Universitatea “Politehnica” din București
Facultatea de Automatică și Calculatoare, Catedra de Calculatoare

LUCRARE DE LICENŢĂ

Comutare de nivel aplicaţie
folosind Session Initiation Protocol

BACHELOR THESIS

Application Layer Handover
using Session Initiation Protocol

Scientific Advisers (Conducători științifici):
As. Drd. Ing. Elena Apostol
Prof. Dr. Ing. Valentin Cristea

Author (Autor):
Mititelu C. Ştefan-Cristian

-2014-

Abstract

During the past few years, mobile, wireless communications greatly surpassed wired communications due to the people's growing need for mobility. Nowadays, almost every person posses at least one mobile device. More and more functionalities are added to the mobile devices in order to create the perfect gadget which will provide more flexibility to the end-user.

With the developing wireless communications, new problems arose and continuous labor is being developed to find better solutions. In this thesis, wireless mobility issues are taken into consideration, Voice over IP concept, SIP signaling protocol and SDP protocol are presented. Furthermore, this thesis considers the handover issues that occur while moving between different types of wireless networks. Various handover types will be explained, analyzed and compared, focusing on application layer handover using SIP. Practical handover scenarios will be pictured, discussed and gradually improved to obtain seamless handover. The open source SFLphone software was chosen as a base platform for implementing soft, horizontal, seamless handover using SIP as signaling protocol. The phone's architecture and functionality will be illustrated and explained, followed by the implementation description of the proposed Mobility Manager solution.

Contents

List of Abbreviations

UPB	University Politehnica of Bucharest
CS	Computer Science
IEEE	Institute of Electrical and Electronics Engineers
ISDN	Integrated Services Digital Network
IAX	Inter-Asterisk eXchange
VoIP	Voice over IP
SIP	Session Initiation Protocol
SDP	Session Description Protocol
RTP	Real Time Protocol
PSTN	Public Switched Telephone Network
QoS	Quality of Service
GPS	Global Positioning System
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
SCTP	Stream Control Transmission Protocol
TLS	Transport Layer Security
URI	Universal Resource Identifier
HT	Handover Time
HTL3	Handover Time at Layer 3
HTL6	Handover Time at Layer 6
LQI	Link Quality Indicator
AP	Access Point
NAT	Network Address Translation
IPC	Inter Process Communication

List of Figures

Chapter 1

Introduction

Context

From radio waves to infrared and microwaves, personal, mobile phone devices to GPS satellites, we are revolutionizing communications. Wireless communications are becoming more and more common than ever before due to the need of flexibility in nowadays society. With wireless technology developing at an exponential rate, a lot of new problems arise, problems that never happened before in a static, wired communication. In a wireless environment, bandwidth fluctuations, packet loss and packet collision happen at a higher rate compared to wired environment because of highly unpredictable shared medium. Also the amount of memory buffers in a mobile device is limited by the current technology. Moreover, the devices must know to passively or actively scan and connect to existing networks.

Motivation

The growing necessity of individuals for mobility lead to a boom in the mobile device's industry, and, additional features needed to be developed to satisfy their needs. As Figure 1.1 illustrates, one might want to be reached at both laptop and mobile phone at different time periods or might want to walk outdoors while maintaining a video call, or still communicating with an endpoint while business traveling. All of the above are interesting mobility features that are not taken for granted. Continuous work is being made in the mobile domain so it can offer more and more features for the society.

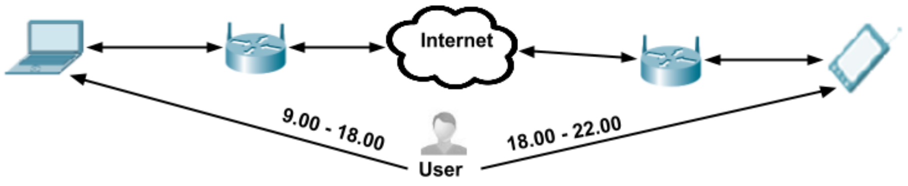


Figure 1.1: Mobile device usage

One of the problems that arises in the wireless communications is the signal attenuation as the device moves away from the emitter. According to Figure 1.2, the signal strength decreases with the square distance from the emitting source, therefore mobile devices must perfect the signal detection and reception. Although infrared or bluetooth technology is at hand, radio technology is more often used because data could be transmitted over large distances.

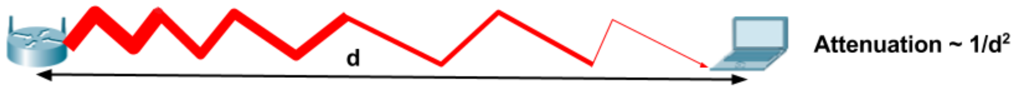


Figure 1.2: Attenuation of radio signal

Furthermore, collision is one of the important issues that must be solved in wireless networks. For example the hidden terminal problem, pictured in Figure 1.3, will result in signal collision when terminal A sends at the same time with terminal B.

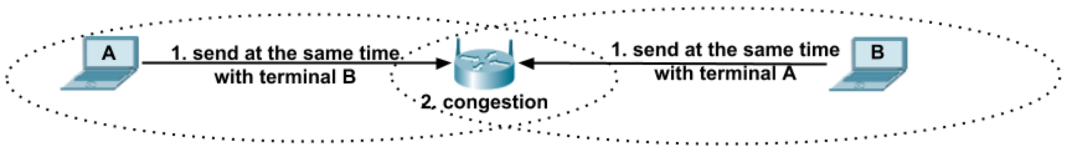

A
1. send at the same time with terminal B
2. congestion
1. send at the same time with terminal A
B

Figure 1.3: The hidden terminal problem in wireless networks

Moreover, the wireless environment is very unsecured because anyone with a proper antenna could capture exchanged messages, as shown in the Figure 1.4. Hence, additional algorithms must be used to encrypt data sent over wireless links. The current standard is Advanced Encryption Standard (AES) which is a trade-off between the time needed for encryption plus decryption and the encryption strength.

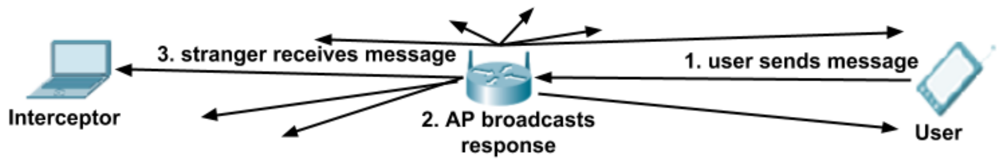


Figure 1.4: Wireless signal interception

At last, another important issue that arises in the wireless environment is the capability of a device to maintain services while roaming from current network to a different network, regardless of the data-link layer. For example, let's consider a device with both a Wifi and a 3G interface, scenario pictured in Figure 1.5. Whilst no Wifi network is around, the device handles services via the 3G interface, at a rate provided by the 3G protocol. When a better signal is found (i.e. Wifi), the device will switch on the Wifi interface, process called handover or handoff. There are different handover mechanisms, varying from data-link layer[3] to application layer[12]. Efforts have been made for the handover to happen very fast such that the end user not to experience unpleasant interruptions.

Even if some of the above problems are solved by using different methods, research is being developed to find better solutions, thus improving the wireless environment reliability and making it a sustainable mean of communication.

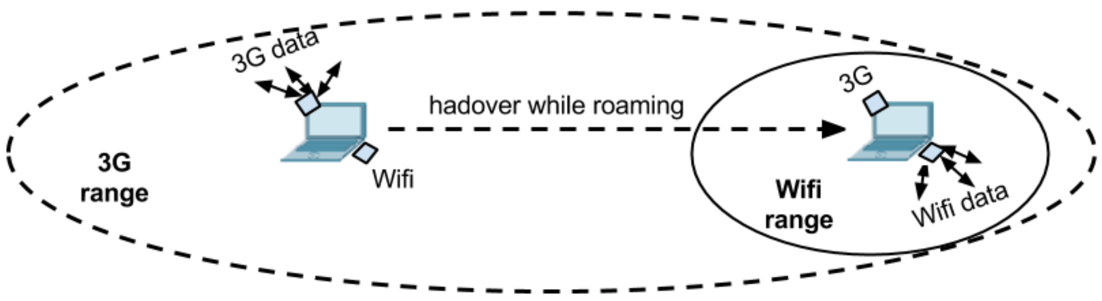


Figure 1.5: Handover from 3G network to Wifi network

Contribution

This thesis document focuses on the last of the issues described above, namely application layer handover using Session Instantiation Protocol[1][7]. The greatest advantage of this type of handover is that while switching to new network, the application can renegotiate session's parameters, depending on the available bandwidth, signal strength and communication protocol, thus establishing higher quality sessions. Seamless, horizontal, soft handover will be described and implemented in this document, using an open source softphone. This implies that while the video call is developing, user will not notice the handover between two networks.

Summary

In the **State of the Art** chapter, mobility issues will be discussed, followed by an overview of the most important protocols used in establishing and streaming of a session. VoIP will introduce the concept of IP telephony and weight the advantages and disadvantages of packet switched networks over circuit switched networks. The Session Instantiation Protocol will be thoroughly discussed, offering a strong idea about the functionality of the protocol and the design of SIP networks. Session Description Protocol, used in SIP message body, will be also presented, finishing with a brief overview of Real Time Protocol.

The **Related Work** chapter will present and describe the handover types, followed by examples of specific protocols which support handover at different layers in the OSI protocol stack. Here, Mobile IP, Mobile Stream Control Transmission Protocol, Multipath TCP and SIP protocols will be analyzed and compared, illustrating the message flow in a typical handover scenario. Afterwards, some example of existing softphones' features will be provided, followed by a description of the design and features of SFLphone softphone, used in this thesis.

Next, the **Architecture** chapter will picture the modules added to the original softphone and how they interact with the initial modules. Each new developed module will be described, showing its part in performing handover.

Subsequently, the **Implementation Details** chapter will dig into how each module is actually implemented, which programming language is used, how it manages to accomplish a specific task in the handover process and how the classes and functions interact (pseudocode might be used when needed).

Following, **Case Study and Experiments** chapter will describe, analyze and compare common handover scenarios. Each scenario is described and pictured accordingly, followed by specific measurements' analysis and interpretation. Firstly, the most basic type of handover will be sketched, followed by additional improvements that will eventually lead to the best handover type that could have been implemented.

Finally, the **Conclusion and Future Work** chapter summarizes the concepts presented throughout the entire thesis and presents some optimizations and further improvements that are to be done.

Chapter 2

State of Art

In this chapter, the concepts of mobility will be introduced, followed by an overview of protocols used in the mobility context. The chapter is divided into five sections, the first section describing the context of mobility and the following sections describing some of the most important protocols used when talking about mobility.

2.1 The Mobile Environment

The use of mobile, wireless technologies are becoming more and more widespread nowadays and thus mobility issues arise. One of the mobility issues is that a device will not receive packets as soon as it leaves the current network because of the destination address in the IP header of the receiving packet. However, mobility does not necessary refer to a person movement; it might refer to switching to a better signal. Take for example a 3G device which might switch to a better signal cell due to fluctuations in the sensitive wireless environment, even though the device is stationary. There are several types of mobility, classified either by category of user or by availability of services. The former comprises Terminal Mobility, Personal Mobility, Application Mobility, Session Mobility and Role Mobility, while the latter comprises Continuous mobility and Discrete mobility, as this paper [21] informs. Following, each type of mobility will be briefly described.

Category of user mobility:

-*Terminal Mobility* refers to a single device (laptop, phone) capable of moving between different IP network subnets while offering all the subscribed services. For example Mobile IP[12] is a network layer terminal mobility mechanism which will be discussed in the next section. Mobile Stream Control Transmission Protocol[20] is a transport layer terminal mobility mechanism which will also be discussed in the next chapter.

-*Personal Mobility* refers to the capability of a person to access the network or to be accessed by the network without depending on the gateway or the device it uses, while offering all the subscribed services. Therefore, a person could manage its programme so it can be contacted at the company PC by day and at the personal phone by night.

-*Application Mobility* refers to the capability of relocating applications (i.e. processes) while moving to a new terminal, so a person can access that application regardless the device.

-*Session Mobility* refers to maintaining the current sessions while either persons, applications or terminals are moving from one IP network to another. For example the conference session is maintained when switching from voice-video enabled to voice-only enabled terminals.

-*Role Mobility* refers to the switching between two or more roles of a person. For example consider a person who has two jobs and thus, two ”roles” namely manager and employee; role mobility allows switching between services that come with each role. It is a relatively new concept of mobility.

Availability of services mobility:

-*Continuous mobility* refers to maintaining services while moving. It is met in mobile phone networks and it is desired to expand the entire planet. It uses the Session Mobility concept.

-*Discrete mobility* refers to maintaining services only on limited areas like offices or classrooms. Services are not maintained between two areas. Session Mobility is also to hold session between two areas.

The Mobile Environment consists of different types of networks, varying from traditional telephony networks, Wifi, WIMAX, 3G, UMTS, LTE. Maybe the currently most popular wireless technology is the Wifi technology. The Wifi networks could be grouped in infrastructure and ad-hoc networks, as depicted in Figure 2.1. In infrastructure networks the message flow is managed by a central entity, while in ad-hoc networks every node participates in forwarding messages and no coordinating entity is needed. Thirteen channels are supported in the $2.4G$Hz band from which three of them are independent and do not interfere, namely channel 1, 6, 11. Also the standard has been extended in the 5GHz band for higher transmission speed and more independent channels.

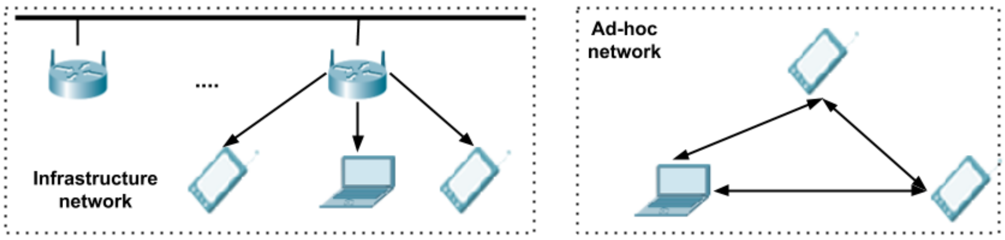


Figure 2.1: Wifi network types

Telephony[24] represents the signals transmitted over a medium to make possible the spoken communication between two or more persons. The first telephony networks’ designs followed two patterns shown in the Figure 2.2. In meshed networks each end-point was linked to all other end-points in the network. This design did not scale, the number of network links growing exponentially. Switched networks design reduced the number of links in the network by using a centralized approach.

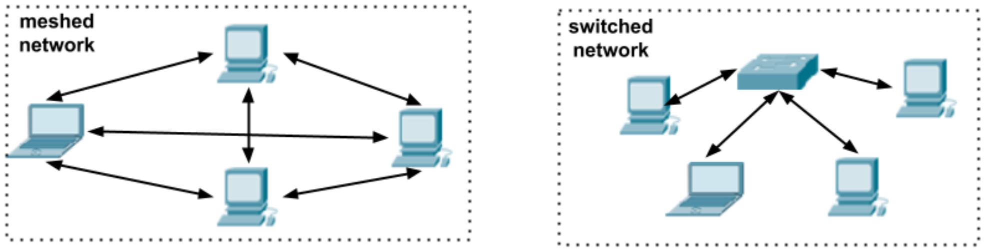


Figure 2.2: Meshed vs Switched networks

Wifi telephony[2] networks uses Wifi protocol to transmit Voice over IP messages. The advantage over other types of wireless telephony is the higher transmission rates which results in enhanced media quality. Also improves mobility and can be easily integrated with other types of telephony networks like traditional Public Switched Telephone Network[2].

2.2 Voice over IP - VoIP

VoIP[24] is not exactly a protocol. It could be considered a standardized, common language which all VoIP devices must speak. VoIP family comprises a couple of signaling protocols which are used to notify the beginning and the end of a call. Some examples of signaling protocols and their intended usage are shown in Table 2.1. Compared to real-time traffic, the signaling protocol traffic is sent in a short and bursty manner. Usually, a single phone supports just one of the signaling protocols. When choosing a certain signaling protocol, the engineer must take into consideration the interoperability and compatibility issues. Thereby, a packet switched network protocol is used in the present thesis, namely Session Initiation Protocol.

Protocol	**Signaling scenarios**	**Maintainer**	**Legacy support**
H.323	Telephony, video	ITU-T	✓
SIP	Telephony, video, IM	IETF	×
IAX	Telephony	Digium,Inc	×
SCCP	Telephony	Cisco Systems	×
H.248	Telephony, gateway control	IETF + ITU-T	✓
MGCP	Telephony, gateway control	IETF	✓

Table 2.1: VoIP signaling protocols

There are two main types of networks, based on the approach used to transport data between network end-points:

-*Circuit Switched* network transports data based on a dedicated channel which guarantees a certain bandwidth. The network nodes are called switches (e.g. Crossbar Switch). Once the circuit is established, the data transfer is occurring at high speeds, making Circuit Switched network suitable for transmitting analogue signals. The same channel (i.e. network nodes) cannot be used at the same time by two different sessions and time is needed for the circuit establishment. An example of this type of network is the Public Switched Telephone Network (PSTN[24]) which is actually the old fashion telephony network.

-*Packet Switched* network transports data based on the data header content attached by sender end-point and removed by the receiver end-point. It was designed to transmit bursty data. The network nodes are called routers. The routing process is accomplished either by virtual circuits, when the packets always take the same route to the destination, emulating Circuit Switched network, or by datagrams when the packets always take different routes, therefore ensuring load-balancing. This network is more efficient because there are no idle resources. Being less expensive, the nodes can also buffer received data such that no transmission rate conversion is needed. The drawbacks are the higher delay, the overhead due to data header, the impossibility of transmitting a certain data packet unless it is next in the transmitting queue. The Internet is an example of this type of network.

VoIP transmits data over the Packet Switched network. Some advantages of VoIP are the lower cost of the VoIP devices, which can use the existing data packet network, not needing a separate infrastructure for telephony. Secondly, the integration of telephony with computers is very easy using VoIP. Finally VoIP networks are easy administered, scale very well, and can be used for long distance communication, over Internet. As a drawback, VoIP messages can be discarded with the network traffic increase, as long as no Quality of Service (QoS[9]) is implemented.

2.3 Session Initiation Protocol - SIP

Session Initiation Protocol[1][7] protocol is an application layer protocol and is used for establishing, ending as well as modifying sessions in a packet switched network like Internet. SIP could be considered an envelope transporting the details regarding a certain session (usually SDP message). Being independent of the handled session type and the way of describing sessions, it has a variety of applications from games to video conferences. SIP supports TCP, SCTP, UDP, TLS as transport layer protocols. Some of its features resemble two other application layer protocols, HTTP and SMTP. Being ASCII encoded, like SMTP, makes it very easy to debug. Also SIP header fields resemble SMTP *"To:"*, *"From:"*, *"Subject:"* field format. The resource identification is made using SIP Universal Resource Identifier (URI), similar to HTTP.

2.3.1 SIP message

The SIP messages can be classified into request methods and response messages. The first category includes the INVITE, ACK, BYE, CANCEL, OPTIONS, REGISTER, PRACK, SUBSCRIBE, NOTIFY, PUBLISH, INFO, REFER, MESSAGE, UPDATE methods. The second category is further partitioned into six types of responses displayed in Table 2.2.

Response Type	**Particularity**	**Example**
1xx Provisional	Indicates that the server has not a definitive response	100 Trying, 180 Ringing, 181 Call is Being Forwarded, 183 Session in Progress
2xx Successful	Indicates that a certain method was successfully accepted	200 OK, 202 Accepted, 204 No Notification
3xx Redirection	Indicates next action to be done to solve the request	300 Multiple Choices, 301 Moved Permanently, 302 Moved Temporarily
4xx Client Failure	Indicates either request bad sintax or the specific server could not process a valid request	400 Bad Request, 403 Forbidden, 408 Request Timeout, 410 Gone, 484 Address Incomplete, 489 Bad Event
5xx Server Failure	Indicates that the specific server could not process a valid request	500 Server Internal Error, 501 Not Implemented, 502 Bad Gateway
6xx Global Failure	Indicates that no server could process a valid request	600 Busy Everywhere, 603 Decline, 604 Does Not Exist Anywhere

Table 2.2: SIP Response Classes

A SIP message consists of a SIP header, which contains a list of header fields, some of them being mandatory, and a SIP body, which usually is a SDP message, describing the ongoing session. The order of the header fields is not important, but it is recommended that routing headers to stay at the top of the list. The format of a header field is case-insensitive, usually following the *"Header: info;params"* pattern. A brief description of each mandatory header field in a SIP header is granted in the below Table 2.3.

Field	Feature	Example
Via	Each device that creates or forwards SIP messages will append it's own URI here	Via:SIP/2.0/UDP 1.2.3.4:5060;branch=b323aef Via:SIP/2.0/UDP 5.6.7.8:5060;branch=x1l2cz Via:SIP/2.0/UDP proxy.ro:5060;branch=x1zbl
To	Identifies the receiver	To: sip:elena.apostol@upb.ro
From	Identifies the sender	From: sip:stefan.mititelu@upb.ro
Contact	Identifies end user device	Contact: <sip:stefan.mititelu@11.22.33.44>
Call-ID	Used to identify each SIP session	Call-ID: 45382142
Cseq	Incremented with each SIP request sent	Cseq: 1 REGISTER
Max-Forwards	Decremented by each server to prevent routing loops	Max-Forwards: 70

Table 2.3: SIP mandatory header fields

One of the important header fields in a SIP message is *Via* header field because the routing process is based on its value. As Figure 2.3 illustrates, each time the message passes through a Proxy server, described in the following 2.3.3 section, the proxy may stamp its own URI in a Via header to indicate the route of the response. When receiving a response, Proxies check if the first Via header corresponds to their URI, removes it and forwards the message to the next URI indicated by the next Via such that the messages returns on the same path. To be noted that is not mandatory for Proxies to update the Via header field, it is just a method of tracing back the session initiator and forcing the message to pass through certain Proxies.

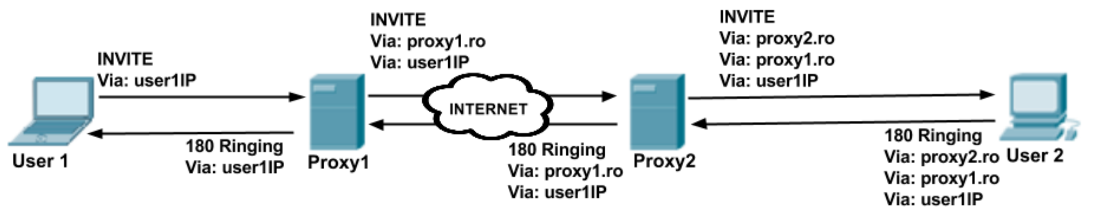
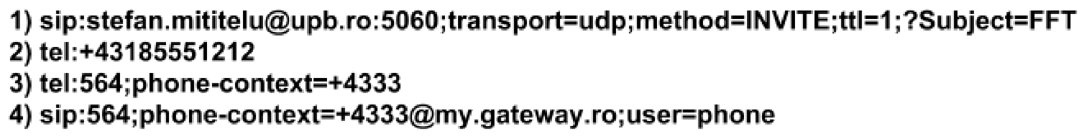


Figure 2.3: SIP Via header field

2.3.2 SIP URI

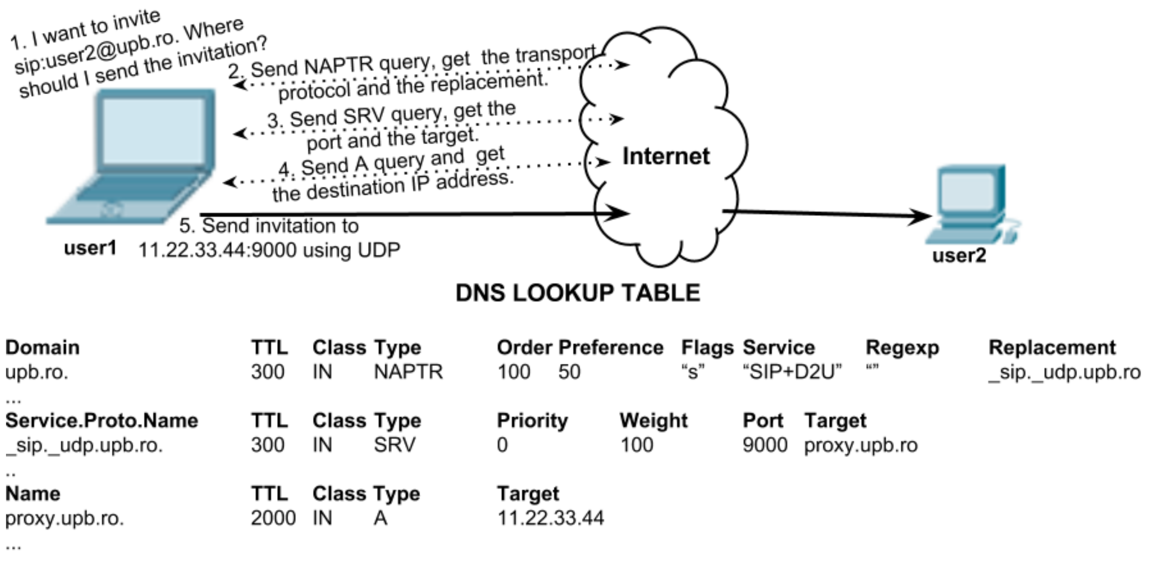
As introduced above, SIP uses Universal Resource Identifier as a **logical address** for a certain user. The same URI can be used to access various devices, registered by a user. The general form of an URI is *"scheme:scheme-specific-attributes"*, without any spaces in it. The URI schemes supported by SIP are *sip, sips, im, pres, tel.* The sip and sips specific URI format is *"protocol:user@host:port;params"*. Take as an example the first line in the Figure 2.4. Header fields can be added in the request by specifying them after the "?" symbol. The next two lines are examples of *tel* URI schemes. Line two is an example of global telephone number where +4 is the country code, and the rest is the telephone number. Line three is an example of local telephone number where +4 is the country code, 333 is the zone code and 564 is the telephone number. The tel URI can be easily converted to sip URI according to line four in the Figure 2.4. Note the *user=phone* parameter which makes the transition between the two URI schemes.



1) sip:stefan.mititelu@upb.ro:5060;transport=udp;method=INVITE;ttl=1;?Subject=FFT
2) tel:+43185551212
3) tel:564;phone-context=+4333
4) sip:564;phone-context=+4333@my.gateway.ro;user=phone

Figure 2.4: URI scheme examples

Given a particular SIP URI, the endpoint must infer where and how to send the SIP message, namely the transport protocol, port number and destination address. If it already has numeric host IP address then UDP is used for sip URI scheme and TCP is used for the sips URI scheme and the well known 5060 port is used. If the SIP URI does not have a numeric host IP address, then DNS **NAPTR** query is issued to discover the transport layer protocol to be used (*SIP+D2U* for UDP and *SIP+D2T* for TCP). Further DNS **SRV** query is sent which will result in the proper port number and possibly the numeric IP address. If needed, DNS **A** or **AAAA** query is issued to find out the IP address. Subsequently, the SIP message is built and sent to the desired endpoint, as Figure 2.5 displays.


1. I want to invite sip:user2@upb.ro. Where should I send the invitation?
2. Send NAPTR query, get the transport protocol and the replacement.
3. Send SRV query, get the port and the target.
4. Send A query and get the destination IP address.
5. Send invitation to 11.22.33.44:9000 using UDP
Internet
user1
user2
DNS LOOKUP TABLE

Domain	TTL	Class	Type	Order	Preference	Flags	Service	Regexp	Replacement
upb.ro.	300	IN	NAPTR	100	50	“s”	“SIP+D2U”	“”	_sip._udp.upb.ro
...									

Service.Proto.Name	TTL	Class	Type	Priority	Weight	Port	Target
_sip._udp.upb.ro.	300	IN	SRV	0	100	9000	proxy.upb.ro
..							

Name	TTL	Class	Type	Target
proxy.upb.ro.	2000	IN	A	11.22.33.44
...				

Figure 2.5: SIP URI resolve

When TCP is used as a transport layer protocol, no reliability problems can occur because TCP is connection oriented and can handle packet loss. While using UDP, SIP reliability mechanisms such as retransmission timers and positive acknowledgements are activated. Provisional responses are never transmitted reliable. However, there are different approaches for INVITE and non-INVITE methods. When non-INVITE method is sent, the T1 timer is released and doubles with every retransmission, until it reaches the T2 threshold at which point the T1 value stops increasing and the retransmission continues until a final response is received. Similar, when INVITE request is sent, the T1 timer is released and doubles with every retransmission until it reaches the 64 * T1 threshold, at which point the transmission stops and the invited endpoint is considered unavailable. Take a look at the below Figure 2.6 for a better understanding.

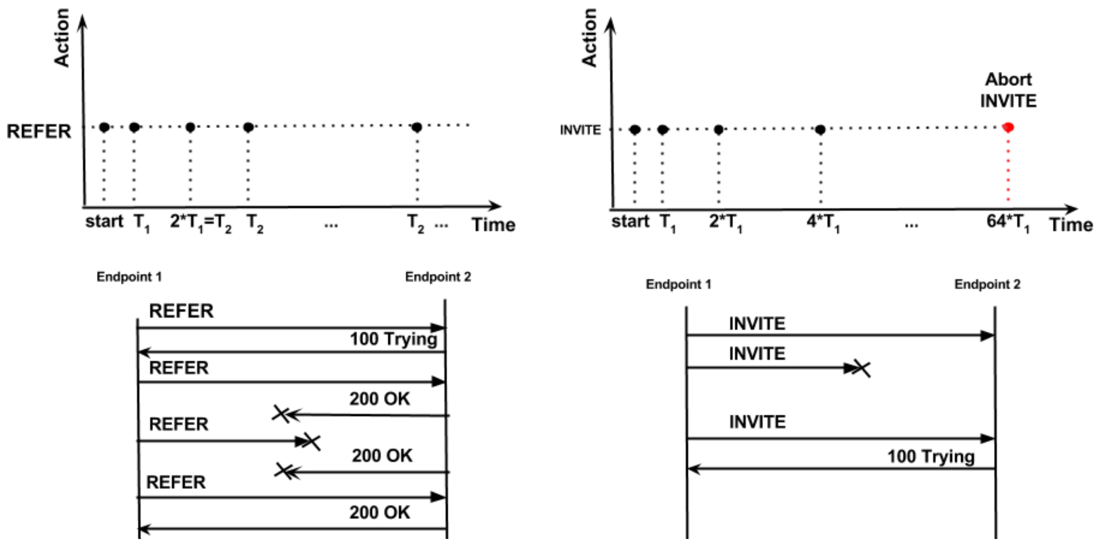


Figure 2.6: SIP retransmission timers and messages

Because of the non-routable, private IP address classes, SIP has difficulties when trying to communicate behind private networks. The solution was the usage of *"received"* and *"rport"* Via header parameters as depicted in Figure 2.7. Moreover, protocols such as STUN[7] or ICE[7] were developed to accomplish NAT traversal.

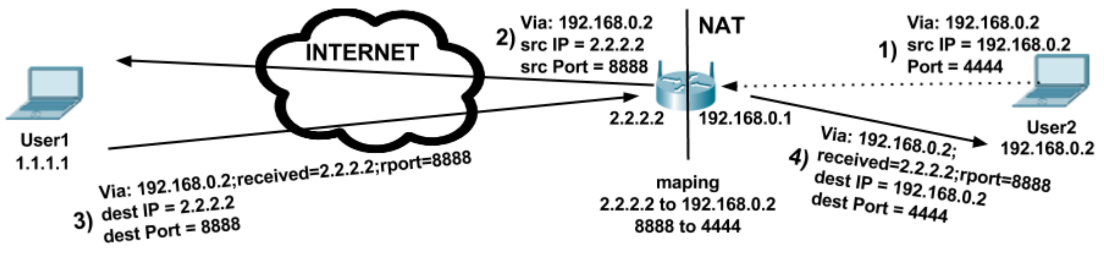


Figure 2.7: SIP NAT traversal

2.3.3 SIP entities

In a complex SIP network, eight types of SIP entities could be encountered: User Agent, Presence Agent, Back-To-Back User Agent, Gateway, Proxy Server, Redirect Server, Registrar Server and Location Server. An overview of these entities and the design of the a SIP network will be presented next, up to the end of this section.

SIP User Agent - UA

The User Agent is the SIP entity which directly interacts with the user by receiving commands through a specific interface, manipulating sessions and rendering results. It is mandatory for an UA to maintain the established sessions, to understand Session Description Protocol used for describing sessions and to implement responses for unknown requests. SIP User Agent comprises both a *User Agent Client (UAC)* which sends the requests and a *User Agent Server*

(UAS) which sends the responses. It can be implemented both in software, without specific hardware support, called softphone, or using optimized hardware, called hardphone. Figure 2.8 shows some examples of SIP User Agents.

Figure 2.8: SIP User Agents

SIP Presence Agent - PA
The Presence Agent is a special type of User Agent which is used to gather information about devices, for instance devices that register or willingly publish information about their presence. Also, the Presence Agent is used in SUBSCRIBE-NOTIFY request-response message flow.

SIP Back-To-Back User Agent - B2BUA
Back-To-Back User Agent is another special type of User Agent which can act as a mediator between the conversation of two different UAs, by receiving, converting and re-sending SIP messages. Thereby, the endpoints need to know only about the B2BUA address to exchange SIP messages. The drawback of this scheme is the higher latency of SIP traffic as well as the single point of failure of the SIP transactions. A solution could be the geographical distribution of B2BUAs.

SIP Gateway
The SIP Gateway is one more special type of User Agent which is capable of converting one signaling protocol to another, thus connecting SIP networks with other network types (PSTN) as pictured in Figure 2.9. The Gateway is the end of media and signaling path and the beginning of other media and signaling path. Is divided into *Media Gateway (MG)* which conducts the media stream and *Media Gateway Controller (MGC)* which conducts the signaling.

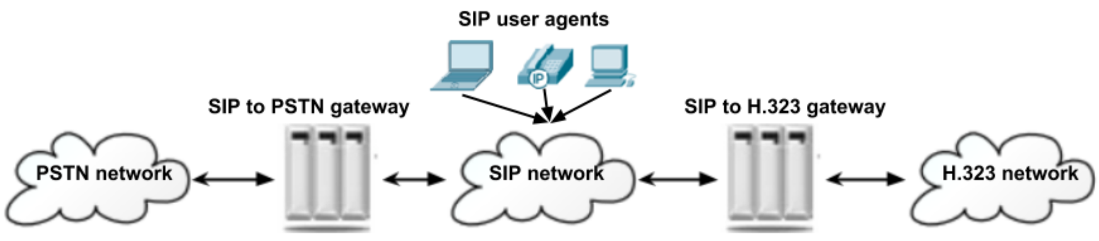


Figure 2.9: SIP Gateways

SIP Proxy Server
The Proxy Server is responsible for responding to or redirecting the received SIP message. It can access a Location Server to obtain the address where to redirect the message. Though it can modify some existing headers of the received message, it never builds and sends requests by itself (excepting the ACK or CANCEL request) and has no media capabilities. The most important feature is that Proxy Server does not need to understand the requests to forward them. Thus new request types can be implemented without any changes to the existing network. Also, can implement group addresses which represent public addresses that can be mapped to a set of employees and the proxy will call each one of them until the call is established.

There are three types of SIP Proxies in a VoIP network namely Call Stateful, Stateful and Stateless Proxy. The Call Stateful Proxy is always in the signaling path between two UAs and accounts state information from the beginning to the end of the session. The Stateful Proxy maintains the state information only during the request, and then destroys this informations. An example of Stateful Proxy is the Forking Proxy, depicted in Figure 2.10, which tries to contact a user on all its registered devices and cancels the invitation for all others once the user answered to one specific device. Finally, the Stateless Proxy redirects the message based only on the Via header, does not keep any state information, does not retransmit messages and does not use timers.

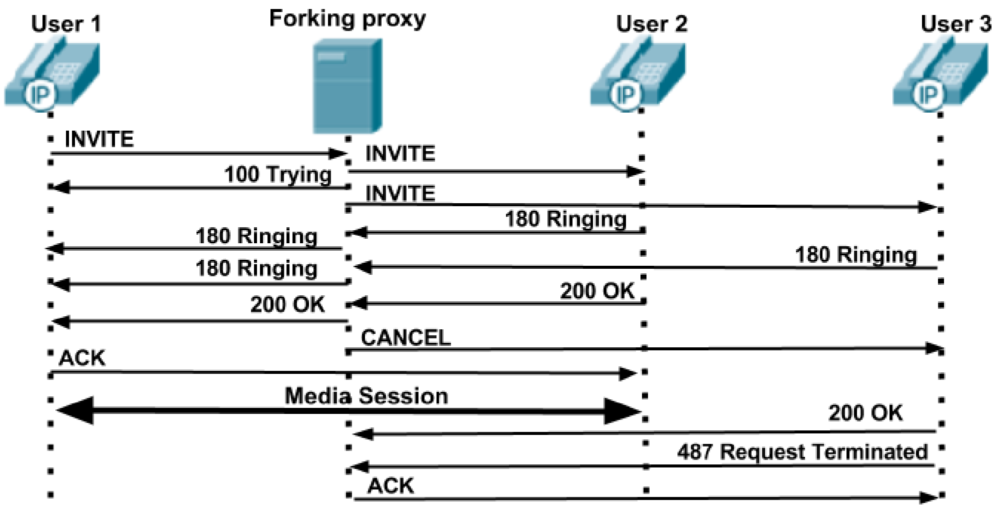


Figure 2.10: Forking Proxy

SIP Redirect Server
The Redirect Server is only responsible for responding to the received SIP message. It can also access a Location Server to obtain the address of the destination endpoint, and can implement group addresses. The new address to which the Redirect Server will redirect the requesting endpoint will be included in the *302 Moved* response message according to Figure 2.11.

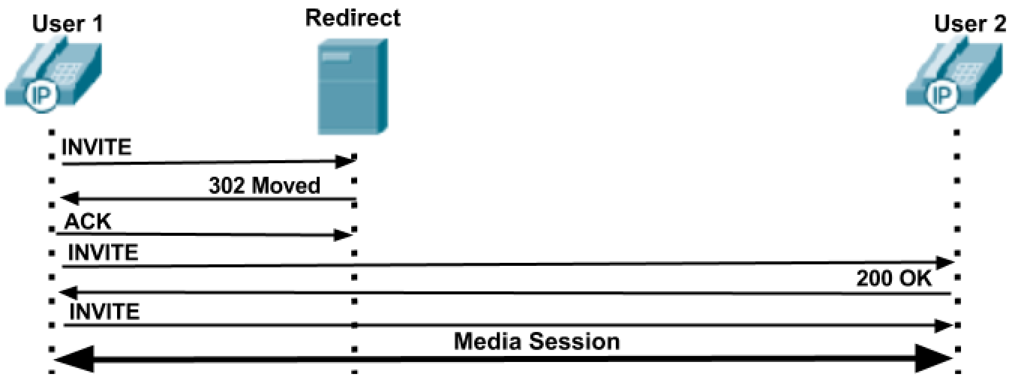


Figure 2.11: Redirect Server

SIP Registrar Server

The SIP Registrar Server allows the user to specify the device on which he/she is currently reachable such that other users to know where to send the call invitation. For example if the user has a mobile phone and a personal computer, each with a certain IP address, he/she could inform the Registrar Server that is reachable at his/hers personal computer between 9 - 17 and at his mobile phone between 17 - 22. This is done by sending REGISTER messages containing the *logical URI to device IP* mapping, according to Figure 2.12. The user can also remove registered devices or add new ones.

Location Server

The Location Server is not actually a SIP entity; it resembles more to a database of user locations. The Registrar Server update the Location Server entries as the user registers or unregisters. Also, the Proxy Server obtains the SIP destination address by querying the Location Server, as pictured in Figure 2.12.

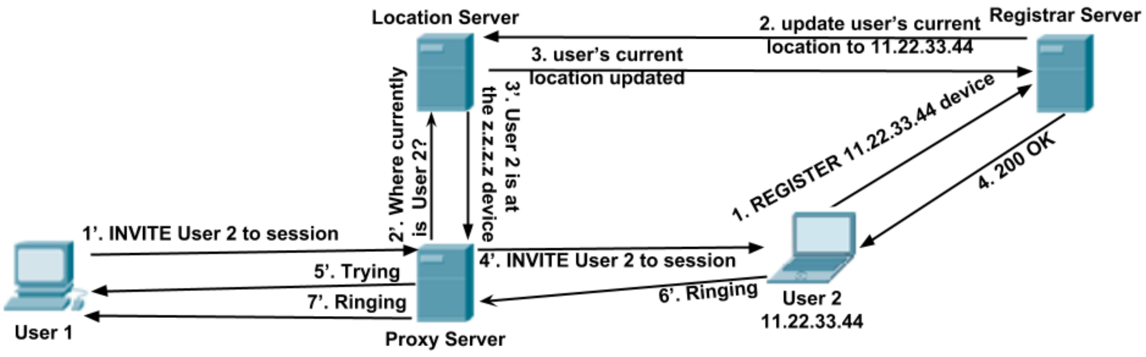

Location Server
2. update user's current location to 11.22.33.44
Registrar Server
3. user's current location updated
2'. Where currently is User 2?
3'. User 2 is at the z.z.z.z device
1. REGISTER 11.22.33.44 device
4. 200 OK
1'. INVITE User 2 to session
5'. Trying
7'. Ringing
4'. INVITE User 2 to session
6'. Ringing
User 1
Proxy Server
User 2
11.22.33.44

Figure 2.12: Location Server

SIP is a scalable protocol due to its architecture design, namely the placement of the servers in the network. The core of the network needs to stay as simple and fast as it can, so Stateless Servers should be placed close to it. While gradually moving away from the core, Stateful Proxies might be introduced, followed by the Call Stateful Proxies, which are the closest servers to the users and can perform intricate routing processes without lagging the core. Figure 2.13 presents the distribution of the SIP servers in the network.

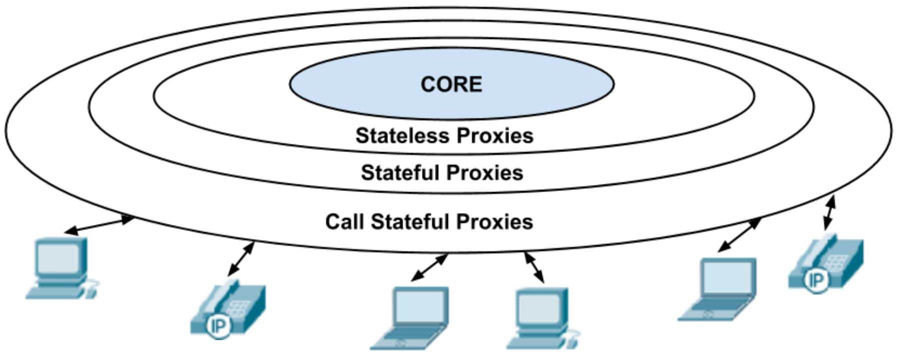


Figure 2.13: SIP architecture

2.4 Session Description Protocol - SDP

Session Description Protocol[1][7] is a static media description protocol, with strict parsing rules, used to describe media initialization parameters. Being an ASCII encoded protocol makes SDP messages more human readable. SDP message consists of a number of fields (Table 2.4), each on a new line, some of them being mandatory.

Field	**Mandatory**	**Feature**	**Example**
v=	✓	Version number	v=0
o=	✓	Session owner	o=stefan 2777634435 2898841234 IN IP4 130.43.2.1
s=	✓	Session name	s=Licence
c=	✓	Connection information	c=IN IP4 11.22.33.44
t=	✓	Session timer	t=2777634435 29196034512
m=	×	Media information	m=audio 49170 RTP/AVP 0
a=	×	Media attributes	a=rtpmap:0 PCMU/8000 a=recvonly
b=	×	Bandwidth information	b=CT:144
i=	×	Session information	i=Information
e=	×	E-Mail address	e=stefan@upb.ro
p=	×	Phone number	p=+4-333-112-1234

Table 2.4: SDP Field examples

Each line in a SDP message begins with a field abbreviation followed by ”=” symbol and a list of parameters separated by single spaces, and ends in *CRLF*. Note there is no space before or after the ”=” symbol: *field=param1 param2 param3 .. paramN*. For instance let’s consider a simple SDP message:

- v=0
- o=mititelu 3250742366 3250742366 IN IP4 11.22.33.44
- s=Licence diploma
- c=IN IP4 110.220.220.110/127
- t=2877631875 2879633673
- m=audio 49172 RTP/AVP 0 6 99
- c=IN IP4 222.222.222.222/127
- a=rtpmap:0 PCMU/8000
- a=rtpmap:6 DVI4/16000
- a=rtpmap:99 iLBC
- m=video 23422 RTP/AVP 31
- a=rtpmap:31 H261/90000

v= represents the current version number of the SDP protocol; the current version is 0.

o= offers information about the session originator. The meaning of each parameter, from left to right, is: username (horizontal dash if none), session-id (unique generated number), version (number which increases if the session changes), network-type (IN for Internet), address-type (IP4 or IP6 addresses) and finally the dotted decimal originator's IP (could also be a DNS resolvable hostname).

s= represents the session name; it is a mandatory field which may contain 0 or more characters as a parameter.

c= offers media connection information. The meaning of each parameter, from left to right, is: network-type (IN for Internet), address-type (IP4 or IP6 addresses) and the connection-address (the dotted decimal host's IP of the following media streams; could also be a DNS resolvable hostname). The connection information could be **session level** (it applies to the entire SDP message) or **media level** (it applies only to the specific media session, ignoring the session level connection information).

m= offers media session information. The meaning of each parameter, from left to right, is: media (could be video, audio, message, text, application), port (port number of the media stream), transport (could be either transport layer protocol or RTP profile, discussed in the next section), format-list (media codecs also present in the RTP header, Payload Type field). Media codecs between *0 - 95* are static payload types (each number is associated with a well defined codec). The format-list interval *96 - 127* represents dynamic payload types, meaning that same codec might be represented by a different number on different hosts. Thus, further attribute is needed to specify the codec.

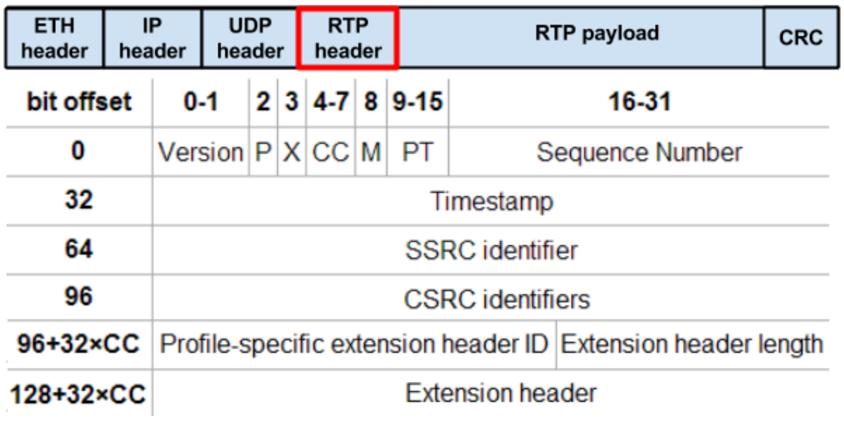
a= offers further information about the previous media session (attributes of the above media session). Note that there could be more than one attribute per media session. Attributes could also be **session level** or **media level**.

To modify a session, either UA must send a *re-INVITE* message with a new SDP. If the **o=** field in the SDP message remains unchanged, no session change is issued. If the version number in the o= field is incremented, the SDP message is re-parsed by the destination UA. The new SDP must include all media lines from previous SDP. To destroy an existing media, the port number in the **m=** header field must be set to 0. New media can be added at the end of the previous SDP message.

2.5 Real Time Protocol - RTP

Real Time Protocol[22] is the protocol used to send voice, video or other media datagrams over IP, using UDP as transport layer protocol. It is an unidirectional protocol and it dictates how the datagrams arrive from source to destination. Uses intricate prediction and interpolation algorithms for the end user not to notice slight packet loss. Also RTP offers support for media encryption. All the information about the RTP media session is contained in the SDP message body, as shown in section 2.4.

The most important factors regarding the quality of the real time media are **packet loss rate** and **end-to-end latency**. Depending on the codec used, more or less lost packets can be recovered. For instance some codecs might be using intricate prediction algorithms, based on the neighbouring packets, to recover the lost packets. Other codecs might be using error correction codes for recovery. Because real time media is very sensitive to latency, end-to-end latency must be kept under 150 milliseconds for the end users not to notice disturbing effects. There are various latency sources in the path between two endpoints like encoding and decoding of data, packetization process, buffering process and especially the transport over the network. Efforts are made to minimize each of the above by developing smart codecs, finding the best payload size to minimize packetization latency, find the best buffer size depending on jitter variations and minimize transport latency using Quality of Service (QoS[9]). Figure 2.14 shows the RTP header followed by a concise description of the header fields.


ETH header
IP header
UDP header
RTP header
RTP payload
CRC

bit offset	0-1	2	3	4-7	8	9-15	16-31
0	Version	P	X	CC	M	PT	Sequence Number
32	Timestamp						
64	SSRC identifier						
96	CSRC identifiers						
96+32×CC	Profile-specific extension header ID						Extension header length
128+32×CC	Extension header						

Figure 2.14: RTP Header

Version indicates the RTP version, 2 being the current version.
Marker (M) when set, indicates the end of a complete video frame.
Payload Type (PT) indicates the codec used in SDP **m=** field.
Sequence Number header field is an unique identifier of a RTP packet and is used to detect and discard out-of-order packets.
Timestamp header field indicates when the respective RTP packet must be played in a specific stream. Thus variable delay (jitter) can be detected.
Payload Type header field indicates the codec used to encode payload data.
SSRC header field is a unique number which identifies a specific actor in a given stream.
CSRC header field is a unique number which identifies a mixer, used in conference media.

Figure 2.15 illustrates the steps of RTP media processing. First of all, data provided by a analogue to digital converter (microphone) is sampled at a certain sample rate, dictated by a certain codec algorithm; the codec algorithm is specified in the header of the RTP packet and is also used for decoding the payload. The sampled data is then broken into smaller chunks to fit in the payload (step 2) and the packet is then sent over the Internet, usually using UDP (step 3). As it receives the packet, the destination checks it not to be out of order or to have errors, removes the RTP header (step 4) and buffers the payload (step 5); the buffer size might adapt due to variable delay (jitter). After decoding the buffered data using the codec algorithm in the payload type header field (step 6), the receiver replays the original sound using a digital to analogue converter (step 7).

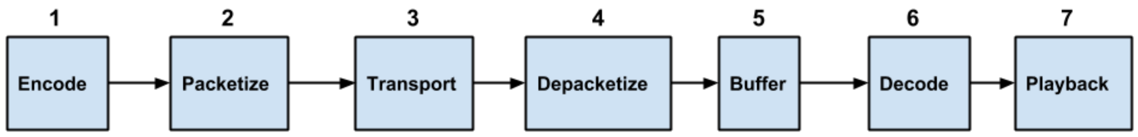


Figure 2.15: RTP processing steps

Real Time Control Protocol (RTCP[6]) is the control protocol associated to RTP, used to transmit control informations like number of packets lost or packet jitter. Usually, RTCP uses the next port number after RTP port.

A RTP profile is a specification of a series of protocols. It could be considered as a protocol of protocols, as proven in Table 2.5. SDP uses some RTP profiles in describing media sessions. For example, RTP/AVP is the most commonly used profile and specifies UDP as a transport protocol, even port number for RTP streams and odd port number for RTCP.

Name	**Standard**	**Feature**
RTP/AVP	RFC 3551	Minimal audio and video control RTP
RTP/SAVP	RFC 3711	Secure RTP
RTP/AVPF	RFC 4585	Feedback RTP
RTP/AVPF	RFC 5124	Secure RTP extended for RTCP feedback

Table 2.5: RTP Profiles

Chapter 3

Related Work

This chapter is structured in three main sections and presents the previous work that has been done in the mobility domain. It begins with the handover concept and provides some implementation examples of existing handover mechanisms at different network layers in the OSI stack model. Following, it provides some examples of existing softphones and the handover mechanisms they implement (if any), ending with the working details and default features of the softphone used in this document.

Due to the need of flexibility and mobility in nowadays society, the demand of wireless devices greatly increased, therefore new mechanisms needed to be developed for these devices. One of the most important mechanism is the handover mechanism because it supports device mobility such that no user involvement is needed when transitioning between networks. Bellavista, Corradi and Foschini were first who made, in their paper[4], a clear classification of the existing handover mechanisms, as displayed in the Table 3.1.

Criteria	**Related features**	**Examples**

Handover awareness	Horizontal/Vertical	Horizontal = switch between two Wifi networks
		Vertical = switch between Wifi and 3G networks

	Soft/Hard	Soft = simultaneous connections to more networks Hard = only one connection to a network
	Proactive/Reactive	Proactive = switch before disconnection occurs Reactive = switch after the communication is down
Qos awareness	Latency	Time need to disconnect and re-connect
	Patcket loss	Lost packets during handover
	Content adaption	Adapting to the bandwidth of the new technology
Location awareness	Micro/Macro/Global	Micro = move between two APs whithin same subnet, with no address change Macro = move between two APs in different subnets, with address change Global = move between two APs in different subnets, with address change and re-Authentication
	Service rebind	Reconnect to services while in foreign area
	Content transfer	Move context information while in freign area

Table 3.1: Handover types

The following sections will present and compare network layer, transport layer and application layer handover mechanisms. Throughout this section, some abbreviation of the mobile environment entities are used :

-*Mobile Node (MN)* is the device which moves and switches between networks. MN has two network interface cards, one used for the home network and another used for the foreign network.

-*Correspondent Node (CN)* the device which communicates with MN. CN may move but for simplicity let's consider it stationary.

-*Home Agent (HA)* is the default gateway in the MN's home network. HA must implement all the basics of a router, including tunneling; may provide a DHCP server.

-*Foreign Agent (FA)* is the default gateway in the foreign network. FA must implement all the basics of a router, including tunneling; may provide a DHCP server.

3.1 Mobile IP - MIP

Mobile IP[13] is a network layer protocol designed by IETF and provides a layer three handover mechanism for devices which move from one network to another. Mobile IP was designed to support seamless and continuous Internet connectivity. It is best suited for active users who might move alot, thus switching between networks very often. Mostly useful in wireless networks, it is also found in wired networks. For a better understanding let's consider the wireless networks because the most roaming devices are wireless.

Suppose MN is placed in the home network. It connects to the home network using the home interface and it gets a home IP address either by DHCP or static IP. This home address is never deleted or changed, even when MN is in the foreign network. According to Figure 3.1, the message generated by MN is sent to HA which forwards it to CN. The response travels back to HA which forwards it back to MN. This is the situation of classic routing and there is no need of additional interface.

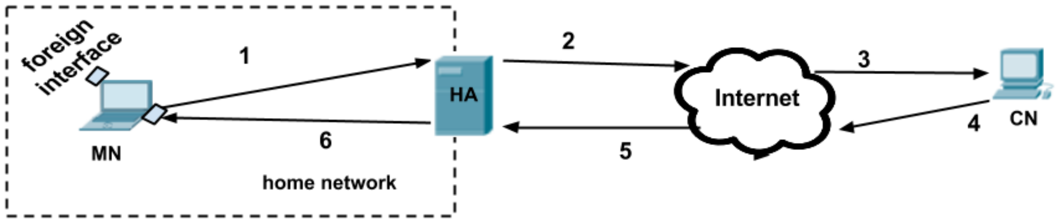


Figure 3.1: MIP home network message flow

As MN moves and arrives in the foreign network an IP is assigned to its foreign interface, either by DHCP or by static IP. This address is called *Care-of-Address (CoA)* and changes within different foreign networks. MN updates its location by sending a Registration Request to HA which is now aware of the Care-of-Address. The message generated by MN still has home IP address as its source address and CN address as its destination address. Because routing occurs based on destination address, the message is send through foreign interface to the default router i.e. FA which forwards it to CN. However, CN does not know that MN is roaming and is not aware of the MN's foreign location; MN is a static node to him, thus the response is sent to HA. Following, HA adds the CoA IP address over the current destination IP and then forwards it to FA(i.e performs IP tunneling). The latter performs routing table lookup based on the tunnel header, drops the tunnel header of the message and forwards it back to MN. Figure 3.2 depicts all of the above.

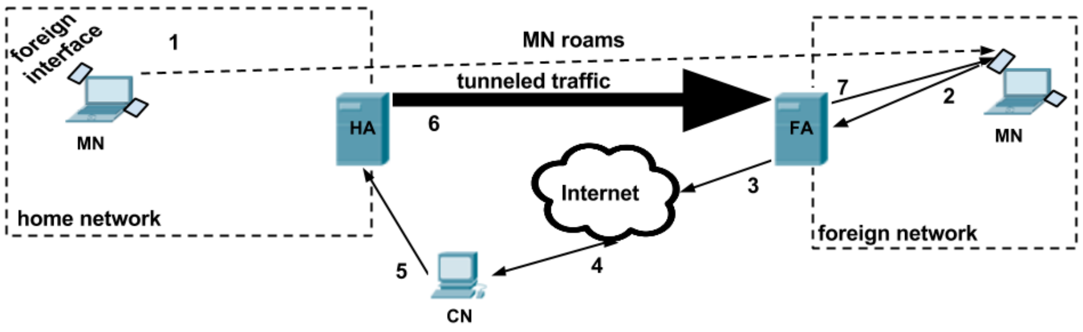


Figure 3.2: MIP foreign network message flow

The design flaw of the Mobile IP is that IP is considered both location identifier and host identity. Because of the detoured traffic from CN to MN through HA, triangular routing problem arises. To perform tunneling, about 20 bytes must be added to each packet which decreases the size of the useful payload. In addition, the detour brings in an unwanted latency, not acceptable in real time traffic. To overcome this problems, the Route Optimization(RO) was proposed. However, this solution has also some disadvantages because CN must know to receive, store CoA and tunnel messages.

3.2 Mobile Stream Control Transmission Protocol - mSCTP

mSCTP[20] is a transport layer protocol, designed by IETF to support multihoming. Multihoming is an interesting feature that allows stations to have more than one IP address and connect to more than one network at the same time. Despite multihoming, mSCTP uses only one primary path to send data in a certain session while on the other paths only retransmissions can be sent. The SCTP failover mechanism is used to detect the availability of all paths. A counter registers the number of consecutive timeouts for the primary path and a heartbeat algorithm is used for the secondary paths. If the counter reaches a certain threshold, the primary path is considered down and a new one is chosen from the existing secondary paths. mSCTP also allows adding and deleting addresses as well as changing the current primary path.

Consider a multihomed MN which has both a 3G and a Wifi interface (3.3). Suppose no APs are nearby, therefore the current primary path is through the 3G interface. As the MN moves, it detects and connects the AP, obtains an IP address through DHCP and adds it to the mSCTP address pool. Next, the MN has to inform the CN about the new address by sending a special control message which must be confirmed by the CN. At this point, the MN can send data for the same session through the Wifi interface, using the second path, but will receive data only on the first path. To switch the entire session on the second path another control message must be sent, which contains the new IP address and informs the CN about the change.

mSCTP protocol is very useful in terminal, session mobility and is able to perform soft session handover or to hold the session state during handover. Also provides redundancy by keeping a list of backup IP addresses. On the other hand, traffic is not load balanced because it is sent only on one path while the others are kept for backup. To overcome this inconvenience,

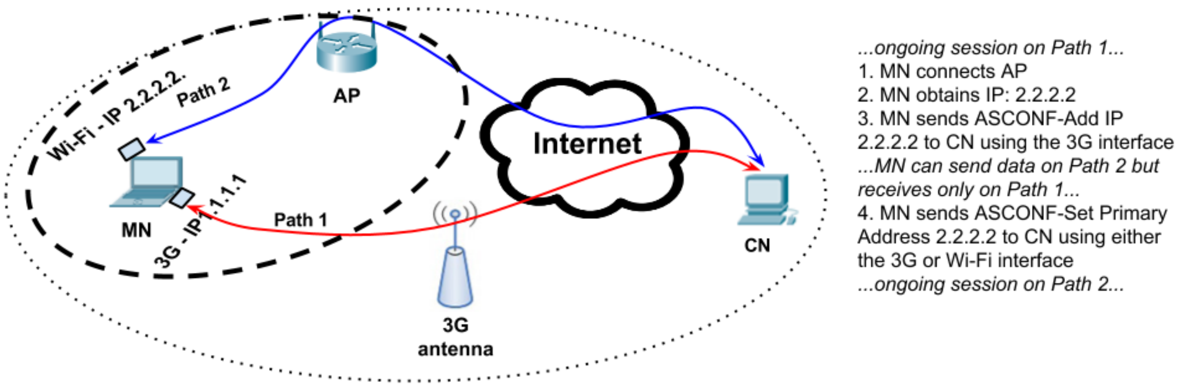

Wi-Fi - IP 2.2.2.2.
Path 2
AP
Internet
MN
3G - IP 1.1.1.1
Path 1
CN
3G antenna
...ongoing session on Path 1...
1. MN connects AP
2. MN obtains IP: 2.2.2.2
3. MN sends ASCONF-Add IP 2.2.2.2 to CN using the 3G interface
...MN can send data on Path 2 but receives only on Path 1...
4. MN sends ASCONF-Set Primary Address 2.2.2.2 to CN using either the 3G or Wi-Fi interface
...ongoing session on Path 2...

Figure 3.3: mSCTP handover message flow

Multipath TCP was developed, presented in the next section. Moreover, mSCTP has NAT problems because it assumes that all the addresses in the address pool are routable.

3.3 Multipath Transmission Control Protocol - MPTCP

MPTCP[8] is a transport layer protocol also designed by IETF to support multihoming. In addition to SCTP, it supports data transfer on multiple paths at the same time. Therefore, the total throughput is the sum of the throughput on each path. It is compatible with standard TCP and it can be used to perform seamless session handover.

After the connection to the new AP and the attainment of a IP address, MN sends a SYN Join message to initiate the new path and waits for the SYN Join acknowledge. Once acknowledgement received, data for the same session can be sent simultaneously on both paths, as pictured in Figure 3.4.

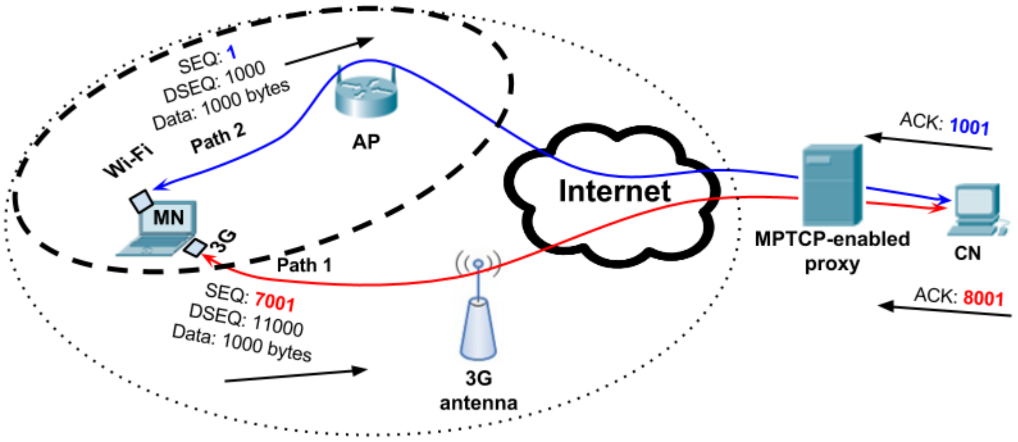


Figure 3.4: MPTCP message flow

3.4 Session Instantiation Protocol handover

SIP is an application layer protocol, it can be used for terminal, personal, session and service mobility and is described in more detail in the 2.3 section of this document. Compared to other types of handover, it has the advantage of re-negotiating session parameters based on current network bandwidth. For example, when switching from 3G to Wifi network, renegotiating media parameters is made possible to increase media quality due to higher Wifi bandwidth. There are two SIP handover mechanisms which make use either of re-INVITE or REFER methods.

re-INVITE based SIP handover

After the MN has obtained an IP address, a *REGISTER* message, containing the new IP address, must be sent to the Registrar Server to update the current device location. Afterwards, the device must inform the CN about its new location by sending a *re-INVITE* message containing the *"Contact: URI"* header field, which resolves to the newly assigned IP. Furthermore, the re-INVITE message body contains an updated SDP message with modified *c=* line, which redirects the RTP session to the new location, process depicted in Figure 3.5. The advantage of this method is the seamless handover performance.

REFER based SIP handover

The approach based on the *REFER* method also updates device's location using a SIP *REGISTER* message. Following a SIP *REFER* message is sent to CN, through the old proxy, which contains the new MN location. When receiving the request method, CN accepts it by replying with a *202 Accepted* response and sends a new invitation to the MN's updated location, waiting for his answer. After the call has been accepted, the old session is tore down and the new session is established, as pictured in Figure 3.5. This approach does not perform seamless handover because the media call is reinitiated and all the media dependencies must be restarted.

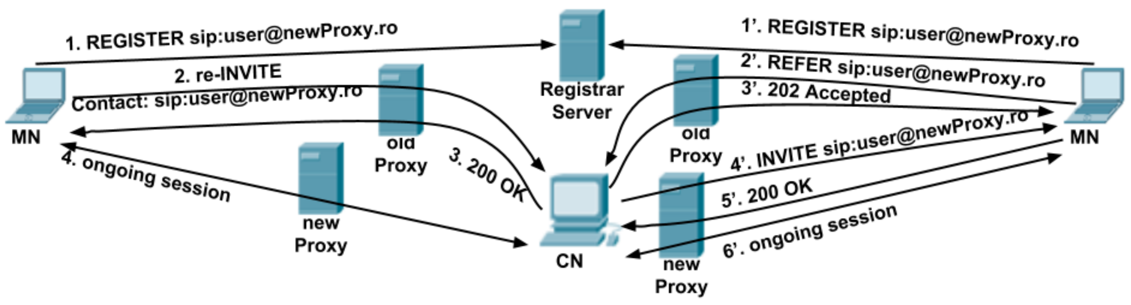

1. REGISTER sip:user@newProxy.ro
2. re-INVITE
Contact: sip:user@newProxy.ro
MN
4. ongoing session
old Proxy
new Proxy
3. 200 OK
Registrar Server
CN
1'. REGISTER sip:user@newProxy.ro
2'. REFER sip:user@newProxy.ro
3'. 202 Accepted
old Proxy
4'. INVITE sip:user@newProxy.ro
5'. 200 OK
6'. ongoing session
new Proxy
MN

Figure 3.5: SIP handover message flow

The handover methods chosen to be implemented and analyzed in this thesis are application layer handover using Session Instantiation Protocol. The main reason for choosing this handover mechanisms is that the ongoing media parameters could be adjusted after handover. This implies that when performing hard handovers, namely switch between different wireless technologies, proper codecs will be used, specific for that technology, thus maintaining reasonable media quality. For example while performing handover from Wifi to 3G the bandwidth information and media codecs might change such that the session can still carry on.

3.5 Existing Softphones' Features

Skype

Maybe the most popular softphone currently on the market is Skype[19], currently developed and maintained by Microsoft. It is a free, closed source product which has support for a variety of operating systems. Its features comprises both audio and video calling, peer to peer and group calling, call forwarding and call history instant messaging, file transfer and, of course, authentication. There are other features such as worldwide landline calling or SMS message transmission that needs to be payed for. However, no handover mechanisms are still implemented. Instead, a mere mechanism that will try, for a few seconds, to reconnect the session after the call disconnected. Even if a second interface is connected and can access Internet, Skype does not switch to the new interface but hopes that the initial interface will recover. Consider the scenario of an initial call that is currently held on *wlan0*. By suddenly deactivating *wlan0* and connecting *eth0*, Skype will not switch to *eth0* but will try reconnecting the session on *wlan0* and will eventually end the session.

Google Hangouts

A service offered by Google, accessible by Gmail or Google+ websites by providing the Hangouts[15] feature. This feature allows users to message or video chat, take and exchange pictures and videos very easy. It allows organization of Hangouts parties, by inviting your Google+ friends which are basically groups where all of the above can be performed and which can be broadcasted and seen by non-Google+ users. International calling between Hangouts users are free. Compared to Skype, it allows free landline calling throughout entire U.S. and Canada as well as free SMS messages through Google Voice[16]. Moreover, Hangouts group video conference allows up to ten members without perceiving any fee while on Skype one of the user must be premium user. While Hangouts permits file sharing through Google Drive, Skype can perform file sharing over IM or video calls. Hangouts integrate with Youtube and Google+ whilst Skype integrate with Facebook social media. However, Hangouts has not yet implemented the presence information feature, while on Skype the status can be set and published. Nevertheless, neither of them implements handover mechanism.

Linphone

Linphone[17] is a free, open source VoIP softphone which uses SIP signaling protocol for session establishment. It is available for Linux, Windows and Mac based operating systems as well as for Android or Blackberry mobile devices. The provided features include hold, transfer, resume of the call, instant text messaging, authentication. It handles NAT issues using STUN servers and supports a variety of audio and video codecs and uses a graphical user interface to interact with the user. Despite all of the above, no handover mechanisms are implemented at all, not even a partial workaround.

SFLphone - used for implementing seamless handover

SFLphone[18] is an Linux and Android based open source softphone, under the GNU GPL version 3 license, designed and developed by the entire global community, maintained by a Canadian company named Savoir-faire Linux. Being implemented in C++, it uses standard *pjsip* library, written in C, to manage SIP message creation and transmission. Supporting both SIP and IAX2 as signaling protocol, SFLphone can handle both audio, video and conference calls implementing a variety of audio and video codecs. Amongst other features, it supports authentication, re-INVITE messages, signaling and voice encryption using TLS, hold, record, auto answer and redirect call, peer to peer calls and instant messaging and also records call history. It does not implement any handover mechanism.

SFLphone was selected for further handover extension because of its straightforward implementation which allows to easily understand and modify the existing code. It is also well organised into modules which simplifies the further implementation. Moreover, all the current implementation details needed can be found on the dedicated wiki. Furthermore, questions can be asked or previous solutions can be found in the mailing list. Also the github repository provides the latest release of the SFLphone softphone.

SFLphone is divided in two main entities, a client and a daemon. The SFLphone client mediates the communication between the user and the daemon. It directly interacts with the user either through graphical user interface or command line interface which allows call automation. The client to daemon communication is accomplished using the D-Bus IPC interface which will be discussed later in this chapter. After receiving the command from the user, the SFLphone daemon makes use of pjsip library to execute the command, namely to create and send the SIP message. SFLphone architecture is presented in Figure 3.6.

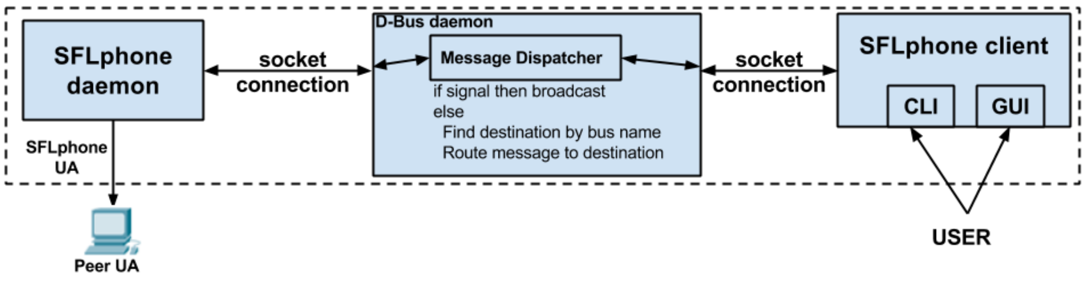


Figure 3.6: SFLphone architecture

The D-Bus interface was designed for the communication between the same desktop applications or between sessions and operating system, but might also be used for other purposes like reliable open connections. A D-Bus daemon handles all the current D-Bus connections identified by a bus name. Facilitates the interaction between daemon and non-daemon processes either through methods or signals, specified by particular interfaces. The methods have a return type, might have several in or out arguments to send or receive data from the daemon and can be invoked by the non-daemon processes. The signals are mere notification that a specific event has happened and can be either true or false. An XML file must be created in order to specify the interfaces, methods and signals used. In this way, the SFLphone client interacts with SFLphone daemon through the D-Bus interface.

Asterix[14] is a free, open source software that acts like a Private Branch Exchange (PBX[24]), namely a server which make possible the communication between two or more softphones. Capable of routing calls through the current private network, it can also connect and forward calls to PSTN or VoIP networks. It supports a variety of VoIP protocols like SIP, MGCP, H323 as well as traditional circuit-switched network protocols like ISDN or SS7. It may behave either like a Registrar server or Gateway or both at the same time. SFLphone makes use of Asterix both as a Registrar and Proxy server.

Chapter 4

Architecture

In this chapter two main architecture types will be described and compared, followed by high-level description of the modules implemented. Our mobility solution is called Mobility Manager, its architecture being pictured and described in Figure 4.2. This chapter ends with the advantages of the decentralized Mobility Manager over the alternative.

4.1 Centralized approach

When deciding about the architecture of the Mobility Manager, two major designs are to be taken into consideration, either centralized or decentralized. The former assumes a separate entity which has information about the device's location and can predict device movement with certain probability based on previous location data. The device constantly sends information to the external Mobility Manager, using special communication protocols. Beside consuming bandwidth for the control messages, if somehow those message are lost, by filtering for example, the location information is lost and thus handover fails. On the other hand, this method is very easily debugged and administered. This type of approach is better suited for Internet Service Providers, which can easily monitor their clients, as depicted in Figure 4.1.

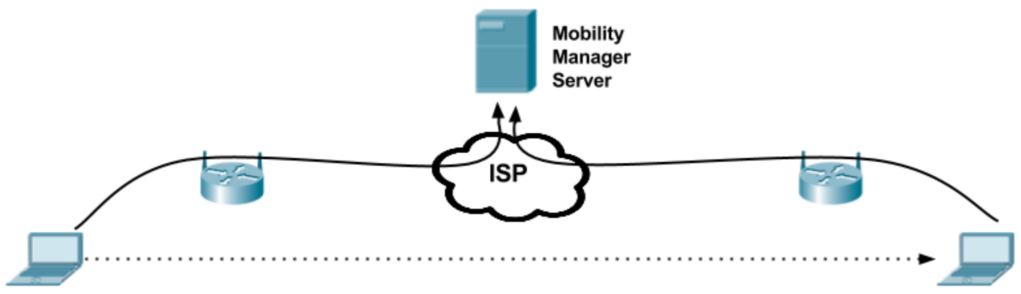


Figure 4.1: Centralized approach

4.2 Decentralized approach

The other, decentralized approach, assumes each SFLphone softphone handles the handover by his own, namely gather signal information about nearby AP, decides and makes handover. Nevertheless, all the softphones need to be patched with the Mobility Manager software which could be considered an additional feature to the SFLphone softphone. The solution is best suited for small, home networks. The current document focuses on the decentralized approach as it offers a simpler solution, yet the handover effect is the same. Moreover, no special communication protocol or intricate algorithms needs to be implemented, no special entity is needed and no bandwidth is consumed for the control messages.

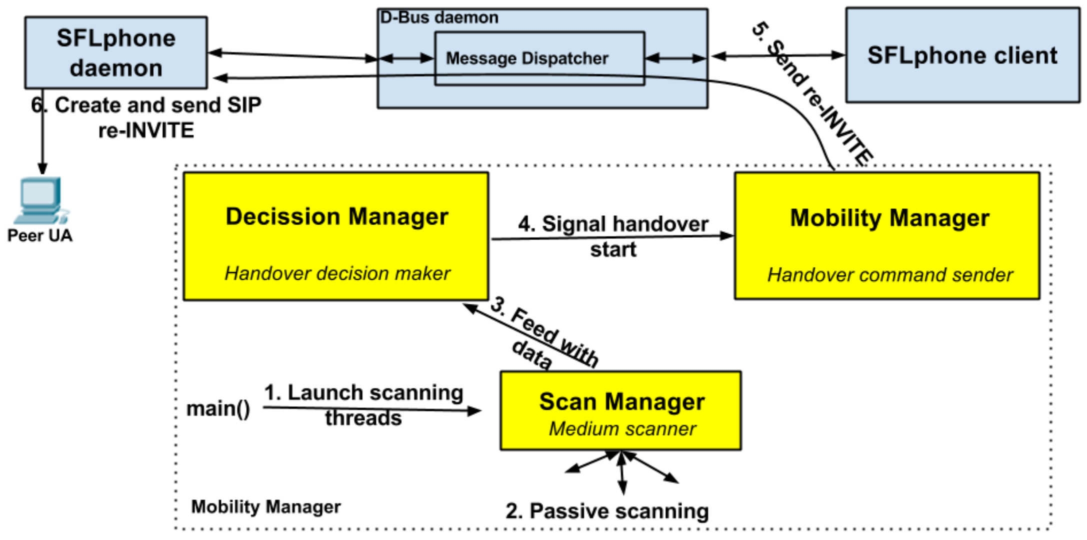


Figure 4.2: Decentralized approach

Figure 4.2. pictures the decentralized architecture of the Mobility Manager and the interaction with the SFLphone softphone. Notice the modular design of the Manager, which makes the integration with the original softphone, the addition or replacement of modules and the debugging easily done. There are three modules that compose the Mobility Manager namely Scan Manager, Decision Manager and Mobility Manager, each with his well defined role. The Scan Manager can be considered a Layer 1 OSI module which uses the Wifi interfaces to scan the environment for existing access points (AP). The presence information as well as details regarding an AP can be deduced based on the beacon frames constantly broadcasted by the AP. The information is buffered and then transmitted to the Decision Manager module. Afterwards, the Decision Manager applies a decision algorithm to the received data, and retains the outcome. Moreover, this module also handles the state of the interfaces namely *UP*, *SCANNING* and *ACTIVE*. If the handover decision has been taken, Mobility Manager module is notified. The last module handles both Layer 3 and Layer 6 OSI handover. After connecting to the new AP and obtaining new IP address, the Session Mobility Manager uses special D-Bus interface, to send handover command to the SFLphone daemon. Following, the daemon creates and sends the SIP message to the peer User Agent to inform it about the handover that has just happened. Finally, the peer User Agent is aware about the updated location and the session may carry on.

The Mobility Manager module is an essential component in implementing horizontal, seamless handover, further divided into two submodules. The first one, handles the network layer handover namely connecting to the new AP based on the essid and password received from Decision Manager, obtaining new IP address through DHCP and updating routing table information. The second submodule deals with the application layer handover namely SIP handover. Provided the newly assigned IP address, this submodule sends the handover request to the daemon, in same way the SFLphone client interacts with the daemon, namely through the D-Bus interface. Although combined into one module, these two submodules can be considered Network and Application Mobility Managers.

The Scan and Decision Manager could have been combined into one big module, as an alternative. By doing this, the Mobility Manager's flexibility might have decreased as well as the current ease of debugging. The modular approach allows further changes or even substitution of the modules. Therefore, the type of information buffered by Scan Manager can be modified or the decision algorithms used can easily be improved or replaced and metrics used to decide about handover could be changed. The above reasons are enough to argue that those two modules should stay separated.

Chapter 5

Implementation Details

In this chapter the implementation details will be discussed for each of the module presented above. The handover solution presented in this thesis is implemented in C programming language. There are three *.c* files, each representing a Mobility Manager's module. In the beginning, the implementation of message exchange over the D-Bus interface will be discussed, followed by an example of how to add new methods or signals to an existing interface. Afterwards, the base implementation of SFLphone's modules will be briefly described, followed by the analysis of each Mobility Manager's module along with comparison with alternate implementations.

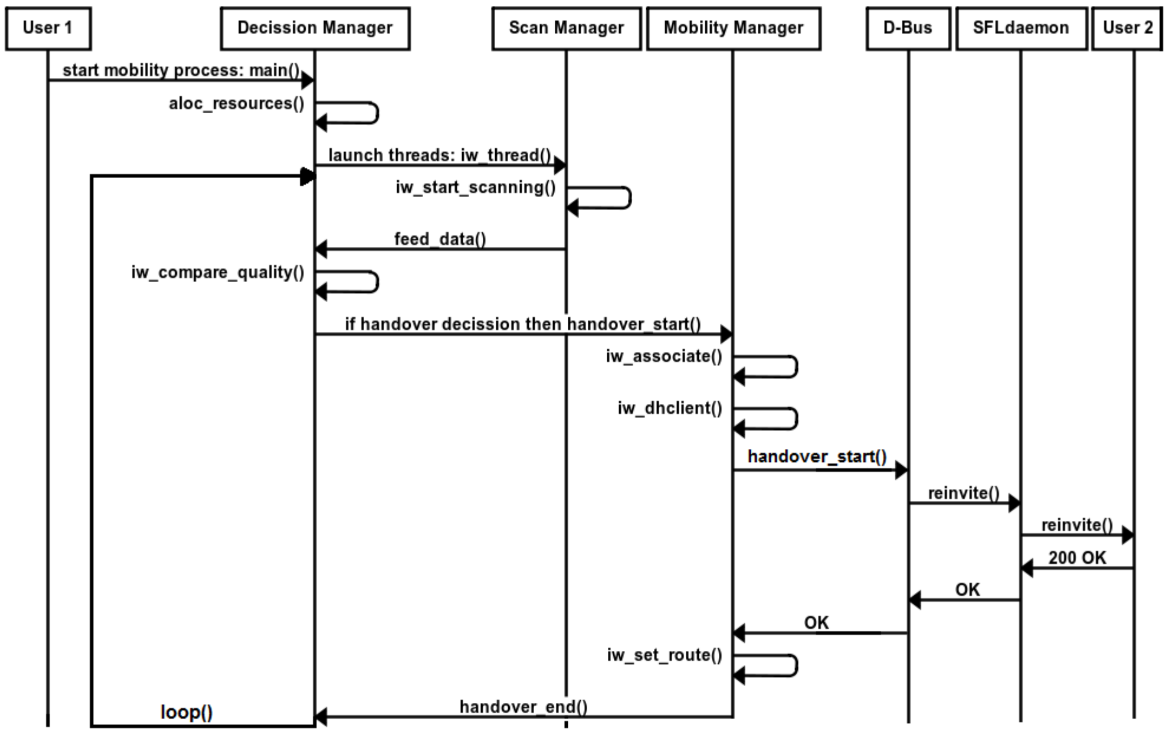


Figure 5.1: Module functionality and inter-operability

In the above Figure 5.1, the main implemented functions are presented along with the interractions between modules. A new Mobility Manager process is started, which looks after the handover that might occur. Based on the feedback from the Scan Manager, Decision Manager determines the proper switch point and calls Mobility Manager's functions to perform handover. To be noted that SFLphone daemon actually creates and sends the SIP re-INVITE message, Mobility Manager just calls the remote daemon's method.

5.1 D-Bus

D-Bus introduces the concepts of **Native Object** and **Object Path**. For example, an instance of *java.lang.Object* is an native object while */home/miti/licence/object* is an object path, namely a filesystem path that maps a native object, thus allowing remote processes to refer to it. The trend is to name the object paths according to the domain name used, for maintaining the modular approach. Each object has its own methods and signals. Methods are basically functions which may have input or output parameters and can be called for a certain object, whereas signals are broadcasted by objects to warn the concerned listeners.

An object can implement one or more interfaces which comprises specific methods and signals. The interface is identified by a string directly mapped to the specific programming language interface. The object-interface, interface-method and interface-signal are in a one-to-many relationship, as shown in Figure 5.2. Another concept introduced is **D-Bus Proxy**, which represents a native object used to refer a remote object and simplifies the way of calling remote methods.

Each time a new process connects to the D-Bus daemon, a new bus connection between the two is created and an unique bus name is assigned to it. This allows other processes to communicate with the initial process through the newly created bus connection, by referring the bus connection name. Apart from routing messages, the bus names are used to detect when a process ends or crashes.

Finally, the processes connecting to the D-Bus daemon can be either servers or clients. The server will listen to a socket, represented by an address, while the client connects to that socket through the D-Bus daemon. Figure 5.2 describes how a client can access server methods.

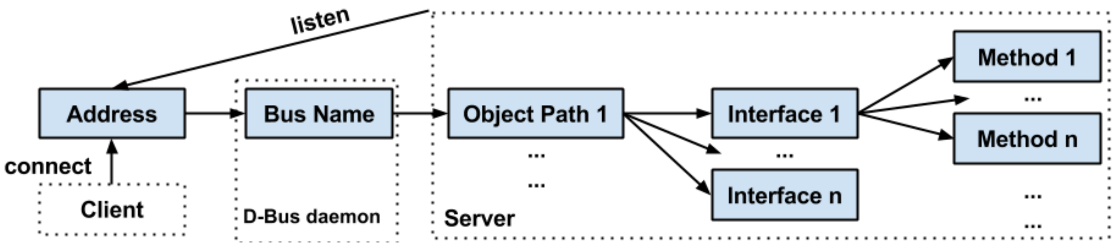


Figure 5.2: D-Bus functionality

The D-Bus object path, interfaces, methods and signals assigned to that object can be defined in an XML file. Only one object can be defined in one XML file. Following, *dbus-binding-tool* is used to transcribe *.xml* file in a *.c* file with the function prototypes defined. By specifying the transcription mode parameter, namely *–mode=glib-client* or *–mode=glib-server*, client or server side code can be generated. Take for example the below XML files imported from SFLphone, each file defining one object with one interface and one method, for each interface.

```
<?xml version="1.0" ?>
<node name="/callmanager-introspec">
  <interface name="org.sflphone.SFLphone.CallManager">
    <method name="reinvite">
      <arg type="s" name="callID" direction="in"> </arg>
      <arg type="s" name="contact" direction="in"> </arg>
      <arg type="b" name="reinviteSucceeded" direction="out"/>
    </method>
  </interface>
</node>

<?xml version="1.0" ?>
<node name="/configurationmanager-introspec">
  <interface name="org.sflphone.SFLphone.ConfigurationManager">
    <method name="sendRegister">
      <arg type="s" name="accountID" direction="in"></arg>
      <arg type="b" name="enable" direction="in"></arg>
    </method>
  </interface>
</node>
```

The SFLphone bus name is *org.sflphone.SFLphone*, the object paths follow the */org/sflphone/SFLphone/ObjectName* pattern and the interface are identified by *org.sflphone.SFLphone.InterfaceName*. The communication between SFLphone client and daemon is made possible using two main objects defined in two XML files, as sketched above. The *callmanager-introspec.xml* file describes the implementation of **CallManager** interface which handles call methods like place, accept, hold, transfer or hangup call. In the *configurationmanager-introspec.xml* file the **ConfigurationManager** interface is described, which controls the account management including register, add new accounts or get accounts details. Another important XML file, *instance-introspec.xml*, describes the **Instance** interface methods used to register a new processes to the bus daemon.

5.2 Mobility Manager

The *mobility.c* file contains the code needed for Mobility Manager to accomplish both network and application layer handover. The further presented implementation will describe how Mobility Manager module works interacts with other modules.

To perform network layer handover, the module must associate to the new AP, obtain IP address and set default route. Using modified *iwlib* library functions, the module connects to the AP using the essid and password, when necessarily. The association is successful only if the new AP has no encryption or is WEP encrypted with the *"aaaaaaaaaaaaa"* password. At this point, two interfaces will be active and connected to two different APs. The IP address is obtained using the *dhclient* system function. Finally, the default route is updated through the newly associated interface, using a special function that can either add or remove routes from the routing table. Basically, three functions are executed when performing Layer 3 handover:

```
iw_associate(interface, AP);
iw_dhclient(interface);
iw_set_gateway(network, mask, via)
```

To communicate with SFLphone daemon, Mobility Manager module must use the same D-Bus communication methods SFLphone client uses. Therefore, some methods from the two main XML files are imported. In addition, reinvite method was additionally appended in the *callmanager-introspec.xml* file on both Mobility Manager and daemon side. The re-INVITE method is shown in the above section5.1 and comprises three parameters: the callID string which identifies the current SIP session, the contact string which is the new IP address to which handover is performed and a boolean variable to indicate whether or not the reinvitation succeeded.

After building prototype communication methods using the above tool, the Mobility Manager needs to establish a new connection with the specific bus name. This is done by using both D-Bus API and generated instance functions. The below example registers a new process so it can communicate with the SFLphone daemon:

```
connection = dbus_g_bus_get(DBUS_BUS_SESSION, error);
instance_proxy = dbus_g_proxy_new_for_name(connection, SFL_bus_name,
                 SFL_object_path, SFL_interface_name);
org_sflphone_SFLphone_Instance_register(instance_proxy, getpid(),
                 process_name, error);
```

After connecting to the SFL daemon, Mobility Manager module can call remote objects' methods from SFLphone daemon's side. When calling *reinvite()* or *transfer()* method with appropriate parameters, the specific function in the SFL daemon process is called. The function extracts the contact IP address from the parameters and starts creating a SIP INVITE or REFER messages. The message contains the *Contact* header field with the new contact IP and has updated *Via* header field also containing the contact IP. Moreover, the new updated SDP message contains connection information related to the new contact address previously extracted from the Mobility Manager's request. Following, the daemon sends the message to the destination, which updates the peer's location. Now, the media session is has been switched and application layer handover is completed. The functions called to accomplish application layer handover are listed below:

```
iw_get_new_ip(if_name)
iw_get_call_id(call_proxy, callID)
iw_reinvite(call_proxy, callID, newContact) or
iw_transfer(call_proxy, callID, transferPeer)
```

5.3 Scan Manager

The Scan Manager's functionality is implemented in *scan.c* file which contains the code executed by the launched scanning threads. For each interface, a scanning thread is created and starts executing the given function. The communication between Scan Manager module and Decision Manager module is made possible using shared memory access, as depicted in Figure 5.3. Each given thread argument is actually a structure which contains references to a memory zone, also read or written by the Decision Manager.

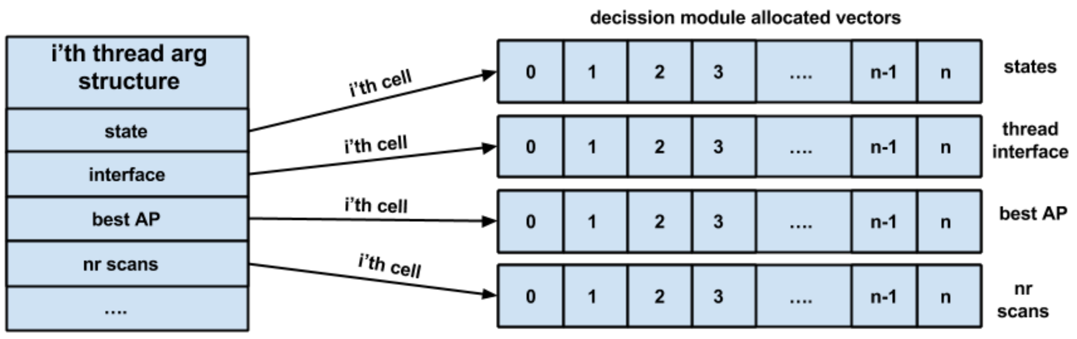


Figure 5.3: Interaction between Scan and Decision

In a while loop, the thread checks for its current state, namely *SCANNING* or *ASSOCIATED*. If the thread is in *SCANNING* state, medium scanning is performed using a modified *iwlib* function, followed by the selection of the best AP signal quality received, which is transmitted to the Decision Manager by updating the shared memory location. If thread is *ASSOCIATED*, only the signal quality received from the currently connected AP is measured and updated.

Even if no synchronization mechanism is used, there are no concurrency problems in reading the correct thread's state because the thread never ends, cycling in a while loop, and will eventually read the updated state. Moreover, the thread never transits directly between the two main states but passes through an intermediary state, which is an extra safety mechanism, as depicted in Figure 5.4. Moreover, the Mobility Manager process never returns, it loops forever, thus Decision Manager will eventually read the updated scanned values.

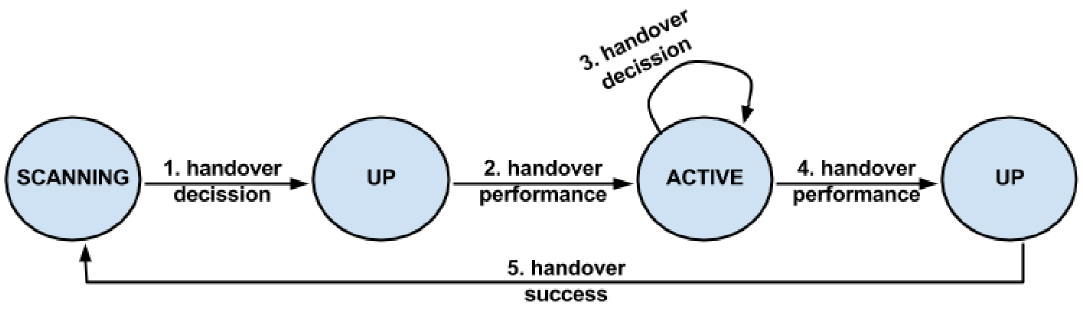


Figure 5.4: Thread state transition at handover time

5.4 Decision Manager

The *decision.c* file contains the decision algorithm used to compare the interfaces' qualities and to trigger handover. Initially, all the Wifi interfaces are in *SCANNING* state. After performing initial scan, the signal qualities are compared by the Decision Manager. The interface with the highest signal quality received is associated with the specific AP and its state is switched to the *ASSOCIATED* state by the decision module, passing through *UP* state, as shown in the above Figure 5.4. If some of the signal qualities are equal, the interface with the highest number is connected.

Following, threads are scanning, according to their present state and signal qualities received are continuously compared. Handover decision is taken when the difference between the best quality received amongst the scanning interfaces is greater than the quality of the associated interface plus a threshold value, a number of scans in a row, as depicted in Figure 5.5. Afterwards, Mobility Manager is notified and the handover process described above begins. Some of the functions implemented by the decision module are listed below:

```
iw_get_state(if_name)
iw_set_state(if_name)
iw_compare(scanned_cells, if_associated)
```

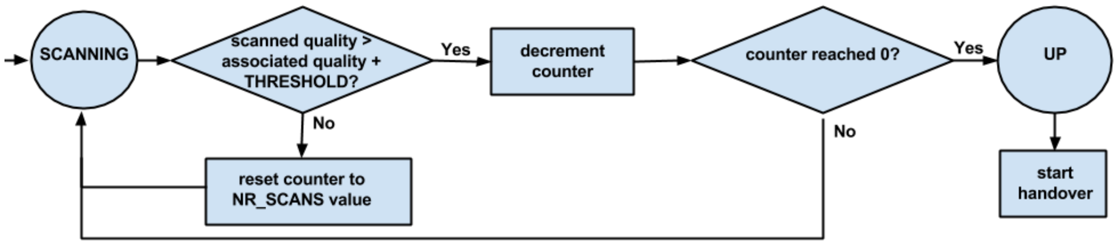

SCANNING
scanned quality > associated quality + THRESHOLD?
Yes
No
decrement counter
reset counter to NR_SCANS value
counter reached 0?
Yes
No
UP
start handover

Figure 5.5: Decision algorithm

All of the above modules are called in the *main()* function of *main.c* file. Here, the shared memory is allocated, all the interfaces are brought up, scanning threads are created and launched, handover decision is taken and handover is engaged based on the above implemented functions. The main loops forever and never returns; the scan, decision and handover actions are endlessly repeated. Defined threshold value and number of times the quality difference must be positive can be easily adjusted.

Chapter 6

Case Study and Experiments

This chapter will present three main handover scenarios, each of them being described and pictured accordingly, followed by the measurements and charts which mirror the specific experiments made. The Handover Time (HT) which comprises the Handover Time at Layer 3(HTL3) and the Handover Time at Layer 6(HTL6), will be measured for each scenario. For simplicity, all of the following scenarios simulate the Internet in a private local area network meaning that no NAT techniques are used for SIP signaling. In the end, a comparison between presented scenarios will be pictured and discussed.

6.1 Scenario 1

The first scenario, shown in Figure 6.1, presents the most basic type of handover, namely hard application layer handover using SIP, which has the drawback of closing current session and then starting a new one. The Mobile Node has only one Wifi interface on which both environment and interface scanning is performed.

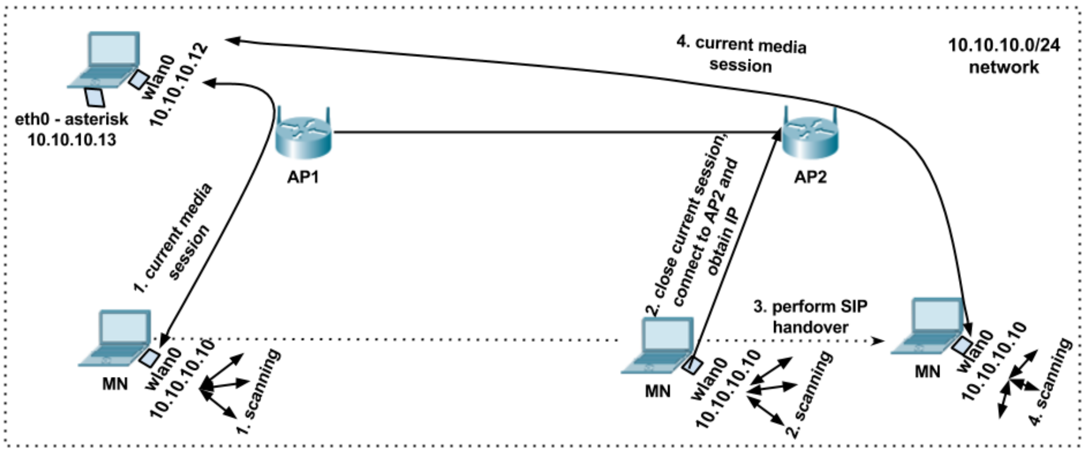


Figure 6.1: Scenario 1

Initially, the interface is scanning the environment and is not connected to any AP. After detecting AP1 and connecting to it, IP address has to be obtained through DHCP and routing table information needs to be updated. At this point, Scan Manager performs both environment and interface scanning and buffers signal quality information about all the APs it detects, including the currently connected AP. Afterwards, this information is sent to Decision Manager module which compares the value received with the signal quality from the currently connected AP. While moving towards AP2, the initial signal quality is decreasing while the new signal quality increases. According to the above decision algorithm, if the difference between the signal quality overpasses a certain threshold in favour of AP2, three scans in a row, handover must be performed by informing the Mobility Manager module. Following, the current call will be terminated, and network layer handover will be performed, namely connecting to AP2 and establishing Internet connectivity. Next, new call session will be established by sending SIP INVITE message to CN which will accept the invitation and media will start flowing again.

The signal measurements proportional to distance are presented in the Figure 6.2. On the OX axis, the crossed distance between APs is represented whilst on the OY axis, the measured Link Quality Indicator (LQI) is accounted. LQI was introduced by IEEE, initially in ZigBee networks; it is a value calculated based on the signal strength and the rate of errors received. While moving away from the AP1 and towards AP2, the quality1 decreases and quality2 increases. When a threshold value is reached, the Decision Manager starts counting down in how many consecutive scans the difference between received qualities is maintained greater than that threshold. When the counter reaches zero, after a certain number of scans in a row, the handover process starts, as described above. The counter is reset to initial value if the quality difference is not maintained until counter reaches zero. In this situation, the handover time measured from the ending of the old session to the beginning of the new session is at least 8 seconds, which is definitely noticeable for the user. Further improvement will be discussed in Scenario 2 and 3.

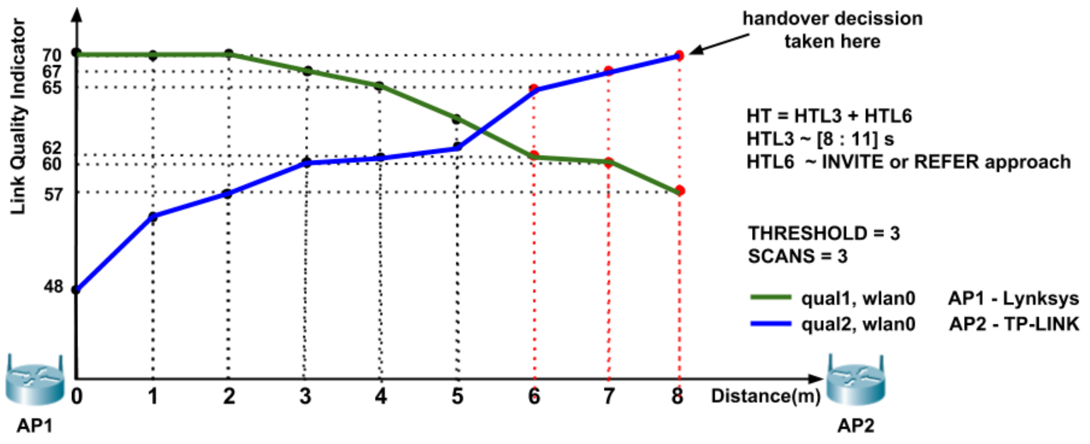


Figure 6.2: Scenario 1 measurements

6.2 Scenario 2

In this scenario, one of the SIP handover mechanisms will be performed and analyzed, namely SIP call transfer approach. The Mobile Node is equipped with two interfaces, therefore it can connect up to two APs at the same time. Additionally, the two interfaces can connect to the same AP at the same time.

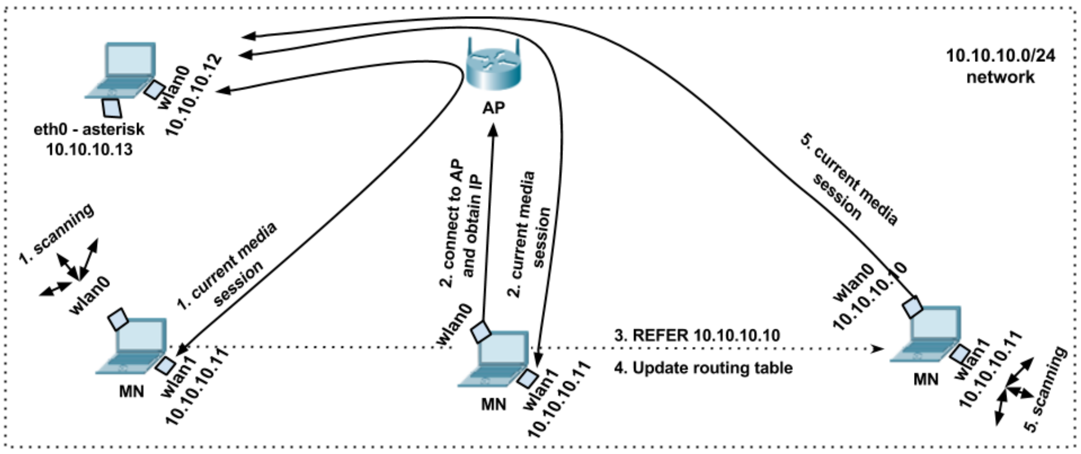


Figure 6.3: Scenario 2

In the beginning, both interfaces are scanning the medium for AP beacon frames. The fastest interface (here *wlan1*) will connect to the AP while scanning is still performed on the other. The Scan Manager sends this scanning data to the Decision Manager. Whilst moving away from the AP the signal quality received on *wlan1* constantly diminishes but the same signal is detected by *wlan0* with a higher quality. The decision algorithm is the same as presented above, namely if the difference between signal qualities is greater than a certain threshold in favor of the scanning interface, three consecutive scans, than Layer 3 handover performance begins. First of all, *wlan0* connects to the same AP and obtains IP address through DHCP. Notice that the session is still active and packets are sent on *wlan1* interface. After the network connection establishment, application handover is initiated by transferring current media session. Thus, Mobility Manager module sends *REFER* messages through *wlan1* containing the newly connected *wlan0* IP address, as explained in the above section 3.4. CN accepts the *REFER* request by sending a *202 Accepted* response and *INVITEs* MN to a new call, on *wlan0* interface, which leads to a new media session establishment and the end of old session. Figure 6.3 displays the above process.

The measurements of the of signal quality relative to the distance are shown in the Figure 6.4. Again, on the OX axis the distance from the AP is labeled, whilst on the OY axis the measured Link Quality Indicator (LQI) from the AP is represented. To be observed that signal qualities received by both interfaces are both decreasing but at different rates. As described in the previous scenario, when the threshold is attained and counter reaches a certain value, handover is performed.

The advantage brought by the second interface is a substantial Layer 3 handover time decrease, from 8 seconds to at most 0.5 second (i.e. the ammount of time needed to set a default route), because the current media session is maintained while the other interface (*wlan0* in this

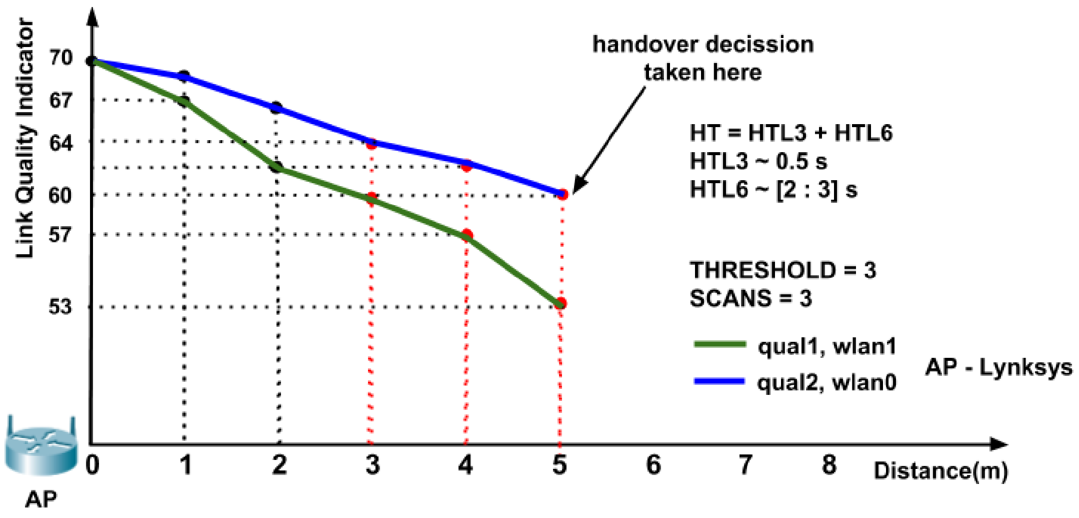


Figure 6.4: Scenario 2 measurements

scenario) connects to the AP and obtains IP. The Layer 6 handover time depends on Mobility Manager's answering time and restarting the media dependencies, which commonly counts from 2 up to 3 seconds. Nevertheless, the user is still capable of perceiving the handover because of the required call answer and restart of media dependencies. In the next setup one last improvement will be discussed and analyzed with the purpose of further decreasing the delay to almost zero, thus performing seamless handover.

6.3 Scenario 3

The last scenario is depicted in Figure 6.5. Mobile Node has two Wifi interfaces but no simultaneous connections to the same AP are permitted. This setup is the most representative for this thesis, as it performs seamless, soft, application layer handover using SIP.

In the first instance, no interfaces are connected. The initial connection of the MN is based on which interface performs scanning faster, here *wlan1* being the fastest one. Thus, *wlan1* is connected to the AP1 and can establish media session, while the other keeps searching for access points to connect. As the MN moves, Scan Manager detects, through *wlan0*, the presence of the AP2, and periodically sends the signal quality to the Decision Manager. The latter compares the received quality with the signal quality received from the currently connected AP1. If the difference between the connected AP1 quality and scanned AP2 quality is greater than a certain threshold, e.g. three consecutive scans, handover decision is made and Mobility Manager module is notified. Afterwards, the last module connects the *wlan0* interface to AP2 and obtains IP address through DHCP. Notice that session flow is still kept unchanged, through *wlan1*. Finally, the session handover is initiated by sending re-INVITE with the updated location, and the session is switched on *wlan0* whilst *wlan1* is scanning in the end. The re-INVITE message must contain the *VIA* and *Contact* SIP headers as well as the SDP's *c=* header updated to the new location.

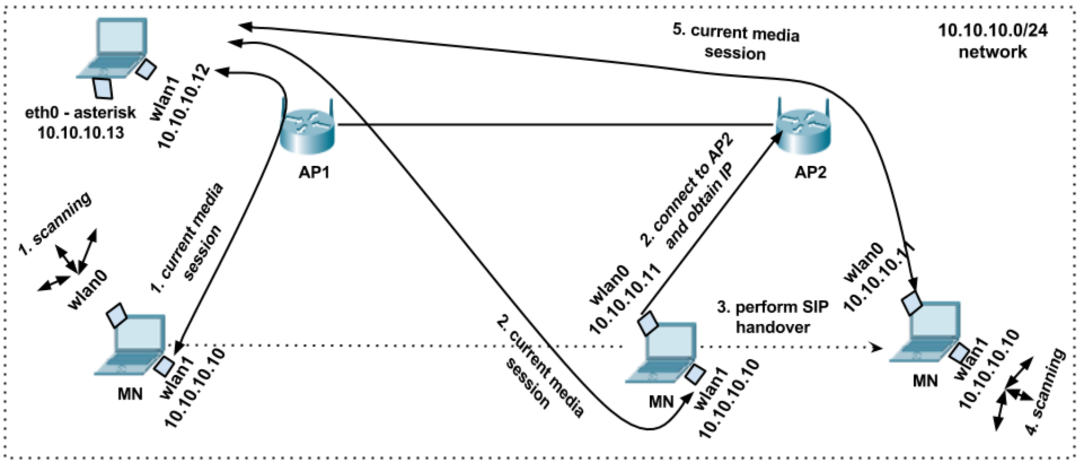


Figure 6.5: Scenario 3

In the following Figure 6.6, the variation of signal quality received from both APs is depicted, while MN is roaming. Like in previous scenarios, the distance between APs is represented on the OX coordinate and the Link Quality Indicator measurements are depicted on the OY coordinate. Notice the diminishing signal quality received from AP1 and the growing signal quality from AP2, as the MN moves towards the latter. The decision algorithm is the same as in previous scenarios but, the measured handover transition time is less than 0.5 seconds which is the best handover time obtained. This approach does not cause a great degree of discomfort and the user perceives almost no discontinuity in the communication flow. Another advantage of this handover method is the adjustment of the media parameters in the SDP message while switching between networks. For example, when moving away from the AP and switching to 3G techology, media attributes and codecs used will adapt such that the call is maintained at a reasonable quality. Vice versa, when switching back to the Wifi techonlogy, the quality of the media reverts to its initial value. This is the ultimate handover mechanism which can be implemented using SIP protocol.

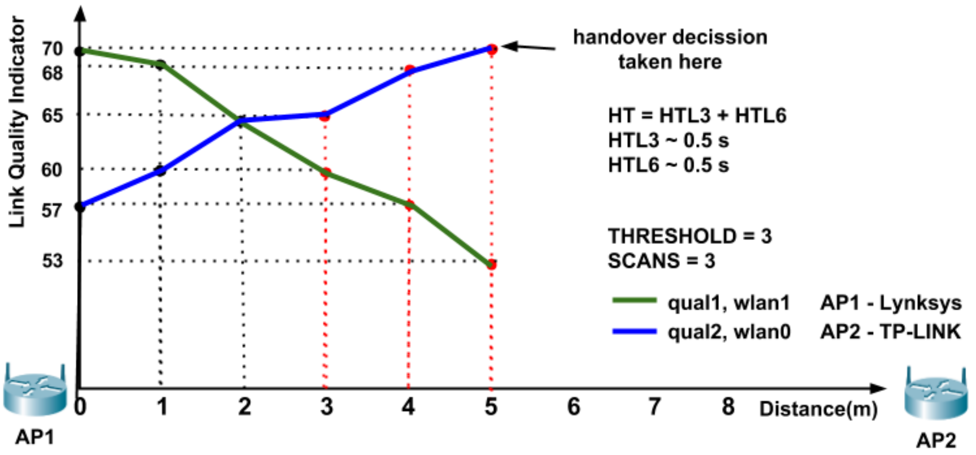


Figure 6.6: Scenario 3 measurements

As a conclusion, Figure 6.7 presents the handover delay comparison for each of the above scenarios. As illustrated, the measured minimum time needed to perform handover using only one wireless interface is 10 seconds. When an additional Wifi interface is used, the HTL3 drops down to around 0.5 seconds due to the ability to associate the second interface while mainatining session on the first one. Further Handover Time improvement is accomplished due to the SIP re-INVITE handover approach which lowers the delay under 1 second.

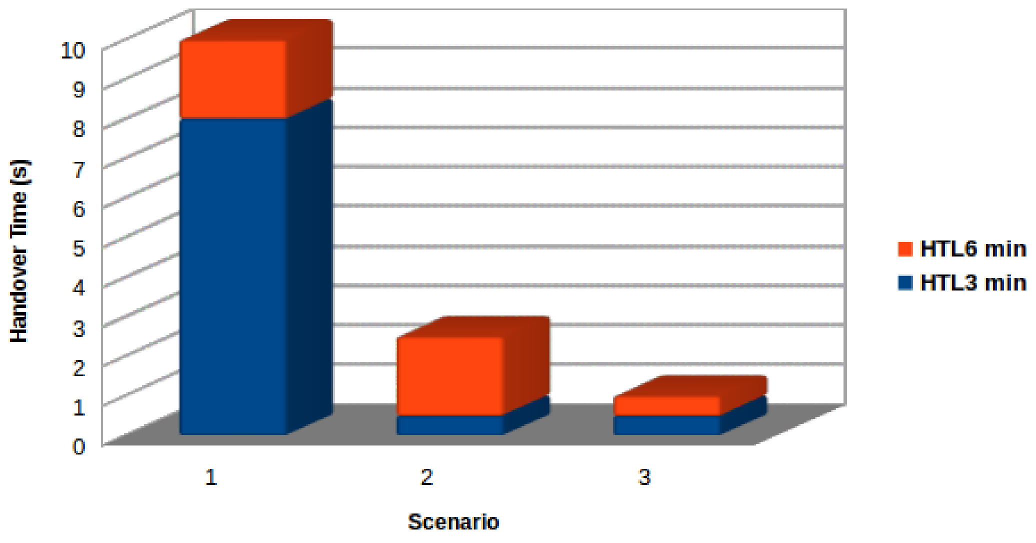


Figure 6.7: Scenario Handover Time comparison

Chapter 7

Conclusions and Future Work

In the above chapters, this document presented some of the most important issues that were encountered when trying to communicate over wireless environment, focusing on mobility issues, namely handover management. After explaining the problem, different types of handover were analyzed and compared. The purpose of this thesis to perform application layer handover and the advantage over other types of handover was justified. Real-life mobility scenarios were pictured, discussed and analyzed, providing some diagrams based on the measurements made.

Before presenting the above problem, an overview of protocols used throughout this document were discussed, including VoIP, SIP, SDP, RTP. Session Instantiation Protocol functionality and organization of SIP networks were comprehensively described because it is the signaling protocol that supports application layer handover.

Following, optimizations and future fork are being presented. In the near future, refactoring will be done, namely changing the imported library function prototypes and structures such that Mobility Manager module will not depend on the *iwlib* library anymore. Instead will be a standalone package which can be integrated with the SFLphone softphone.

The mobility solution provided in this thesis will be further improved to support vertical handover[10][11][25], namely switching between different wireless technologies while moving. Only Wifi to Wifi handovers are possible so far. Moreover, application layer handover will be improved such that media attributes will adjust according to the newly switched technology[23], as presented in the section 3.4. Moreover, the bandwidth metric will be taken into consideration while performing handover, which will be very useful when switching between different wireless technologies. For example switching from 3G to Wifi will result in higher bandwidth available, thus the quality of the media transmitted shall increase.

Ultimately, using latest WiSee[5] technology, calls will be triggered by specific gestures that map to certain users. A receiving antenna can detect wireless signal variations due to the interference between the signals and human body. The interference patterns are always the same no matter the gesturing person. Machine learning algorithms will be used to train the WiSee module which then will be able to lock to a certain user, remember the signal variation patterns for his peers, and then performing calls with respect to the SFLphone softphone interface.

Bibliography

[1] Gonzalo Camarillo. *SIP Demistified.* McGraw-Hill, 2002.

[2] Praphul Chandra and David Lide. *Wifi Telephony.* Elsevier Science and Technology, 2007.

[3] Chung-Ming Huang, Meng-Shu Chiang and Jin-Wei Lin. A link layer assisted fast handoff scheme using the alternative path approach. *Proceedings of Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference*, 1, 2006.

[4] Paolo Bellavista , Antonio Corradi and Luca Foschini. Context-aware handoff middleware for transparent service continuity in wireless networks. *Proceedings of Pervasive and Mobile Computing*, 3:439 – 466, 2007.

[5] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota and Shwetak Patel. Whole-home gesture recognition using wireless signals. *Proceedings of the 19th annual international conference on Mobile computing networking*, pages 27 – 38, 2013.

[6] Kashibuchi, Taleb, Jamalipour and Nemoto. A new smooth handoff scheme for mobile multimedia streaming using rtp dummy packets and rtcp explicit handoff notification. *Proceedings of Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, 4:2162 – 2167, 2006.

[7] Alan B. Johnston. *Understanding the Session Instantiation Protocol.* Artech House, 2009.

[8] J Kellokoski. Real-life multipath tcp based make-before-break vertical handover. *Proceedings of Computers and Communications (ISCC), 2013 IEEE Symposium*, 2013.

[9] Rajiv Kumar and Rajesh Khanna. Quality of service approach in umts-wlan handover. *Green Computing and Communications (GreenCom), 2012 IEEE International Conference*, pages 692 – 695, 2012.

[10] Johann MĂArquez-Barja , Carlos T. Calafate , Juan-Carlos Cano Pietro Manzoni. An overview of vertical handover techniques: Algorithms, protocols and tools. *Proceedings of Computer Communications*, pages 985 – 997, 2011.

[11] Stefano Salsano , Luca Veltri , Gianluca Martiniello and Andrea Polidoro. Seamless vertical handover of voip calls based on sip session border controllers. *Proceedings of Communications, 2006. ICC '06. IEEE International Conference*, pages 2040 – 2047, 2006.

[12] K.Durairaj , S.Jeevanandham , M.Rajkumar and R.Selvakumar. Multi interface mobility management with sip and mip (to avoid handoff delay). *International Journal of Computer Science and Mobile Computing*, pages 184 – 191, 2014.

[13] Abdul Nasir and Mah-Rukh. Internet mobility using sip and mip. *Information Technology: New Generations, 2006. ITNG 2006. Third International Conference*, pages 334 – 339, 2006.

[14] Asterisk Home Page. Retrieved July, 2014 from `http://www.asterisk.org/`.

[15] Google Hangouts Home Page. Retrieved July, 2014 from `https://www.google.com/hangouts/`.

[16] Google Voice Home Page. Retrieved July, 2014 from `https://www.google.com/chat/voice/`.

[17] Linphone Home Page. Retrieved July, 2014 from `http://www.linphone.org/`.

[18] SFLphone Home Page. Retrieved July, 2014 from `http://sflphone.org/`.

[19] Skype Home Page. Retrieved July, 2014 from `http://www.skype.com/en/`.

[20] Maximilian Riegel and Dr. Michael TĂŒxen. Mobile sctp - transport layer mobility management for the internet, 2002.

[21] Carlos Garcia Rubio. Transport-layer mobility using msctp, 2009.

[22] Esa Piri , Tiia Sutinen and Janne Vehkaper. Cross-layer architecture for adaptive real-time multimedia in heterogeneous network environment. *Proceedings of Wireless Conference, 2009. EW 2009. European*, pages 293 – 297, 2009.

[23] Peyman TalebiFard and Victor C.M. Leung. Efficient multimedia call delivery over ip-based heterogeneous wireless access networks. *Proceedings of Pervasive and Mobile Computing*, pages 439 – 466, 2008.

[24] Ted Wallingford. *Switching to VoIP*. O'Reilly Media, 2005.

[25] Xiaohuan Yana , Y. Ahmet ĆekercioÄlua and Sathya Narayananb. Integrated mip-sip for ims-based wimax-umts vertical handover. *Proceedings of Telecommunications (ICT), 2012 19th International Conference*, pages 1 – 6, 2012.