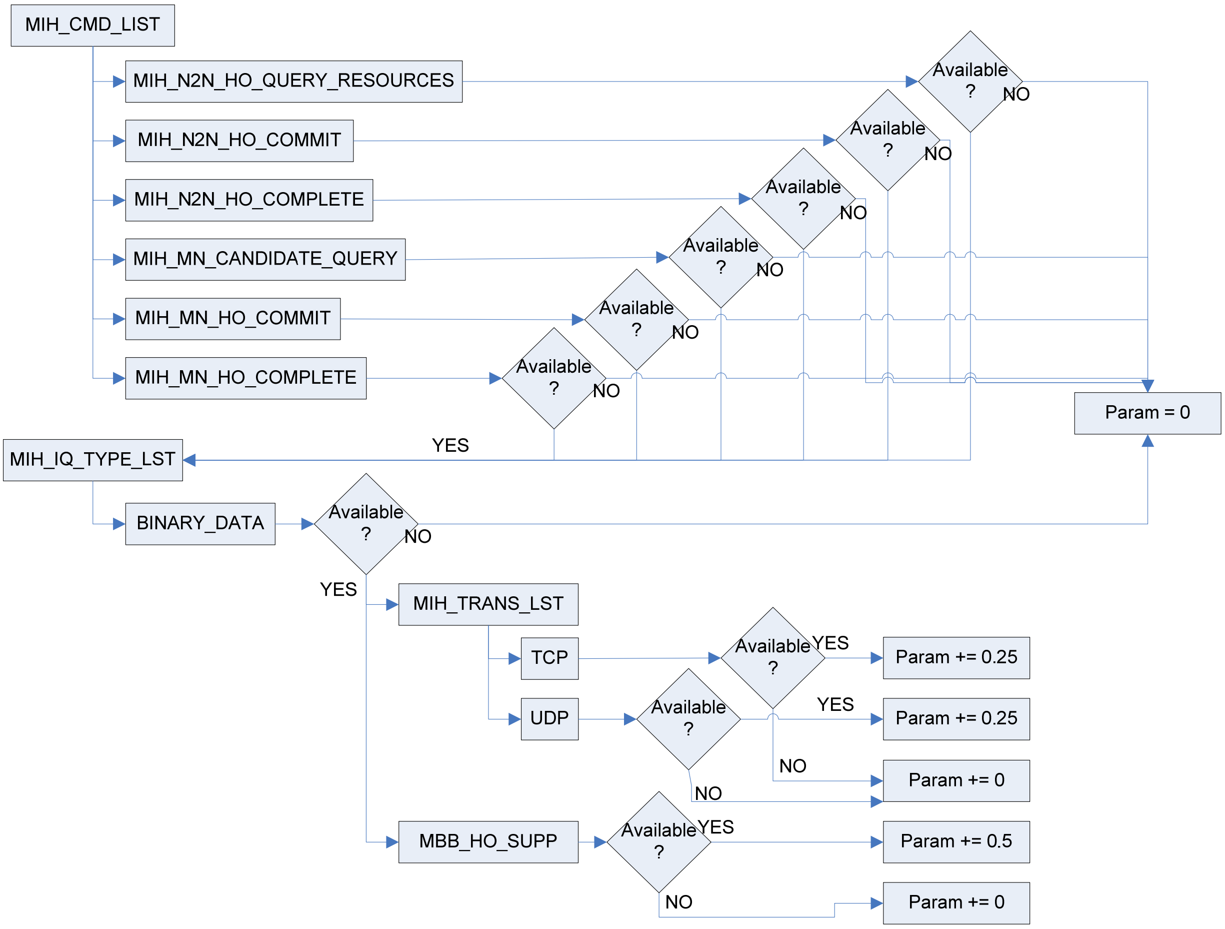


Figure 3.37: Candidate decision schema

At the end, the parameters got from Candidate Link Decision and Candidate Decision are multiplied to get a final decision parameter for each discovered or candidate PoA. The PoA which receives the best punctuation will be the communication target.

3.4.2.2.2. Handover decisions

A handover procedure is started when the service received during a communication is getting degraded. For a MIH handover to take place, it is necessary to be previously attached to a PoA able to provide continuity of service. Again, this procedure is divided in two subroutines: one is a Candidate Link Decision, and works as described in the previous subsection, and the other one is a MIIS decision. Each sub-procedure returns a parameter which has been previously pondered in function of the elements used in these routines, as well as the bootstrapping decision schema does. Finally, these parameters are considered to establish the handover decision. The figure 3.38 shows a schema of the handover decision system.

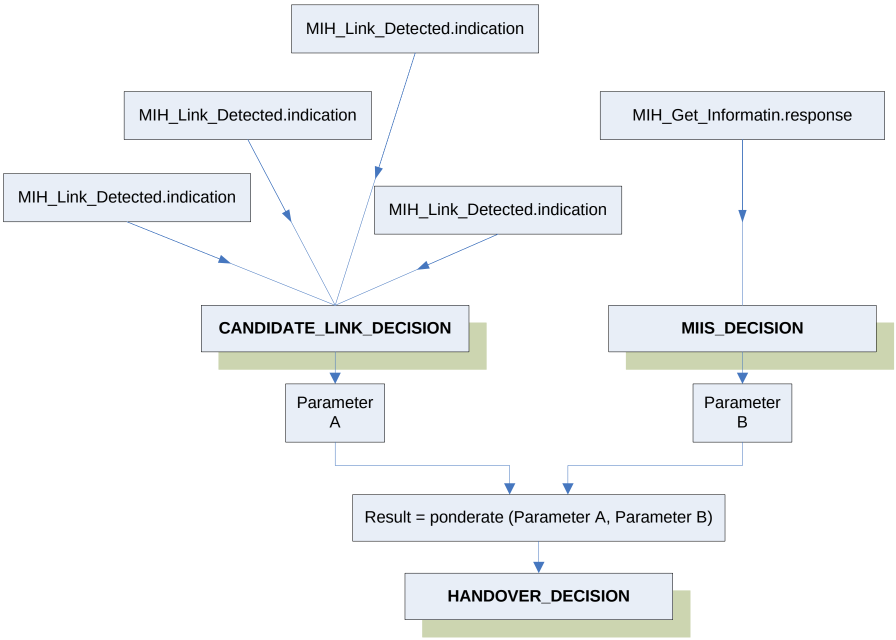

MIH_Link_Detected.indication
MIH_Link_Detected.indication
MIH_Get_Informatin.response
MIH_Link_Detected.indication
MIH_Link_Detected.indication
CANDIDATE_LINK_DECISION
MIIS_DECISION
Parameter A
Parameter B
Result = ponderate (Parameter A, Parameter B)
HANDOVER_DECISION

Figure 3.38: Handover decision schema

The main difference between bootstrapping and handover schemas is the number of messages received in each one. In a bootstrapping decision, the Candidate Decision is performed ranging from the MIH_Capability_Discover response messages received in the MN. The MN receives as many response messages as PoAs are surrounding it. The MIIS decision is done through the reception of just a MIH_Get_Information response message.

The Candidate Link Decision works as well as commented in the previous clause. On the other hand, the MIIS Decision is probably the most complex decision subsystem of them all. It is based in the Information Elements responded from a MIIS to a query message from the MN. In the current scenario, just a few IEs are required in response from the MIIS to take decisions of handover. Other IEs are not needed because the information they provide is included in MIH_Link_Detected indication messages. The rest of IEs do not fit in the current framework. The information query to the MIIS demands a set of IEs which will cause in a hierarchy of IEs in the response message. In particular, an IE Container List of Networks will be queried, and it will carry all the suitable IEs from surrounding networks and the PoAs within. The following table indicates the requested IEs in capitals and the reason why they are used or not.

Information Element	Required or not
Network-specific Information Elements	
IE_NETWORK_TYPE	needed to allow roaming compatibility
IE_OPERATOR_ID	needed for roaming partnership
IE_SERVICE_PROVIDER_ID	not used in the current framework
IE_COUNTRY_CODE	not used, not relevant in the current framework
IE_NETWORK_ID	needed to determine the attached or target network
IE_NETWORK_AUX_ID	is always the same for the required network type
IE_ROAMING_PARTNERS	needed for roaming partnership
IE_COST	needed to take HO decisions regarding to cost of access
IE_NETWORK_QOS	needed to know the quality of the data flow
IE_NETWORK_DATA_RATE	not used, included into capabilities
IE_NET_REGULAT_DOMAIN	not used
IE_NET_FREQUENCY_BANDS	redundant
IE_NET_IP_CFG_METHODS	needed to determine upper layer parameters
IE_NET_CAPABILITIES	included in LINK_DET_INFO
IE_NET_SUPPORTED_LCP	not used in the current framework
IE_NET_MOB_MGMT_PROT	not defined an upper layer management protocol
IE_NET_EMSERV_PROXY	not used in the current framework
IE_NET_IMS_PROXY_CSCF	not used in the current framework
IE_NET_MOBILE_NETWORK	not used in the current framework
PoA-specific information elements	
IE_POA_LINK_ADDR	needed to know the PoA MAC address
IE_POA_LOCATION	mandatory IE (not used in the current framework)
IE_POA_CHANNEL_RANGE	mandatory IE (redundant in the current framework)
IE_POA_SYSTEM_INFO	mandatory IE (redundant in the current framework)
PoA-specific higher layer service information elements	

IE_POA_SUBNET_INFO	needed to know L3 parametres
IE_POA_IP_ADDR	mandatory IE (redundant, included in subnet info)

Table 3.2: Available IEs

The mechanism to set a parameter works different in this sub-system. The parameter to be returned is initially set to one, and it varies its value depending on the information provided into the IEs. The final value ranges between 0 and one.

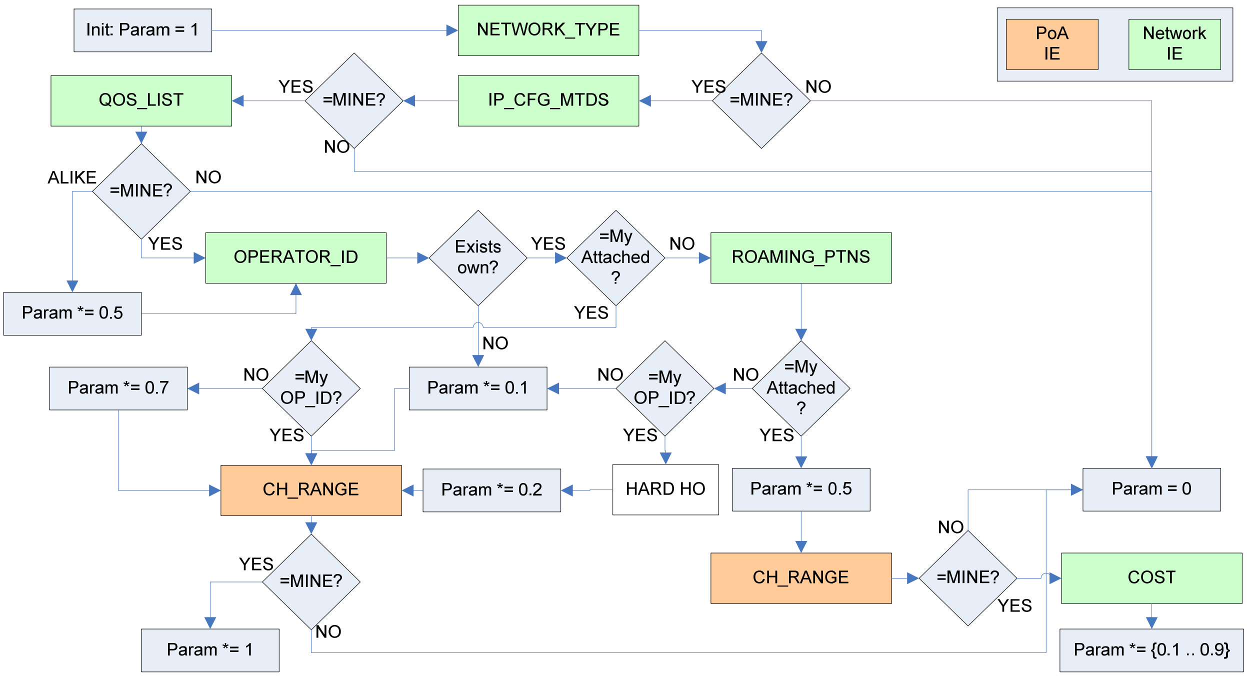


Figure 3.39: Received IE decision flow

The decision flow mixes the IEs provided into a IE_Container_Network and the IE_PoAs within. Remember that a network can be composed of many PoAs, each of them with different features. The process is repeated for each existing PoA provided into the information query response frame:

* Number_of_iterations = add(Network_i*Number_of_PoAs_per_Network_i)

The implemented system penalizes most the peers that do not provide QoS enough, a valid set of configuration methods, a type of network equal than the currently used (see that both MN interfaces are physically IEEE 802.11), or do not match with any of the needed channels of the requested range. These PoAs or networks are directly discarded, setting the decision parameter to one. The rest of the evaluation is done pondering the information got from the operator, the roaming partners and the cost of access to the network. The whole decision flow is shown in the figure 3.39.

The figure 3.40 shows the schema followed to determine the punctuation arising from the comparison between demanded and offered quality of service. The blocks represent the set of QoS demanded or offered ranging from QoS class 0 (highest QoS) to QoS class 5 (lowest QoS) [39]. The yellow block is the set of QoS queried by the MN, whereas the red block represents the same for the offered by the PoA. In this procedure, two decisions are taken: first, the decision parameter is modified in function of the coincidences between QoSs, and second, a degree of QoS is selected. Three basic situations can take place: one, the requested set of QoS is higher than the set of QoS offered by the PoA; second, some QoS offered by the PoA matches with the demanded by the MN, but not the highest demanded; and three, the best QoS demanded by the MN is also offered by the PoA. The first situation drives to a deficient assignation of QoS resources, and so the decision parameter is set to 0. The second one is a sub-optimal situation, where the MN gets one degree of service into the range demanded, but not the best, and so the decision parameter is set to 0.5. In the last case, the assignation is optimal, and the decision parameter is set to one.

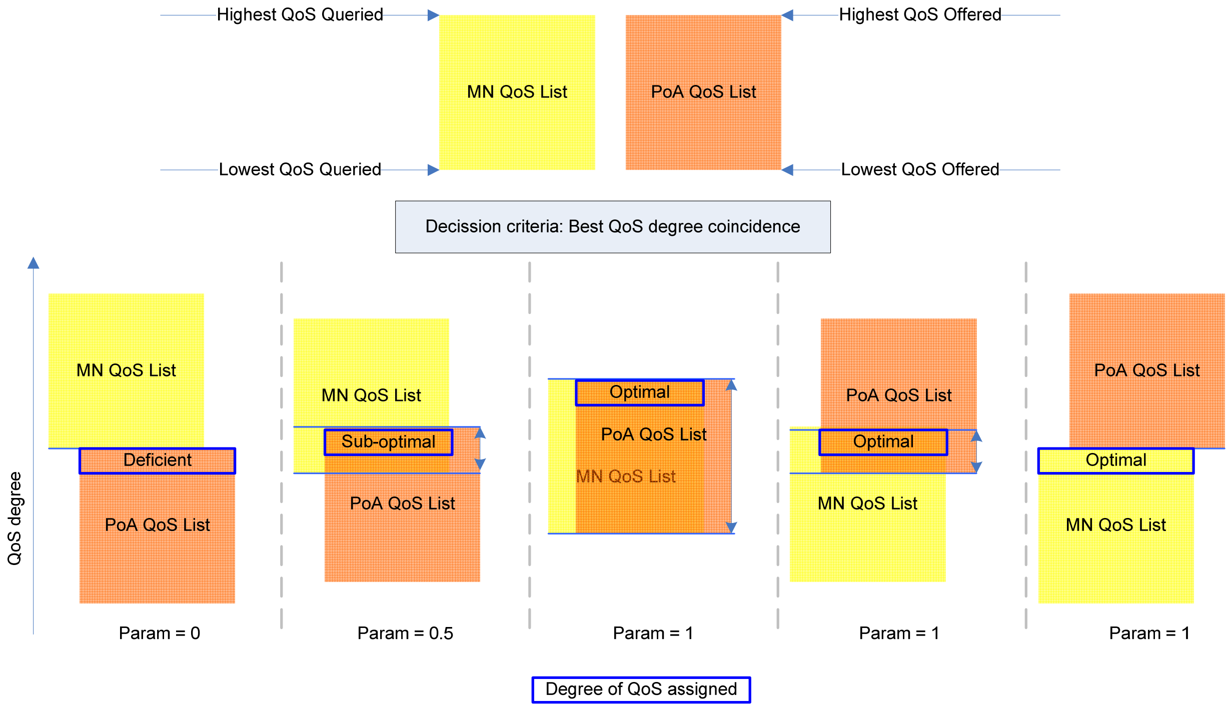


Figure 3.40: QoS decision schema

Moreover, another subsystem into the MIIS decision subsystem is implemented. It is designed to choose the decision parameter in function of the cost of each access network. This procedure does the following: the information provided into the corresponding received IE, namely the cost unit (day, month, year…), the cost value (value per cost unit) and the cost currency are combined to determine the cost for acceding the reviewed network along a determined period of time. The decision parameter is set to a value ranging from 0 to one, depending on the stated result. The figure 3.41 illustrates this procedure.

Summarizing, the parameters got from Candidate Link Decision and MIIS Decision are multiplied to get a final decision parameter for each candidate PoA. The PoA which receives the best punctuation will be the target for a handover of the current communication.

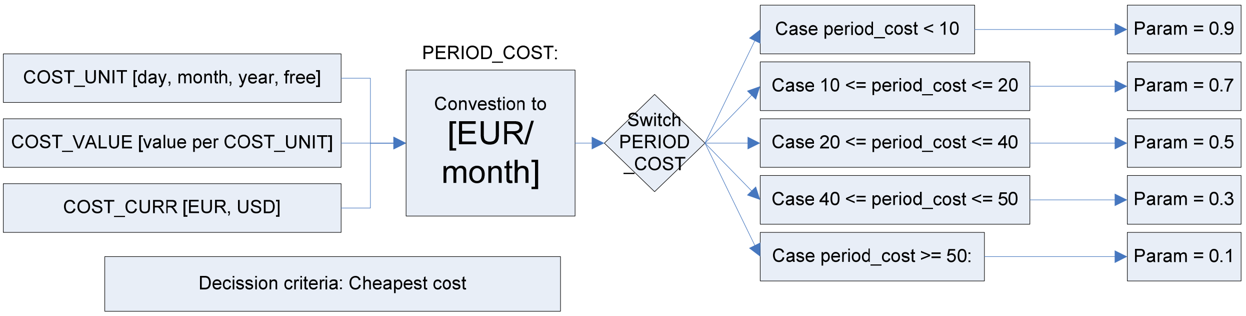

COST_UNIT [day, month, year, free]
COST_VALUE [value per COST_UNIT]
COST_CURR [EUR, USD]
PERIOD_COST:
Convestion to
[EUR/
month]
Switch
PERIOD
_COST
Decission criteria: Cheapest cost
Case period_cost < 10
Case 10 <= period_cost <= 20
Case 20 <= period_cost <= 40
Case 40 <= period_cost <= 50
Case period_cost >= 50:
Param = 0.9
Param = 0.7
Param = 0.5
Param = 0.3
Param = 0.1

Figure 3.41: Parameter evaluation for access cost

3.4.2.3. MN Signal Strength related event triggering

The signal strength related events are generated at the lowest layers of the implemented protocol stack. A module has been designed to be aware of the received signal level from the attached PoA, performing a continuous monitor of this value. This system is capable of reporting the events which are generated due to crossing determined signal thresholds. The following lines will provide a qualitative description of the sub-procedures carried out into this module, relating them with some of the explanations from previous subsections.

As stated before, two main thresholds are defined in function of the quality of the received signal: roaming threshold and clear channel threshold. Moreover, a third threshold above the previous ones is defined in order to

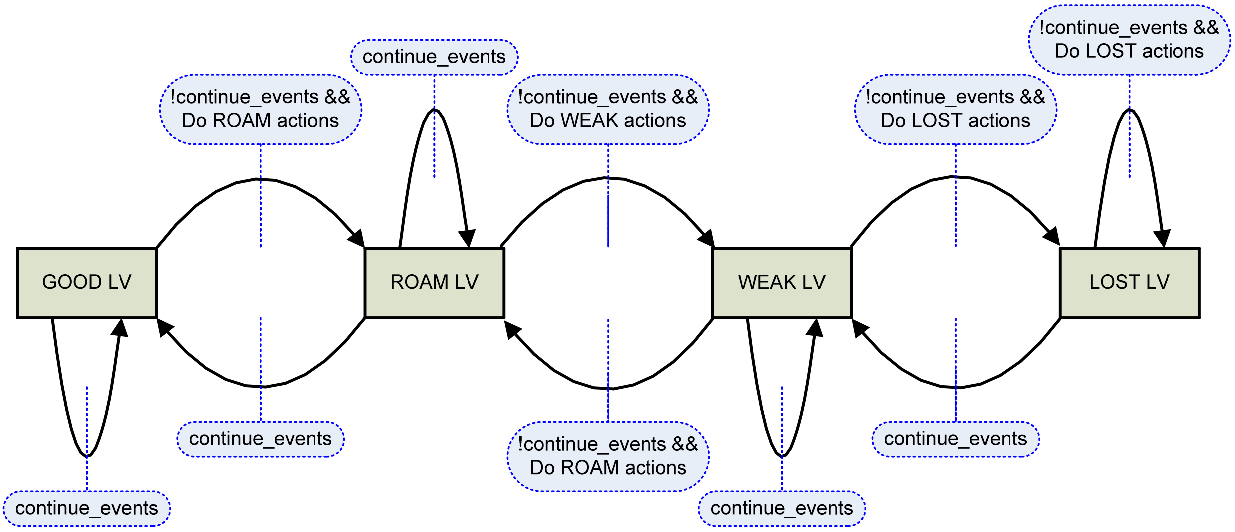


Figure 3.42: Actions taken regarding to the signal strength sub-range changes

decide if the MIH connection is lost or not. So, four sub-ranges are defined between these thresholds, namely *good level*, *roam level*, *weak level* and *lost level*, from most strong received signal to least. In the current implementation, it has been presumed that changes in signal level are smooth, so it cannot hop from a given range to another one which is not adjacent. The actions taken when each different range is met depend just on the previous state. The figure 3.42 represents the changes between sub-ranges of signal level. Grey boxes represent the four stated sub-ranges, and the blue ones are the actions taken regarding to the state change.

The *roam level* related actions prepare the indication carried within a Link_Parameters_Report indication primitive in the MIH Link SAP, indicating that a certain threshold has been crossed, that the stated threshold is for the received signal strength, and which are the quantitative values of this threshold and the parameter measured. This event marks the beginning of a handover procedure.

The *weak level* related actions prepare the information carried within a Link_Going_Down indication primitive in the MIH Link SAP, indicating that a certain threshold has been crossed and also which link is the expected to go down, reporting the expected time for this to happen. Moreover, the reason why the network is expected to go down is included. This event triggers the continuation of the initiated handover procedure.

The *lost level* related actions prepare the information carried within a Link_Down indication primitive in the MIH Link SAP, indicating that a certain threshold has been crossed and also which is the link that has gone down, reporting the link address of the old access router and the reason why the link connection has been lost. Once this primitive has been triggered, the upper level entities will start again a bootstrapping MIH discovery based procedure.

3.4.2.4. MN MAC-MIHF mapping table solution

MIHF is defined as a 2.5 layer protocol or as a protocol parallel to the levels 2 and 3 of the OSI protocol stack. This means that in the context of a vertical handover between two WLAN hotspots, the MIHF packets come packed into MAC/LLC IEEE 802.11 std. link layer packets. The proposed test-bed scenario shows a clear boundary between the MAC addresses of the WLAN devices of the PoAs and their MIHF identifiers. Both interfaces of the MN are also related to the same MIHF identifier within the peer. No procedures are defined under the IEEE 802.21 standard to bound MAC and MIHF identifiers within the same peer. Nevertheless, the current scenario eases this task and so a procedure is proposed under this implementation.

The primitives declared into IEEE 802.21 std. do not refer to addresses different than MIHF identifiers when a message has to be routed (i.e; they do not contain IP address, MAC address or any other type of address different from the MIHF identifier). The implemented solution into MN code is a mapping table at a registry which bounds a MAC address with its correspondent MIHF identifier. Two functions are defined to keep the registry updated and to retrieve an associated Mac address:

```
* u8 store_POA_TUPLE( POA_TUPLE poa_tuple, POA_TUPLE_LIST *list );
```

This function stores the selected pair of identifiers into the list of the MN data base. The parameter `poa_tuple` is the pair MAC address – MIHF identifier and `list` is the data base that maintains all the received pairs mapped in a table. The return value is 1 if the indicated pair has been stored, or 0 otherwise.

* `u8 retrieve_mac_addr( u8 *mac_addr, MIHF_ID *mihf_id, POA_TUPLE_LIST *list );`

This function retrieves the associated MAC address to the indicated MIHF identifier from the given list. `mac_addr` is the buffer where the MAC address, if found, is stored; `mihf_id` is the identifier whose associated MAC address is searched, and `list` is the data base where the MAC address has to be searched. The function returns 1 if the associated MAC address is found or 0 otherwise.

The process of registration in this table is executed at every Link_Detected indication, during the MIH Discovery procedure. A similar process is implemented in the PoAs. Actually, it is a system that generates an automatic destination MAC address in response to the last received source MAC address for a given transaction. The solution into the MN could be moved to the PoAs in future developments in order to avoid a possible mess of addresses in a multiple roaming MN scenario.

3.4.3. PoA features

3.4.3.1. PoA addresses resolution

The point of attachment has to deal with different situations to complete parts of the MIHF discovery procedures which are not covered under the standard specification. One is an advice of presence which a MN needs to detect the existing PoAs. The other is the way a PoA establishes a communication with another PoA within the same network or not. The following sub-clauses provide a description of the solutions adopted to solve these issues.

3.4.3.1.1. PoA MIHF identifier for MN MIH discovery procedure

As mentioned in previous sections, a MN needs to discover the surrounding APs in order to determine which provides MIH compatibility, and which identifier they use to exchange this protocol messages. In this case, it is not enough with a MIH capability advice within the beacon frames. A PoA MIHF identifier has to be supplied somewhere. This can be done introducing some support protocol (existing or not) establishing a dialog between MN and PoA, or by directly getting the PoA identifier from a management IEEE 802.11 frame such as the mentioned beacon frame. For reasons of ease of implementation and agility in the process, the second method has been implemented in the current development. An implementation considering security issues should consider performing an intermediate MIHF discovery protocol, but by now this work falls out of the current design purposes.

3.4.3.1.2. Wired network to network communication engine

When different handover decisions need to be communicated between a MN and a remote PoA through the network, a communication mechanism needs to be introduced between the serving PoA and the remote PoA. In this particular case, the system uses the following set of exchanged messages between PoAs:

* MIH_N2N_HO_Commit Request and Response,

* MIH_N2N_HO_Confirm Request and Response.

The idea of communicating two peers in a wired IP network is not new. It is based in a client-server relationship, using a TCP transport protocol. Both PoAs involved in a network to network communication are composed by the client and server elements. The peer needing to query the other throws an instance of the client, which sends the corresponding MIH message encapsulated into a TCP frame to the server in the remote PoA. The server in the remote peer responds to the received message once the upper layer response has been composed and is available. Once the source client receives the response it passes it to the upper layers and the client instance ceases to exist. The servers in each PoA are continuously running and waiting for new TCP connections.

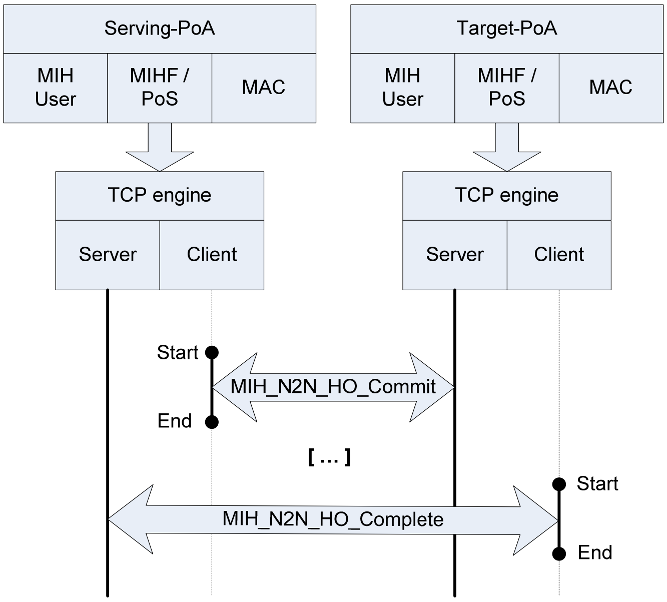


Figure 3.43: Server-client network to network communication model

The principal problem comes up in the procedure to discover which IP direction in a peer is associated to the corresponding MIHF identifier within a certain PoA. Again, the solution can be based in a network discovery protocol, with a similar behavior to ARP (Address Resolution Protocol) or RARP (Reverse ARP). Developing a protocol of this nature is not under the scope of the current work, and so a simpler alternative explained has been implemented.

The proposed solution is based on a table which establishes a relationship between a certain MIHF identifier and its associated IP address i.e. a lookup table. This table is information previously introduced in each PoA dependencies. It is not the best choice to solve this problem due to the difficulty in updating the table once an identifier is changed; nevertheless, it does the required work without an additional effort. When a PoA needs to send a message to another one over a TCP transport, it just has to look up which is the associated PoA IP direction in this table and establish a network socket with the remote peer. A resolution protocol for this issue should be considered in future developments.

3.4.3.2. PoA Registrar

The point of attachment is provided of a data base where a registry of MNs is maintained. Every time a new MN wants to get registered with a PoA, it sends a MIH_Register Request query message. If the required conditions are met, the PoA allows this specific MN to be registered in its Registrar data base and a MIH_Register response message with success status is sent back to the MN. From this moment on, the MN can freely exchange messages with its associated PoA as long as it needs it. In the current implementation, it is decided that no messages different from those provided by MIMS can be exchanged if the registry process is not completed successfully or, in other words, the MN has not been previously registered in the required PoA. It discards any packet different from those coming from management service if the source mobile node has not been registered previously.

A registry subscription expires once the MN has completed a handover to another PoA, receiving the MIH_N2N_HO_Complete message. In this case, a procedure within the MIH Registrar searches the MN MIHF identifier in the corresponding data base, and once it has been found its related information is deleted.

It must be commented that this subscription should be upgraded each certain period of time in order to keep a MN registered in the attached PoA’s data base. A field within the MIR_Register Response message is provided to indicate the MN the time interval in which subscription is valid. By default, the current implementation supplies a timer interval of 0, which means that the subscription period is undefined.

3.4.3.3. PoA MIIS implementation

As stated, it has been decided that the MIIS would be placed in the same host as the PoA. This entity has been designed as a MIHF module which provides this service, and as a User level entity which automatically generates responses to the received information service queries coming from the MN. The higher layer mechanism provides a set of IEs that have been responded and coded in TLV format, which is the selected type of IE representation for the current development.

3.4.3.3.1. Hierarchal IE encapsulation

On one hand, the TLV coding format is defined in standards as a protocol which provides information within a determined structure. The information unit is formed by a *Type* field which has a fix width, a *Length* field with a variable width and a *Value* field also with a variable width. In the current development, this kind of encapsulation is used by the User MIIS in order to provide the stated information elements.

On the other hand, the definition of these information elements provides a complex but flexible way to transport information to the entity that has queried them. As stated in Chapter 2, different containers of IES can be queried. Commonly, it makes no sense querying isolated IEs. In the current development, the MN demands an IE which is a container list of networks, carrying indeed three lower levers in the IE hierarchy. The next level is a container of IEs related to a single network which can be isolated IEs or containers of the next level. These new level containers are particular for each PoA in the current network, and are formed by different IEs which are single for each PoA. The figure 3.44 shows the hierarchy of information elements and containers of information elements.

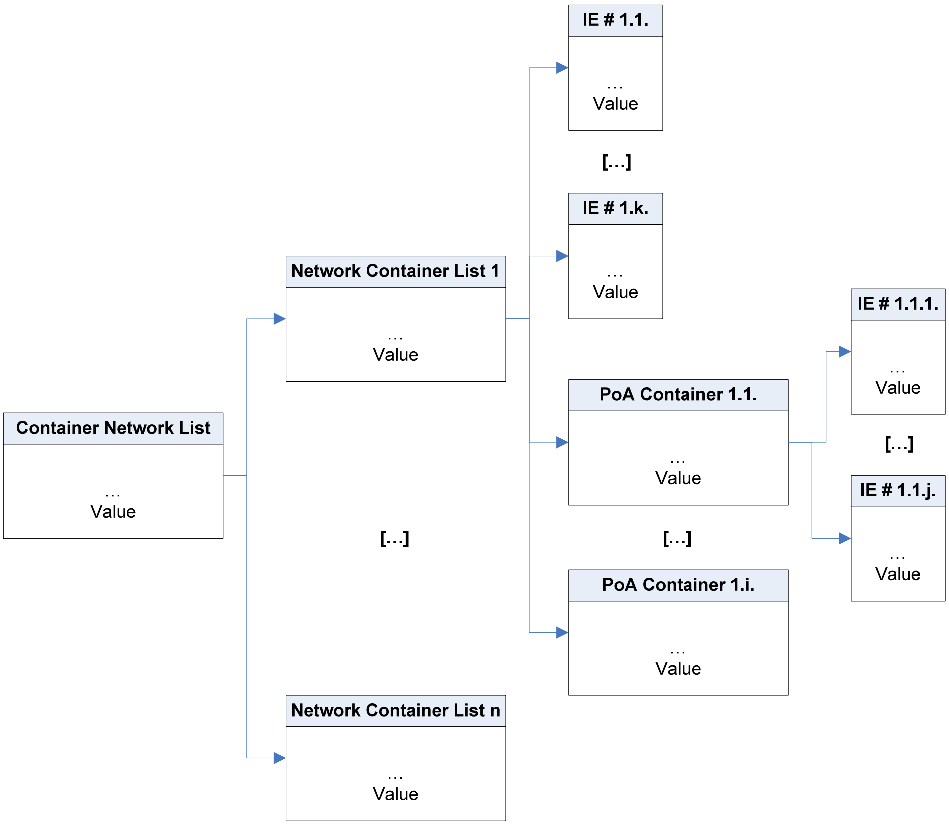


Figure 3.44: Hierarchal IE encapsulation

The development of the engine which provides the IEs as response has supposed a difference regarding to the implementation of the rest of the response engine in the PoA entity. The encapsulation of these IEs is done at user level, which means that most of the complexity of this procedure has been ported to this level. The principal difficulty is the size of the information generated. The encapsulation of single IEs does not produce any problem. It follows the general schema of almost all the TLV encapsulation in the applications. The difference appears when the volume of information exceeds 128 bytes per encoded IE. All encapsulations have less than this cipher, except most of those provided by the MIIS. The width of *Length* field in an IE or other TLV encoded element is just a byte. In general, the information within a container of IEs has a size larger than 128 bytes, so the width of the *Length* field needs to be larger than a byte. For this issue, specific structures and procedures have been coded in order not to lose the flexibility provided by the query/response method in the MIIS. This IE encoded information is handled as follows: the single IEs use a short structure, namely short container, which provides a byte of width to indicate the Length of the information within the packet, and a buffer of 128 bytes for the Value field; whereas the IE containers use a long structure, namely long container, which provides two bytes of width to indicate the Length of the information within the packet, and a buffer of 65536 bytes for the *Value* field. It has been calculated that the longest query responses fill a response network container list of about 500 bytes in the current development, so nine or more bits are needed to encode the *Length* field in natural binary format. Note that in a determined peer, these lengths are handled as natural binaries coded as unsigned chars or unsigned integers (in C/C++ notation), which is different from the coding of the *Length* field in the TLV procedure. At the end, a procedure takes the information into the programming language structures and codes them all as strings of bytes, performing a part of the payload of the MIH message.

3.4.3.4. AP beacon stuffing

The access point needs to provide an extended set of information to become a PoA, advising to nearby MNs that it has MIH compatible capabilities. This can be done by adding more IEEE 802.11 information elements into the beacon frames. The current subsection explains the way it has done via modifying properly the HostAP User Space Daemon (Hostapd).

Many files with different purposes can be found within the sources of Hostapd. Amongst them, there are *beacon.c* and *beacon.h*. These files provide the procedures which perform the contents of beacon and probe- response frames. In particular, *beacon.c* is composed of many different functions following this schema:

```
* u8 *function(struct hostapd_data *hapd, u8 *eid);
```

Where `eid` is a pointer to a generic address of memory. In this case, `eid` determines whether it has to write in the header or tail positions of a beacon or probe response frame. The pointer to `hostapd_data` structure is a label for a piece of memory that will attach more information to the header or to the tail inside the function. Functions following this pattern return a pointer to the next address of memory suitable to be filled at header or tail positions.

Following these premises, two functions are implemented in *beacon.c* in order to provide information about MIHF functionalities within the PoA. Each of them introduces a different information element in beacon frames thrown from an AP. These information elements are specific for beacons in IEEE 802.11 standard, and are added as tags at flexible header part of this type of frames. These tags use a type of identifier which is still unused, so only specific applications such as the MIH daemon for the MN can recognize them. The implemented functions are a copy of these ones with similar functionality within the beacon frames, but with the proper contents.

The new tags introduced are the MIHF identifier, solving by this way the initial MIHF identifier discovery issue, and a set of flags providing the information within the triggered Link_Detected indications into the MN stack. Further details on modifications in source code will be given in the following chapter.

3.4.4. Structure of the developed applications

In this part of the chapter, a description of the structure of the applications code will be provided. The following sections supply a qualitative description of the entities performing the design of both MN and PoA daemon applications running in each peer. A little code is shown in the following lines, which is just the prototype of some relevant functions. The aim is to embody the principal idea about the mechanisms implemented in every layer of the design that conform these applications.

3.4.4.1 Mobile node

3.4.4.1.1. Common structures and definitions

The code composing the mobile node frequently has to access to a set of data types, structures or procedures that are common to the different layers. In this implementation, there is defined the format of the data that is handled within the messages (as stated in IEEE 802.21 standard) and primitives invoked from the different entities. Moreover, different constants and macros are declared for its use as well as the values that the variables of these types can take. In the current implementation, the file containing the data types defined in the annex F of this standard is *21_data_types.h*. The set of constants, macros and structures commonly used is defined in *defines.cc* and *defines.h* files, together with the frequently used system libraries.

In order to ease the byte representation of the most basic data types, different variables have been declared. Typical sets of bytes are composed by one, two, four or eight bytes, and so they have been declared as `u8`, `u16`, `u32` and `u64` being unsigned data types of the corresponding lengths.

One of the most used patterns within the code is the followed by a MIH message. For this purpose, a structure containing the different fields of its header and payload has been declared. The header has been defined identically as in standard pages. The payload, which is a set of variable information in TLV format, has been declared as a buffer of memory. Its contents will be defined by different functions along the process of message composition. The name of the corresponding structure is `MIH_MESSAGE`.

Moreover, the file *defines.h* provides the declaration of structures such as the different containers needed to deal with MIIS, namely `CONTAINER`, which is the structure containing the binary received information from MIIS; `PoA_CONTAINER`, which is the structure containing the received information about a PoA from MIIS; `NETWORK_CONTAINER`, structure containing the received information about a Network from MIIS; and NETWORK_LIST_CONTAINER, the structure containing the received information about a set of networks from MIIS.

Finally, different structures have been declared in order to ease the operation of the current application. For instance, `LINKS_DISCOVERED` and `CAPABILITIES_DISCOVERED` are two structures used to store the information from MIH Discovery and MIH Capability Discovery procedures, respectively. The structure called `Serving_PoA_Information` stores the parameters required by the application regarding to the current attached PoA. Another example are the structures called `PoA_TUPPLE` and `PoA_TUPPLE_LIST`, which are the pair MIHF identifier – MAC address from a determined PoA, and the pattern of a data base containing several of them, respectively.

3.4.4.1.2. Layer 2

3.4.4.1.2.1. Hardware manager

This entity contains the basic procedures used to set the underlying interfaces as well as getting its parameters and some information about the received information. In particular, it can be divided into four main blocks: one has a set of procedures used to initiate the physical devices once the application has been started, another is a set of threaded functions able to perform the peer discovery processes, there is also a couple of procedures which perform authentication and association processes in IEEE 802.11, and finally a set of other tools dedicated to perform other operations related to the device management. Moreover, the structures containing the format of the headers at radio and MAC levels are defined by this entity. The files composing the hardware manager are *link_layer.cc* and *link_layer.h*. A summarized schema of this module can be seen in figure 3.45.

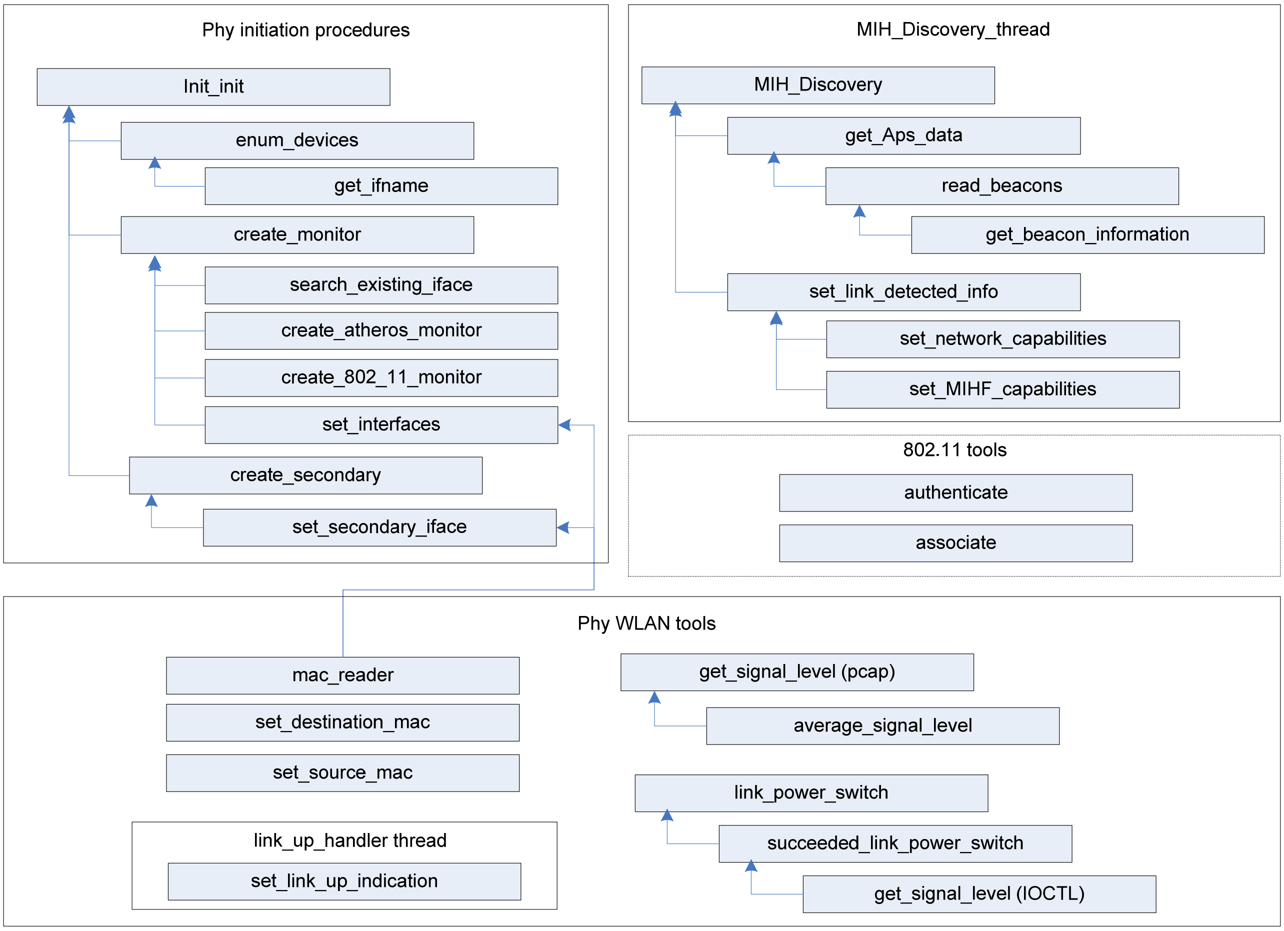


Figure 3.45: Summarized MN hardware manager coded schema

The physical initiation procedures are a set of functions called at startup, initiated with the `init_init` procedure. Its aim is to set properly the existing physical devices by determining which will take the role of a primary interface (if able to be set as monitor) and which will be an alternative interface. Three main procedures are called within this initiator. Their prototypes are the following:

* `void enum_devices (int      skfd, iface_data *ifc);`

Where `skfd` is a socket file descriptor to communicate user-space with the kernel, and `iface_data` is the structure of data requested to be filled with the required information of each virtual interface. This function uses other procedures which finally perform an *ioctl* call to request the needed info. The information of each interface is stored at */proc/net/wireless* folder within the file system.

* `bool create_monitor( char *mon, iface_vec ifc, u8 ii );`

Where `mon` is the destination name of the monitor interface concatenated with the current WLAN interface name described in `ifc[ii].iface_name`. The procedure searches via `search_existing_iface` whether the resulting name does exist as a monitor interface. If it does not exist, it is created through one of the two procedures designed to create monitor interfaces: `create_atheros_monitor`, for Atheros devices running under MADWiFi drivers, or `create_802_11_monitor`, for generic mac80211 device drivers. Finally, the configured interfaces are registered into a structure which determines which is the monitor interface designed for the IEEE 802.21 communication, and which is the managed interface handling the IEEE 802.11 protocol. The returned value determines if the interfaces have been set properly.

* `void create_secondary( iface_vec ifc, u8 ii );`

Where the MAC address under `ifc[ii].iface_name` is registered and will be the optional MAC address of the current multi-radio device.

The MIH Discovery thread is a complex procedure controlled by upper layers via thread handlers such as *mutex*, condition and boolean variables. It triggers, when it is run, the corresponding Link Detected indications for each surrounding PoA. This is done examining the set of beacon frames received using the tools supplied by *pcap* library. Two main procedures can be found here:

* `u8 get_APs_data( char *mon_iface, u8 nAllowedMacs );`

Where `mon_iface` is the identifier of the current interface handling the IEEE 802.21 protocol, and `nAllowedMacs` is the minimum number of MAC interfaces needed to be discovered. The value of return indicates how many PoAs have been discovered. Two options are possible: if the MIH Discovery procedure is done in the middle of a handover, the current function will not return until the number of discovered PoAs were the same as the quantity provided by the MIIS query response; if the MIH Discovery procedure occurs at bootstrapping time, it will return the PoAs discovered once, unless PoA no PoA is discovered (the MIH Discovery procedure would take place again, then). This function relies on the information provided by the function `read_beacons` (uses *pcap* habilities), which provides the information of each beacon via `get_beacon_information` (examines the contents into a beacon frame).

* `void set_link_det_info( LINK_DET_INFO &LinkDetInfo); void set_link_det_info( LINK_DET_INFO &LinkDetInfo, u8 itr );`

This is an overloaded function which allows calling it with different contents in its header. The first one sets the common parameters for all the indications needed to generate, and the second one is called after having received and determined the particular parameters of each of the discovered PoAs. `LinkDetInfo` is the data contained within the Link_Detected indication MIH Link SAP primitive, and the `itr` variable is the iterator used to choose the information regarding to each PoA. This function relies on `set_network_capabilities` and `set_MIHF_capabilities` to complete its operation.

Moreover, a pair of functions is designed to complete the IEEE 802.11 authentication and association procedures. They are called authentication and association respectively, and perform the corresponding actions that a user would do by introducing the corresponding *ifconfig* and *iwconfig* or *iw* commands via command line:

* `int authenticate( char *ifname, char *essid );`

Where `ifname` is the identifier of the interface which is going to be authenticated and `essid` is the name of the ESS the given interface is going to be authenticated with. The value of return is 0 if the authentication has succeeded or -1 otherwise.

* `int associate( char *ifname );`

Where `ifname` is the identifier of the interface which is going to be associated via getting a dynamic IP address from the AP. The value of return is 0 if the association has succeeded or -1 otherwise.

At last, a set of tools is provided in order to facilitate wireless devices handling. The following is a list of prototypes of the most relevant ones and a brief description of them:

* `void mac_reader( char *ifname, u8 *mac );`

This function gets the MAC address from the interface given in `ifname`, and stores that in `mac`. See reference GNU mac-changer 0.1.5 free software for further details that have inspired this procedure.

* `float get_signal_level( int skfd, char *ifname );`

This function reads and reports the received signal strength from a socket file descriptor `skfd` of a given interface called `ifname`. It uses an *ioctl* call to get this value, and is used by each triggered Link_Detected indication. The returned value is the received signal level from the corresponding monitored AP. See references on `iwconfig.c` and `iwlist.c` for further details that have inspired this procedure.

* `int get_signal_level( u8 idleTime, char *ifname, char *Essid );`

This function gets the signal level from beacon frames. It is useful to avoid difficulties with mac80211 dependent devices. The previous defined routine often returns 0 for a mac80211 device (almost until Linux 2.6.29 kernel versions), which is not a valid value. This routine is used for monitoring the state of the current link. `Essid` is the ESS which is going to be monitored, and `idleTime` determines the period of time between two samples. This function relies on `average_signal_level` to determine the average received signal strength in order to smoothen the received wave-form and avoiding some undesired ping-pong effects due to fast fading in signal. The returned value is the smoothed received signal strength.

* `int set_channel( char *ifname, int freq );`

This procedure takes the interface designed as `ifname` and sets it to the chosen channel called `freq` via *ioctl* calls. The returned value is 0 if the channel has been set properly or 1 otherwise.

* void set_source_mac(u8 *Header80211, u8 *source_mac);

Sets the source MAC address source_mac into an IEEE 802.11 header in Header80211.

* void set_destination_mac(u8 *Header80211, u8 *destination_mac);

Sets the destination MAC address destination_mac into an IEEE 802.11 header in Header80211.

* void set_Link_Up_indication (LINK_TUPLE_ID &LinkIdentifier, LINK_ADDR &OldAccessRouter, LINK_ADDR &NewAccessRouter, IP_RENEWAL_FLAG &IPRenewalFlag, IP_MOB_MGMT &MobilityManagementSupport);

Sets the information within a triggered Link Up indication into the corresponding MIH Link SAP. The regarding information had been previously supplied within the last received indications, and so the contents need just to be filled. This function is run into an ephemeral thread whose name is link_up_handler and is called from an upper layer.

* void link_power_switch(switch_t onoff);

Finally, this function helps Link_Actions request and confirm MIH Link SAP commands to switch on or switch off a given interface. The passed onoff parameter determines if the interface has to be switched on or off. This function relies on succeeded_link_power_switch and get_signal_level (the one that uses an *ioctl* call) to complete the information within the response primitive generated.

3.4.4.1.2.2. Event manager

The event manager is an entity which handles and reports the generated events regarding to the received signal level from the working wireless interface. It consists on a threaded core basically implemented in *events.cc* and *events.h* files, but takes some of the basic methods for handling wireless interfaces implemented in *link_layer.cc* and *link_layer.h*.

The main thread monitors the events that happen regarding to the changes in signal strength in the received signal. It determines, through a set of support functions, when to stop monitoring and start triggering actions which lead to calls to the MIH Link SAP. A set of functions is designed to deal with signal level: get_signal_level, switch_signal_level and classify_signal_level. The prototypes and functionalities are the following:

* int get_signal_level(u8 idleTime, char *ifname, char *Essid) (explained in the previous subsection);

* int classify_signal_level(int signal_strength);

Where signal_strength is the measure of the current received signal in [dBm] and provided by get_signal_level function. The return value is the sub-range where this level is classified.

* bool switch_signal_level(int &siglevel, u8 &LastState, LINK_TUPLE_ID &LinkId);

Where siglevel is the range provided by the previous function, LastState is the sub-range the signal was when the previous measurement was done, and LinkId is a part of the needed information from the link above to complete the corresponding triggered MIH Link SAP primitive, if proceeds. The returned value determines whether the thread has to continue monitoring or it has to stop temporary waiting for some actions to be taken.

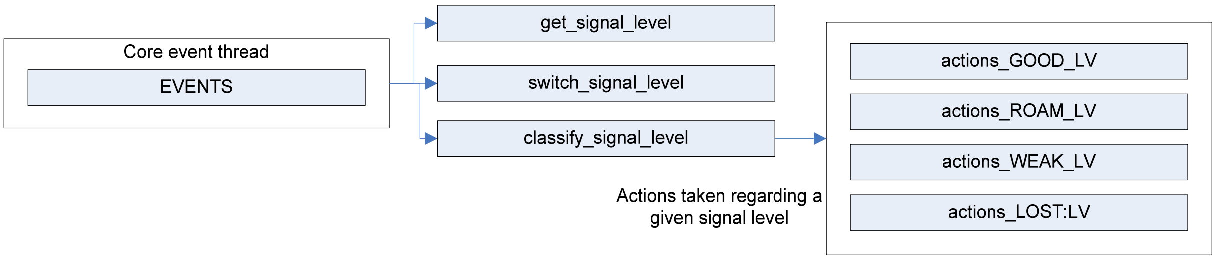


Figure 3.46: Summarized MN event manager coded schema

Moreover, there are four procedures which implement the actions taken regarding to the corresponding sub-ranges of signal when it proceeds. These functions carry the required calls to the proper MIH Link SAP primitives. Their prototypes follow the next profile:

* u8 actions_SUBRANGE_LV(required_parameters);

Where SUBRANGE refers to GOOD, ROAM, WEAK and LOST sub-ranges, and required_parameters are some of the parameters within the triggered SAP primitives into these functions. The return value is just the current sub-range.

Note that a set of thread handlers are provided to allow external entities re-starting the main thread of this module once the system needs to monitor the received signal strength again.

3.4.4.1.3. Layer 2.5

3.4.4.1.3.1. MIHF

MIHF is a logical entity that facilitates handover decision making. MIH Users make handover decisions based on inputs from the MIHF. This layer provides a common interface to upper layers and independent from the media below. Files composing this module are *MIHF.cc* and *MIHF.h*. Moreover, a set of procedures is provided into a *data_type_parsers.cc* and *data_type_parsers.h* in order to facilitate the conversion from a determined variable to a buffered element, and vice versa, for the destination MIH message payload.

This entity consists on a four-threaded core which represents each of the four services implemented in the MN (MIMS, MIES, MICS and MIIS). Once a local event from the underlying layer occurs, a command from an upper layer or a command from a remote MIHF arrives to the local MIHF, the corresponding thread is activated and triggers the required SAP primitive to cede the involved information to the correspondent entity. Note that this block is provided of a data segment with the MIHF level variables which compose the received and sent messages and primitives; and a signaling segment which has the required *mutexes*, condition variables and flags to handle the behavior of the of the multi-threaded core.

A set of functions has been implemented in order to parse the incoming MIHF messages from remote peers. Its aim is to convert the binary data into the payload of a MIH message into the corresponding MIHF data type for each message. The prototypes of these functions are the following:

* `int DATA_TYPE_received_MIH_message( MIH_MESSAGE Msg );`

Where `received_MIH_message` is applicable to the denomination of each of the received messages, and `MIH_MESSAGE Msg` is a structure containing the whole MIH message (header included). The value of return indicates if the data extraction has succeeded or not.

Another set of functions is also implemented to do just the reverse action. The outgoing messages into programming structures need to be converted to buffers of data in order to be handled by the *pcap* library and to be sent to remote MIHF peers. The prototype of these functions is the following:

* `MIH_MESSAGE MESSAGE_sent_MIH_message( MIH_MESSAGE_data );`

Where `sent_MIH_message` is applicable to the denomination of each of the sent messages, and `MIH_MESSAGE_data` are the data handled into each sent message. The value of return is a `MIH_MESSAGE`, which has a structured header, but a buffered payload.

The services requiring messages to be received from remote peers (MICS, MIMS and MIIS) use a function which implements the needed methods. Actually, the function used has the following prototype:

* `MIH_MESSAGE receive(u16 TransactionID, u8 ServiceID, u8 OperationCode, u16 ActionID, bool ignore_bcst);`

Where the elements related to the MIH header are used to determine which kind of message needs to be received as response, and the boolean `ignore_bcst` is a flag used to determine if broadcast packets are needed to be ignored. The return value is the `MIH_MESSAGE` with a structured header, but a buffered payload.

Additional functions are used by receive function in order to ease modularity. `Read_80221_packet` is the function used to extract the MIH message from the whole incoming physical packet. At the same time, it relies on a parser for the MIH header and a parser for the message identifier.

Moreover, additional procedures have been described in order to ease the codification or decodification of *Length* and *Value* fields of structures composed by both elements. For instance, `define_length_of_Length` is a function which implements the *Length* field of a TLV. It is returned as an array from 1 to the needed `u8` elements describing the width of the *Value* field in the TLV, and it also returns the length as an `u32` variable passed by reference. Furthermore, `define_length_of_Value` is another function which is implemented to find out the total number of bytes that an element occupies. This function is applicable to data types which have *Length* and *Value* fields.

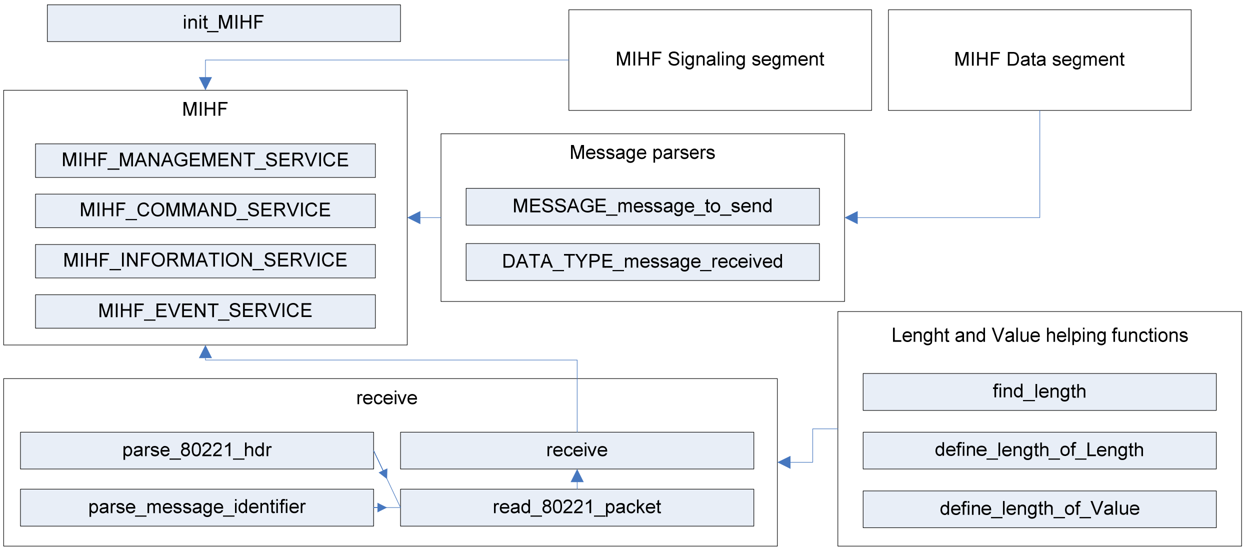


Figure 3.47: Summarized MN MIHF coded schema

Finally, some features about the data parsers should be pointed out. These data parsers are used by message parsers in order to ease the coding. Similar schemas are repeated in each message or SAP primitive, so a file summarizing them is mandatory. These functions have been divided into two groups: one provides conversion from the internal data into the SAP to the same data within a buffer ready to be used as the payload into a MIH message, and the other one does the reverse action extracting a data structure from a buffered payload. The headers can be summarized as follows:

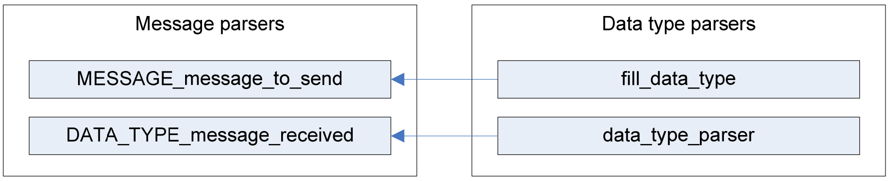


Figure 3.48: Summarized MN MIHF parsers coded schema

`*u8 fill_data_type( u8 *pu8, DATA_TYPE data );`

This is a converter from data to buffer, where pu8 is a pointer to the position of the buffer to be filled, and data is a generic way to design one of the data types defined under IEEE 802.11 standard, as well as data_type. The returned value is the number of bytes written into the buffer.

* u32 data_type_parser(DATA_TYPE *data, u8 *pu8);

This is a converter from buffer to data, where pu8 is a pointer to the next position of the buffer to be read, and data is a generic way to design one of the data types defined under IEEE 802.11 standard (as well as data_type). The returned value is the number of bytes read from the buffer. Moreover, the following prototype is an alternative for a reduced set of data types:

* DATA_TYPE *data_type_parser(u8 *pu8, u32 *offset);

The elements are the same, but in different position. The offset indicates the number of bytes read from the buffer, and the value of return is the data type read from the buffer.

3.4.4.1.3.2. MIH Register

The MIH register is the entity in charge of implementing the system to keep associated a determined MIHF identifier with its corresponding MAC address. The files composing this entity are *MIH_Register.cc* and *MIH_Register.h*. The methods provided within are available for upper layers and are useful to register a remote MIHF PoA in case it could be necessary.

This module is basically composed of a dynamic data base which stores structured data that bounds MIHF identifiers with MAC addresses from remote peers. This list is acceded via two functions which allow introducing new tuples of elements and reading them, in case it is needed. These are their prototypes:

* u8 store_POA_TUPLE(POA_TUPLE poa_tuple, POA_TUPLE_LIST *list);

Where poa_tuple is the pair of addresses of the different layers, and list is the dynamic data base. The returned value is 0 if the element is present in the list or 0 otherwise.

* u8 retrieve_mac_addr(u8 *mac_addr, MIHF_ID *mihf_id, POA_TUPLE_LIST *list);

Where mac_addr is the MAC address and mihf_id is the MIHF identifier of the remote MIHF PoA; and list is the dynamic data base. The function looks up for the given MAC address and returns 1 if it has been found, or 0 otherwise.

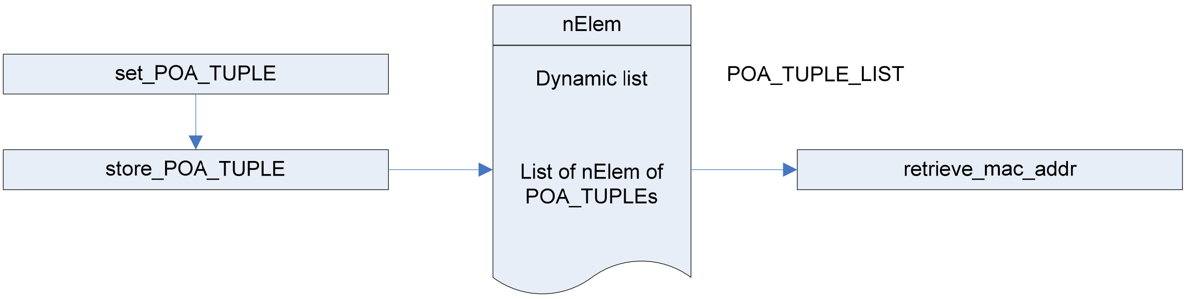


Figure 3.49: Summarized MN MIH Register coded schema

3.4.4.1.3.3. State machines

This entity gathers the state machines defined in Chapter 8 of IEEE 802.11 standard. The implementation proposed is based in classes that can be instantiated and can be run as independent threads. A set of common functions declared in the standard is also provided and used into these classes as friend functions. The files containing the implementation of this entity are `state_machines.cc` and `state_machines.h.`

The approach is the following: there is a base class, called Transaction State Machine (TSM) which implements the common attributes and methods for all the different instances of the required state machines, including the elements for a Transaction Timer State Machine (TTSM), Ack Requestor State Machine (AReqSM) and Ack Responder State Machine (ARspSM). This base class is not allowed to be instantiated directly. Instead, two derived classes are defined: Transaction Source State Machine (TSSM) and Transaction Destination State Machine (TDSM). These classes are instantiable and provide a set of methods which can be used externally (encapsulation design) to access the required member variables.

The instances of the referred classes are entities which run independently as threads, and so they are executed through their corresponding handler once they have been instantiated and they are required via a public method. These handlers have the structure of a thread and execute the routine containing the “Run” code. The proceeding is the next: once one TSSM or TDSM has been instantiated and initially set, it is started. The method that starts the thread calls the handler of the corresponding thread and then it executes the routine with the main contents of the state machine. Once the state machine ends (the thread finishes its execution), the handler exits the thread and so the thread can be joined.

The three of the blocks composing the TSM have defined different methods for different purposes. First, a set of three methods is provided for procedural handling. Their headers are the following:

* `void Run_state_machine();`

It is the routine to be executed when the thread is started. The clause `state_machine` stands for `AReqSM`, `ARspSM` or `TTSM`.

* `void Start_state_machine();`

It is the call used to start the thread. The clause `state_machine` stands for `AReqSM`, `ARspSM` or `TTSM`.

* `void Join_state_machine();`

It is the call used to end the thread. The clause `state_machine` stands for `AReqSM`, `ARspSM` or `TTSM`.

Second, a set of methods is defined in order to embrace the actions executed in each state of the state machines. Third, a set of functions is provided to handle the secure access and modification of the attributes via *mutex* encapsulation. Last, a set of methods is provided to get and modify the stated attributes once they have been safely acceded.

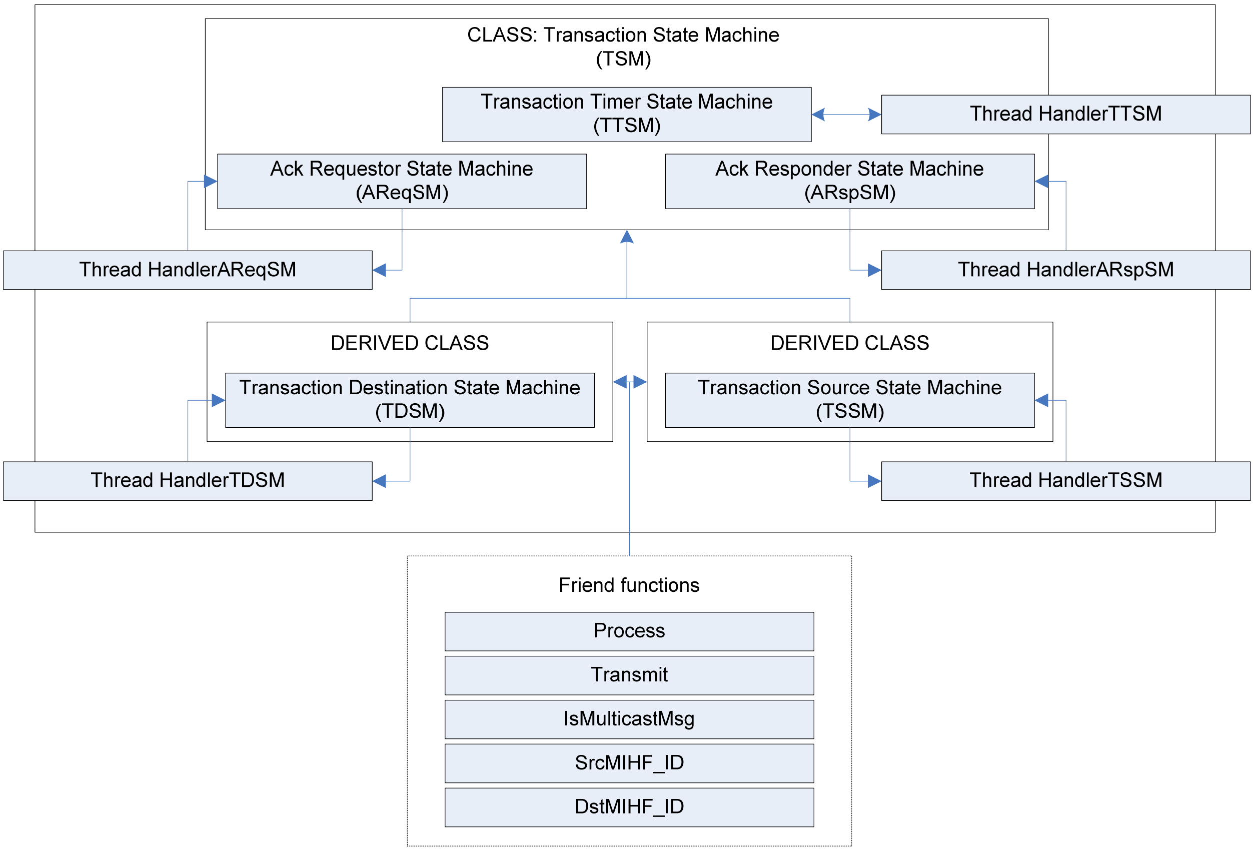


Figure 3.50: Summarized MN State Machines coded schema

A brief description of the so called inter-state procedures that have been used within the state machines as friend functions is given next. The operation matches with the provided at IEEE 802.21 standard pages:

* `BOOLEAN Process(MIH_MESSAGE MsgIn, MIH_MESSAGE MsgOut, BOOLEAN IsMulticast);`

This procedure processes the incoming message passed as an input variable. A value of `TRUE` is returned if an outgoing message `MsgOut` is available in response to the incoming message `MsgIn`. Otherwise, a value of `FALSE` is returned.

* `void Transmit( MIH_MESSAGE MIH_msg );`

This procedure transmits the message passed as the input variable `MIH_msg` by the wireless channel via *pcap* related proceedings.

* `BOOLEAN IsMulticastMsg( MIH_MESSAGE MIH_msg );`

This procedure outputs `TRUE` if the input message `MIH_msg` has a multicast destination MIHF_ID. Otherwise, it outputs `FALSE`.

* `MIHF_ID *SrcMIHF_ID( u8 *pu8, u32 *offset );`

This procedure obtains a source Identifier TLV from the message passed as the input, which is pointed by `pu8`, and returns the value of the TLV, understanding MIHF identifier by value. The parameter `offset` is the number of bytes read regarding to `pu8` positioning.

* `MIHF_ID *DstMIHF_ID( u8 *pu8, u32 *offset );`

This procedure obtains a destination Identifier TLV from the message passed as the input, which is pointed by `pu8`, and returns the value of the TLV, understanding MIHF identifier by value. The parameter `offset` is the number of bytes read regarding to `pu8` positioning.

3.4.4.1.4. Upper layers

3.4.4.1.4.1. MIH user

The MIH User is the entity in charge to take handover decisions. The intelligence of the system in the MN is allocated in this layer; so many different decisions are performed here besides the handover related ones. The MIH User entity is composed of *MIH_user.cc* and *MIH_user.h* files. Its intelligence relies on the bootstrapping and decision manager entities, which are part of the user level and will be explained in the next subsection.

The principal feature of the MIH User entity is that it has allocated the main procedure for the application. This is the first function called once the daemon has been started, and has the aim of initiating and enabling the subsystems that compose the application. The actions performed are the following: first, it gets the basic system parameters from lower layers; then it initiates these layers (link level, MIHF level and User level, in this order); afterwards it starts the threads that compose the core of the application (MIH User, MIHF services, MIH Discovery and Event manager threads); and finally, it performs the initial MIH Discovery.

This entity is provided of a segment of data with different purposes. First, there is a part where variables related to thread handling are declared, such as those used to synchronize the MIH user thread and the MIH Discovery procedures. There is also a set of IEEE 802.21 variables which are part of the own user level parameters (initiated with the corresponding MIH User routine) and a set of variables of the same nature used to fill the payload of the outgoing MIH SAP primitives or extracted from those ones.

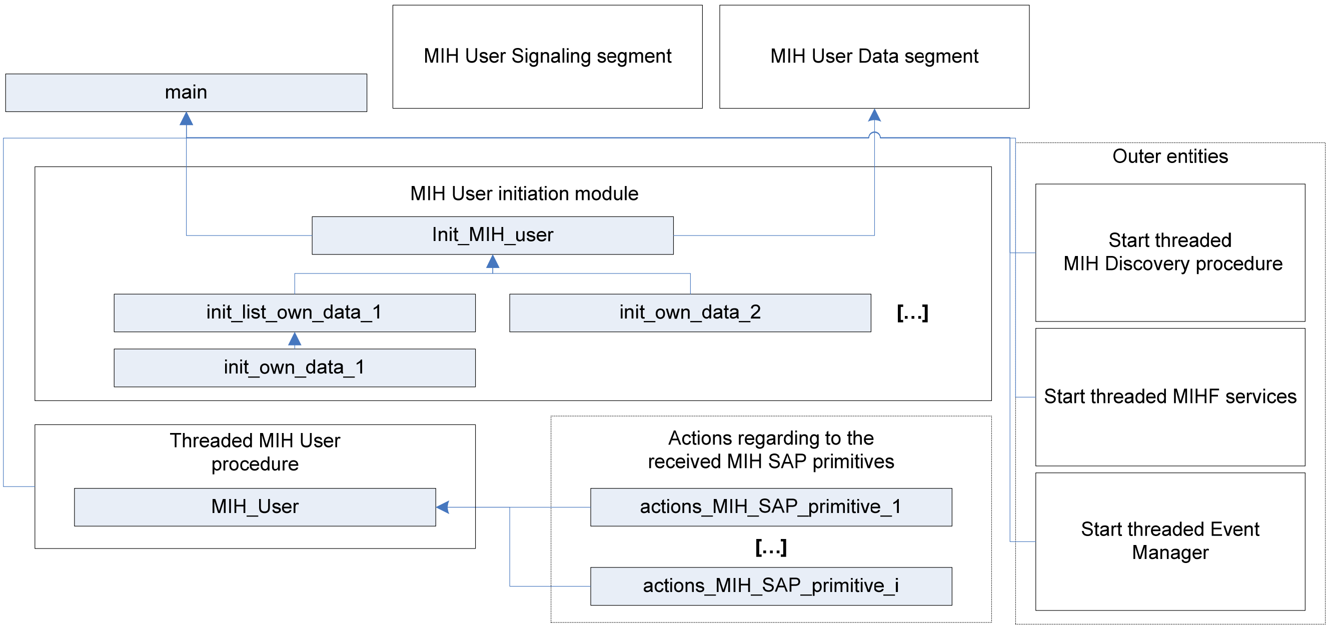


Figure 3.51: Summarized MN MIH User coded schema

The block that starts the MIH User uses an extended set of functions in order to initiate the corresponding information of the top layer. The own MIH User information is defined within those functions, so any change in the MIH user characterization should be introduced on them. Further developments should consider introducing a Management Information Base (MIB) and extracting the required information from it. By now, this is the prototype of the used functions:

* `void init_own_data( DATA_TYPE data1, … );`

Where `data1` is the own data initiated (can be one or more), and `own_data` stands for the type of IEEE 802.21 data which corresponds to `data1`. A variation of this profile is the following:

* `DATA_TYPE init_own_data( DATA data1, … );`

The difference is that the information passed within the parenthesis refers to heterogeneous data not necessarily being defined under the standard and the returned value is the own data initiated (which is in IEEE 802.21 format).

Sometimes, these functions are also used by intermediate functions which request a list of elements of the same nature. Their prototype is the following:

* `LIST<DATA_TYPE> init_list_own_data( DATA data1, … );`

Where the information passed within the parenthesis refers to heterogeneous data not necessarily being defined under the standard and the returned value is the list of own data initiated (which is in IEEE 802.21 format).

The MIH User is controlled by a threaded management block which handles the received MIH SAP primitives by triggering the corresponding MIH user action and calling to a determined MIH SAP primitive, if needed. There is an action defined for each received primitive and for a set of given conditions. These actions have been implemented as functions which follow the next prototype pattern:

```
* DATA actions_MIH_SAP_primitive( DATA data1, … );
```

Where the data within parenthesis can be an IEEE 802.21 data or not, and so the returned value. The inner procedures are heterogeneous, and so they depend on the conditions given at a determined moment.

3.4.4.1.4.2. Bootstrapping and target decision manager

The decision manager is composed of two basic entities which handle the bootstrapping and handover decisions. These two entities use three sub-proceedings, one individual for each one and one that is shared. The bootstrapping module is provided by the services of another module which performs the candidate link decision, and another one which performs the candidate decision. Moreover, a set of procedures is provided with the aim of setting the required configuration after bootstrapping. The handover decision module is fed by candidate decision and MIIS decision sub-procedures. A set of procedures is also provided in order to set the required configuration after a decision has been taken. The code of these entities is given in *decision_manager.cc* and *decision_manager.h* files.

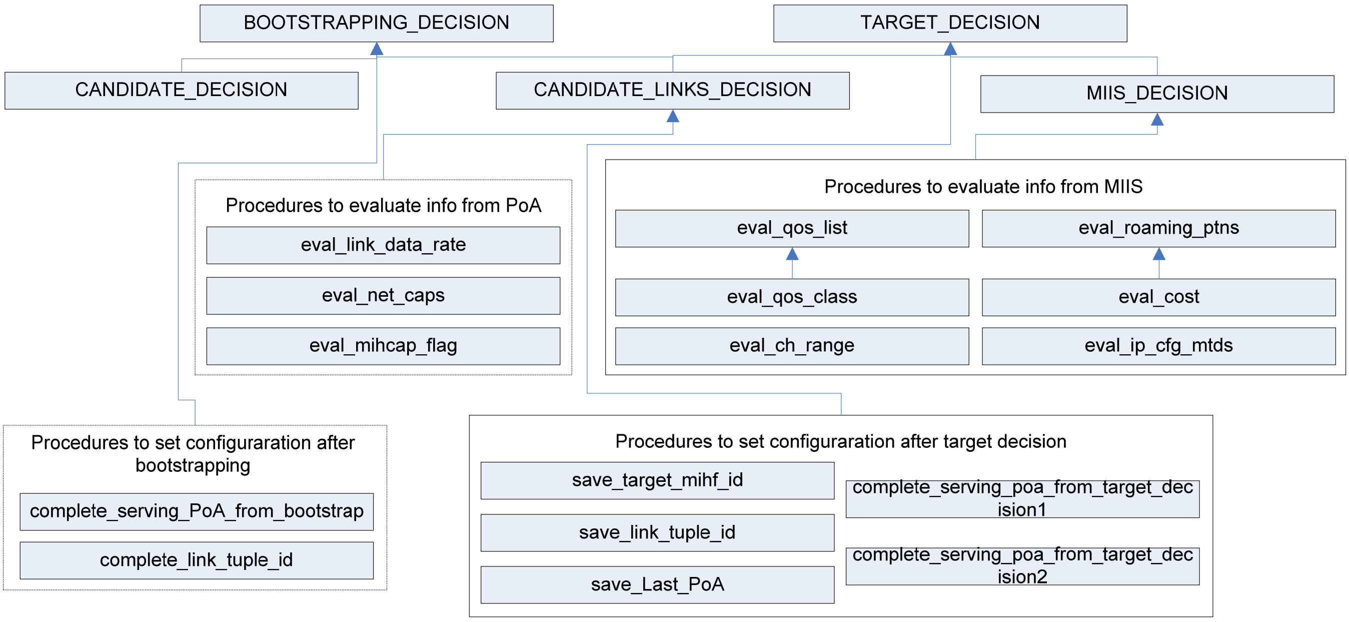

BOOTSTRAPPING_DECISION
TARGET_DECISION
CANDIDATE_DECISION
CANDIDATE_LINKS_DECISION
MIIS_DECISION
Procedures to evaluate info from PoA
eval_link_data_rate
eval_net_caps
eval_mihcap_flag
Procedures to evaluate info from MIIS
eval_qos_list
eval_roaming_ptns
eval_qos_class
eval_cost
eval_ch_range
eval_ip_cfg_mtds
Procedures to set configuraration after bootstrapping
complete_serving_PoA_from_bootstrap
complete_link_tuple_id
Procedures to set configuraration after target decision
save_target_mihf_id
complete_serving_poa_from_target_decision1
save_link_tuple_id
complete_serving_poa_from_target_decision2
save_Last_PoA

Figure 3.52: Summarized MN decision entities coded schema

The set of procedures called in the candidate links decision module are used to evaluate the information within the received beacons from those APs which are compatible with the IEEE 802.21 standard. The prototypes follow the next pattern:

```
* DATA eval_capability( DATA *data1, … );
```

Where the `data` into parenthesis can be an IEEE 802.21 data or not, and so the returned value.

The set of procedures called within the MIIS decision module are used to evaluate the information that the query to the information server provides through MIH Get Information messages. The prototypes follow the same pattern as candidate links decision ones.

3.4.4.1.5. SAPs

The SAPs are implemented as a set of functions that can be called from the adjacent layers of the stack. This implementation considers a media independent SAP, the MIH_SAP, and a media independent SAP, the MIH_LINK_SAP. These entities are defined in *MIH_SAP.cc* and *MIH_SAP.h* files, and in *MIH_LINK_SAP.cc* and *MIH_LINK_SAP.h* files, respectively. Moreover, *SAP_parser.cc* and *SAP_parser.h* are files provided in order to ease the task of transmitting a determined data type form those included in the header of a SAP primitive to the corresponding layer of the stack. The structure of the functions contained as SAP parsers is the following:

```
* DATA_TYPE *copy_data_type( DATA_TYPE *data );
```

Where `data` stands for the data needed to be transmitted to a higher or lower layer, and `data_type` is the corresponding IEEE 802.21 type for this data. The returned value is the copy of the data. The following structure is also followed as alternative:

```
* void copy_data_type( DATA_TYPE copy_data, DATA_TYPE data );
```

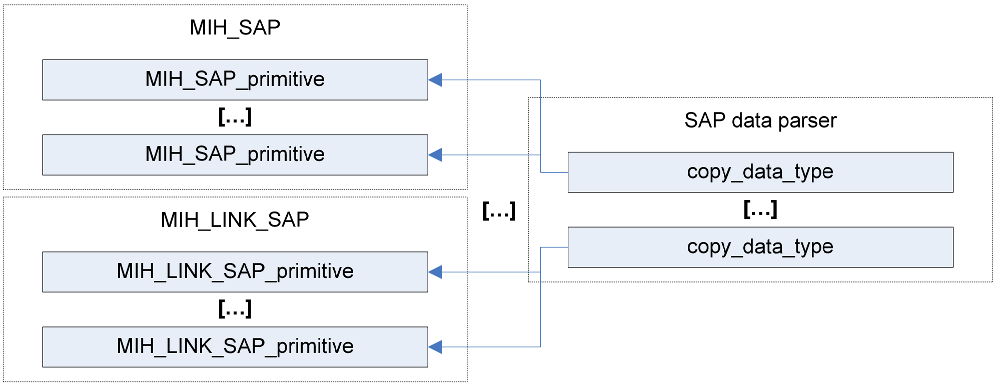


Figure 3.53: Summarized MN SAPs coded schema

Where `copy_data` is the destination copy and `data` is the source data.

3.4.4.1.5.1. MIH link SAP.

MIH_LINK_SAP is the entity containing the MIH Link SAP primitives. These primitives follow the next prototype pattern:

* `void mih_link_sap_primitive( DATA_TYPE_1 data_1, … ) ;`

Where `mih_link_sap_primitive` is the name of the corresponding primitive, and `data_1` and the subsequent data are the IEEE 802.21 data transmitted from layer to layer.

3.4.4.1.5.2. MIH SAP

MIH_ SAP is the entity containing the MIH SAP primitives. These primitives follow the next prototype pattern:

* `void mih_sap_primitive( DATA_TYPE_1 data_1, … ) ;`

Where `mih_sap_primitive` is the name of the corresponding primitive, and `data_1` and the subsequent data are the IEEE 802.21 data transmitted from layer to layer.

3.4.4.2. Point of attachment

3.4.4.2.1. Common structures and definitions

The PoA uses a set of common structures and definitions which are also common with those used for the MN. Some slight differences are included such as a reduced set of particular macros and constants which are particular to a PoA. In future developments, these common structures and definitions should be unified under a unique set of files common for both MN and PoA. These files receive the same name as those in the MN, namely *21_data_types.h*, *defines.cc* and *defines.h*.

3.4.4.2.2. Layer 2

3.4.4.2.2.1. Hardware manager

This module provides a reduced set of functionalities which are common with those used in the corresponding entity of the mobile node. There is a subset of functions which allow initiating the wireless interface needed to provide IEEE 802.21 services. Its functionality is similar to the same module in MN, but simpler because just one interface is handled. Moreover, this block provides the set of headers required for the transmitted and received messages and a set of routines used to ease MAC address management. The files containing these resources are *link_layer_support.cc* and *link_layer_support.h*.

The procedures within the physical initiation are almost the same as those stated for the MN. The only change is the part that regards to the number of physical devices used. The block that provides support for MAC addresses handling has the same `set_source_mac` and `set_destination_mac` as in MN code. The other pair of functions are new and are characterized as follows:

* void store_packet(u8 *pu8, u8 *store);

Where pu8 points to the buffer containing a packet received with *pcap*, and store is a pointer to a buffer which will be the container of the received MAC packet. The function saves a copy of the last packet remaining received in a buffer used by a *pcap* capture for future extraction of its MAC addresses.

* void retrieve_source_mac(u8 *stored_packet, u8* source_mac);

Where stored_packet is the buffer containing the packet saved by the previous procedure above and source_mac is a pointer to the data where the source MAC address of the given packet will be saved.

This set of functions is used by the module of transmission defined by Transmit procedure, which is within the transaction state machines explained in a previous subsection.

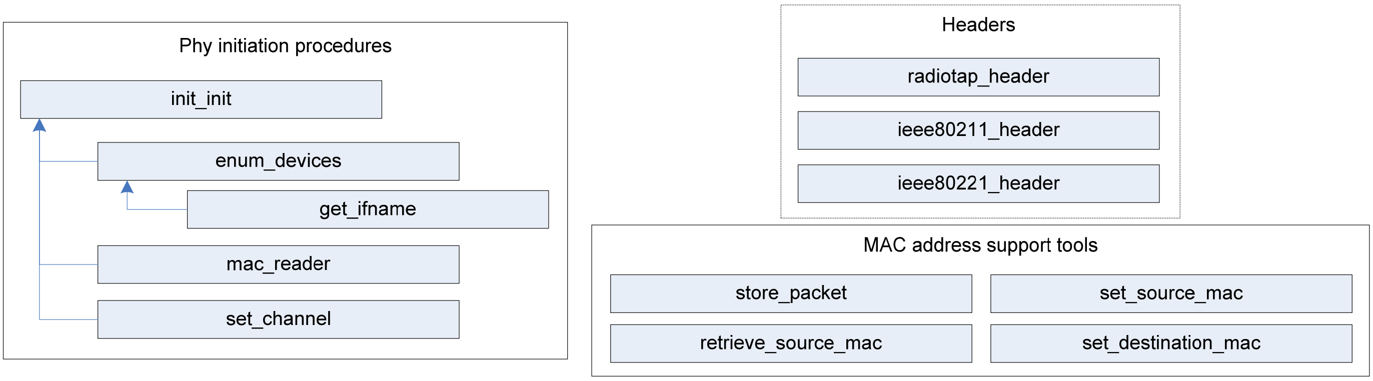


Figure 3.54: Summarized PoA Hardware Manager coded schema

3.4.4.2.3. Layer 2.5

3.4.4.2.3.1. MIHF

The MIH function in the PoA has several similarities with the so called implemented in the mobile node. The set of functions used to receive MIH messages and the support functions that supply help in determining length and value fields are almost the same. The structure of the procedures used to perform MIH messages is also the same, as well as the MIH data parsers included. There is also a segment of variables which control the synchronization between threads performing the core of the MIHF, and a segment of MIH data for the incoming and outgoing messages or data within MIH SAP. The main differences come with the implementation of the MIHF core and the TCP/IP over the wired network capabilities. The files that compose this MIH function are *MIHF.cc* and *MIHF.h*. The IEEE 802.21 data and message parsers are included into *data_type_parsers.cc* and data *type_parsers.h*.

The MIHF core works slightly different respect to the same entity in the MN due to the fact that the PoA acts as a passive element that responds to the messages received whereas the MN is an entity which initiates all the transactions. The implementation is based in a main thread called `MIHF_SERVICE_MANAGER` which starts MIMS, MICS and MIIS services. Moreover, this thread executes two procedures: the first one is an always active threaded TCP server which waits for possible incoming messages from the wired network; and the second one is a procedure which also executes a threaded routine expecting to receive a message from the wireless media. This thread uses the `receive` primitive, which has been commented within the code of the MN. This implementation allows being aware of the messages received by both interfaces at the same time, and responding them or triggering the required MIH SAP primitive in case it is required.

Finally, it should be noticed that another introduced element is a TCP client. This entity is called by the `Transmit` routine defined in the state machines module and is used to begin a transaction with a remote PoA. This element exists only when a wired transaction is required.

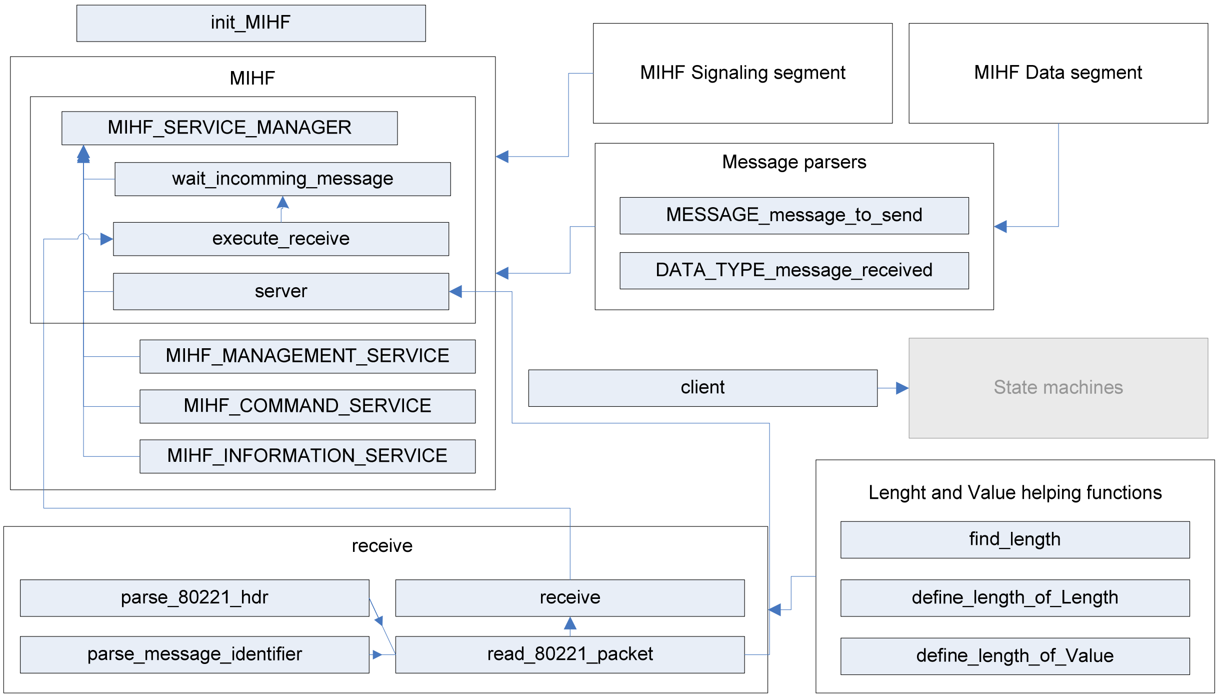


Figure 3.55: Summarized PoA MIHF coded schema

3.4.4.2.3.2. MIH Registrar

This entity is composed of two basic units: the Registrar data base itself and the mapping table for the surrounding PoAs. The files containing these structures are *registrar.cc* and *registrar.h*.

The Registrar is an array of pointers to structures which stores the MIHF identifier and the link address of a registered MN. The elements of this data base are instances of a class which are generated and deleted depending on the MNs that are registered or unregistered, respectively. A general sequence number for the registrar and a particular queue identifier for each registered element are given in order to ease MIH Registrar management.

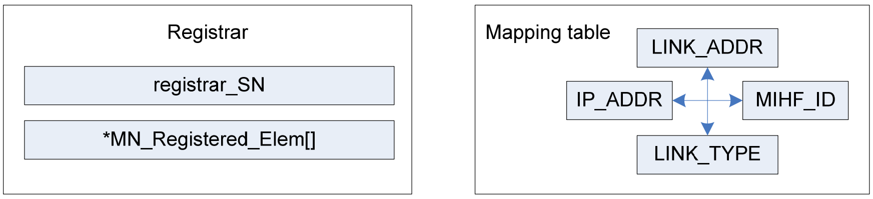


Figure 3.56: Summarized PoA Registrar coded schema

The mapping table is a pre-defined registry which bounds the link address, MIHF identifier and the IP address of the surrounding PoAs. This table must be filled dynamically when a new PoA is required to be contacted from a determined network to another one. Nevertheless, as stated in previous sections, this involves introducing a PoA discovery protocol, which is out of the scope of this work.

3.4.4.2.3.3. State machines

The implementation of the entity containing the transaction state machines is identical to the one stated for the mobile node. In fact, the development was initially set out for the mobile node implementation, whereas the PoA had a simpler one, but not fully functional and IEEE 802.21 alike. Finally, the development of this entity in the MN was moved to the PoA. Just a few differences had to be introduced in the Transmit procedure in order to distinguish and act according to a transmission by the wireless channel or over a wired TCP/IP protocol. The files containing this entity are *state_machine.cc* and *state_machine.h*.

The prototypes for the functions which compose the transmission unit are the following:

* `void WirelessTransmit( MIH_MESSAGE MIH_msg );`

* `void WiredTransmit( MIH_MESSAGE MIH_msg );`

Where `MIH_msg` stands for the MIH message which is going to be transmitted, in both cases. These two functions are used by `Transmit` procedure, which is defined in chapter 8 of IEEE 802.21 standard.

3.4.4.2.4. Upper layers

3.4.4.2.4.1. MIH user

The MIH user at the PoA has the same functionality as commented for the MN. In fact, it is composed by the same blocks of procedures but adapted to the received information from the particular MIH SAP primitives and introduced in it. No intelligence mechanisms have been defined for PoA implementations (because the decisions are taken in the MN) and so the MIH user generates the corresponding response to each received request. The files composing this entity are *MIH_User.cc* and *MIH_User.h*. The upper layer called MIIS exists in parallel to this entity in the protocol stack and will be explained in the following subsection.

Amongst the set of initiator functions used by the `init_MIH_user` procedure, there are two of them that can be highlighted. One is `init_map_table`, which is the function in charge of introducing the correspondences between link address, MIHF identifier and IP address of a given PoA in the surrounding networks. The other one is init_IpServAddr, which assigns an IP address to the server module of the local PoA. The rest of them set the initial values of the MIH user IEEE 802.21 parameters in the local PoA.

The MIH user is also provided of the main procedure of the application. As well as in the MN, it calls the main initiation procedures of each layer and after that starts the required threads for the entities running. In this case, just a pair of threads is needed: the thread which handles the MIH User queries and triggers the corresponding responses via the corresponding action procedure, and the thread which starts the MIHF_SERVICE_MANAGEMENT. As stated, the MIHF services are started within this procedure, so the main function is freed from this task. The figure 3.57 below summarizes the structure of the MIH user entity in the PoA.

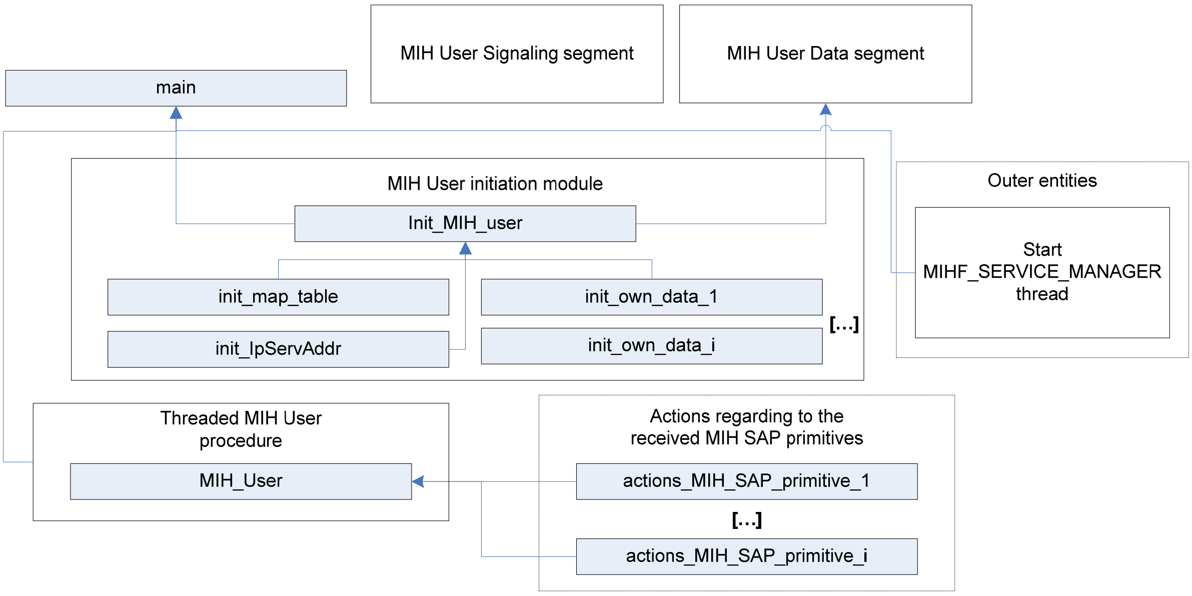

MIH User Signaling segment
MIH User Data segment
main
MIH User initiation module
Init_MIH_user
init_map_table
init_IpServAddr
init_own_data_1
init_own_data_i
[...]
Outer entities
Start MIHF_SERVICE_MANAGER thread
Threaded MIH User procedure
MIH_User
Actions regarding to the received MIH SAP primitives
actions_MIH_SAP_primitive_1
[...]
actions_MIH_SAP_primitive_i

Figure 3.57: Summarized PoA MIH User coded schema

3.4.4.2.4.2. MIIS user.

The MIIS user is an entity which provides the queried information from the MN. This is composed of a segment where the information elements regarding to the surrounding networks are declared, a procedure which initiates all of them, and a function to handle them. The files composing this entity are *MIIS_user.cc* and *MIIS_user.h*.

The segment of data is characterized by the declaration of the IEs that each network can provide. An IE_INFO structure is declared at the top of a list of IEs that can be queried for each network. This structure is filled when the MIIS is initiated via the corresponding function. First, some information is introduced into this structure, and after it is introduced in the payload of its corresponding information element, as well as its type and length fields.

A function is defined into this entity whit the purpose of composing the response container of IEs in response to a MIH_Get_Information request primitive from a MN. The function has the following prototype:

```
* CONTAINER set_BIN_RESPONSE( RPT_TEMPL IncomingRequest );
```

Where IncomingRequest is the template of IEs queried from the MN, and the returned value is a TLV container with the required encapsulation of the demanded IE hierarchy. This function introduces first the IES for a determined PoA in a determined network, and continues with the following IEs in the related PoAs of the same network. This mechanism is reproduced for the information of each surrounding network. When all networks have been reviewed, the final container is composed and then it is returned. This function relies in another one called insert_IE, which mechanizes the task of introducing the queried IE into the target container of IEs.

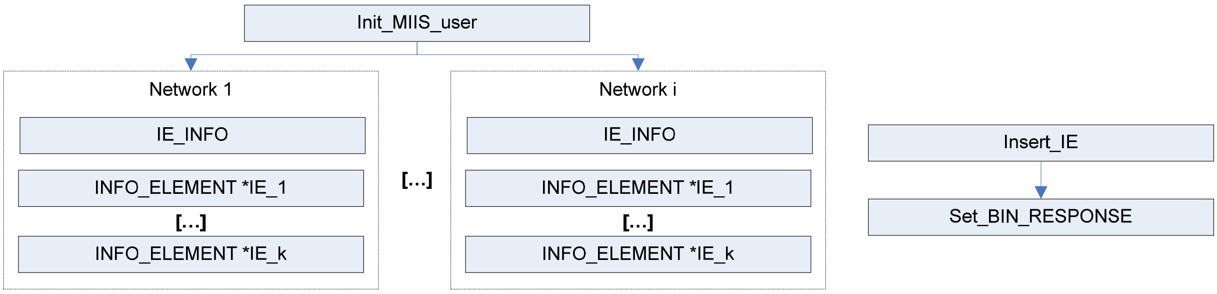


Figure 3.58: Summarized PoA MIIS User coded schema

3.4.4.2.5. MIH SAP

This SAP is implemented as a set of functions that can be called from the adjacent layers of the stack. The implementation considers just a media independent SAP, the MIH_SAP. This entity is defined in *MIH_SAP.cc* and *MIH_SAP.h* files. The files *SAP_parser.cc* and *SAP_parser.h* should be introduced in further developments in order to ease the task of transmitting a determined data type form those included in the header of a SAP primitive to the corresponding layer of the stack. Nevertheless, these SAP parsers are also implemented and its pattern is the same as stated for the SAPs regarding to the MN. The primitives for this SAP also follow the same pattern than the defined for the MN.

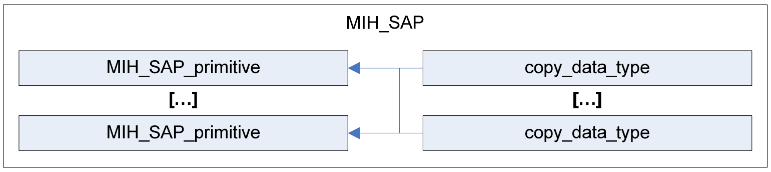


Figure 3.59: Summarized PoA MIH SAP coded schema

4. Validation and Results

In the present chapter, the applications developed are tested in order to validate its operation. First of all, a description of the installation process is provided, as well as the hardware and software requirements involved. After that, an example of a trace followed by the output debug console is provided and commented in order to validate the system behavior. Last, it is checked how the communication continuity is followed by tracking the path that a flow of packets follow and measuring the most representative times between events and other relevant metrics.

4.1. Installation

4.1.1. Installation requirements for MN and PoA daemons

4.1.1.1. Requirements

The following are the hardware and software requirements to install the developed applications into the peers in the role of a MN and a PoA. The system features in terms of CPU processor and required memory are not included in the next lines.

Common MN and PoA software requirements:

* OS Linux, kernel 2.6.28 *wireless-testing* branch from GIT repository or later.

* System libraries: *libpthread*, *libpcap*, *libiwlib* and *libncurses*.

MN particular software requirements:

* *iw* tool for *mac80211* WLAN device management.

MN particular hardware requirements:

* Two WLAN NIC capable of multiple interfacing and monitor mode. They can be an Atheros device using the old MADWiFi driver or another NIC whose driver is included into *mac80211* stack.

PoA particular software requirements:

* User-space Hostapd daemon, CRDA and related kernel entities (*mac80211* apps: *cfg80211*, *nl80211*, …)

* *dhcp3-server* packet from the repository.

PoA particular hardware requirements:

* A WLAN NIC settable to AP via Hostapd whose driver **is** included into mac80211 stack.

4.1.1.2. Test-bed used

The next sub-section provides the basic information about the machines used in the current test-bed and some key points about the assigned configuration to each of them. The basics about the peer features are given emphasizing the configuration of the WLAN devices they use.

4.1.1.2.1. Mobile node

In this case, the MN is composed of a notebook computer with two WLAN interfaces in order to provide seamless handover when it is required. Table 4.1 summarizes the most relevant features of it.

Role	Mobile Node	
Device	Notebook Acer TravelMate 3000 Series Intel Pentium M processor, 1.73 GHz 1 GB RAM	
Operating System	Debian 5.0, kernel 2.6.28	
WLAN extensions	WLAN device 1	WLAN device 2
Associated wireless MAC address	00:02:6F:2F:04:74	00:1C:f0:10:0E:6E
Associated wireless IP (private) address	Dynamically assigned (DHCP)	Dynamically assigned (DHCP)
Wireless hardware	Adapter Atheros AR5001X+ rev. 01 IEEE 802.11g Wireless LAN (miniPCI)	D-Link System DWA-140 802.11n Adapter
Wireless associated controller	ath_pci (MADWiFI) or ath5k	Ralink rt2870sta (Linux)
MIHF identifier	ruben@mn.net	

Table 4.1: MN configuration

The daemon running on the MN has been tested in both Ubuntu and Debian OS, providing best results in the last one. Problems with DHCP association occur in Ubuntu when more than one interface needs to be attached to different networks at the same time. In Ubuntu, it has not been possible to get a WLAN associated to an AP and the second WLAN associated to another AP simultaneously. Debian solves this bug, which does not depend on the MN daemon.

The running kernel on the current MN OS does not provide full support for the D-Link adapter in its *mac80211* stack. So, this device cannot provide multiple interfaces allowing a monitor interface and a common managed interface at the same time. Later versions (kernel 2.6.31 and later) permit multiple interfacing through *iw* tool. The MN daemon is ready to work dealing with all the *mac80211* devices as equal.

The driver which manages the Atheros WLAN in the current mac80211 kernel stack is *ath5k*. Previous versions of this controller (*ath_pci*) included in the MADWiFi project did not have an open source at all and its development has been abandoned. However, the daemon is ready to handle this module.

4.1.1.2.2. PoAs

In this case, PoAs are mounted on personal computers. They use the same type of NIC, which is settable as an AP and its configuration can be customized. Table 4.2 summarizes the most relevant features of these hosts.

Role	PoA 1	PoA 2
Device	Intel Pentium 4, 3 Ghz 2 GB RAM	NEC PlaTIC 2008 Bull Intel Core 2 vPro 4 GB RAM
Operating System	Ubuntu 8.10 or Ubuntu 9.04, kernel 2.6.30-rc4 *wireless-testing* branch	
WLAN extensions	WLAN device	WLAN device
Associated wireless MAC address	00:22:15:61:86:E1	00:22:15:61:86:95
Associated wireless IP (private) address	192.168.5.1	192.168.6.1
ESSID	whoah	yheah
Wireless hardware	Asus WL-138G V2. Network Controller Broadcom BCM4318 rev. 02 IEEE 802.11g Wireless LAN (PCI)	
Wireless associated controller	b43 + hostapd	
Associated wired IP (ethernet) address	147.83.47.181	147.83.47.179
MIHF identifier	whoah@poa.net	yheah@poa.net

Table 4.2: PoA configuration

Now, the chosen OS is Ubuntu. Versions 8.10 and 9.04 of this OS have been tested with similar results. Seemingly, the only difference that affects the current design resides in the allocation of the file containing the information of the network interfaces.

Differences between hardware are not appreciated in execution time. It has to be highlighted that originally the NEC computer was not available. Instead, there was a Fujitsu-Siemens computer running a slow Pentium 4 processor which lead to problems with the management of memory resources. These problems ended by introducing the NEC computer, which has more available capacity than the Fujitsu-Siemens machine.

4.1.2. Installation guidelines

4.1.2.1. Previous work: a new Linux OS installation

Before starting, it is necessary to get the sources of the current kernel version and have them allocated into /usr/src directory. It can be done downloading them via download manager such as *Synaptic* (GUI), *apt-get* (CLI) or *aptitude* (CLI). Once we have them, a *.config* file must be generated. This is possible by introducing:

```
    # make menuconfig
```

Maybe the system requires downloading *libncurses* development library before executing this instruction. The easiest way to obtain it is by searching it at *Synaptic* search bar and confirming its installation once it has been found. Note that just by entering and exiting the graphical menu to configure the kernel options, a new *.config* file has been generated at the current directory. The C language API *libncurses* is required to design applications with simple graphical components.

Installing the latest kernel version does not assure having a working wireless card. It is possible to see a wlan0 device by using wireless tools, but often it is not possible to bring it up because the interface does not exist into devices folder. To solve this problem, it is necessary the use of a tool called firmware cutter. The next line should be introduced into the shell command line to make b43 driver work in a proper way:

```
    # apt-get install b43-fwcutter
```

A firmware cutter is a tool used to extract the needed information from a driver designed to work in another OS (mainly in Windows OS) and to make it work into a Linux system. In particular, *b43-fwcutter* is the tool used to get the information from a Broadcom driver. Another popular firmware cutter is *ndiswrapper*.

4.1.2.2. Wireless-testing kernel installation from GIT repository

The wireless-testing kernel provides the latest updates for the WLAN devices handled under the *mac80211* stack. First, download its sources in */usr/src/* folder and uncompress them, if necessary:

```
    /usr/src# git clone
    git://git.kernel.org/pub/scm/linux/kernel/git/linville/wireless-testing.git
```

Build the configuration file .config following one of these two methods:

Method A) if the kernel needs to be configured manually:

```
    /usr/src# make menuconfig
```

First, it is needed to have installed *libncurses* package. Once into the menu, exit and save without modifying anything.

Method B) if the last working configuration of the kernel is needed to be kept, the working *.config* file must be copied:

```
/usr/src# cp /usr/src/Linux-source-KERNEL-VERSION/.config .config
```

Here, KERNEL-VERSION is the current kernel version. Then, the kernel has to be set with the old configuration, answering everything by default (press ENTER until the configuration process has finished):

```
/usr/src# make oldconfig
```

Next make kernel, modules, and install all of them:

```
/usr/src# make && make modules && make modules_install && make install.
```

Now it can be found the following files that correspond to the new kernel:

* */boot/vmlinuz-KERNEL_VERSION*

* */boot/System.map-KERNEL_VERSION*

And here KERNEL-VERSION stands for the new kernel version.

The file *initrd.img* is mandatory and required to map the modules. This file is built by this way:

```
/usr/src# mkinitramfs -o /boot/initrd.img-KERNEL-VERSION
/lib/modules/KERNEL_VERSION
```

Here, KERNEL-VERSION stands for the new kernel version.

Last, to access to the system via the new kernel it is necessary to modify the boot loader, which in this case is Grub, through one of the following methods:

Method A) by hand: copy the last working entry in */boot/grub/menu.lst* and update it with the files generated for the new kernel:

* kernel name: */boot/vmlinuz-KERNEL-VERSION*

* initrd: */boot/initrd.img-KERNEL-VERSION*

Here, KERNEL-VERSION is the new kernel version.

Method B) automatically: rename */boot/grub/menu.lst* to make the system do not recognize it at the next startup and save it in case of problem:

```
# /usr/src# mv /boot/grub/menu.lst /boot/grub/menu.lst.old
```

Build a new menu with the new data at */boot*:

```
# /usr/src# update-grub
```

Accept when the manager asks for a new menu to be created. Now restart the computer:

```
# /usr/src# reboot
```

If everything has succeeded, a new kernel version should be available to be chosen at startup. Grub menu should provide a new entry to select booting the new kernel in the current distribution.

4.1.2.3. Create a new AP with Asus WL-138G v2 PCI card

To create a new AP with the stated NIC, first the required software should be downloaded at */usr/src* folder. The programs are the following: Wireless regdb, CRDA, *libnl*, Hostapd and *iw*. They are available in GIT repository:

```
#   git clone git://git.kernel.org/pub/scm/linux/kernel/git/linville/wireless-
regdb.git

 # git clone git://git.kernel.org/pub/scm/linux/kernel/git/mcgrof/crda.git

 #    git clone git://git.kernel.org/pub/scm/libs/netlink/libnl.git

 # git clone git://w1.fi/srv/git/hostap.git

     # git clone http://git.sipsolutions.net/iw.git
```

These command lines download the stable versions, because they work right and do not require bug fixes. Development versions can cause some problems or require patches to be installed. Unstable versions are currently not used.

Once obtained the stated software, it must be installed. Begin installing *libnl*:

```
# cd libnl

 # autoconf

 # ./configure --prefix=/usr

 # make && make install

 # cd ..
```

Then, install CRDA. It requires *m2crypto* and *libgcrypt-dev* packages that can be obtained via download manager:

```
# cd wireless-regdb

 # mkdir -p /usr/lib/CRDA

 # cp regulatory.bin /usr/lib/CRDA/

 # cd ..

 # cd CRDA
 # make && make install

 # cd ..
```

Now proceed installing *iw*:

```
# cd iw

 # make && make install

 # cd ..
```

Finally, install Hostapd. It requires having installed *libssl-dev* package previously:

```
# cd hostap/hostapd

 # cp defconfig .config
```

The configuration of the Hostapd daemon must be set prior to launch the process. It is done through editing *.config* file doing the following:

```
# nano .config
```

1) First, uncomment:

```
CONFIG_DRIVER_NL80211=y
```

2) Now replace the following paths with the stated information. Do not forget to uncomment the useless lines:

```
LIBNL=/usr

 CFLAGS += -I$(LIBNL)/include

 LIBS += -L$(LIBNL)/lib
```

3) Then add:

```
CONFIG_LIBNL20=y
```

4) Last, uncomment:

```
CONFIG_IEEE80211N=y
```

Exit from *nano* editor and save. The last step is compiling and installing the program:

```
# make && make install

 # cd ../..
```

Once installed Hostapd, the desired parameters to run it need to be defined. The following sub-section provides some guidelines to do it and to run this application.

4.1.2.4. Configure and start Hostapd

First, it is needed to go to the directory which hosts Hostapd:

```
# cd /usr/src/hostap/hostapd
```

And now copy the configuration file to */etc*, folder which has most of the configuration files of the system:

```
# cp hostapd.conf /etc/
```

Edit the configuration file as it is needed. These are the basic parameters to configure:

```
interface=wlan0 ← mac80211 interface for devices using it.
 driver=nl80211  ← nl (netlink)
 ssid=whoah ← name of the generated private network.
 hw_mode=g  ← a, b, g are the options.
 channel=7  ← important, otherwise hw_mode=a. The channel must be 11 or less.
 auth_algs=1  ← open system authentication
```

Now exit the editor and save the modifications. NetworkManager and all related processes or other applications designed to manage the network connectivity must be suppressed in order to avoid conflicts:

```
# killall NetworkManager
```

Now, Hostapd is ready to be executed from shell:

```
# hostapd -dd /etc/hostapd.conf
```

Here, −dd means that the daemon shows by console the events and actions that occur when Hostapd is being executed. The path is where the configuration file is hosted in.

By this moment on, the computer which has Hostapd correctly running is ready to act as AP. It provides connectivity within the LAN designed in the configuration file. The assignation of the network IP addresses must be done by hand at each AP. To get IP addresses dynamically, a DHCP server must be installed in each computer which is in the role of an AP. The following sub-section explains how to do it.

4.1.2.5. Install and configure a DHCP server

One suitable package which can provide DHCP capabilities is *dhcp3-server*. It must be installed using a download manager, for instance:

```
# aptitude install dhcp3-server
```

Now it is needed to configure the file */etc/default/dhcp3-server* in order to indicate that *wlan0* is the associated interface to the DHCP service:

```
INTERFACES=”wlan0”
```

The configuration of the subnet parameters must be set into */etc/dhcp3/dhcpd.conf* file as follows:

```
ddns-update-style none;

option domain-name-servers 147.83.2.3, 147.83.2.10;

default-lease-time 86400;

max-lease-time 604800;

authoritative;

subnet 192.168.3.0 netmask 255.255.255.0 {

 range 192.168.3.2 192.168.3.254;

 option domain-name-servers 147.83.2.3, 147.83.2.10;

 option routers 147.83.3.1;

 option broadcast-address 192.168.3.255;

 default-lease-time 86400;

 max-lease-time 604800;

}
```

Once configured, the private network address must be associated to an interface. For instance:

```
# ifconfig wlan0 192.168.3.1
```

At the end, the process that starts the DHCP service can be launched:

```
# /etc/init.d/dhcp3-server start
```

In addition, in order to enable connectivity with outer networks the system must be configured to allow forwarding packets by setting to 1 the required file:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Finally, it is needed to configure *iptables* in order to handle network address translations (NAT) in a proper way. The *iptables* command is the following:

```
# iptables -t NAT -A POSTROUTING -o eth0 -j MASQUERADE
```

The command is used to map the addresses of packets that go through interface *eth0* to an outer network. The target MASQUERADE is used to identify a source address that is assigned dynamically, and the address translation is applied on these ones.

4.1.2.6. Hostapd and IEEE 802.21 compatibility: beacon stuffing

In order to provide MIH capability awareness, the daemon Hostapd must be modified. If Hostapd is running, it must be stopped. Then, *beacon.c* file must be modified by adding the following lines just before *includes* section, which is allocated at line 30 of this file (approximatedly):

```
#define IEEE80221_LINK_MIHCAP_FLAG_ES           0x0001
#define IEEE80221_LINK_MIHCAP_FLAG_CS           0x0002
#define IEEE80221_LINK_MIHCAP_FLAG_IS           0x0004
#define IEEE80221_NET_CAPS_SECURITY             0x0008
#define IEEE80221_NET_CAPS_QOS_CLASS_0          0x0010
#define IEEE80221_NET_CAPS_QOS_CLASS_1          0x0020
#define IEEE80221_NET_CAPS_QOS_CLASS_2          0x0040
#define IEEE80221_NET_CAPS_QOS_CLASS_3          0x0080
#define IEEE80221_NET_CAPS_QOS_CLASS_4          0x0100
#define IEEE80221_NET_CAPS_QOS_CLASS_5          0x0200
#define IEEE80221_NET_CAPS_INTERNET_ACCESS      0x0400
#define IEEE80221_NET_CAPS_EMERGENCY_SERVICE    0x0800
#define IEEE80221_NET_CAPS_MIH_CAPABLE          0x1000
u8 * hostapd_802_21_supported_caps(u8 *eid)
{
    u8 *pos = eid;
    /* Save to byte to store Type and Length fields before. */
    u16 *caps = (u16 *) (pos + 2);
    u16 PoA_caps = 0x0000;
    PoA_caps |=  IEEE80221_LINK_MIHCAP_FLAG_IS |
                 IEEE80221_LINK_MIHCAP_FLAG_CS |
                   IEEE80221_NET_CAPS_QOS_CLASS_4 |
                    IEEE80221_NET_CAPS_QOS_CLASS_5 |
                 IEEE80221_NET_CAPS_INTERNET_ACCESS |
                 IEEE80221_NET_CAPS_MIH_CAPABLE;
    *caps = PoA_caps;
    pos = (u8 *) (caps + 1);
    eid[1] = pos - eid - 2;
    eid[0] = 0xaf;     /* Tag 175, Reserved Tag Number . */
    return pos;
}
u8 * hostapd_802_21_MIHF_ID(u8 *eid, char *mihf_id)
{
    u8 *pos = eid;
    /* Save to byte to store Type and Length fields before. */
    char *name = (u8 *) (pos + 2);
    memcpy(name, mihf_id, strlen(mihf_id)+1);
    pos += strlen(mihf_id) + 2;
    eid[1] = pos - eid - 2;
    eid[0] = 0xbf;     /* Tag 191, Reserved Tag Number . */
    return pos;
}
```

Note that these functions provide a couple of beacon tags which allow defining IEEE 802.21 capabilities. The call to these functions must be introduced into `ieee802_11_set_beacon`, which is the function that builds the beacon frame. The precise point of insertion is into *Variable Tags* section in beacon frame. They have been inserted just after *Extended Supported Rates* in the daemon implementation:

```
    /*80221 CAPABILITIES*/
    tailpos = hostapd_802_21_supported_caps(tailpos);
    char name[] = “whoah@poa.net”
    tailpos = hostapd_802_21_MIHF_ID(tailpos, name);
```

Note that this modifies the beacons that can be read in the network nodes through passive scanning. In case an active scan is used via probe-request frames, it is needed to reproduce the same operation into *handle_probe_req* function, which is defined just under the previous one.

The file *driver_nl80211.c* in must be modified in order to define the frame types which can be captured by the monitor interface used by Hostapd. The following line, that is located around line 1023, must be commented:

```
/* NLA_PUT_FLAG(flags, NL80211_MNTR_FLAG_COOK_FRAMES); */
```

And to replace the commented line, the following lines must be typed:

```
NLA_PUT_FLAG(flags,  NL80211_MNTR_FLAG_FCSFAIL  |  NL80211_MNTR_FLAG_PLCPFAIL  |
NL80211_MNTR_FLAG_CONTROL | NL80211_MNTR_FLAG_OTHER_BSS);
```

Note that the latest sources of Hostapd may not include the file *driver_nl80211.c*. This is the reason why the modification of these flags should be done in other extern source file. These flags are standard into the Linux kernel, and are defined in *nl80211.h* and used *in cfg80211.h* files.

Now the correct behavior of the APs through Hostapd can be checked by running the daemon and analyzing the new tags included into beacon frames through a network sniffer such as Wireshark or TCPdump. The connectivity can be checked by associating a station to these APs through a graphical or command line network manager such as Network Manager or *dhclient*.

4.1.3. Quick start reference for MN and PoA configuration

Now that all the required elements have been installed in their respective hosts, there only thing that rests is to establish the proper configuration of each element. The following lines provide the pattern to configure the system once the required entities have been installed in their respective MN and PoAs.

4.1.3.1. MN configuration

First, the user in the MN should use root privileges. This can be done by using a super user console or by switching to super user mode. Then, all the processes related to Network Manager or similar must be disabled if they have not been ended before. The MN application will act as network manager. Finally, with the Realteck Wi-Fi device plugged in a USB port, execute the following lines:

```
user:~$ su root

[enter password for root]

user:~# killall NetworkManager

user:~# ifconfig eth0 down
```

```
    user:~# ifconfig wlan0 up

    user:~# ifconfig wlan1 up      ← see the way your wireless interfaces are named
```

Note that the current implementation does not consider handling wired interfaces. Modifications on the MN program are not difficult to introduce, but have been left for further improvements. However, the current development has been focused on wireless interfaces management.

Normally, it is useful to check the kernel route table to see if the routes are correct or if there exist not desired routes. These last ones should be deleted in order to avoid routing conflicts.

4.1.3.2. PoA configuration

The steps to configure the PoA are the following: first, get root privileges. Then, go to the folder which contains the modified code for Hostapd and start this daemon with the customized configuration file. Then, in another window and also with root privileges, configure the IP address of the PoA. Start *dhcp3-server* daemon, which provides DHCP server capabilities. Then, allow forwarding IPv4 packets and finally update the NAT table in order to allow queries to hosts which are not within the local network.

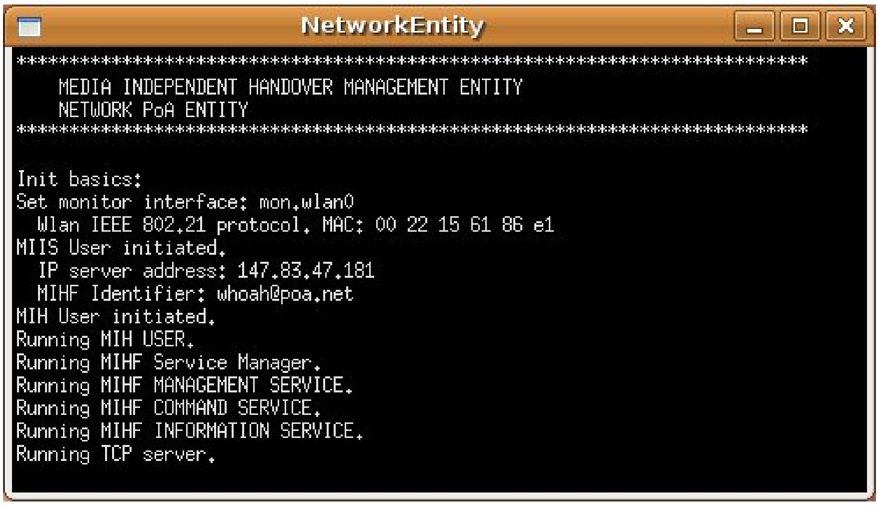
```
    user:~$ su root

    [enter password for root]

    user:~# cd /usr/src/hostap.poa/hostapd/

    .../hostapd:~# ./hostapd -dd ./hostapd.conf

    user:~# ifconfig wlan0 192.168.3.1

    user:~# /etc/init.d/dhcp3-server start

    user:~# echo 1 > /proc/sys/net/ipv4/ip_forward

    user:~# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

The lines in the box abobe are a summary of previous sub-sections in order to indicate how to run the PoA every time it is needed to be started. It must be considered that the IP address is different in each PoA, so it is not recommended to assign the same address to each of them.

4.2. Operation Validation

4.2.1. Operation Validation for MN and PoA daemons

Once the required applications have been installed in the system, the daemons can be started. The behavior of the involved applications is described through a set of screenshots coming from their debugging output. In the mentioned screenshots the two basic actions performed in order to handle the connection are described: the bootstrap decision and the handover decision.

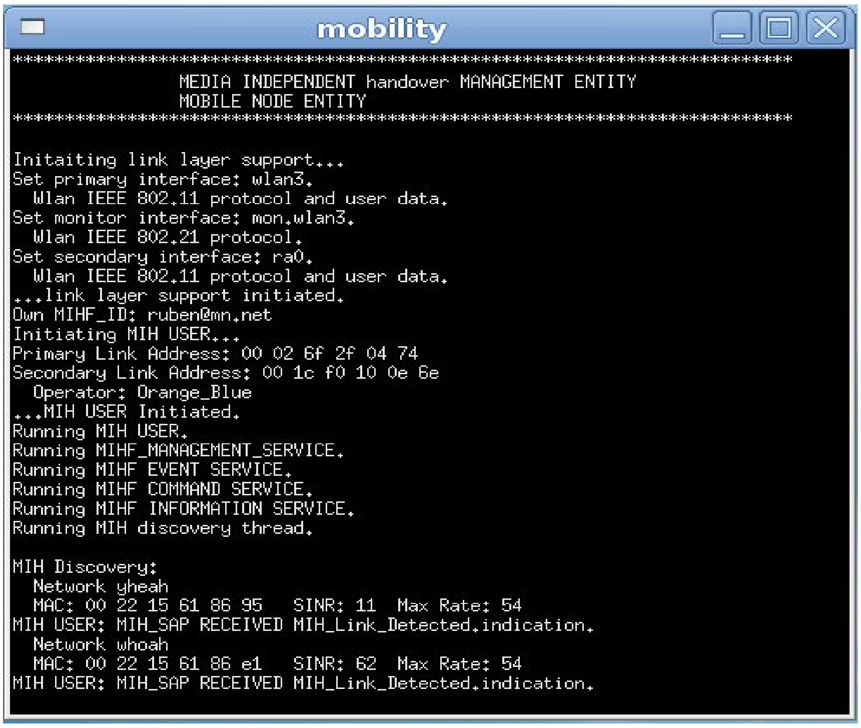


```
***************************************************************************
    MEDIA INDEPENDENT HANDOVER MANAGEMENT ENTITY
    NETWORK PoA ENTITY
***************************************************************************

Init basics:
Set monitor interface: mon.wlan0
  Wlan IEEE 802.21 protocol. MAC: 00 22 15 61 86 e1
MIIS User initiated.
  IP server address: 147.83.47.181
  MIHF Identifier: whoah@poa.net
MIH User initiated.
Running MIH USER.
Running MIHF Service Manager.
Running MIHF MANAGEMENT SERVICE.
Running MIHF COMMAND SERVICE.
Running MIHF INFORMATION SERVICE.
Running TCP server.
```

Figure 4.1: PoAs daemon startup process

The scenario is the one described at the beginning of this chapter. There are the two PoAs interconnected through the laboratory network (although it does not matter the way they are engaged in this demonstration) and a MN. To sum up, the behavior is the following: the MN starts the daemon and so it requests the best PoA to get bounded; after a successful engagement the received signal strength decreases and a MIH assisted handover takes place. The whole process is described in depth hereinafter with the help of some screenshots.

Before the MN daemon starts executing, both PoAs are supposed to be running. Figure 4.1 depicts the behavior of the PoAs during their startup processes. It can be seen the name assigned to the monitor interface which will

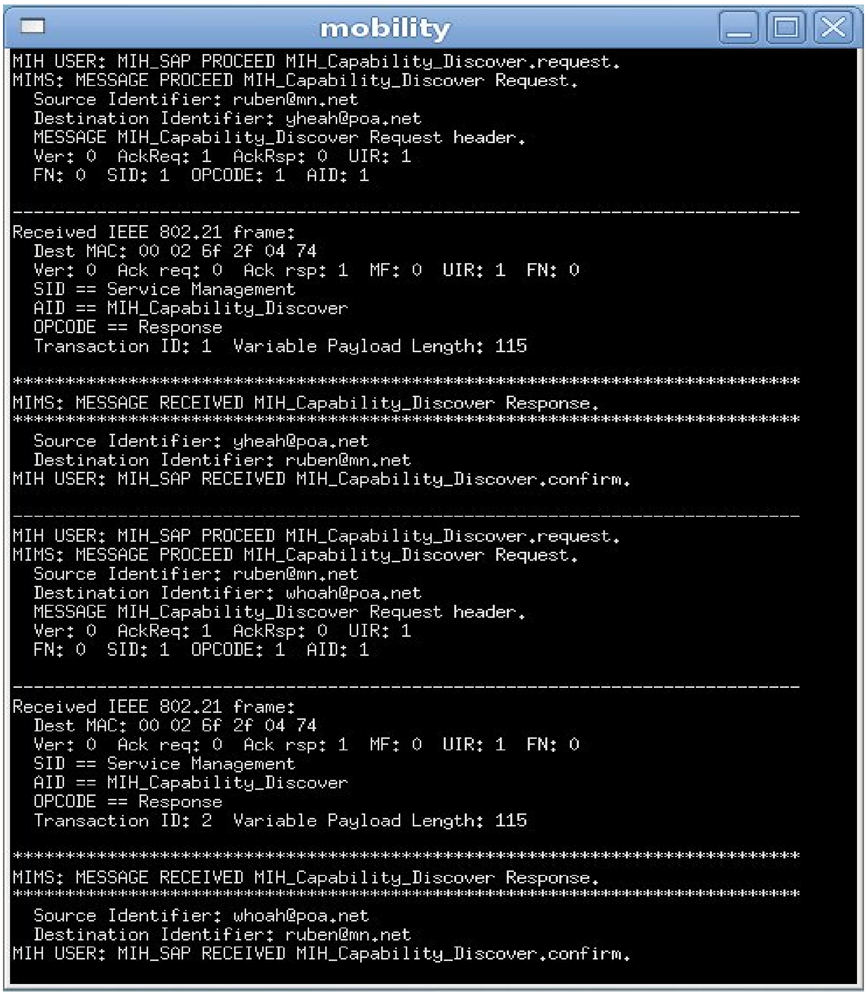


```
***************************************************************************
                MEDIA INDEPENDENT handover MANAGEMENT ENTITY
                MOBILE NODE ENTITY
***************************************************************************

Initaiting link layer support...
Set primary interface: wlan3.
  Wlan IEEE 802.11 protocol and user data.
Set monitor interface: mon.wlan3.
  Wlan IEEE 802.21 protocol.
Set secondary interface: ra0.
  Wlan IEEE 802.11 protocol and user data.
...link layer support initiated.
Own MIHF_ID: ruben@mn.net
Initiating MIH USER...
Primary Link Address: 00 02 6f 2f 04 74
Secondary Link Address: 00 1c f0 10 0e 6e
  Operator: Orange_Blue
...MIH USER Initiated.
Running MIH USER.
Running MIHF_MANAGEMENT_SERVICE.
Running MIHF EVENT SERVICE.
Running MIHF COMMAND SERVICE.
Running MIHF INFORMATION SERVICE.
Running MIH discovery thread.

MIH Discovery:
  Network yheah
  MAC: 00 22 15 61 86 95   SINR: 11  Max Rate: 54
MIH USER: MIH_SAP RECEIVED MIH_Link_Detected.indication.
  Network whoah
  MAC: 00 22 15 61 86 e1   SINR: 62  Max Rate: 54
MIH USER: MIH_SAP RECEIVED MIH_Link_Detected.indication.
```

Figure 4.2: MN daemon startup and MIH discovery

handle the MIH protocol, being *mon.wlan0* in both cases. It can also be appreciated the MIHF identifier of those entities as well as the MAC address of their NICs and the IP address used to handle the wired connection through a TCP connection.

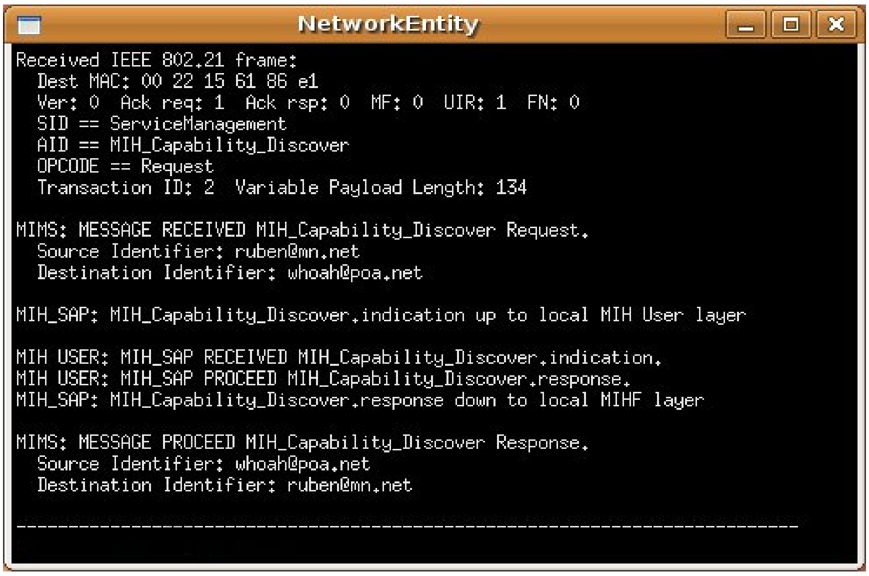
Figure 4.2 sums up the basic parameters that the MN daemon gathers when it is initiated and the discovered PoAs during the MIH Discovery process. It can be seen that the primary interface, which is the data bearer, is called by the system as *wlan3*, whereas the monitor interface which handles the MIH protocol is called *mon.wlan3*. The other data bearer is the *ra0* interface, which corresponds to the secondary interface. The startup also indicates the MAC addresses of those interfaces, as well as the MIHF identifier used to describe the local MIH entity. The MIH Discovery process shows through MIH_Link_Detected indications that two networks have been discovered: network *whoah* and network *yheah*.



```
MIH USER: MIH_SAP PROCEED MIH_Capability_Discover.request.
MIMS: MESSAGE PROCEED MIH_Capability_Discover Request.
   Source Identifier: ruben@mn.net
   Destination Identifier: yheah@poa.net
   MESSAGE MIH_Capability_Discover Request header.
   Ver: 0  AckReq: 1  AckRsp: 0  UIR: 1
   FN: 0  SID: 1  OPCODE: 1  AID: 1

---------------------------------------------------------------------------
Received IEEE 802.21 frame:
   Dest MAC: 00 02 6f 2f 04 74
   Ver: 0  Ack req: 0  Ack rsp: 1  MF: 0  UIR: 1  FN: 0
   SID == Service Management
   AID == MIH_Capability_Discover
   OPCODE == Response
   Transaction ID: 1  Variable Payload Length: 115

***************************************************************************
MIMS: MESSAGE RECEIVED MIH_Capability_Discover Response.
***************************************************************************
   Source Identifier: yheah@poa.net
   Destination Identifier: ruben@mn.net
MIH USER: MIH_SAP RECEIVED MIH_Capability_Discover.confirm.

---------------------------------------------------------------------------
MIH USER: MIH_SAP PROCEED MIH_Capability_Discover.request.
MIMS: MESSAGE PROCEED MIH_Capability_Discover Request.
   Source Identifier: ruben@mn.net
   Destination Identifier: whoah@poa.net
   MESSAGE MIH_Capability_Discover Request header.
   Ver: 0  AckReq: 1  AckRsp: 0  UIR: 1
   FN: 0  SID: 1  OPCODE: 1  AID: 1

---------------------------------------------------------------------------
Received IEEE 802.21 frame:
   Dest MAC: 00 02 6f 2f 04 74
   Ver: 0  Ack req: 0  Ack rsp: 1  MF: 0  UIR: 1  FN: 0
   SID == Service Management
   AID == MIH_Capability_Discover
   OPCODE == Response
   Transaction ID: 2  Variable Payload Length: 115

***************************************************************************
MIMS: MESSAGE RECEIVED MIH_Capability_Discover Response.
***************************************************************************
   Source Identifier: whoah@poa.net
   Destination Identifier: ruben@mn.net
MIH USER: MIH_SAP RECEIVED MIH_Capability_Discover.confirm.
```

Figure 4.3: MN capability discover processes

The next step for the MN is performing the capability discover with the discovered PoAs. Figure 4.3 depicts the messages received in the MN when it queries the involved entities. As a pattern for all the exchanged messages,

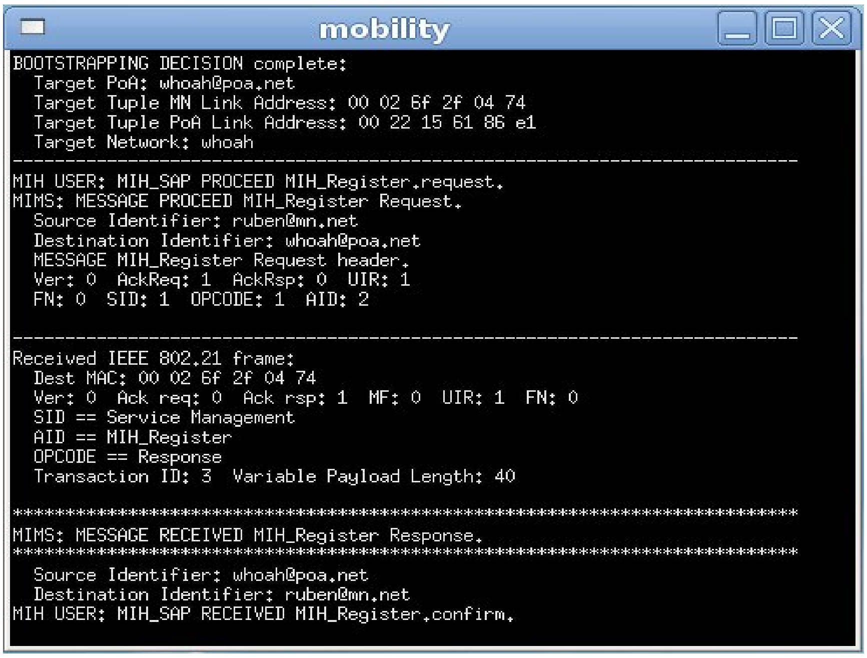


```
Received IEEE 802.21 frame:
  Dest MAC: 00 22 15 61 86 e1
  Ver: 0  Ack req: 1  Ack rsp: 0  MF: 0  UIR: 1  FN: 0
  SID == ServiceManagement
  AID == MIH_Capability_Discover
  OPCODE == Request
  Transaction ID: 2  Variable Payload Length: 134

MIMS: MESSAGE RECEIVED MIH_Capability_Discover Request.
  Source Identifier: ruben@mn.net
  Destination Identifier: whoah@poa.net

MIH_SAP: MIH_Capability_Discover.indication up to local MIH User layer

MIH USER: MIH_SAP RECEIVED MIH_Capability_Discover.indication.
MIH USER: MIH_SAP PROCEED MIH_Capability_Discover.response.
MIH_SAP: MIH_Capability_Discover.response down to local MIHF layer

MIMS: MESSAGE PROCEED MIH_Capability_Discover Response.
  Source Identifier: whoah@poa.net
  Destination Identifier: ruben@mn.net

----------------------------------------------------------------------------
```

Figure 4.4: PoAs MIH Capability Discover processes

the output highlights the received messages by writing their identifier between two lines of asterisks. Prior to that, the basic information included in the header is also supplied. The output also shows the MIHF identifiers of the source and destination peers involved in this transaction before the name of the received message is depicted, and also the actions the MIH take after receiving the message. Commonly these actions are indications or confirmations to the MIH user layer. Once the corresponding actions are taken, the next taken steps are shown. Generally, they are MIH user decisions or instructions to the MIHF layer in order to generate new outgoing messages or to handle the link layer. In this particular case, the MN queries the discovered PoAs and receives their responded information. The MIH User performs MIH_Capability_Discover request and indicates that it has received a MIH_Capability_Discover confirm SAPs in both cases.

Figure 4.4 shows the corresponding request messages received at one PoA and the response generated. The other

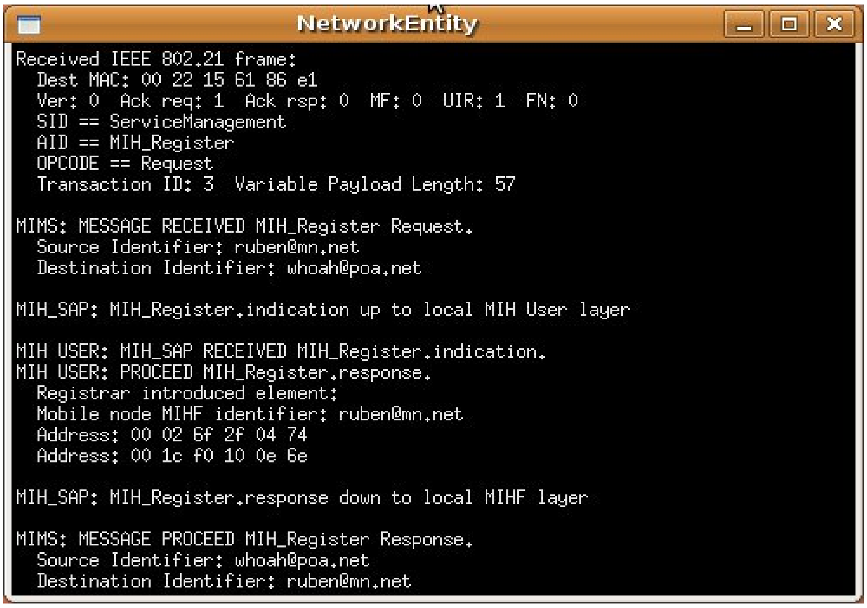


```
BOOTSTRAPPING DECISION complete:
  Target PoA: whoah@poa.net
  Target Tuple MN Link Address: 00 02 6f 2f 04 74
  Target Tuple PoA Link Address: 00 22 15 61 86 e1
  Target Network: whoah
----------------------------------------------------------------------------
MIH USER: MIH_SAP PROCEED MIH_Register.request.
MIMS: MESSAGE PROCEED MIH_Register Request.
  Source Identifier: ruben@mn.net
  Destination Identifier: whoah@poa.net
  MESSAGE MIH_Register Request header.
  Ver: 0  AckReq: 1  AckRsp: 0  UIR: 1
  FN: 0  SID: 1  OPCODE: 1  AID: 2

----------------------------------------------------------------------------
Received IEEE 802.21 frame:
  Dest MAC: 00 02 6f 2f 04 74
  Ver: 0  Ack req: 0  Ack rsp: 1  MF: 0  UIR: 1  FN: 0
  SID == Service Management
  AID == MIH_Register
  OPCODE == Response
  Transaction ID: 3  Variable Payload Length: 40

****************************************************************************
MIMS: MESSAGE RECEIVED MIH_Register Response.
****************************************************************************
  Source Identifier: whoah@poa.net
  Destination Identifier: ruben@mn.net
MIH USER: MIH_SAP RECEIVED MIH_Register.confirm.
```

Figure 4.5: MN Bootstrap decision and MIH Registry processes

PoA has the same behavior as the depicted in the figure. The general behavior of the PoAs is simpler than MN's, but follows the same pattern as the described for the MN. In this case, it can be seen how a MIH_Capability_Discover message request arrives to the current PoA and so the corresponding MIH SAP is triggered. The MIH User outputs the actions taken, that in this case is the invocation to the MIH_Capability_Discover response SAP. The MIHF outputs the source and destination MIH identifiers of the current transaction, which are the MN and the local PoA respectively.

At this point, the MN proceeds with Bootstrapping Decision and MIH Registry processes. These actions are depicted in figure 4.5. In this case, the MN chooses to register with *whoah@poa.net* MIHF. As seen in figure 4.2, this is the PoA that supplies the best quality of signal. In fact, in the current implementation the only thing that differentiates both PoAs is the received signal strength at the MN. Figure 4.5 also shows which are the tuple MAC addresses of the current link and the corresponding ESS identifier of the access network for Bootstrapping Decision.

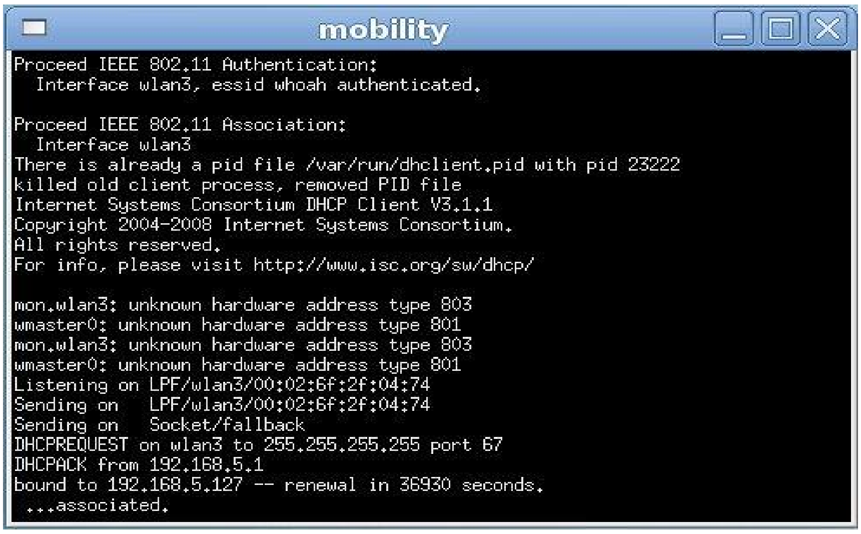


```
Received IEEE 802.21 frame:
  Dest MAC: 00 22 15 61 86 e1
  Ver: 0  Ack req: 1  Ack rsp: 0  MF: 0  UIR: 1  FN: 0
  SID == ServiceManagement
  AID == MIH_Register
  OPCODE == Request
  Transaction ID: 3  Variable Payload Length: 57

MIMS: MESSAGE RECEIVED MIH_Register Request.
  Source Identifier: ruben@mn.net
  Destination Identifier: whoah@poa.net

MIH_SAP: MIH_Register.indication up to local MIH User layer

MIH USER: MIH_SAP RECEIVED MIH_Register.indication.
MIH USER: PROCEED MIH_Register.response.
  Registrar introduced element:
  Mobile node MIHF identifier: ruben@mn.net
  Address: 00 02 6f 2f 04 74
  Address: 00 1c f0 10 0e 6e

MIH_SAP: MIH_Register.response down to local MIHF layer

MIMS: MESSAGE PROCEED MIH_Register Response.
  Source Identifier: whoah@poa.net
  Destination Identifier: ruben@mn.net
```

Figure 4.6: Serving PoA MIH Resgistry processes

The exchange message for registry is the same as described previously for the preceding messages. The output generated by the PoA illustrates the same pattern as the depicted in the former received message, but with MIH_Register response. In the example case, the registration is done correctly and so no failure messages are printed by the output, as seen in figure 4.6.

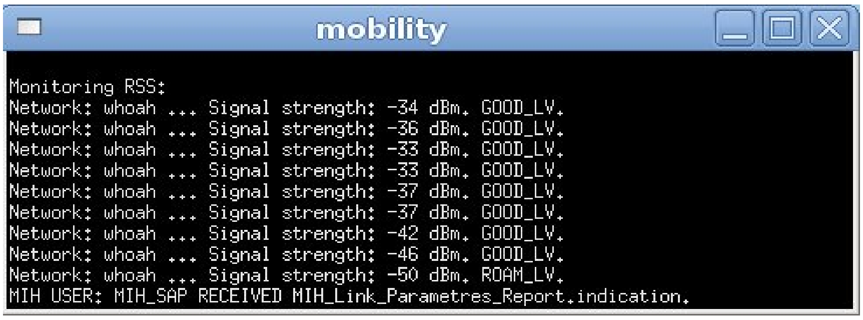
Once registered at MIH level, the MN proceeds with authentication and association processes which correspond to IEEE 802.11 protocol. Figure 4.7 depicts its behavior, which can follow different patterns depending on the conditions the association process has been done. In the current example the association process is done almost instantaneously because the interface was previously associated in former tests, and no DHCP discovery processes are needed in order to obtain the required entity to supply a new address. The figure shows that the remote DHCP server just confirms the current IP address assigned to the MN.



```
Proceed IEEE 802.11 Authentication:
  Interface wlan3, essid whoah authenticated.

Proceed IEEE 802.11 Association:
  Interface wlan3
There is already a pid file /var/run/dhclient.pid with pid 23222
killed old client process, removed PID file
Internet Systems Consortium DHCP Client V3.1.1
Copyright 2004-2008 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

mon.wlan3: unknown hardware address type 803
wmaster0: unknown hardware address type 801
mon.wlan3: unknown hardware address type 803
wmaster0: unknown hardware address type 801
Listening on LPF/wlan3/00:02:6f:2f:04:74
Sending on   LPF/wlan3/00:02:6f:2f:04:74
Sending on   Socket/fallback
DHCPREQUEST on wlan3 to 255.255.255.255 port 67
DHCPACK from 192.168.5.1
bound to 192.168.5.127 -- renewal in 36930 seconds.
 ...associated.
```

Figure 4.7: MN IEEE 802.11 authentication and association

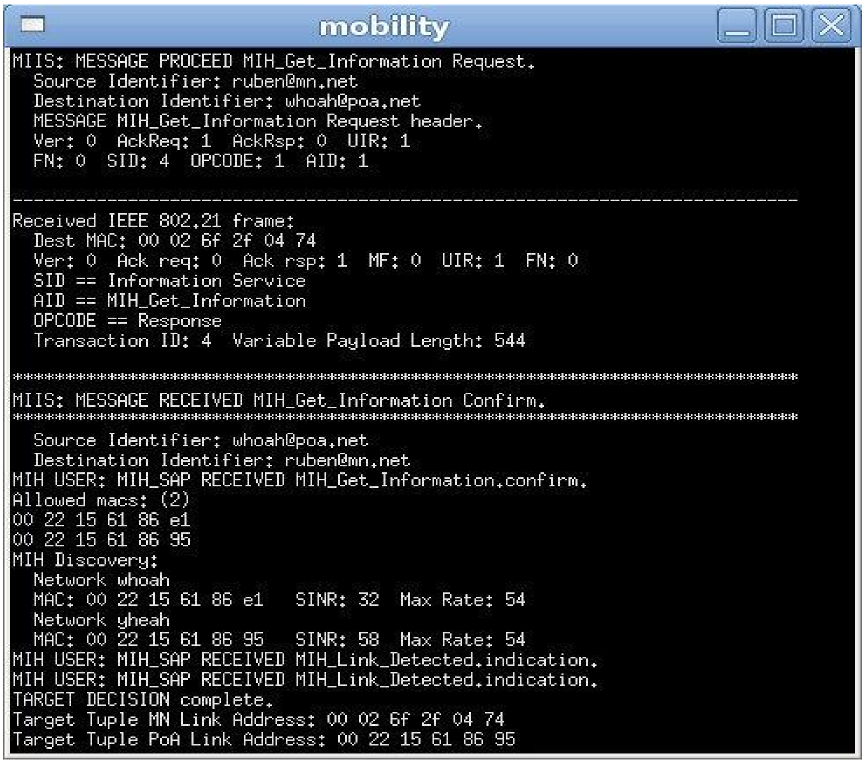
After that, the MN starts monitoring the received signal strength that comes from the attached PoA. In this case, after some seconds of a good level of received signal, the MN has started to move and so it receives a weaker signal. Figure 4.8 shows this degradation and how MIH_Link_Parameters_Report indication is triggered once the signal strength has gone under a determined threshold. In this case, the threshold has been set to 50 [dBm], which is useful for this test but not for a real environment because in fact it is considered a very good level.



```
Monitoring RSS:
Network: whoah ... Signal strength: -34 dBm, GOOD_LV.
Network: whoah ... Signal strength: -36 dBm, GOOD_LV.
Network: whoah ... Signal strength: -33 dBm, GOOD_LV.
Network: whoah ... Signal strength: -33 dBm, GOOD_LV.
Network: whoah ... Signal strength: -37 dBm, GOOD_LV.
Network: whoah ... Signal strength: -37 dBm, GOOD_LV.
Network: whoah ... Signal strength: -42 dBm, GOOD_LV.
Network: whoah ... Signal strength: -46 dBm, GOOD_LV.
Network: whoah ... Signal strength: -50 dBm, ROAM_LV.
MIH USER: MIH_SAP RECEIVED MIH_Link_Parametres_Report.indication.
```

Figure 4.8: MN Received Signal Strength monitoring

Having received this indication, the MN proceeds with a MIH query to the information server through the serving PoA. The message exchange schema is the same as in previous cases. Figure 4.9 depicts this action. Once received the corresponding response, the MN application outputs the MAC addresses of the discovered PoAs which are allowed in the MN to trigger Link_Detected indications that compose the MIH Discovery process. The figure shows

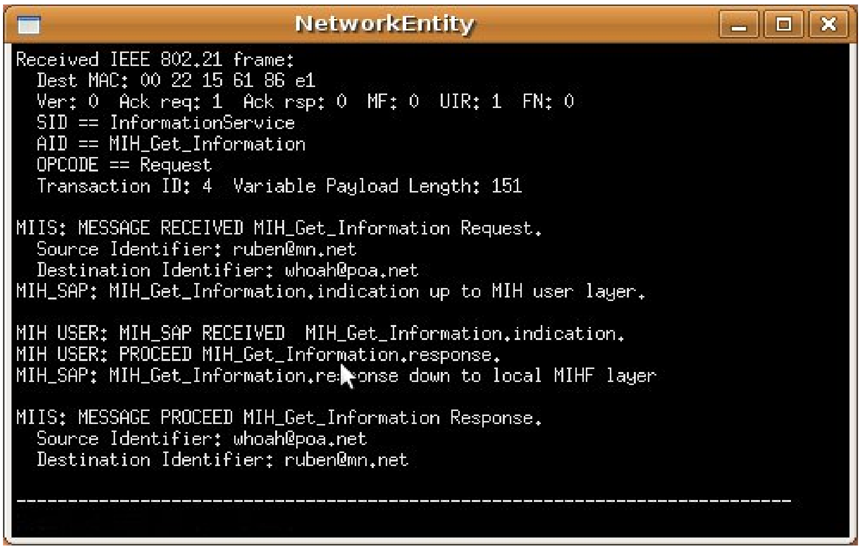


```
MIIS: MESSAGE PROCEED MIH_Get_Information Request.
  Source Identifier: ruben@mn.net
  Destination Identifier: whoah@poa.net
  MESSAGE MIH_Get_Information Request header.
  Ver: 0  AckReq: 1  AckRsp: 0  UIR: 1
  FN: 0  SID: 4  OPCODE: 1  AID: 1

----------------------------------------------------------------------------
Received IEEE 802.21 frame:
  Dest MAC: 00 02 6f 2f 04 74
  Ver: 0  Ack req: 0  Ack rsp: 1  MF: 0  UIR: 1  FN: 0
  SID == Information Service
  AID == MIH_Get_Information
  OPCODE == Response
  Transaction ID: 4  Variable Payload Length: 544

****************************************************************************
MIIS: MESSAGE RECEIVED MIH_Get_Information Confirm.
****************************************************************************
  Source Identifier: whoah@poa.net
  Destination Identifier: ruben@mn.net
MIH USER: MIH_SAP RECEIVED MIH_Get_Information.confirm.
Allowed macs: (2)
00 22 15 61 86 e1
00 22 15 61 86 95
MIH Discovery:
  Network whoah
  MAC: 00 22 15 61 86 e1   SINR: 32  Max Rate: 54
  Network yheah
  MAC: 00 22 15 61 86 95   SINR: 58  Max Rate: 54
MIH USER: MIH_SAP RECEIVED MIH_Link_Detected.indication.
MIH USER: MIH_SAP RECEIVED MIH_Link_Detected.indication.
TARGET DECISION complete.
Target Tuple MN Link Address: 00 02 6f 2f 04 74
Target Tuple PoA Link Address: 00 22 15 61 86 95
```

Figure 4.9: MN MIIS query and Target Decision processes

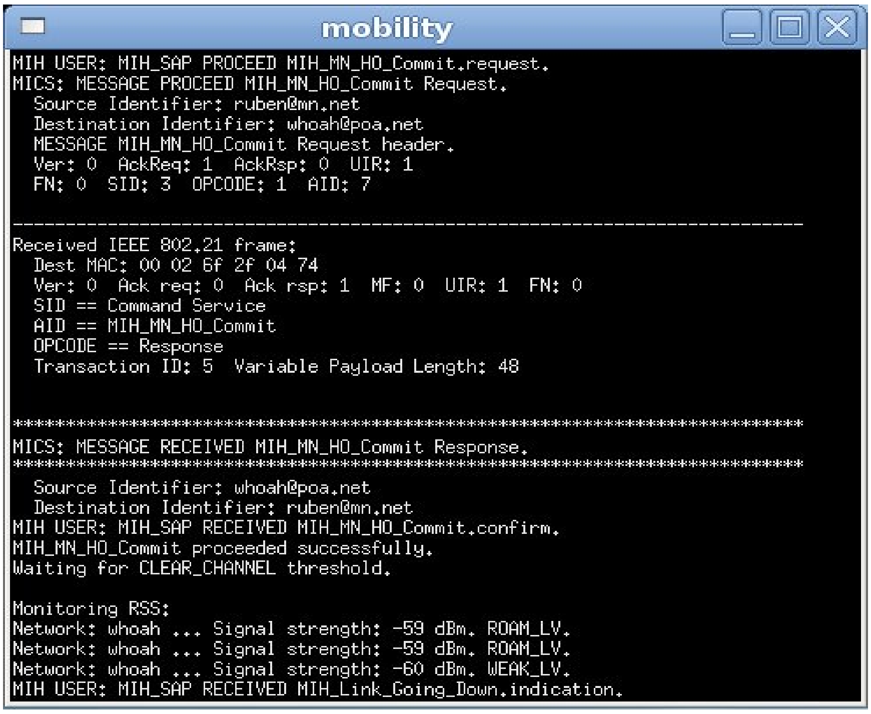
the results of this process, and as it can be seen, the network *yheah* is in this case the one that provides better quality of signal. As explained in previous chapters, the information from both MIH Discovery process and from the MIIS is combined in order to make a decision. In the current example, the resources that both PoAs can provide are the same, so at the end the decision relies on the received signal strength. The result of the Target Decision is showed by printing both source and destination MAC addresses of the target link.

Figure 4.10 shows the corresponding MIIS query response thrown by the output of the PoA which is analog to the output of the previous incoming messages.



```
Received IEEE 802.21 frame:
  Dest MAC: 00 22 15 61 86 e1
  Ver: 0  Ack req: 1  Ack rsp: 0  MF: 0  UIR: 1  FN: 0
  SID == InformationService
  AID == MIH_Get_Information
  OPCODE == Request
  Transaction ID: 4  Variable Payload Length: 151

MIIS: MESSAGE RECEIVED MIH_Get_Information Request.
  Source Identifier: ruben@mn.net
  Destination Identifier: whoah@poa.net
MIH_SAP: MIH_Get_Information.indication up to MIH user layer.

MIH USER: MIH_SAP RECEIVED  MIH_Get_Information.indication.
MIH USER: PROCEED MIH_Get_Information.response.
MIH_SAP: MIH_Get_Information.response down to local MIHF layer

MIIS: MESSAGE PROCEED MIH_Get_Information Response.
  Source Identifier: whoah@poa.net
  Destination Identifier: ruben@mn.net

----------------------------------------------------------------------------
```

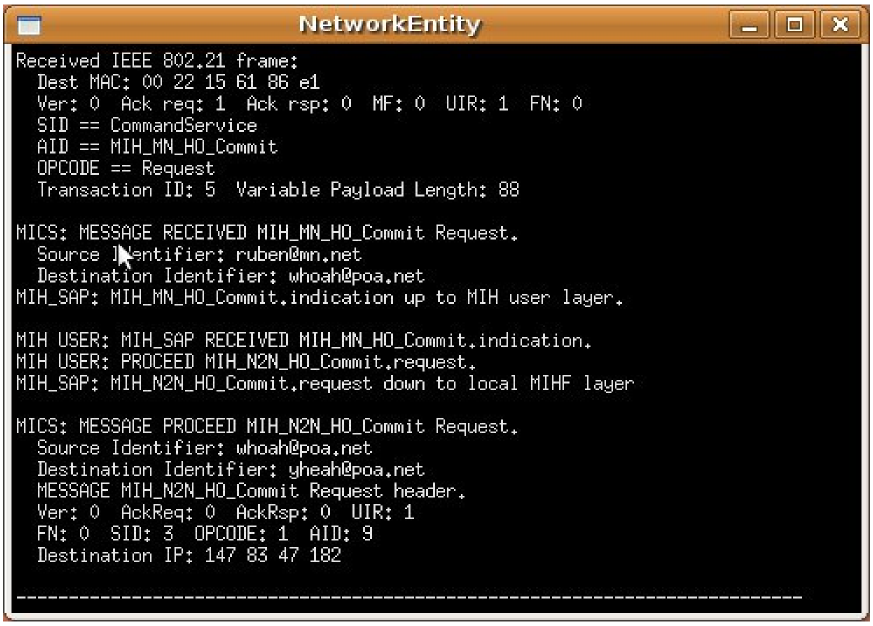
Figure 4.10: Serving PoA MIIS response procedures



```
MIH USER: MIH_SAP PROCEED MIH_MN_HO_Commit.request.
MICS: MESSAGE PROCEED MIH_MN_HO_Commit Request.
  Source Identifier: ruben@mn.net
  Destination Identifier: whoah@poa.net
  MESSAGE MIH_MN_HO_Commit Request header.
  Ver: 0  AckReq: 1  AckRsp: 0  UIR: 1
  FN: 0  SID: 3  OPCODE: 1  AID: 7

---------------------------------------------------------------------------
Received IEEE 802.21 frame:
  Dest MAC: 00 02 6f 2f 04 74
  Ver: 0  Ack req: 0  Ack rsp: 1  MF: 0  UIR: 1  FN: 0
  SID == Command Service
  AID == MIH_MN_HO_Commit
  OPCODE == Response
  Transaction ID: 5  Variable Payload Length: 48

***************************************************************************
MICS: MESSAGE RECEIVED MIH_MN_HO_Commit Response.
***************************************************************************
  Source Identifier: whoah@poa.net
  Destination Identifier: ruben@mn.net
MIH USER: MIH_SAP RECEIVED MIH_MN_HO_Commit.confirm.
MIH_MN_HO_Commit proceeded successfully.
Waiting for CLEAR_CHANNEL threshold.

Monitoring RSS:
Network: whoah ... Signal strength: -59 dBm. ROAM_LV.
Network: whoah ... Signal strength: -59 dBm. ROAM_LV.
Network: whoah ... Signal strength: -60 dBm. WEAK_LV.
MIH USER: MIH_SAP RECEIVED MIH_Link_Going_Down.indication.
```

Figure 4.11: MN MIH handover commit and RSS monitor processes

After having made the decision, the MN starts the handover procedure by alerting the target PoA via the serving PoA. Figure 4.11 depicts the followed pattern when the MN sends a MIH_Capability_Discover request message. Once the response of the corresponding transaction is received with the required resources being assigned, the MN waits for a signal degradation to take the next actions (the next threshold is set to -60 [dBm]). As it can be seen, the MN monitors the received signal strength until it crosses a new threshold. The MN then triggers

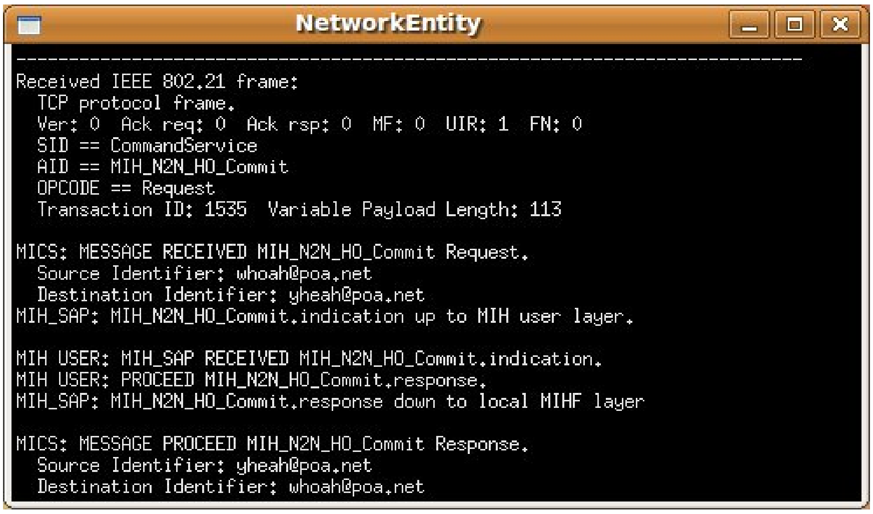


```
Received IEEE 802.21 frame:
  Dest MAC: 00 22 15 61 86 e1
  Ver: 0  Ack req: 1  Ack rsp: 0  MF: 0  UIR: 1  FN: 0
  SID == CommandService
  AID == MIH_MN_HO_Commit
  OPCODE == Request
  Transaction ID: 5  Variable Payload Length: 88

MICS: MESSAGE RECEIVED MIH_MN_HO_Commit Request.
  Source Identifier: ruben@mn.net
  Destination Identifier: whoah@poa.net
MIH_SAP: MIH_MN_HO_Commit.indication up to MIH user layer.

MIH USER: MIH_SAP RECEIVED MIH_MN_HO_Commit.indication.
MIH USER: PROCEED MIH_N2N_HO_Commit.request.
MIH_SAP: MIH_N2N_HO_Commit.request down to local MIHF layer

MICS: MESSAGE PROCEED MIH_N2N_HO_Commit Request.
  Source Identifier: whoah@poa.net
  Destination Identifier: yheah@poa.net
  MESSAGE MIH_N2N_HO_Commit Request header.
  Ver: 0  AckReq: 0  AckRsp: 0  UIR: 1
  FN: 0  SID: 3  OPCODE: 1  AID: 9
  Destination IP: 147 83 47 182

---------------------------------------------------------------------------
```

Figure 4.12: Serving PoA receives request for handover and sends it to target PoA

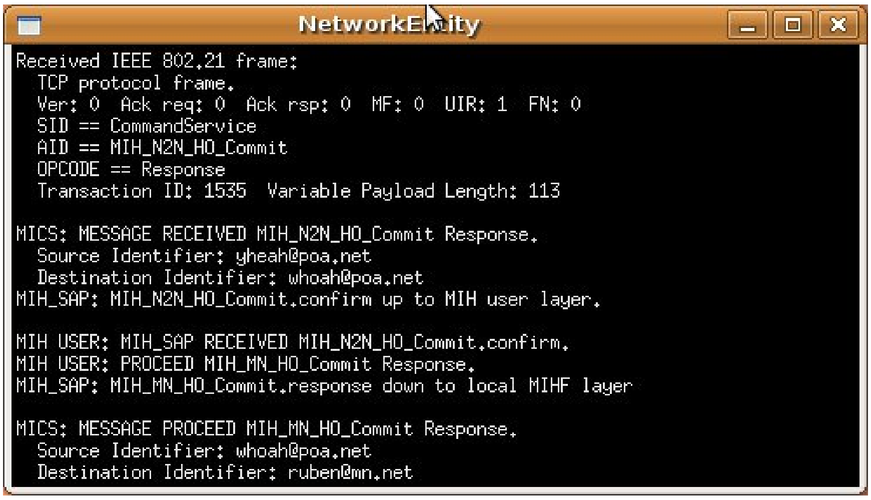
MIH_Link_Going_Down indication and gets ready to perform new actions.

Before this indication is triggered, the message exchange has flown through serving and target PoAs. Figure 4.12 depicts the reception of the query in the serving PoA and how it sends the query to the target PoA through a TCP connection by the wired network. It can be appreciated that the output of the serving PoA prints the IP address of the target PoA when the TCP connection is used. Figure 4.13 describes the reception of the message in the target PoA and the associated response during the current transaction. Finally, Figure 4.14 depicts the reception of the MIH_N2N_HO_Commit response and how it retransmits the required information to the MN, which is the element that initiated the described transaction.



```
--------------------------------------------------------------------------
Received IEEE 802.21 frame:
  TCP protocol frame.
  Ver: 0  Ack req: 0  Ack rsp: 0  MF: 0  UIR: 1  FN: 0
  SID == CommandService
  AID == MIH_N2N_HO_Commit
  OPCODE == Request
  Transaction ID: 1535  Variable Payload Length: 113

MICS: MESSAGE RECEIVED MIH_N2N_HO_Commit Request.
  Source Identifier: whoah@poa.net
  Destination Identifier: yheah@poa.net
MIH_SAP: MIH_N2N_HO_Commit.indication up to MIH user layer.

MIH USER: MIH_SAP RECEIVED MIH_N2N_HO_Commit.indication.
MIH USER: PROCEED MIH_N2N_HO_Commit.response.
MIH_SAP: MIH_N2N_HO_Commit.response down to local MIHF layer

MICS: MESSAGE PROCEED MIH_N2N_HO_Commit Response.
  Source Identifier: yheah@poa.net
  Destination Identifier: whoah@poa.net
```

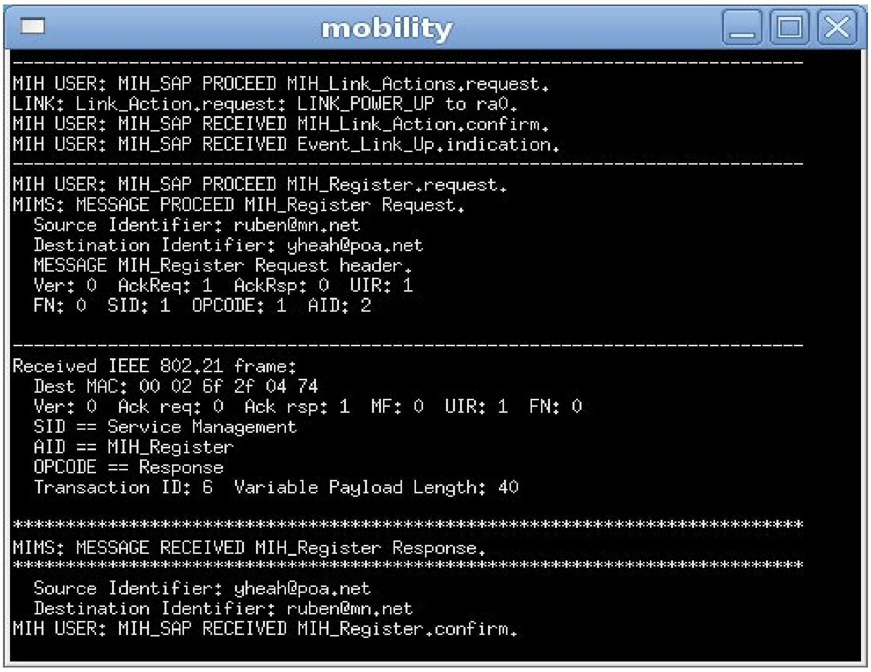
Figure 4.13: Target PoA reception of MIH handover request and response generated



```
Received IEEE 802.21 frame:
  TCP protocol frame.
  Ver: 0  Ack req: 0  Ack rsp: 0  MF: 0  UIR: 1  FN: 0
  SID == CommandService
  AID == MIH_N2N_HO_Commit
  OPCODE == Response
  Transaction ID: 1535  Variable Payload Length: 113

MICS: MESSAGE RECEIVED MIH_N2N_HO_Commit Response.
  Source Identifier: yheah@poa.net
  Destination Identifier: whoah@poa.net
MIH_SAP: MIH_N2N_HO_Commit.confirm up to MIH user layer.

MIH USER: MIH_SAP RECEIVED MIH_N2N_HO_Commit.confirm.
MIH USER: PROCEED MIH_MN_HO_Commit Response.
MIH_SAP: MIH_MN_HO_Commit.response down to local MIHF layer

MICS: MESSAGE PROCEED MIH_MN_HO_Commit Response.
  Source Identifier: whoah@poa.net
  Destination Identifier: ruben@mn.net
```

Figure 4.14: Serving PoA receives response to handover commit and sends it to MN

Once the MIH_MN_HO_Commit confirmation has been received in the MN, it sends to the link layer the required command to enable the remaining NIC in which the communication will be continued. In figure 4.15 it can be seen that the MIH User invokes MIH_Link_Actions SAP primitive in order to perform a LINK_POWER_UP to the interface

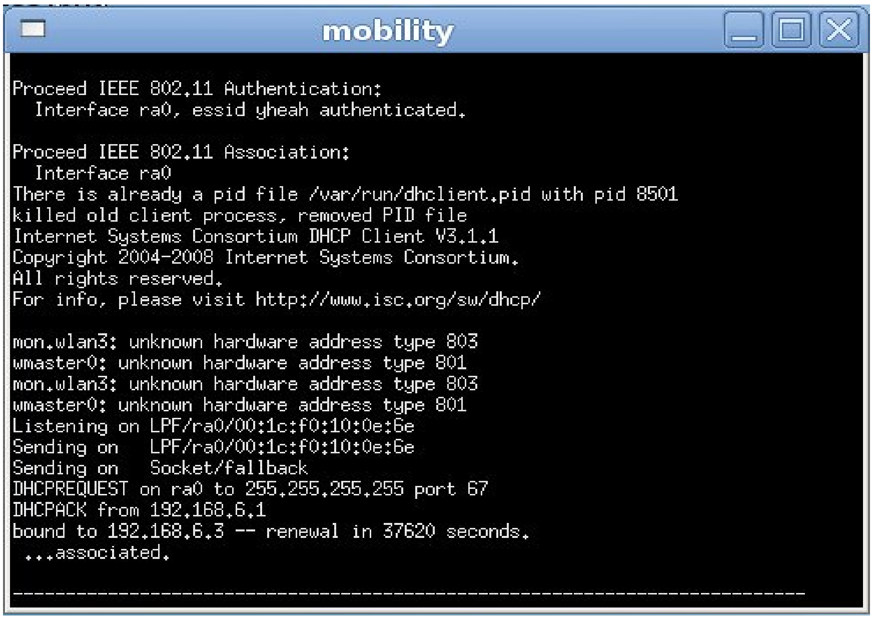


```
--------------------------------------------------------------------------------
MIH USER: MIH_SAP PROCEED MIH_Link_Actions.request.
LINK: Link_Action.request: LINK_POWER_UP to ra0.
MIH USER: MIH_SAP RECEIVED MIH_Link_Action.confirm.
MIH USER: MIH_SAP RECEIVED Event_Link_Up.indication.
--------------------------------------------------------------------------------
MIH USER: MIH_SAP PROCEED MIH_Register.request.
MIMS: MESSAGE PROCEED MIH_Register Request.
  Source Identifier: ruben@mn.net
  Destination Identifier: yheah@poa.net
  MESSAGE MIH_Register Request header.
  Ver: 0  AckReq: 1  AckRsp: 0  UIR: 1
  FN: 0  SID: 1  OPCODE: 1  AID: 2

--------------------------------------------------------------------------------
Received IEEE 802.21 frame:
  Dest MAC: 00 02 6f 2f 04 74
  Ver: 0  Ack req: 0  Ack rsp: 1  MF: 0  UIR: 1  FN: 0
  SID == Service Management
  AID == MIH_Register
  OPCODE == Response
  Transaction ID: 6  Variable Payload Length: 40

********************************************************************************
MIMS: MESSAGE RECEIVED MIH_Register Response.
********************************************************************************
  Source Identifier: yheah@poa.net
  Destination Identifier: ruben@mn.net
MIH USER: MIH_SAP RECEIVED MIH_Register.confirm.
```

Figure 4.15: MN performs LINK_POWER_UP and proceeds MIH Registry processes

ra0. After that, the MN proceeds with the MIH Registry process with the new network, as well as it did in Bootstrapping Decision. Now, the target PoA is *yheah@poa.net*, which is the one that supplies better conditions. The MIH Registry process is the same as described after Bootstrapping Decision but involving the new PoA, so the corresponding figure to illustrate how it happens in the target PoA is no required now.

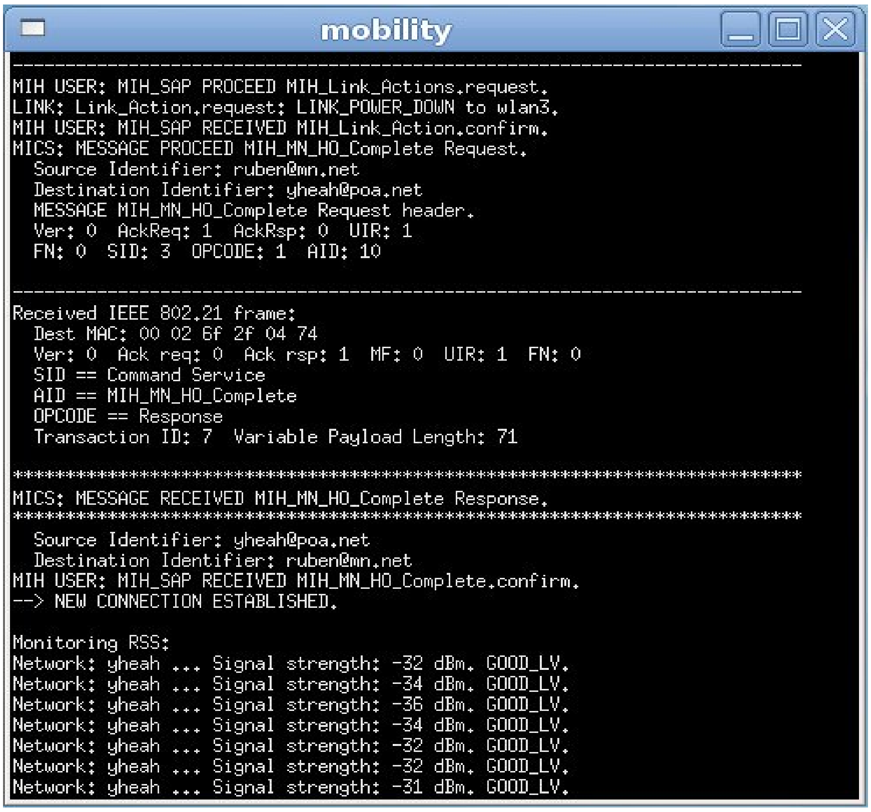
Figure 4.16 shows how the MN gets attached now to the target PoA through an IEEE 802.11 authentication and association. In this case, the interface that handles the data connection is *ra0*, and it gets attached to the *yheah* network in the same manner the interface *wlan3* did with *whoah* network.



```
Proceed IEEE 802.11 Authentication:
  Interface ra0, essid yheah authenticated.

Proceed IEEE 802.11 Association:
  Interface ra0
There is already a pid file /var/run/dhclient.pid with pid 8501
killed old client process, removed PID file
Internet Systems Consortium DHCP Client V3.1.1
Copyright 2004-2008 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

mon.wlan3: unknown hardware address type 803
wmaster0: unknown hardware address type 801
mon.wlan3: unknown hardware address type 803
wmaster0: unknown hardware address type 801
Listening on LPF/ra0/00:1c:f0:10:0e:6e
Sending on   LPF/ra0/00:1c:f0:10:0e:6e
Sending on   Socket/fallback
DHCPREQUEST on ra0 to 255.255.255.255 port 67
DHCPACK from 192.168.6.1
bound to 192.168.6.3 -- renewal in 37620 seconds.
 ...associated.

--------------------------------------------------------------------------------
```

Figure 4.16: MN Authentication and association with the target PoA

After being authenticated and associated, the MN is ready to switch the interface. It is done by switching off the old interface. Data can follow an alternative route because there is a new connection established with the target access network that now becomes the serving network. Figure 4.17 depicts how the MN successfully closes *wlan3* by sending MIH_Link_Actions request to the old local interface with the instruction LINK_POWER_DOWN.

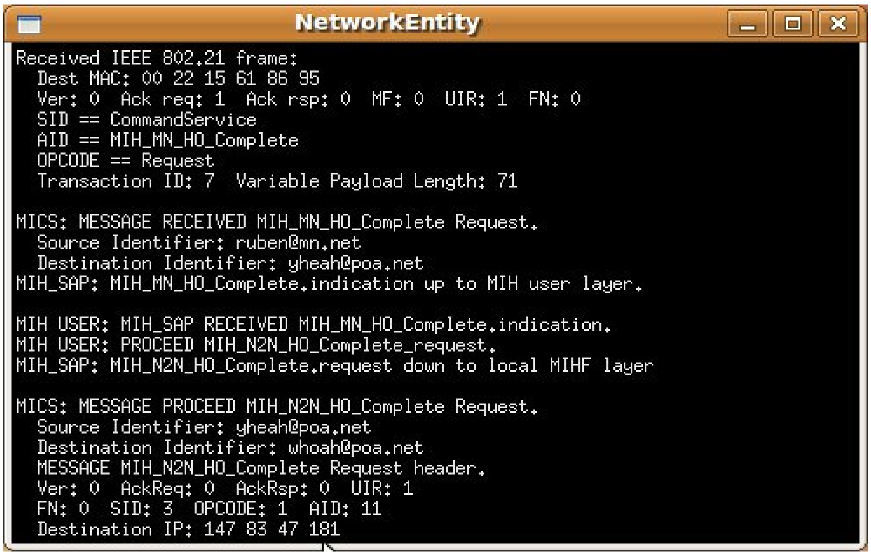


```
------------------------------------------------------------------------
MIH USER: MIH_SAP PROCEED MIH_Link_Actions.request.
LINK: Link_Action.request: LINK_POWER_DOWN to wlan3.
MIH USER: MIH_SAP RECEIVED MIH_Link_Action.confirm.
MICS: MESSAGE PROCEED MIH_MN_HO_Complete Request.
  Source Identifier: ruben@mn.net
  Destination Identifier: yheah@poa.net
  MESSAGE MIH_MN_HO_Complete Request header.
  Ver: 0  AckReq: 1  AckRsp: 0  UIR: 1
  FN: 0  SID: 3  OPCODE: 1  AID: 10

------------------------------------------------------------------------
Received IEEE 802.21 frame:
  Dest MAC: 00 02 6f 2f 04 74
  Ver: 0  Ack req: 0  Ack rsp: 1  MF: 0  UIR: 1  FN: 0
  SID == Command Service
  AID == MIH_MN_HO_Complete
  OPCODE == Response
  Transaction ID: 7  Variable Payload Length: 71

************************************************************************
MICS: MESSAGE RECEIVED MIH_MN_HO_Complete Response.
************************************************************************
  Source Identifier: yheah@poa.net
  Destination Identifier: ruben@mn.net
MIH USER: MIH_SAP RECEIVED MIH_MN_HO_Complete.confirm.
--> NEW CONNECTION ESTABLISHED.

Monitoring RSS:
Network: yheah ... Signal strength: -32 dBm. GOOD_LV.
Network: yheah ... Signal strength: -34 dBm. GOOD_LV.
Network: yheah ... Signal strength: -36 dBm. GOOD_LV.
Network: yheah ... Signal strength: -34 dBm. GOOD_LV.
Network: yheah ... Signal strength: -32 dBm. GOOD_LV.
Network: yheah ... Signal strength: -32 dBm. GOOD_LV.
Network: yheah ... Signal strength: -31 dBm. GOOD_LV.
```

Figure 4.17: MN LINK_POWER_UP and MIH Registry processes

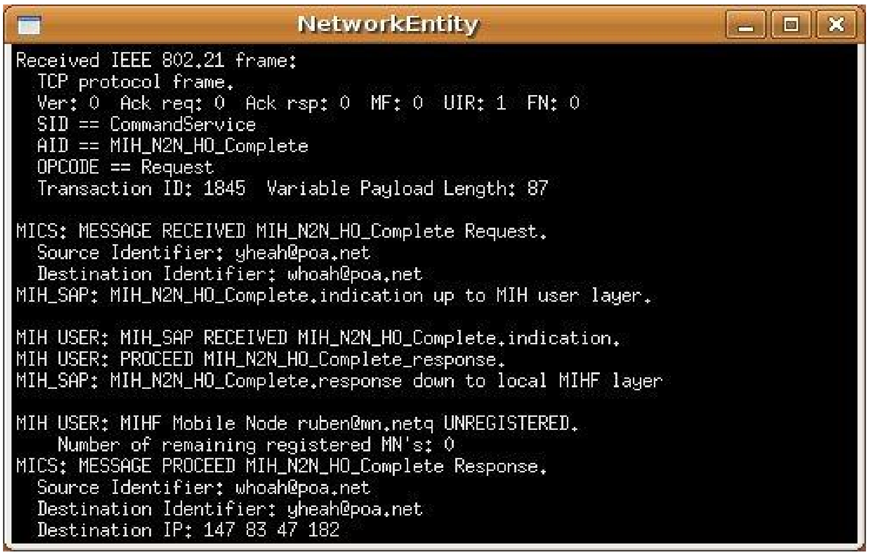
Once the response SAP is received in the MIH User, the handover completion procedure starts. Now the MN sends MIH_MN_HO_Complete request to the new serving PoA in order to release resources in the old network. The followed schema is the same as the one seen for MIH_MN_HO_Commit, but target and serving PoAs exchange their roles.

In figure 4.17 it can be seen how the MN sends the request message to *yheah@poa.net*, which is currently the PoA that provides IEEE 802.11 connectivity to the network. Figure 4.18 shows how this PoA receives the mentioned message and then it sends the corresponding MIH_N2N_HO_Commit request to *whoah@poa.net*, which is the old PoA. Here, the destination IP of the wired link of the old PoA can be observed. Figure 4.19 shows how the MIH_N2N_HO_Complete request message arrives to the old PoA and so it unregisters the MN MIH. The number of associated MNs is printed on the output of the debugger and the word UNREGISTERED is clearly shown. This entity transmits the corresponding MIH_MN_HO_Complete response message to the MN in order to confirm and close the current transaction. In figure 4.17 it can be seen how the MN receives this response message. In the output, the NEW CONNECTION IS ESTABLISHED is printed out. This means that the handover process has been finished and the old resources have been released. Form this moment on, the MN begins to monitor the quality of the new link, which is the current data carrier.



```
Received IEEE 802.21 frame:
  Dest MAC: 00 22 15 61 86 95
  Ver: 0  Ack req: 1  Ack rsp: 0  MF: 0  UIR: 1  FN: 0
  SID == CommandService
  AID == MIH_MN_HO_Complete
  OPCODE == Request
  Transaction ID: 7  Variable Payload Length: 71

MICS: MESSAGE RECEIVED MIH_MN_HO_Complete Request.
  Source Identifier: ruben@mn.net
  Destination Identifier: yheah@poa.net
MIH_SAP: MIH_MN_HO_Complete.indication up to MIH user layer.

MIH USER: MIH_SAP RECEIVED MIH_MN_HO_Complete.indication.
MIH USER: PROCEED MIH_N2N_HO_Complete_request.
MIH_SAP: MIH_N2N_HO_Complete.request down to local MIHF layer

MICS: MESSAGE PROCEED MIH_N2N_HO_Complete Request.
  Source Identifier: yheah@poa.net
  Destination Identifier: whoah@poa.net
  MESSAGE MIH_N2N_HO_Complete Request header.
  Ver: 0  AckReq: 0  AckRsp: 0  UIR: 1
  FN: 0  SID: 3  OPCODE: 1  AID: 11
  Destination IP: 147 83 47 181
```

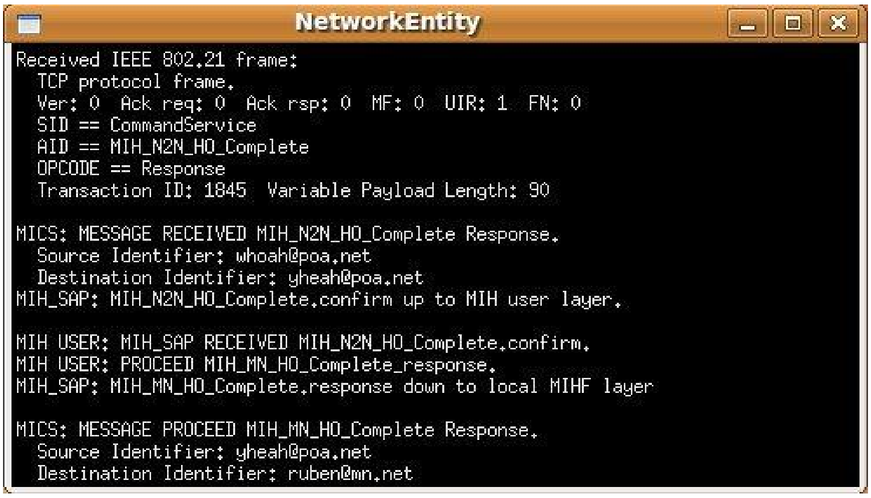
Figure 4.18: New Serving PoA receives MIH handover completion and sends it to old PoA



```
Received IEEE 802.21 frame:
  TCP protocol frame.
  Ver: 0  Ack req: 0  Ack rsp: 0  MF: 0  UIR: 1  FN: 0
  SID == CommandService
  AID == MIH_N2N_HO_Complete
  OPCODE == Request
  Transaction ID: 1845  Variable Payload Length: 87

MICS: MESSAGE RECEIVED MIH_N2N_HO_Complete Request.
  Source Identifier: yheah@poa.net
  Destination Identifier: whoah@poa.net
MIH_SAP: MIH_N2N_HO_Complete.indication up to MIH user layer.

MIH USER: MIH_SAP RECEIVED MIH_N2N_HO_Complete.indication.
MIH USER: PROCEED MIH_N2N_HO_Complete_response.
MIH_SAP: MIH_N2N_HO_Complete.response down to local MIHF layer

MIH USER: MIHF Mobile Node ruben@mn.netq UNREGISTERED.
    Number of remaining registered MN's: 0
MICS: MESSAGE PROCEED MIH_N2N_HO_Complete Response.
  Source Identifier: whoah@poa.net
  Destination Identifier: yheah@poa.net
  Destination IP: 147 83 47 182
```

Figure 4.19: Old PoA receives MIH handover completion and performs response

This has been an example of operation of MN and PoA entities in the current test-bed. Different situations can be observed depending on the given conditions. For instance, when a handover is initiated, if the quality of the link is restored before the second threshold has been crossed, the handover is aborted and it is indicated in the output with the message SIGNAL STRENGTH RESTORED, HANDOVER ABORTED. It is also common to see some messages repeated at both outputs. It occurs when a determined message has not arrived to its destination and the transaction needs to be repeated. Not being so typical, rejection messages in PoAs' outputs can be observed in case the MN tries to access a PoA without being registered before.



```
Received IEEE 802.21 frame:
  TCP protocol frame.
  Ver: 0  Ack req: 0  Ack rsp: 0  MF: 0  UIR: 1  FN: 0
  SID == CommandService
  AID == MIH_N2N_HO_Complete
  OPCODE == Response
  Transaction ID: 1845  Variable Payload Length: 90

MICS: MESSAGE RECEIVED MIH_N2N_HO_Complete Response.
  Source Identifier: whoah@poa.net
  Destination Identifier: yheah@poa.net
MIH_SAP: MIH_N2N_HO_Complete.confirm up to MIH user layer.

MIH USER: MIH_SAP RECEIVED MIH_N2N_HO_Complete.confirm.
MIH USER: PROCEED MIH_MN_HO_Complete_response.
MIH_SAP: MIH_MN_HO_Complete.response down to local MIHF layer

MICS: MESSAGE PROCEED MIH_MN_HO_Complete Response.
  Source Identifier: yheah@poa.net
  Destination Identifier: ruben@mn.net
```

Figure 4.20: New Serving PoA receives MIH handover completion and advertises MN

Moreover, it is also defined a useful set of flags for both MN and PoAs which aid in program debugging. They are Boolean variables which set to 1 print out some kind of information, and set to 0 inhibits its output. One of the most useful of them is the flag which prints out the current state of the local state machine in the MN or PoA. This information is used to control and debug the synchronism of the involved sate machines during a determined transaction. Another flag which has helped the development is the one that prints out the hexadecimal contents of an IEEE 802.11 or Ethernet received packet. This flag is an aid for data encapsulation into the sent and received messages. There are also included other flags which provide information about the requested IEs to the MIIS in both entities involved, and a flag which prints out the SAP being used and the source and receiver of it.

4.2.2. Round Trip Time (RTT) tracking for ping program

4.2.2.1. Introduction and explanation

Next, some tests have been carried out in order to evaluate a set of system features. The first test has the aim to analyze the performance of a simple application that generates continuous traffic between two peers along the network during a handover from PoA to PoA. In this case, one peer is the MN and the other one is a remote entity bounded to the laboratory core network. The *ping* program is the user application that supports this performance.

The *ping* [40] application works as follows: a peer generates a network IP packet with a determined payload length (its content is not important). Its packet name is *ping* and it is addressed to a selected IP peer somewhere in the network, which generates another similar packet in response. This response is called *pong* and it is received by the source peer. The ping application calculates the Round Trip Time (RTT), which is the time since the outgoing packet has been sent until the incoming packet has been received. This program can be set in order to be reproducing this cycle with the desired time cadency and packet size. When this program finishes, the average, maximum, minimum and standard deviation of the RTT is provided, as well as the percentage of packet losses.

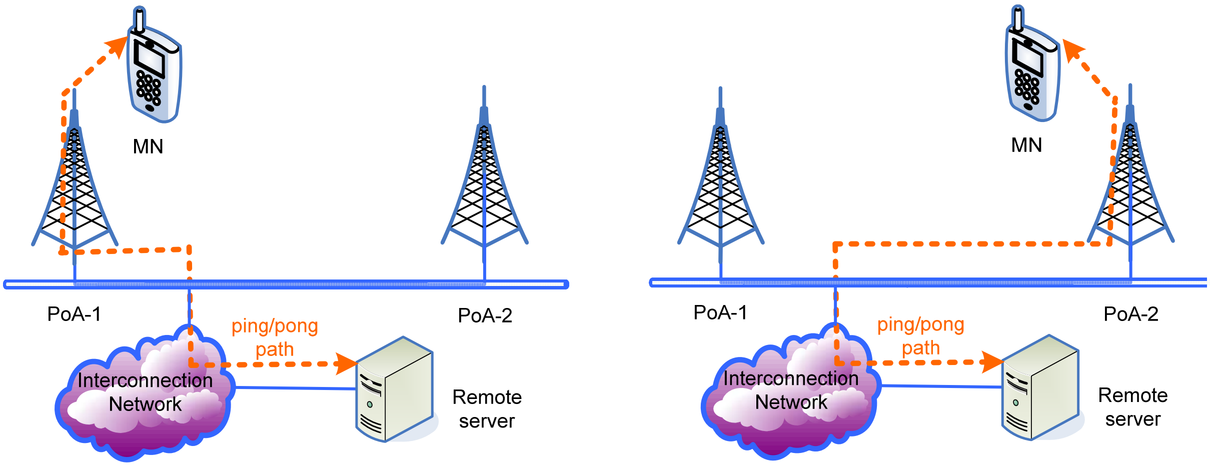


Figure 4.21: Ping tracking during a handover

The situation for this test is the following: a MN gets attached with acceptable quality of signal to a PoA via bootstrapping and is associated to the corresponding AP using the DHCP protocol. In this precise moment, the MN daemon starts running a parallel application which registers into a file the ping RTT responses and events generated in the MN. A simple application in the associated PoA is started to decrease progressively the transmitted power in order to simulate that the MN is going away from its coverage area. During this decrease all the handover events occur and so they can be reviewed and analyzed. The general communication sketch is depicted in figure 4.21.

In order to know the way a packet is routed through the different interfaces of a device at IP level, it is necessary to understand how the kernel works with its IP route table. Every time a new a new DHCP connection is established, the kernel route table is updated with the new routes. When the peer is associated to a new AP, two new routes are added. The first one is intended for the local network, whose packets can be delivered directly to the receiver. The second one is the default route, which define the gateway element to which the packets must be delivered. With two interfaces and in the current application, the kernel route table is updated as described in the following lines.

At bootstrap time and before authentication just an interface is up, but no path is defined. The route table is empty:

Route	Destination	Gateway	Mask	Device

Table 4.3: Empty kernel route table

Once associated to an AP, two routes are defined. The first one provides connectivity to the local network through the AP; and the second one is the default route, which provides connectivity to other networks. When a packet needs to be routed, the system searches the route in the order the routes appear in the table.

Route	Destination	Gateway	Mask	Device
Route 1	192.168.5.0	192.168.5.1	255.255.255.0	wlan0
Route 2	0.0.0.0	192.168.5.1	0.0.0.0	wlan0

Table 4.4: Kernel route table with one active wireless NIC

During HO and after network selection, the second interface is enabled. The association provides two new routes, analog to these stated for the previous one. These routes are added to the route table as follows: the new direct delivery route is always placed before the default routes, then the order it is introduced into the direct delivery sub-group is first the highest address; whereas the new default route just introduced is placed in the first position amongst the default routes.

In case two routes are suitable, the selection is done depending on the order the routes are introduced in this table. In this situation, there exist two routes to external networks, so the first one is chosen due to the order of apparition, as stated some lines before.

Route	Destination	Gateway	Mask	Device
Route 1	192.168.6.0	192.168.6.1	255.255.255.0	wlan1
Route 2	192.168.5.0	192.168.5.1	255.255.255.0	wlan0
Route 3	0.0.0.0	192.168.6.1	0.0.0.0	wlan1
Route 4	0.0.0.0	192.168.5.1	0.0.0.0	wlan0

Table 4.5: Kernel route table with two active wirelesses NIC

When handover is concluded, the first introduced routes are deleted as the first interface is disabled, and so the route table just stores the last introduced routes.

Route	Destination	Gateway	Mask	Device
Route 1	192.168.6.0	192.168.6.1	255.255.255.0	wlan1
Route 2	0.0.0.0	192.168.6.1	0.0.0.0	wlan1

Table 4.6: kernel route table with the remaining active wireless NIC

The RTT of a ping packet depends on several factors such as the number of hops between the instants of sending and receiving this packet, the transmission time for each hop, the time the CPUs need to process these packets and the propagation times. In the given scenario, the hops are fixed but the time the CPUs need to process the packets can be variable due to the load of other simultaneous operations. In case this time was constant (it means, processors exclusively dedicated to handle network traffic) the RTT would be also constant. During a handover, the described situation should be observed but experiencing a difference of a constant in the moment the connection is transferred to the alternative interface and to the target network. This difference is given by the propagation time and the number of hops.

Although the scenario has been intended in order to provide the lower variations as possible, the OSs in each peer use to handle several processes at the same time which leads to a variable time to manage each of them. Frequently, packets that cannot be processed in the moment they arrive are stored in a queue increasing the time they remain in a system.

4.2.1.2. Performed tests

The following tests try to portrait the variation of the RTT in different scenarios [41]. Five different transmission rates for ping packets have been introduced in each scenario. The time interval of each ping has been adjusted to the minimum possible for this application, which is 10ms. The packet size depends on the target transmission rate for each test. The scenarios differ in the way the wired network elements are connected one to each other: first, the PoAs are connected directly one to the other through a network wire, and the ping target is one of them; then, a both network elements are connected to a hub, to which a third entity is also bounded acting as the target of the sequence of pings; finally, the PoAs are connected to the laboratory network, as well as a third entity acting as the target of the ping sequence.

Before proceeding with the experiments, the correct behavior of the elements introduced into the NM is checked in order to obtain a data file with the required information. The ping is set to the default values, which is 1ms of cadency and a packet size of 84 bytes (672 bps, very low transmission rate), and the interconnection of the wired network elements is done through the local laboratory network.

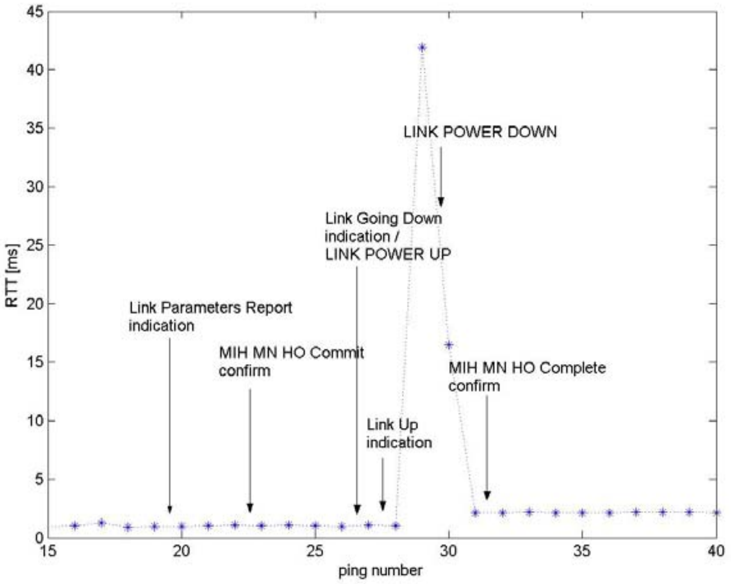


Figure 4.22: Ping tracking during a handover at 672 bps

The features of this test correspond to a standard ping invocation with the aim of checking connectivity between the two ends of the communication. The test considers a very low transmission rate which does not provide a clear idea about the capacity of the system. Instead, this test is useful to see that the connectivity between ends is maintained at L3 during a handover, routing the request and response packets through the two different PoAs.

Figure 4.22 shows a constant RTT of about 1 [ms] until the first interface is switched off. The RTT depicts a peak just before it happens due to a slight delay of representation, error which is corrected in the subsequent tests. This peak represents an increase in the RTT due to the time to process the new route of the outgoing packets. After that, the RTT is reestablished again around 3 [ms]. In order to see further capacities of the system, the transmission rate has been gradually increased in the upcoming tests.

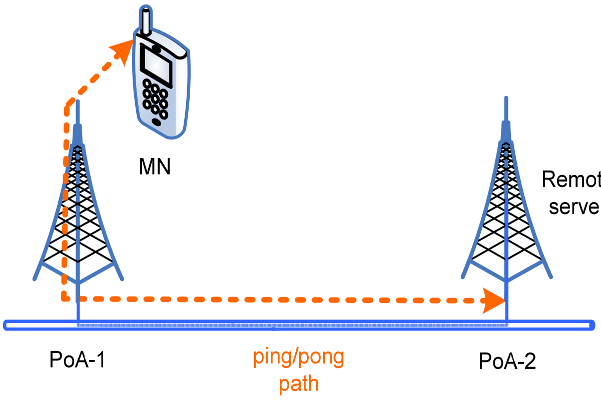


Figure 4.23: Scenario 1, topology using a direct network wire between PoAs

Having fixed the bugs in data collection, the system is ready to provide a set of results. First, the simpler scenario is mounted and prepared to collect data. Figure 4.23 represents the topology described before. The next pages summarize and illustrate some of the tests that have been done in order to check connectivity capacities.

Figures provide a qualitative idea about how the change of interface and network affects to packet transmissions when the handover at MIH level needs to take place. The most significant events are depicted in figures, starting from the trigger of a Link_Parameters_Report indication, which is the beginning of the handover actions, to the end of the association with the new PoA and the switching off of the old interface. The following lines try to sum up the behavior seen in the patterns presented before.

The scenario where the PoAs are bounded one to each other through a network cable and the ping is directed to one of these peers (scenario 1) depicts some interesting features. First of all, it can be seen that for the lowest bitrates (64 kbps and 128 kbps) the RTT average is similar and about 50ms. Both patterns present a peak of RTT just after the interface through which the packets need to be routed is changed. This peak is several times higher than the RTT average at both sides of it resulting in a maximum of 2,7s for the 64 kbps flow and almost 1s for the 128 kbps flow. After PoAs find the path of response to the ping message, the RTT average is restored again.

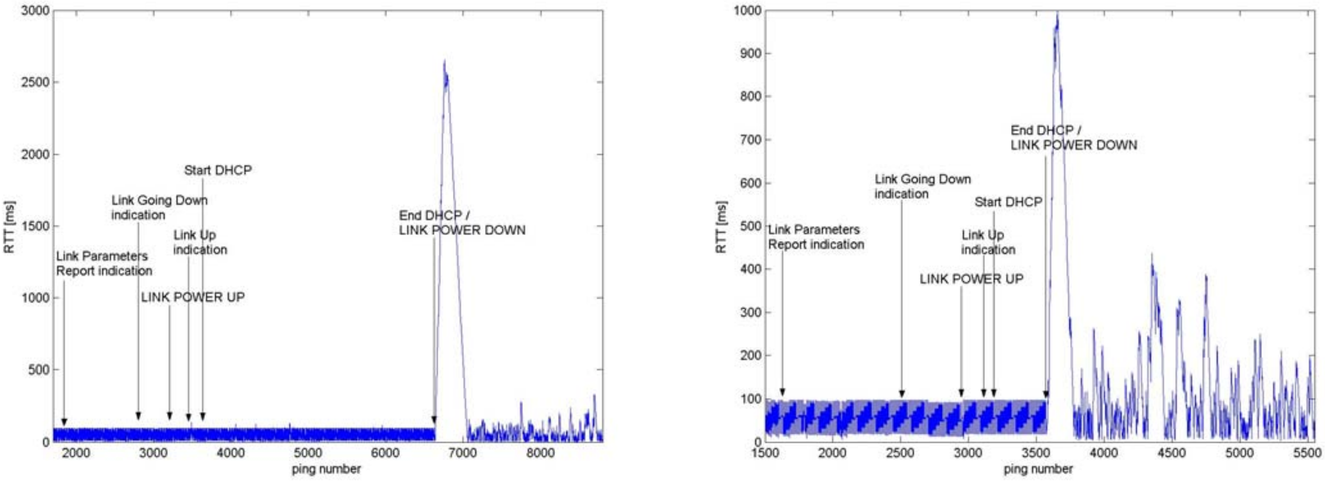


Figure 4.24: Scenario 1, ping tracking during a handover at 64 kbps (left) and 128 kbps (right)

At higher bitrates, the peak disappears and the RTT experiences a significant increase. Values obtained for 256 kbps and for 512 kbps are over 3,5s when the flow of packets changes its path. Even, some packet loss can be appreciated just after exchanging the data flow (straight lines that seem to cut peaks instead of following its shape). The increase of the RTT is due to the amount of extra time needed to process the size of the new flow of packets. The received packets in the PoA remain long times in reception and transmission queues increasing the RTT average in the target networks.

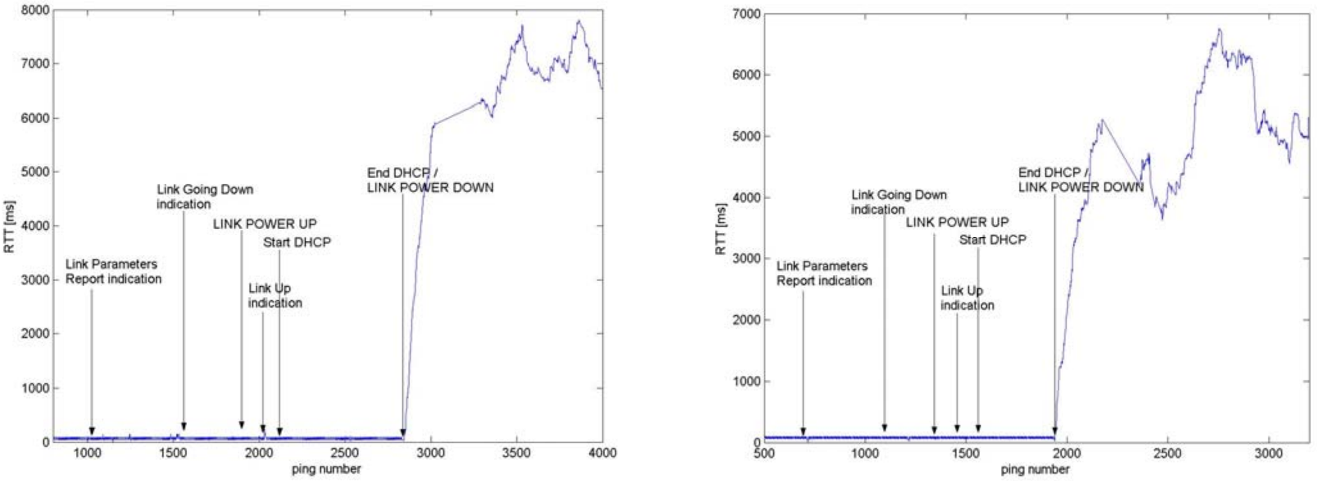


Figure 4.25: Scenario 1, ping tracking during a handover at 256 kbps (left) and 512 kbps (right)

In figure 4.26, the handover performance for a ping tracking at 1024 kbps can be observed. No packet losses seem to occur during handover critical moments, except after Link_Parameters_Report and before Link_Going_Down indications. These losses can be due to time expiration of the response packets caused by the time these ones have been remaining in the destination peer. Before the interfaces have been swapped, the RTT average is about 1,4s which is several times higher compared with previous transmission rates. After swapping, the RTT tends to be increased over 10s.

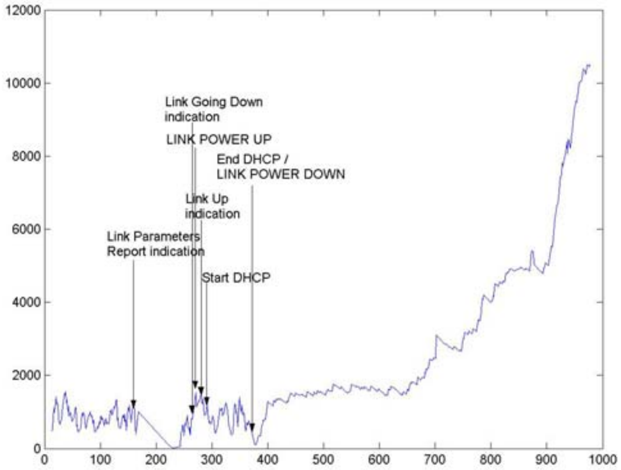


Figure 4.26: Scenario 1, ping tracking during a handover at 1024 kbps

The scenario with the hub in the middle (scenario 2) provides similar results in each test done. The ping now is addressed to a remote server, which is also connected to the hub.

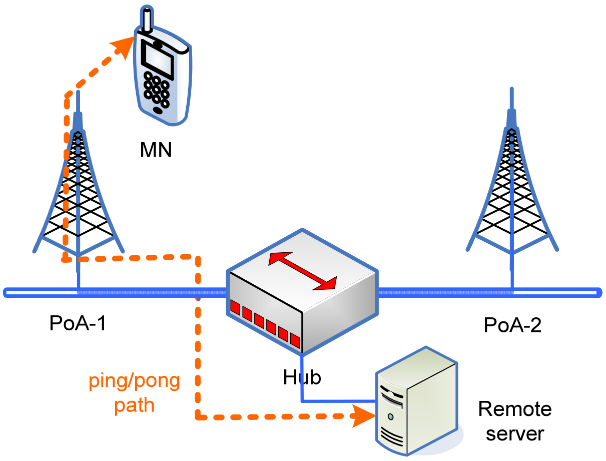


Figure 4.27: Scenario 2, topology using a hub as interconnection element

Prior to handover, RTT average times are almost constant and low with similar values to those obtained in the previous experiment with the same rates. Tests done at 64 kbps, 128 kbps and at 512 kbps tend to unstable RTT times after handover takes place. At 256 kbps and 1024 kbps RTTs tend to be stable. The fact that the behavior of these times differs independently of the bit rates is due to the charge of operations in a determined PoA. Actually, the machines used to act as PoAs share its processor resources with other processes which can lead to the situations depicted in the different figures.

Moreover, the hub is a physical device which retransmits by all its outputs the signal received by all its inputs. Before a handover, a flow of pings arrives to the PoA which is not the target of them. Besides, a flow of pong packets also arrive to this PoA due to the stated reason. It does not affect to the handover performance before swapping the link, but after it happens a flow of pong packets is still arriving from the old PoA to the target PoA due to the delay caused by the processing time and the waiting time of the old PoA. This causes an extra flow of packets in the target PoA which need to be discarded at the same time that the flow of ping packets needs to be responded. These extra operations cause an indirect increase of the RTT.

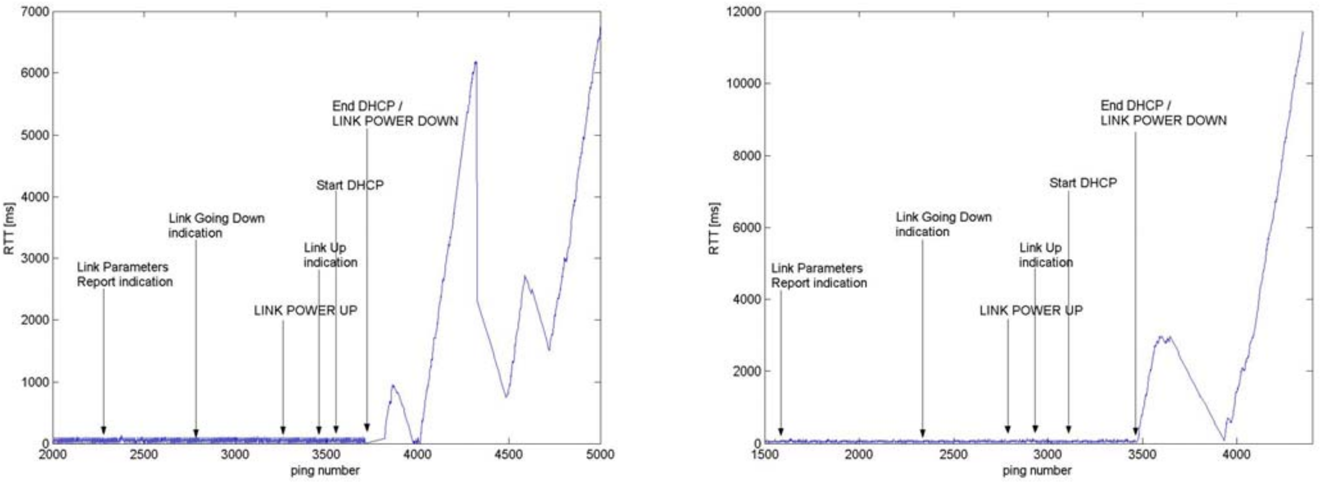


Figure 4.28: Scenario 2, ping tracking during a handover at 64 kbps (left) and 128 kbps (right)

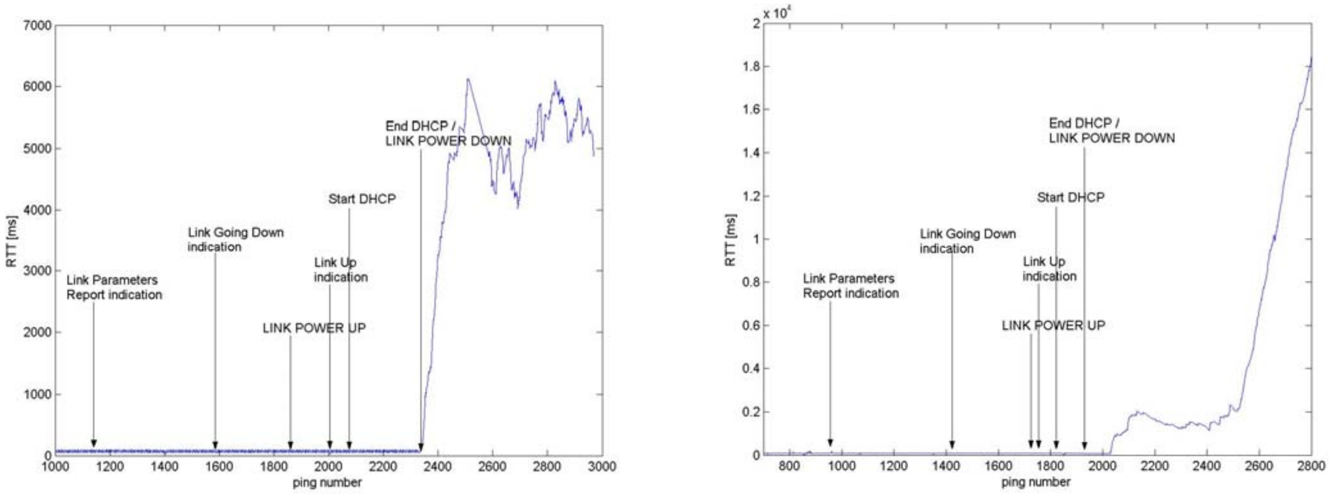


Figure 4.29: Scenario 2, ping tracking during a handover at 256 kbps (left) and 512 kbps (right)

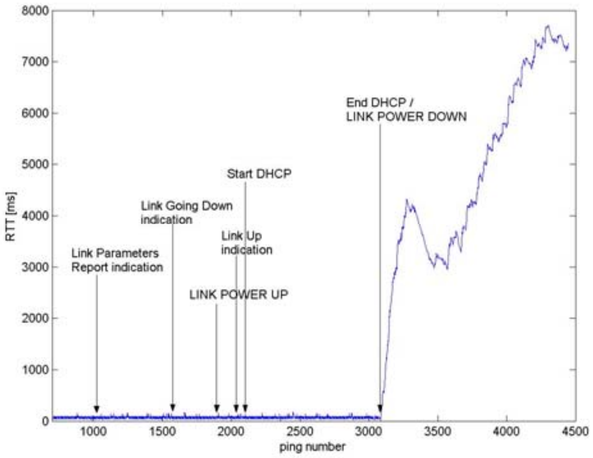


Figure 4.30: Scenario 2, ping tracking during a handover at 1024 kbps

The scenario with the laboratory network as interconnection system between devices (scenario 3) shows a set of results that have similar behaviors.

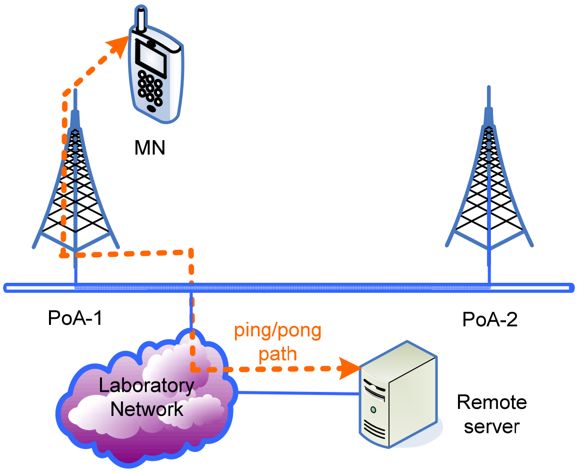


Figure 4.31: Scenario 3, topology using the laboratory network

First of all, all the shown handovers depict almost the same schema: the RTT is approximately constant until the link is swapped, sketching a hop in this precise moment with a new approximately constant RTT. This is the ideal behavior desired for a handover procedure. Particularly now, the figure depicting a handover at 64 kbps shows a pair of gaps just after the communication flow has been swapped. Actually, these gaps are not the absence of pong responses; they are peaks of RTT which correspond to 1,2s and 0,6s respectively. The instants in which they occur indicate processing times due to the new routes. These peaks are instantaneous and disappear quickly because the processor has enough resources to restore a stable flow.

To sum up, it must be highlighted that the experimental results show that the MN starts the handover before the disconnection of the old link. By this way packet losses caused by a broken link and a mandatory DHCP association process, whose timings are variable, are minimized because they happen in parallel to the data communication with an acceptable signal quality. The experiments show certain RTT tracking profiles which are repeated, but there is not a general rule to embrace them given the environment conditions such as processors not exclusively dedicated to network routing management.

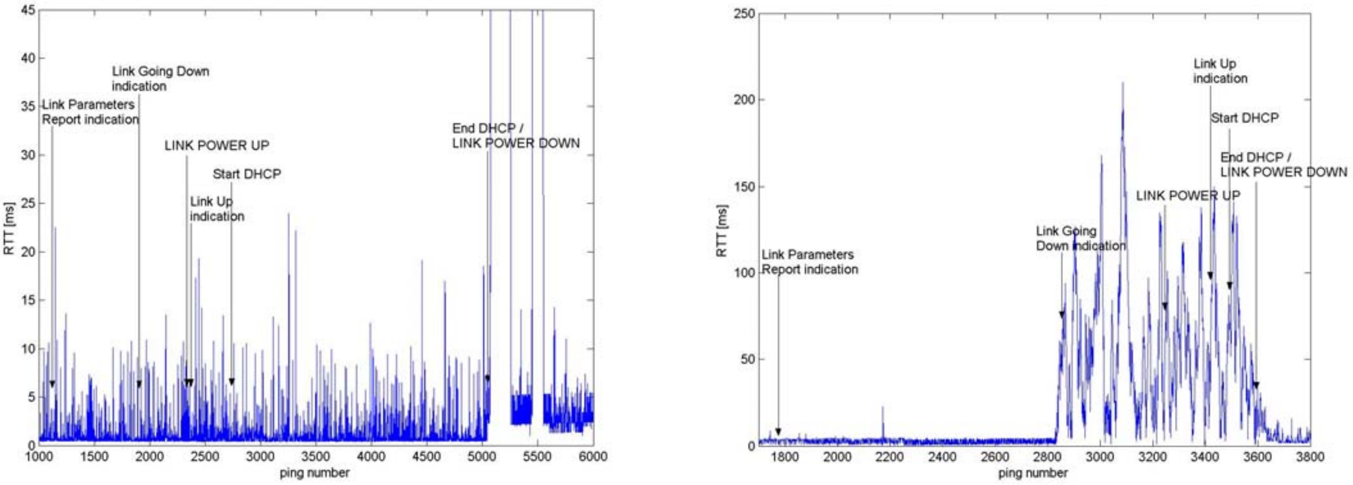


Figure 4.32: Scenario 3, ping tracking during a handover at 64 kbps (left) and 128 kbps (right)

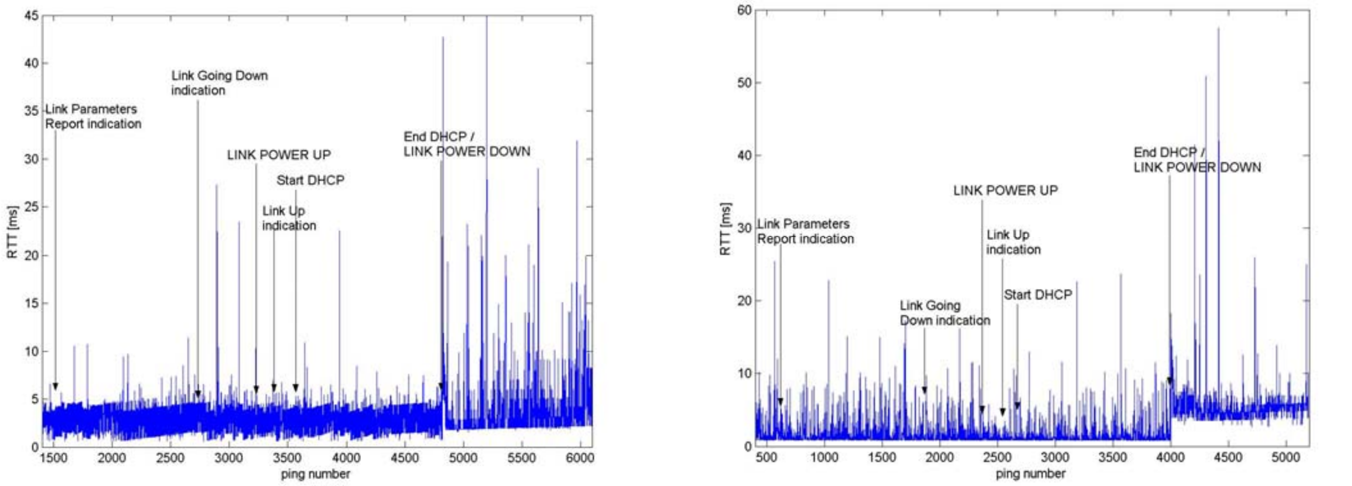


Figure 4.33: Scenario 3, ping tracking during a handover at 256 kbps (left) and 512 kbps (right)

The generated flows of data can be compared to some kind of streaming applications. A flow of 64 kbps is between an acceptable quality of audio streaming and a typical flow of VoIP, which is lower than 40 kbps. A flow of 128 kbps is rather a good quality for audio streaming, which can reach a quality very alike to CD sound. Broadcast radio stations such as *Radio Flaixbac* also transmit its contents through internet using this constant bit-rate. Higher bitrates ranging from 500 to 1500 kbps are commonly used for video streaming. The *Imagenio* IP

TeleVision (IPTV) service of *Telefonica* started to provide television by internet in 2005 with a bit-rate near to 1500 kbps, whereas other poorer video streaming flows require lower transmission rates.

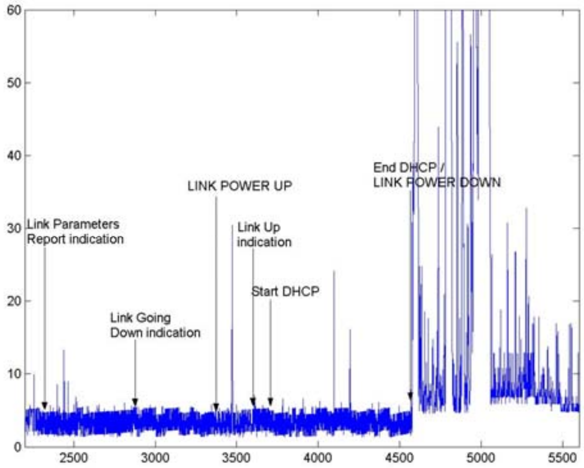


Figure 4.34: Scenario 3, ping tracking during a handover at 1024 kbps

To conclude, it may be interesting to test the system response with applications different from *ping* capable to inject packets in the network. One of those applications is IPERF, and it is capable to emulate UDP data flows with higher cadency between packets than *ping* program. The problem is that the daemons current developed are not ready to handle transport layer connections because this feature does not fall within the scope of this work. For now, this task is left for upper layer applications.

In these experiments, it can be seen that the DHCP association times are very variable and not negligible. As stated, they occur in parallel to data communication through the old link and so the flow is not disrupted. Nevertheless, the time in which a handover needs to take place depends on the MN speed of movement, and if those DHCP associations have a high latency the conditions may change before handover is complete. Note that, RTT timings near to 10 seconds (as obtained in the two first experiments when the MN gets engaged to the target network) are not acceptable for real time traffic, and so a further solution must be found after having identified the source of these delays. In the following sub-section the different timings for handover are characterized in order to study where the communication could fail and how to solve it.

4.2.3. Handover latency and losses

4.2.3.1. Introduction and explanation

In the line of the previous results, it seems that the network topology does not have a decisive influence in service disruption and packet losses. So, the most general scenario has been chosen to do the following test. This scenario is the one that considers the laboratory network as interconnection element between PoAs and the server, and is sketched in figure 4.31.

The main target in the current test is characterizing the most relevant times involved in a handover process. As stated, it has been observed that the DHCP connection has a very variable time depending on the conditions it is performed. Moreover, handovers are done by hand, and timings between signal strength thresholds are dependent on the movement. Both times have been characterized in order to get a clearer idea of the handover latency at pure MIH level. Another thing that must be checked is packet losses. As well as for characterizing timings, the numbers of packets for DHCP and decreasing signal strength intervals have been also measured. [42]

In order to acquire these results, the followed schema is very similar to the schema used for the previous test. This time, besides introducing the output of a ping packet, the instant of time when the program is executed is also introduced. The most relevant events that occur after and before swapping two interfaces for a handover are also saved in the same file. With these elements, all the elements required to do the test are present.

Tests have been done following the same pattern as in the test before. There have been taken 4 handover samples for each different bit-rate in ping program, and timings and packet results have been averaged. To emulate the movement of the MN, the transmitted signal strength from the PoA is decreased using a local application.

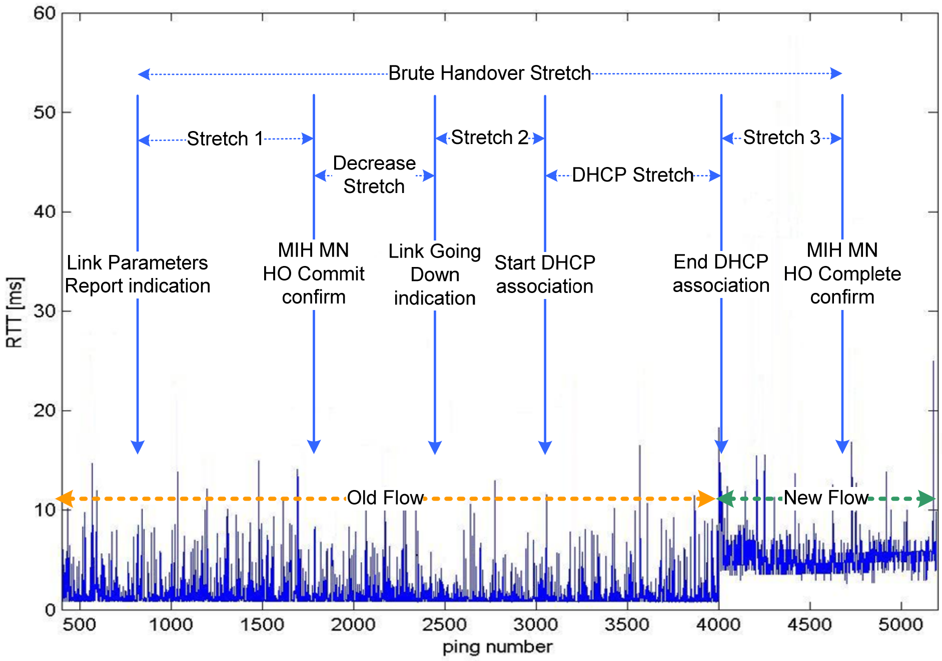


Figure 4.35: Definition of the different time intervals of a handover

4.2.2.2. Performed tests

Figure 4.35 sketches a handover procedure from the previous test with the events that are needed for the current representation. The nomenclature used for this section is indicated. First, the stretch called Brute Handover depicts the interval defined by the beginning of the handover, which is indicated with Link_Parameters_Report indication, and the end of all the handover process, which is determined by MIH_MN_HO_Complete confirm SAP. This interval includes the Decrease and DHCP sub-intervals which have a very variable time and do not depend on the MIH implementation. As stated, these intervals are the time the system waits for signal degradation and the DHCP association processes. Table 4.7 sums up the averaged times for Brute Handover, Decrease and DHCP stretches. The resulting time of handover for the MIH implementation is indicated and is the result of subtracting the Decrease and DHCP times to the Brute Handover times.

Data rate	Brute Tho [s]	T decrease [s]	Tdhcp [s]	Tho [s]
64 kbps	16,28	4,69	7,85	3,73±0,33
128 kbps	26,14	2,31	18,96	4,86±1,18
256 kbps	15,47	1,96	9,08	4,43±0,97
512 kbps	15,85	2,96	8,75	4,15±0,33
1024 kbps	15,36	2,64	8,38	4,33±0,57
AVG	17,82	2,91	10,60	4,30±0,68

Table 4.7: Handover latency

In table 4.8 the averaged number of sent packets during each interval is gathered. Again, the handover packets are the result of subtracting the Decrease and DHCP packets to the Brute Handover packets. The Brute Handover packets are the result of adding the number of packets of the intervals composing it.

Data rate	Brute ho pk	Decr. Pk	Dhcp pk	stretch 1 pk	stretch 2 pk	stretch 3 pk	Ho pk
64 kbps	3649	1305,75	1640,5	214,75	349	139	702,75±60
128 kbps	2512	342,25	1531,25	241,25	267,75	129,5	638,5±61
256 kbps	2832,75	335,5	1837	205,5	320,75	134	660,25±99
512 kbps	1490,75	303	712,75	142,75	160,25	172	475±116
1024 kbps	1078,75	185,5	479,75	176,75	128	108,75	413,5±147
AVG	2312,65	494,4	1240,25				578±96,49

Table 4.8: Number of packets sent (ping) during a handover

Table 4.9 collects the number of received and lost packets during a handover without considering the Decrease and DHCP periods. The results are obtained by combining the information of this table with table 4.8. The number of packet losses is determined by subtracting the number of received pong packets to the number of sent ping packets during the neat handover interval. The resulting average of packet loss is also determined by the quotient of the number of received packets during the neat handover interval by the number of sent packets during the same interval.

Data rate	stretch 1 pk	stretch 2 pk	stretch 3 pk	Ho pk	Total Lost	AVG lost (%)
64 kbps	109,25	330,25	127,25	567±84	136±31	19,64±5,61
128 kbps	95,25	226,25	111,5	433±122	206±83	32,86±15,37
256 kbps	95,5	280,5	123,75	500±120	161±62	24,81±9,99
512 kbps	64,75	141,5	163	369±135	106±22	23,68±8,07
1024 kbps	77,5	120,75	99,25	298±163	116±41	30,62±13,79
AVG	88,45	219,85	124,95	433±125	145±48	26,32±10,56

Table 4.9: Number of received (pong) and lost packets during a handover

From tables above, it can be appreciated that the most consuming operation during a brute handover time is the DHCP association. This time is too long compared to the neat handover time, and should be minimized. The test at 128 kbps shows that the averaged required time has taken almost 20s to perform this action, which is too much and could lead to a broken link due to changes in channel conditions. For example, a person walking at 1,6m/s covers a distance of 32m in 20s. By the law of free space losses, the received signal strength decreases inversely to the square of the distance or, what is the same, it decreases 20xlog(d[m]) [dB]. For the given distance, the receiver may have lost nearly 30 [dB] of received signal strength, which is much more than the allowed for the margins left for signal strength events triggering: by this time, the link may have broken. So, it can be seen that the classical DHCP association is not the most proper way to get attached to a network when mobility is a required feature. The best suitable solution seems to be using mobility protocols, such as those derived from MIP family.

The neat handover time seems to be around 4s without considering DHCP procedures or waiting intervals for signal decrease. This time cannot be easily reduced because it just involves the intervals of transmission of some MIH messages and the time the computers need to process them and to make decisions. As well as for the rest of the results obtained, this time does not depend on the transmission rate the data interface is using.

The average of packets lost during a handover is around the 26%. This average may or may not be acceptable depending on the application. For example, if instead of a ping we had a non real time connection oriented TCP application, these losses would be negligible for a 4s period. For a real time application such an audio or video conference, this average could result in a loss of a second of the conversation, which cannot be considered as a seamless handover at all. Different strategies to improve those losses can be based on the study and adjustment of the RSS thresholds or even considering overlapping during a certain period two flows of data coming from the serving PoA and the old PoA, before the old connection expires. These strategies are left for future work and do not fall within the scope of the current implementation.

5. Future Work and Conclusions

5.1. Future Work

The work does not finish here. Now we have a base to implement applications capable to work under lower layer mobility conditions. Fortunately or not, this is not enough. The developed applications require technical improvements in order to solve some minor implementation issues, as well as more enhancements in order to provide flexibility to future feasible scenarios.

The present section outlines the most relevant aspects that can be completed in future revisions of the current work. It starts with the description of some problems found in the operation of the developed daemons, and continues with elements suitable to be added such as MIH over transport protocols or network controlled handovers. It also provides a summary of the documents that can aid in improving MIHF discovery procedures replacing the current ones. A brief description of the suitable network protocols intended to replace the classical IPv4 is provided next as a way to solve the latency problems for the DHCP problem. Finally, an overview of some application protocols is provided as well as a brief description of the future compatibility with broadcast technologies.

5.1.1. Problems directly associated to the implementation

Although the designed and validated applications fit most of the initial specifications, a few problems have appeared. Some of them have been previously commented, but deserve being mentioned together with solving proposals. After observing the operation of the designed and implemented software, it is possible to expound a set of feasible improvements or enhancements to the applications.

Those proposals to solve the problems that have appeared and to improve the operation and possibilities of the applications have the aim of improving the current implementation by achieving a more robust and efficient set of applications in future versions. By this way, it is possible to enhance the possibilities that the daemons can offer to the users, possibilities that are increased every day with the apparition of new technologies and the demand of new services.

5.1.1.1. Pcap buffer overflow in MN

This problem is directly associated to the implementation. It is manifested as follows: after the MN daemon has been run several times, the execution fails throwing an exception. In the debug output can be read that there has been a *pcap* buffer overflow. It is an uncommon issue, but has to be born in mind if the prototype needs to be improved.

Technical documents about developing with *pcap* tools rarely mention methods to clean up the library buffer or how to improve designs in order to avoid reaching to this type of situation. Is a bit disappointing to see that very similar procedures are implemented in the PoAs using the same *pcap* elements, but the mentioned problem does not appear. The first step would be checking if the MN application throws the same exception being executed on a different hardware. Otherwise, a further research in Internet documentation should be performed in order to find a suitable solution.

5.1.1.2. Thread synchronism between TSM and MIHF

This problem is rarely manifested, and occurs after getting two determined messages in the MN: MIH_MN_HO_Commit Response and MIH_MN_HO_Complete Response. The system enters in a dead-lock situation that has not been resolved yet. The MIHF module gets stopped because the corresponding working instance of the TSM has not been terminated.

The program flow may have reached this situation due to a delay in the state change of the MN TSM instance respect to the received response message. The debugging output throws the reception of the message by part of the MIHF, and also shows the state change within the local state machine. The end of an instance of TSM is indicated at the output when everything has gone right, but in this case this message never arrives. If the instance of the TSM is not suppressed, the corresponding thread of the MIHF module is not unlocked, and so the program cannot continue with is typical flow resting stuck.

Finding a solution to this issue can take some time since this problem rarely appears. The resolution procedure can be based in placing *printfs* around the critical code and see exactly when the involved processes are stopped. A *printf* is a C function that allows printing a clause through the standard output. Once the precise point has been localized, a solution must be found in order to break the dead-lock situation.

5.1.1.3. Modification of CHOICE data type into the applications

When the work started, the available version of IEEE 802.21 standard was still a draft. There, the definition of the data types was not definitive and some modification may be introduced in later amendments. In particular, one data type experienced a slight redefinition that in some points of the implementation caused problems. This data type is known as `CHOICE`.

The standard defines `CHOICE` as a data type that has many other types declared within. Just one of these types is used for a given variable of the type `CHOICE`. The pseudo code declaration would be the following:

```
CHOICE { VAR1, VAR2, … }
```

When a determined variable of the type `CHOICE` gets a value, the type of this value is just one of the types designed as `VARx`. The definition of `CHOICE` is very similar to the existing `union` present in the C/C++ language.

Problems arise when the definition of this data type finishes here. For instance, a variable of the type `CHOICE` can be composed of three different optional variables, namely `VAR1`, `VAR2` and `VAR3`. It is easy to fill the contents of this variable like accessing a field of a C/C++ structure or union. Nevertheless, when this variable needs to be read at a remote peer (it has been transported by the MIH protocol), this entity does not know the type previously selected at origin. The draft 13 of IEEE 802.21 standard did not provide a solution to this issue.

The work started with the definition of this type of variable as a simple *union*. Unfortunately, this problem of ambiguity did not appear until the work was quite advanced. Almost all the definitions of `CHOICE` data types are based in the selection between another data type and a `NULL` data. Checking if the election was one or other is easy: it is needed just to see if the inner data is null. Problems came with the composition of the MIH_Get_Information Request message, which handles some `CHOICE` data types with several options within.

With the given definition of this data type, the resolution of the election taken in the source peer is not trivial in the destination node. A reading of the definitive version of the IEEE 802.21 standard provides the solution: this data type also has a *selector*, a byte which indicates the data that has been selected regarding to the order it is has defined into the `CHOICE` structure.

Changing all the variables defined as `CHOICE` may cause in a revision of almost all the code. So, in order to solve the problem of ambiguity that this data type eventually causes, the decision made is to include this byte just only in the complex structures that do not cause trivial decisions about the chosen element. Almost all of them are included into MIH_Get_Information primitives. Further improvements of the code must consider introducing the stated *selector* in each variable declared as `CHOICE` in order to ease compatibility with other systems.

5.1.1.4. Improvements in communications with the OS kernel

The current development has multiple calls to *ioctls* in order to gather information from the kernel of the system and to set some elements. These *ioclts*, as explained in a previous chapter, are functions that are not well defined at all and some newer mechanisms are trying to replace them in mid and long term.

What happens is that in most cases these *ioctls* have been extracted from the source code of an application such as *wlanconfig* from the *wireless tools* or from other sources of open code. The process of reverse engineering is hard to carry out, and so just a group of the essential calls have been kept in the code. The rest of the required actions that need to communicate with the kernel have been performed by calling directly to the tool needed for that. For instance, the process of association and authentication with a Wi-Fi hotspot has been carried out using the *exec* system call.

In order to develop a more efficient coding, these *execs* should be replaced progressively by other mechanisms of communication with the kernel. Note that the current implementation has limited the use of *ioctls* to some specific functionalities that are a part of existing open source programs which perform more actions than the required for the designed applications.

5.1.2. IEEE 802.21 suitable elements to be added

5.1.2.1. MIH protocol over transport layer

The MIH protocol is commonly defined as a 2.5 layer protocol. This means that the IEEE 802.21 standard provides a cross-layer specification that takes elements from both L2 and L3. In fact, it is intended to provide link layer independency to upper layers. Nevertheless, the original document left a possibility to implement a part or most of the MIH protocol over a transport layer. Note that, for the example of a WLAN, a peer needs to be authenticated and associated prior to the exchange MIH messages. On one hand, this feature would provide great advantages in terms of simplicity of implementation, but on the other hand some capabilities such as many management features need a connectionless exchange of messages.

A part of the work designed to develop rules and standards for MIH over transport layers relies on Mipshop WG from the IETF. Its aim is defining a problem statement for transport over IP of the MIH protocol, and defining delivery of information for MIH services on L3 or above protocols. This allows placing the network information anywhere, not necessarily across the wireless link. It also enables MIH services without the necessity of the corresponding L2 support. Moreover, for the discovery of nodes providing MIH services, current L3 procedures are expected to be used. Existing or new protocols may be used for transport. Last, Mipshop is in charge of defining the L3 discovery component and to address security, a feature which is not within the scope of IEEE 802.21 Standard.

The IETF has published several RFC documents where the definition of the problems that can be found is explained. For instance, the document *MIH Problem Statement* [43] outlines a broad problem statement for the signaling of IEs across a network to support MIH services. A need for a generic transport solution with certain transport and security properties is sketched in this report. Another document, *Design Considerations for MIH Transport* [44], is centered in developing functionality to support MIH services, which are intended to aid IP handover mechanisms between heterogeneous wired and wireless access networks. Finally, the document *Transport of Media Independent Handover Messages Over IP* [45] describes a mechanism for using UDP/IP for the transport of MIH messages between network nodes. MIH messages carry specific L2 information and commands that can be used to optimize handover procedures across different access networks such as WLAN. These are some examples of the documents that aid the development of IEEE 802.21 solutions which particularly in this case are focused in the transport of MIH messages.

5.1.2.2. MIIS as an entity detached to the PoA implementation

In Chapter 3 it was stated that the current design simplifies the implementation by introducing the MIIS into each PoA stack. Although it has the advantage of being easier to implement, this solution does not correspond to a realistic scenario. In one hand, we have a data base difficult to maintain up to date because the contents should be modified by hand every time a change has occurred in every PoA of the surrounding networks. On the other hand, the current development does not take profit of the solutions suggested by the IEEE 802.21 Standard. A centralized MIIS solution must be found.

A MIIS should be an element placed somewhere in the network, not necessarily in the same peer as a PoA. A MIH_Get_Information Request message should be addressed to the PoA the MN is attached to if it has MIIS capabilities available. This feature does not necessarily implicate that the serving PoA has a MIIS into the same host, but can contact to a network peer allowing the service. To do so, a MIH protocol over transport layer (TCP or UDP) should be considered in order to establish a communication between the serving PoA and the queried MIIS. In this case and according to the current design, a determined PoA should be preconfigured with just the IP address of the remote MIIS and with the port the application will use.

In order to update the information available in the MNs once a MIIS has updated its own, the MIH protocol provides a set of primitives and messages to send the new information. These primitives are MIH_Push_Information *request*, *indication* and *confirm* and they provide information upgrades through a set of IEs. The structure of the primitives and messages is very similar to the one described for MIH_Get_Information set.

5.1.2.3. IEEE 802.21 Management Information Base (MIB)

Generally, a MIB is a type of database used to manage devices in a communications network. It is composed of a collection of objects which describe the attributes of the entities in a network. These objects are defined using a subset of Abstract Syntax Notation One ASN.1 called Structure of Management Information Version 2 (SMIv2). This database is hierarchical and entries are addressed through object identifiers.

Simple Network Management Protocol (SNMP) is an UDP-based network protocol for communication between management stations that use MIBs. It is used in management systems providing administrative features. SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. These variables can then be queried and set by managing applications.

Currently, the IEEE 802.21 standard supplies an example of its MIB, defining the configuration of the MIHF entity parameters. An example of one of its objects can be seen in the box below. In this case, the object is the maximum retransmission retries used by the transaction state machines of a peer. It can be seen that it is defined as an unsigned integer of 32 bits and its default value is set to 2 retransmissions. This value can be read and modified, because the access to this object is declared as read-write.

```
dot21PeerMaxRetransmissionCntr OBJECT-TYPE

SYNTAX Unsigned32 (1..255)

MAX-ACCESS read-write

STATUS current
```

```
    DESCRIPTION
    "The maximum number of retransmission retries for MIH messages used for a
    particular peer MIHF."
    DEFVAL { 2 }
    ::={ dot21PeerMihfEntry 12 }
```

The way it has been implemented in the current applications is much simpler. The MIH User of each entity has a segment of data that is referred to the own attributes of the local peer. This information can be recognized because the identifiers of each attribute start by the word “Own”. This information is set before compiling the programs and is assigned through the initiation routines described in chapter 3.

Although the implemented way is simpler because it just uses the tools provided by the programming language, it is not standard and cannot be reused in other systems. The option proposed by the IEEE 802.21 Standard requires setting a complete data base whose elements may not be useful for all the implementations and also should use the SNMP to set and gather its information, but relies on a well known methodology.

5.1.2.4. RDF representation for IEs and SPARQL MIIS query method

As an alternative to the binary representation of the IEs proposed and implemented in the current development, representing them is also feasible through RDF format, which is also accepted in the IEEE 802.21 Std. In particular, the format defined in this standard is the one based in XML. The RDF format is a family of specifications originally designed to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax formats.

The SPARQL query method is also considered under IEEE 802.21 terms as an alternative to the TLV query method for MIIS clients and servers. SPARQL is an RDF query language that allows users to write globally unambiguous queries.

Together the SPARQL and the RDF representation of IEs compose the so called RDF schema. This schema has two feasible versions: the basic schema and the extended schema. The first one should be pre-provisioned in both MIIS client and server. The second one is vendor specific and depends on the implementation. A basic schema is supplied in the annexes of the IEEE 802.21 standard in order to ease implementations.

These features should be considered to be implemented in future revisions of the applications. The target is enhancing possibilities of interoperation with other MIIS devices that allow this type of query method and IEs representation and to decide whether using a method or other is more accurate given a set of conditions.

5.1.2.5. Network controlled handovers

The implemented system was designed bearing in mind a low traffic charge scenario. It assumed that the resources in each point of the network would be enough to handle this traffic. For this reason, the handover policies were placed in the MN instead of the network.

For further developments, higher charge rates must be considered. As well as it happens in cellular telephony, the handover decisions are taken in the network, but aided by the MN. A network initiated handover policy should consider a development alike. The IEEE 802.21 standard has defined a set of messages to handle a network initiated handover, but does not consider aids coming from the MN.

The system should be modified placing the handover policy in the PoAs. It would lead to an easier implementation of the MN code, but at the expense of increasing the complexity of the PoA code. Moreover, a system to handle multiple instances of different simultaneous transactions should be developed in those PoAs.

In this context, for a network initiated handover implementation some ideas should be considered. In one hand, the charge of a determined PoA or network can determine if a new MN admission is possible. In the other hand, the quality of the received signal strength must be considered too in order not to lose the connection. The IEEE 802.21 Standard provides mechanisms for network initiated handovers, and the MIH protocol provides a set of messages to receive events generated in remote peers. By this way, a MN which has triggered a Link_Parameters_Report indication can report this event to the MIHF entity which it is attached using MIH_Link_Parameters_Report indication message. Now, the network can begin the required actions to perform a handover based on the network traffic charge and the requirements and resources needed for the current connection.

5.1.3. Discovery protocols and IETF proposals

A MIH scenario can be composed of a core wired packet switched network and several PoAs providing access to the MN in the wireless access network. Moreover, this network can have other elements such as MIISs, data bases for registry and many other elements related to mobility management at various stack levels. As explained all over this document, the MIH protocol requires exchanging messages between different PoAs. The way these peers can discover themselves and establish a communication is not defined in IEEE 802.21 Standard.

The way the current implementation has solved this issue is not very smart at all. A pre-configured look-up table defines which IP corresponds to the MIHF identifier of the target PoA. This information in each PoA is very static and does not allow a flexible upgrade when new conditions are met such as the introduction of a new PoA in the network or a change in the IP address of one of the associated MIHF identifiers. Again, the IETF proposes different ways to achieve an efficient and centered solution.

The document *IEEE 802.21 Mobility Services Framework Design* [46] provides a solution to discover Mobility Services (MoS) in a MIH compatible network over a transport protocol but between a MN and a PoA. The idea could be extrapolated to MIHF discovery between different PoAs, and is similar to the process a classical network peer would follow with the resolution of the Domain Name Servers (DNS). The stated document proposes querying a DHCP server before the MN is attached to a PoA. The DHCP server responds supplying the IP address of the resolved MIHF. It is curious to see that the proposed transport in this IETF document is done over TCP, and it is very similar to the schema that has been used in the current implementation for message exchange over the wired network.

An analog discovery process to the described in the stated document can be done when a PoA needs to discover another PoA IP address. Moreover, this process can be improved by adding a cache memory within the PoA that stores the relationship between a given MIHF identifier and its corresponding IP address.

Other documents such as *Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Options for IEEE 802.21 Mobility Services Discovery* [47] (also from IETF Mipshop) provide further details on how the DHCP servers should act for MoS discovery and how the whole process should work. Finally, the document *Locating IEEE 802.21 Mobility Services Using DNS* [48] provides a solution to the stated problem by taking profit of the DNS capabilities of a DNS server.

5.1.4. The DHCP latency problem and suitable solutions

As seen in the previous chapter, the time spent to perform a DHCP association is not admissible. Association times around 20 second can lead to a connection lost for a person walking at less than 2m/s. In order to avoid these situations by minimizing the latency of this procedure, other different network protocol must be tested and introduced instead of DHCP. For instance, the fist version of IEEE 802.21 Standard proposed using PMIPv6 in the annexed example message flows, or even draft versions of the same standard proposed using FMIPv6. In both cases, the IP protocol version is IPv6, which provides global IP addresses suppressing many address translation procedures.

5.1.4.1. IP mobility evolution: IPv6, MIPv6 and FMIPv6

IPv6 [49]

The Internet Protocol version 6 (IPv6) is an evolved version of the IPv4 protocol and is designed to replace it in long term. The current IPv4 protocol is about to exhaust its disposable addresses and IPv6 can solve this issue providing a longer range of addresses. IPv4 uses addresses of 32 bit, whereas IPv6 has 128 bit length addresses, which represents an amount of 670 million of billions of addresses for each mm2 in the Earth surface.

This extended range of addresses provides capability for the devices to have a fixed IP address suppressing the need of a dynamic assignment or NAT processes, which increases the latency of operation in the network. The routing procedure works with a system of sub-network prefixes, which define the size of a network. The sub-network is defined by the first 64 bytes and the other 64 bytes can be assigned to each node in a network. The part that corresponds to the node is commonly generated from the MAC address of the associated interface of the peer.

More variations that improve the efficiency of the protocol in terms of latency respect to IPv4 are the following:

* The header of the IPv6 packet has been simplified, removing useless and deprecated fields.

* The IPv6 routers do not fragment packets. Fragmentation and reassembly procedures are only done in source and destination peers.

* The header does not use a checksum field. This checking is left for link and transport layers.

At present, when a user roams through access networks each of them supply a different IP address to the MN. By this way, the user is not able to keep an open session of an application during the change of network. The current model of network does not allow scenarios such as mobile VoIP communications. The MIP protocol is intended to solve problems derived from the absence of mobility capabilities in IPv6.

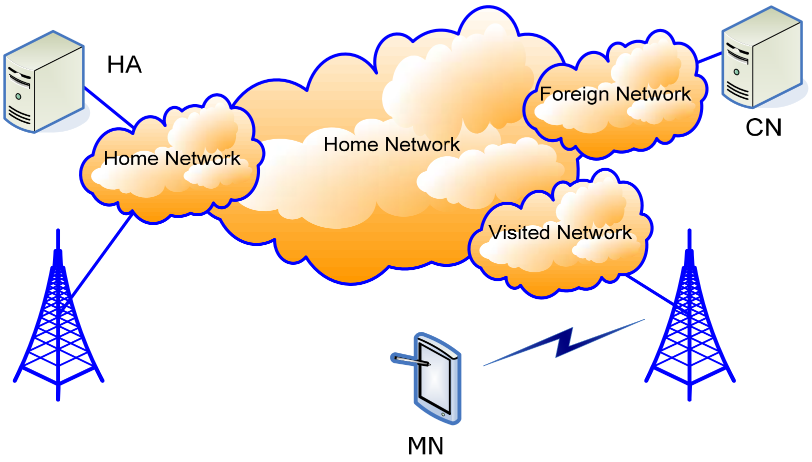


Figure 5.1: MIPv6 network topology

MIPv6 [50]

The IETF developed a new model of connectivity to the Internet which solves the mobility problem stated before. This technology is known as Mobile IP (MIP) and provides a feasible method to support mobility in networks that use the IPv6 protocol. The result is Mobile IPv6 (MIPv6). Specifications for networks based in IPv4 are proposed but they will not probably be developed due to the complexity added basically in authentication of network devices and addresses translation.

A MIPv6 compatible network is composed basically by three elements: the Mobile Node (MN), the Home Agent (HA) and the Corresponding Node (CN). The MN is the user device which is provided of two IPv6 addresses: the Home of Address (HoA), which is a fixed address used in the home network, and the Care of Address (CoA), which is a variable IPv6 address used in foreign networks. The HA is the network element which registers the real location (the CoA) of the MN and binds it to the HoA. The CN is the network element which the MN communicates to through the HoA. In case the MN is located at a visited network, the operation is the following:

1. The MN sends its CoA to the HA once it has been assigned in the foreign network through a Binding Update (BU) message. The HA updates its mapping table by binding the new CoA to the HoA.

2. When the CN wants to communicate with the MN, it uses its HoA, which is known and fixed.

3. The HA encapsulates in MIPv6 packets those received from the CN and sends them to the new CoA of the MN.

4. When the MN needs to respond to the CN, it sends the corresponding packets encapsulated in MIPv6 format to the HA.

5. Once received the response packets, the HA extracts the original packet and forwards it to the CN.

As it can be observed, the path the packets follow may not be optimal because they need to go through the HA. In order to solve this issue, a protocol of route optimization is defined. It works as follows: if the CN is compatible with the MIPv6 protocol, the MN can send a BU message to advertise it of its current CoA. By this way, the CN can send the packets directly to the MN without passing through the HA. This feature suppresses the triangle routing issue and so unnecessary delays disappear.

Using stateless address self-configuration mechanisms and neighbor discovery, MIPv6 does not require DHCP procedures or foreign agents for external links to configure the CoA of the MN. This protocol solves a part of the requirements for L3 mobile devices, but does not provide an optimization of the handover timings. A certain time interval must go by before the totality of the message exchange has taken place from an AP to another. The establishment of a connection is the slowest part due to the stated neighbor discovery, the protocol for acquisition of the new CoA (which involves a duplicated direction detection of 2 seconds), the transmission of the BU to the local agent and the security procedures of MIPv6 before using the new CoA Note also that the BU message requires being assured in order to trust relationship between the MN and HA. This is done through Internet Protocol Security (IPSec) authentication protocol, which introduces a delay in the operation.

FMIPv6 [51]

Mobile IPv6 enables a MN to maintain its connectivity to the Internet when moving from one Access Router (AR) to another, a process referred to as handover. While the handover takes place, there is a period during which the MN is unable to send or receive packets because of link switching delay and IP protocol operations. This *handover latency* resulting from standard MIPv6 procedures, namely movement detection, new CoA (nCoA) configuration, and BU, is often unacceptable to real-time traffic such as VoIP. Reducing the handover latency could be beneficial to non-real-time, throughput-sensitive applications as well.

FMIPv6 defines two mechanisms for handover: *anticipated (or predictive) handover*, which is only based in L3 information, and *tunnel-based (or reactive) handover*, which relies on L2 triggers. The most feasible schema is the first one, which provides make-before-break connections. The operation is the following:

1. A MN needing a handover listens a Router Advertisement from a new AR (nAR) nearby. Then it sends a Router Solicitation Proxy (RtSolPr) to the old AR (oAR) with the identifier of the listened nAR.

2. The oAR composes the nCoA and sends it to the MN together with the IP and link address of the nAR. It simultaneously also sends a Handover Initiated (HI) message to the nAR indicating the oCoA and nCoA

3. The nAR validates the nCoA and stores it at the Neighbor Cache. Then it sends a Handover Acknowledge (HACK) to the oAR indicating that the nCoA is valid. It also sends a PrRtAdv to the MN indicating that a L3 handover is pending.

4. Once received the HACK, the oAR prepares to forward packets establishing a tunnel between oAR and nAR. When the MN receives the PrRtAdv, it sends a Fast BU (FBU) to the oAR.

5. The oAR responds sending a Fast Binding Ackowledge (FBAck) to the nCoA of the MN and to the nAR through the tunnel. The FBAck contains the tunnel lifetime. Now the oAR proceeds forwarding the packets with destination the MN to the nAR through the tunnel.

6. Once received the FBAck, the MN completes the L2 and L3 handover and sends a Fast Network Advertisement (FNA) to the nAR.

7. Finally, the nAR receives the FNA and delivers the stored packets to the MN.

It must be highlighted that apparently MIPv6 as well as FMIPv6 specifications consider just one interface to handle handovers. With a unique interface, it is not possible to maintain a continuous flow of data between the MN and the CN because a change of CoA is needed. FMIPv6 considers a mechanism to store packets and then to deliver them once the L2 and L3 layers in the MN have been configured. Although there are no losses, a service disruption can be appreciated if the required bandwidth for traffic is high. In theory, in a WLAN scenario the time for handover due to FMIPv6 is determined by the WLAN handover latency. The situation for MIPv6 is even worse. The handover latency is estimated to be around of 2 seconds of service disruption, and there exists packet losses because there is not defined a method to store the received packets in the oAR during the handover.

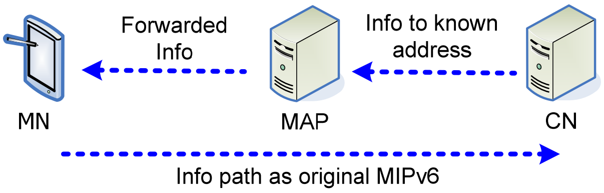


Figure 5.2: HMIPv6 data path

So, it seems that an optimal solution cannot be reached with these two versions of MIP with just an interface. Fortunately, the current implementation considers 2 NICs, so an interface can be configured at the same time the other is still receiving a flow of data. An even simpler mechanism than FMIPv6 could be proposed because forwarded data should not be necessary but just a change of data path. Nevertheless, FMIPv6 can be a feasible solution for mono-modal devices maybe improving its performance with other application layer protocols.

5.1.4.2. Advanced alternatives based in MIP: HMIPv6 and PMIPv6

Hierarchal MIPv6 (HMIPv6) [52]

It has been seen that in MIPv6 the HA intercepts the packets addressed to the MN HoA and tunnels them to the MH CoA using IPv6 encapsulation. Then the MH sends also a BU to its CN, which after that can send packets directly to the MH. Although this protocol optimizes the routing of packets to MNs, it is not scalable.

As the number of MNs increases in a given network, the number of BUs increases proportionally and adds a significant extra traffic overload. The MIPv6 approach is not scalable since the generated signaling load becomes overwhelming as the number of MNs increases in a given network. A hierarchical scheme that differentiates local mobility from global mobility would be more adequate in a wider network scenario.

HMIPv6 is intended to work in scenarios where MNs move fast. That is translated in frequent changes of AP. As seen in MIPv6, to perform a handover the configuration of each involved peer in this operation needs to be updated. The protocol HMIPv6 pretends hiding changes from the rest of the world for a change of AP, that are the nodes the MN is communicating with. This can be achieved by introducing a new network element called Mobile Anchor Point (MAP).

The MAP is intended to be located geographically near the MN. It tracks the current CoA during the time the MN remains under the domain of a given MAP. The MAP listens to the address of the MN that is commonly known and forwards the data to the current CoA.

As shown in figure 5.2, when the data follows the opposite direction in the path, it does not pass through the MAP. This has been left as in the original MIPv6 specification. The current situation does not provide benefits if the MN is fixed without roaming. But if the MN changes constantly of AP, benefits of faster handovers appear. The only node that needs to be updated is the MAP. The rest use the same destination IP address. Note that more levels of MAP hierarchy can be added in order to reduce the handover domain, depending on the size of the network. The world just needs to use the IP address and port of the highest level of the hierarchy.

A hierarchical approach has two main advantages: first, it improves handover performance since handovers occur in a local area, increasing by this way the handover speed and minimizing packet losses; and second, it significantly reduces the mobility management signaling load on the network since the signaling messages corresponding to local handovers do not cross the whole network, they are exchanged locally.

Proxy MIPv6 (PMIPv6) [53]

PMIPv6 is a protocol that allows local mobility for a MN without requiring its participation in the related signaling. The network is in charge of managing the MN mobility on its behalf. MIPv6 signaling is extended and concepts such as HA from the original MIPv6 specification are reused.

PMIPv6 defines a domain which introduces two new network functional entities: Local Mobility Agent (LAM) and Mobile Access Gateway (MAG). The LMA is a local HA for the MN in a PMIPv6 domain. It also acts as an anchor point for the MN home network and handles the bindings of the MN maintaining the reachability to the MN while it is under a FMIPv6 domain. The MAG is an entity commonly located in the AR that is in charge of handling the mobility signaling on behalf the MN. The MAG tracks the movement of the MN, authenticates it and initiates the mobility signaling on behalf the MN. These two entities establish a tunnel allowing the MN to use a prefix address of its home domain. As a result, these entities allow emulating the MN home network in a visited network.

Thus, a MN within a PMIPv6 domain has a unique network prefix assigned by the network, and this prefix is used for each placement within the same PMIPv6 domain. The MN does not need to reconfigure the CoA when it moves into the same domain, but a global mobility management protocol is required if a MN has to perform a handover across different domains.

5.1.5. RTT critical increase in the target network

The experiments have shown that the RTT rises up to an inacceptable level when the MN gets engaged to the destination network, foremost for real-time communications such as audio conference. The nature of this problem is currently unknown, but must be found in order to perform a proper operation. The *ping* program does not modify its behavior, so the problem could be associated to some kind of buffering at source or destination hosts, not even related to the provided implementation.

5.1.6. Higher layer protocols: SIP, HIP and SCTP

In order to develop applications that take profit of the mobility features provided by MIH and MIP layers, some application layer protocols are borne in mind. Applications that could demonstrate the accuracy of the implemented handover system are those based in audio or video streaming, and audio or video conferences.

Some of those applications use higher level protocols such as Session Initiation Protocol (SIP), Stream Control Transmission Protocol (SCTP) and Host Identity Protocol (HIP).

SIP [54]

SIP is a protocol proposed by the IETF for VoIP and other multimedia sessions such as instant messaging, video, online videogames and other services. It is intended to create, modify and terminate sessions between two or more participants. SIP works over transport layer protocols using the port 5060 for both TCP and UDP.

The SIP invitations are used to create sessions and carry the description of the session that allows the participants to reach an agreement about the compatible media system used during a session. SIP uses Proxy servers that aid in routing requests to the current user location, to authorize and authenticate users for services, implement routing policies, and supply services to users. SIP also provides a registry function that allows a user to indicate its current location to be used by a Proxy.

It must be highlighted that SIP is a flexible protocol that allows adding new characteristics maintaining the interoperability with the existing features. Note that SIP provides all the features of the internet telephony.

SCTP [55]

SCTP is a transport layer communication protocol which can be seen as an alternative to TCP and UDP. It provides reliability, flow and sequence control as well as TCP. SCTP optionally allows sending out of order messages, being a message oriented protocol. SCTP provides many advantages over TCP and UDP:

* Multi-homing: one or more ends of the connection can have more than an IP address. This allows transparent reactions against network failures.

* Delivery of chunks within different streams.

* Select and monitor paths, selecting a primary path and verifying constantly the connectivity of each of the alternative paths.

* Validation and consent mechanisms improved in order to protect against flooding attacks by notifying duplicate or lost data chunks.

SCTP was originally intended to transport telephonic Signaling System 7 (SS7) over IP. The target was providing some of SS7 reliability features in IP. Due to its versatility, its use has been proposed in other areas such as the transport of SIP messages. Furthermore, SCTP could be implemented in network systems and applications that deliver voice or data and support quality real-time services such as streaming of video and multimedia.

HIP [56]

At present, IP addresses are used as topological locators (for DNS) and as NIC identifiers. This dual role limits the flexibility of the Internet architecture and makes difficult the IP address renumbering due to mobility. Transport protocols are bound to IP addresses and disconnect when this address changes (the socket changes). In HIP, sockets are bound to HIs rather than IP addresses. IP addresses are used to route packets just as today.

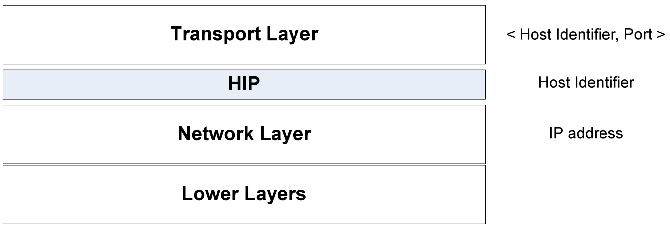


Figure 5.3: HIP layer into IP stack

HIP defines a new global Internet name space. It decouples the name and locator roles currently filled by IP addresses. With HIP, the transport layer operates on Host Identities (HI) instead of using IP addresses as endpoint names. At the same time, the network layer uses IP addresses as pure locators. HIP provides end-to-end opportunistic security, resistance to CPU and memory exhausting denial-of-service (DoS) attacks, mobility and multi-address multi-homing.

5.1.7. Handover in broadcast and multicast technologies [57]

Finally, this section cannot be concluded without referring to broadcast and multicast technologies deserving a handover system such as video or audio broadcasting for mobile devices. At present, devices capable of Digital Video Broadcasting (DVB) for Handhelds (DVB-H), WLAN and UMTS are already available, providing the basis of a broadcast and multicast MIH aided handover.

For instance, in a mobile TV service a program would be delivered to a number of end users. When the popular TV program is no longer available on the Downlink Only (DO) network, there is a need to perform a group handover of a number of MNs from the DO network to the bidirectional networks so that the end users keep watching the TV program.

Some problems with digital broadcasting technologies exist at present. For example, DVB does not provide full coverage in some areas for a mobile TV service. Many of these placements are commonly indoor environments, which could be covered through WLAN solutions. In order to solve issues like this, the IEEE 802.21b Task Group emerges to focus on handovers for broadcast technologies.

5.1.7.1. IEEE 802.21b

DO technologies such as DVB, Terrestrial Digital Multimedia Broadcasting (T-DMB) and Media Forward Link Only (MediaFLO) are becoming more widespread in use. There is a need to support optimized handovers between these technologies and other technologies already supported by IEEE 802.21. Currently, there is no standard which specifies such handovers.

The IEEE 802.21b Task Group handles optimization of handovers between DO networks and bidirectional networks intended for mobile TV service and other broadcast technologies. It appears as an amendment that defines mechanisms to enable the optimization of handovers from DVB / DMB / MediaFLO to WiMAX / WLAN / 3GPP technologies and vice versa. Moreover, a bidirectional channel needs to be intended in order to support interactive services.

Figure 5.4: DMB, MediaFLO and DVB-H logos

IEEE 802.21 defines mechanisms that enable handovers across various technologies such as IEEE 802 and 3GPP/3GPP2. These same mechanisms can be used for handovers with DO technologies. However, this requires enhancements to the IEEE 802.21 specification. These enhancements include extensions of IEEE 802.21 primitives and protocol to support handovers with DO technologies, and the definition of media dependent SAPs and the corresponding mapping of MIH link layer primitives for DO technologies. The scope of the proposals is the following:

* Provide mechanisms to help reducing the latency during handovers between heterogeneous access networks that support IEEE 802.21 and broadcast or DO technologies

* Provide mechanisms to supply information about broadcast services to heterogeneous technologies with IEEE 802.21 MIH protocol exchanges

* Provide the definition of media Independent and media dependent SAPs

5.2. Conclusions

The telecommunications are constantly evolving. This is a fact that can be easily appreciated just by seeing the number of new portable and handheld devices that every day appear in the market, as well as new impressive applications depicted to run “anywhere”. The capabilities of these terminals have been improved providing high transmission data rates at affordable prices.

Because these devices are mobile, it is mandatory to provide them a continuity of service without degradation.. This continuity cannot be obtained without an efficient change of serving cell, what is known as handover. Some technologies such as the cellular telephony define its own methods to perform these handovers. Nevertheless, handovers between different technologies (vertical handovers) such as IEEE 802.11 and IEEE 802.16 are not exploited yet.

The current memory has gathered the most relevant incidences of the development of a system capable to perform handovers between WLAN networks. The main guidelines that have been followed are those stated in the IEEE 802.21 Standard, which is the document that provides the protocol and the mechanisms designed to perform handovers between different access technologies. Although a handover between from an IEEE 802.11 hotspot to another is considered a intra-technology handover (horizontal handover), the IEEE 802.21 Standard is depicted to provide mechanisms also in this scenario.

The work in this project has passed through different stages. At the beginning, the study and comprehension of the standard required a great effort and took lots of hours. This stage required a great effort in research trying to find the proper tools and devices to develop the implementation. It was fairly clear that the OS would be a Linux distribution because this system provides facilities to develop code and also allows sharing knowledge because it is open code. Nevertheless, obtaining proper NICs to build the PoAs was a disappointing issue. At present, this initial research period could have been significantly reduced because the latest OSs are prepared to support AP functionalities with an increasing number of wireless NICs, which have an excellent documentation in *Linux Wireless* Internet web page.

The implementation of the applications for the MN and the PoA was not an easy task. To follow a specification of a standard document can be seen as an advantage in terms of design, but it is absolutely false. The standard provides a set of organized ideas that aids providing a general design scenario, but all the concepts that appear within are abstract and need to be translated in terms of code. And, what is worse, some parts seem to be over-explained whereas others are not covered. So, difficulties arise interpreting this text and also providing solutions to aspects that are not covered but are necessary to build a complete executable application.

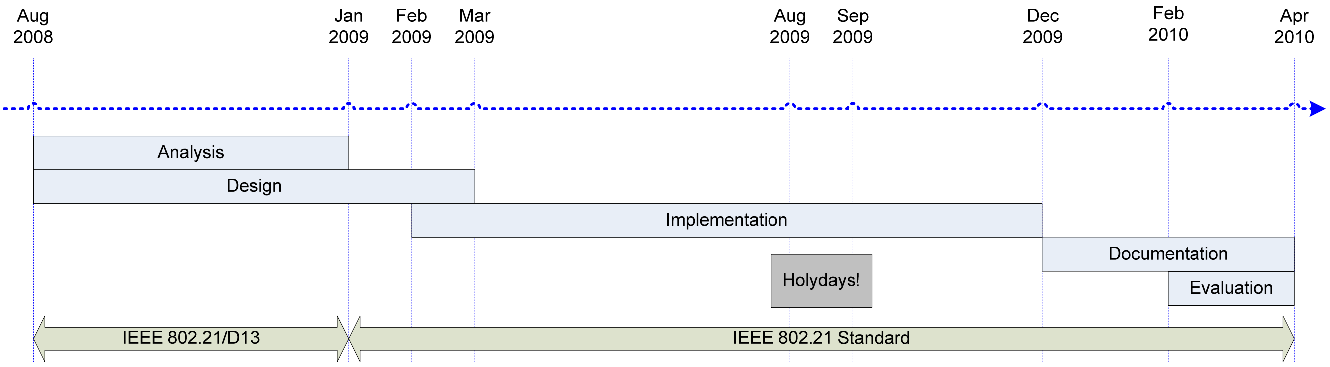


Figure 5.5: Project calendar

The standard does not define the policies to select an AP or another. This feature is not described in any text such as an RFC document, and its definition is left to the operator policies. The current work has been developed with a MIH User supplying an implementation example of these policies. The methods to discover MIHF entities or how to solve their addresses into an IP network are not defined either in the pages of this standard. Instead, this work is left to the IETF, which has proposed many drafts in order to solve different issues related to the management of the MIH protocol in extensive networks.

No message exchange flow is defined in terms of MIH management policies, just the messages involved in a handover. Some examples of different mobility scenarios are provided in the annexes as informative notes. The definition of a whole functional process is required in order to implement an application. To complete the operation of the applications, the implicated exchange of messages for initial network bootstrap and for MIH management (basically MIH registration) has also been defined using the primitives and messages supplied by the standard.

In order to complete the implementation, an upper layer association was introduced aiming to supply IP connectivity. The IEEE 802.21 standard bases its proposals in L3 mobility protocols which theoretically provide great advantages in terms of handover performance. Those protocols are basically evolutions of MIPv6. Nevertheless, some disadvantages are encountered: first, IPv6 networks are not extended yet, and second, MIPv6 derived protocol implementations are not much developed at present and could be difficult to introduce in current applications. Due to the fact that a higher layer development was out of the scope of this work, the decision was to introduce a classical DHCP association after a WLAN authentication.

Once the development and bug fix stage finished, some tests were performed In order to validate the system operation. The tests were based in measures of times of handover and packet losses. The results depicted an acceptable latency of the handover timings without bearing in mind the time spent in DHCP association. Moreover, they showed that this kind of association is not proper for mobility devices because averaged latencies of almost 20s of peak in association procedures can lead to a total change in environmental conditions causing a system fail in terms of handover execution. By this way it justifies a study of feasible L3 mobility protocols candidate to replace this DHCP association. Best candidates are protocols derived from MIPv6 because they provide fast handovers minimizing the signaling on the network and reducing to 0 the packet losses.

To conclude, it must be highlighted that the designed applications work properly. The bootstrap and handover decisions are properly made in the side of the MN, which is the entity in charge of making decisions. The handover latency due to the MIH protocol is very acceptable, but the higher layer handover timings must be improved introducing a smarter protocol of mobility.

A lot of work has to be done still in order to obtain a complete application running under a media player and providing a continuous stream of video while roaming from an AP to another, but thanks to the work developed in this project this moment is now a little bit nearer.

References

[1] “IEEE Standard for metropolitan local area networks. Part 21: Media Independent Handover Services”, IEEE Computer Society, 2009.

[2] “IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks. Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, IEEE Computer Society, 2007.

[3] “Fundamentals of Wireless LANs”, Cisco Systems, july 2006.

[4] R. Akl, D. Tummala and X. Li, “Indoor Propagation Modeling At 2.4 GHz for IEEE 802.11 networks”, IEEE Computer Society. July 2006.

[5] P. Brennen, “A Technical Tutorial on theIEEE802.11 Protocol”, BreezeCOM, 1997.

[6] MªA. Hernández, “Redes Móviles de Tercera Generación”, 2005, http://catedratelefonica.unizar.es/

[7] Pthread, “POSIX thread (pthread) libraries”,

http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

[8] J. Martin Berg, “WiFi Control Plane Overview”, February 2009, http://wireless.kernel.org/

[9] J. Martin Berg, “mac80211 Overview”, February 2009, http://wireless.kernel.org/

[10] Linux Kernel improvements, “Linux Kernel Newbies”, http://kernelnewbies.org/

[11] B.Kerninghan, D.N. Ritchie, “The C Programming Language”, Prentice Hall, 1972

[12] B. Eckel, “Thinking in C++”, Prentice Hall Inc 2nd Edition, 1999

[13] GCC, “GCC, The GNU Compiler Collection”, http://gcc.gnu.org/

[14] g++, “Cprgramming.com, Your Resource for C and C++”,

http://www.cprogramming.com/g++.html

[15] Wikipedia, The Free Encyclopedia, ”Garbage Collection (computer science)”,

http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)

[16] Boehm GC, “A Garbage Collector for C and C++”,

http://www.hpl.hp.com/personal/Hans_Boehm/gc/

[17] Anjuta DevStudio, “Anjuta DesvStudio: GNOME Integrated Development Environment”, http://projects.gnome.org/anjuta/index.shtml

[18] Geany, “Geany Home Page”, http://www.geany.org/

[19] Wikipedia, The Free Encyclopedia, “Official Wiki of Code::Blocks”,

http://wiki.codeblocks.org/

[20] Code::Blocks, “Code::Blocks home page”, http://www.codeblocks.org/

[21] Eclipse, “Eclipse home page”, http://www.eclipse.org/

[22] NetBeans, “NetBeans home page”, http://www.netbeans.org/

[23] Wikipedia, The Free Encyclopedia, “pcap”, http://en.wikipedia.org/wiki/Pcap

[24] TCPdump and Libpcap, “tcpdump/libpcap”, http://www.tcpdump.org/

[25] A. López, “Aprendiendo a programar con Libpcap”, February 2005

[26] Wireshark, “Wireshark home page”, http://www.wireshark.org/

[27] Wikipedia, The Free Encyclopedia, “Endianness”,

http://en.wikipedia.org/wiki/Endianness

[28] ifconfig, “Ifconfig Linux man page”, http://linux.die.net/man/8/ifconfig

[29] Wireless Tools, “Wireless Tools for UNIX”,

http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html

[30] DHCP Client dhclient, “dhclient man page”,

http://linuxcommand.org/man_pages/dhclient8.html

[31] wlanconfig, “wlanconfig Linux man page”,

http://linux.die.net/man/8/wlanconfig

[32] Wikipedia, The Free Encyclopedia, “Inline function”,

http://en.wikipedia.org/wiki/Inline_function

[33] Wikipedia, The Free Encyclopedia, “Daemon (computer software)”,

http://en.wikipedia.org/wiki/Daemon_(computer_software)

[34] Packetspammer, “Penumbra, Wi-Fi Network”, http://penumbra.warmcat.com

[35] Wikipedia, The Free Encyclopedia, “Received Signal Strength Indication”,

http://en.wikipedia.org/wiki/Received_signal_strength_indication

[36] “The MADWiFi project”, http://madwifi-project.org/

[37] T. Melia, D. Corujo, A. de la Oliva, A. Vidal R. Aguiar and I. Soto, “Impact of heterogeneous network controlled handovers on multi-mode mobile device design”, IEEE Society, 2007

[38] V. Paxson and M. Allman, “RFC 2988 - Computing TCP's Retransmission Timer”, NASA GRC/BBN, November 2000

[39] Internet draft, “Y.1541-QOSM -- Y.1541 QoS Model for Networks Using Y.1541 QoS Classes”, AT&T Labs, Alcatel-Lucent, January 6, 2010

[40] Ping, “Linux man page”, http://linux.die.net/man/8/ping

[41] W.Lim, D. Kim, Y. Suh and J. Won, “Implementation and performance study of IEEE 802.21 in integrated IEEE 802.11/802.16e networks”, Computer Communications, Elsevier, October 2008

[42] F. Cacace and L. Vollero, “Managing Mobility and Adaptation in Upcoming 802.21-Enabled Devices” IEEE Society

[43] E. Hepworth, G. Daley, S. Faccin and G. Vivek, “MIH Problem Statement (draft-hepworth-mipshop-mih-problem-statement)”, MIPSHOP Internet-Draft, March 2006

[44] E. Hepworth, “Design Considerations for MIH Transport, draft-hepworth-mipshop-mih-design-considerations-00”, MIPSHOP Internet-Draft, June 2006

[45] A. Rahman, U. Olvera-Hernandez and M. Watfa, “Transport of Media Independent Handover Messages Over IP I-D: draft-rahman-mipshop-mih-transport-00.txt”, MIPSHOP Internet-Draft, June 2006

[46] T. Melia, G. Bajko, S. Das, N. Golmie and JC. Zúñiga, “RFC5677: IEEE 802.21 Mobility Services Framework Design (MSFD)”, Network Working Group, December 2009

[47] G. Bajko and S. Das, “RFC5678: Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Options for IEEE 802.21 Mobility Services (MoS) Discovery”, Network Working Group, December 2009

[48] G. Bajko, “RFC5679: Locating IEEE 802.21 Mobility Services Using DNS”, Network Working Group, December 2009

[49] Wikipedia, The Free Encyclopedia,”IPv6”, http://en.wikipedia.org/wiki/IPv6

[50] M.Díaz and C. Olvera, “Despegando con movilidad IPv6 (MIPv6)”, March 2010.

[51] M. Torrent, X. Pérez and S.Sallent, “A Performance Study of Fast Handovers for Mobile IPv6”, IEEE Society, October 2003

[52] S. Tsiroyannis, “Hierarchical Mobile IP”, April 2002.

[53] Y. Kim, “Overview of Mobile IPv4, IPv6, MIPv6, FMIPv6, HMIPv6, PMIPv6, and 3GPP IMS”, National Institute of Standards and Technology, March 2008

[54] SIP, “VoIP en Español”, http://voip.megawan.com.ar/doku.php/sip

[55] SCTP, “International Engineering Consortium”,

http://www.iec.org/online/tutorials/sctp/index.asp

[56] HIP, “InfraHIP, Infrastructure for HIP”, http://infrahip.hiit.fi/index.php?index=how

[57] B. Simsek, J. Johann, JC. Zúñiga, F.Khatibi, J. Lee and B. Bae, “IEEE 802.21 and Broadcast Handovers”, IEEE Society, March 2008.