MASTER THESIS

TITLE : Medium and mobility behaviour insertion for 802.11 emulated networks

DEGREE: Degree in Telecommunications Engineering and Master in Science in Telecommunication Engineering & Management

AUTHOR: Alberto Martínez Illán

DIRECTOR: PhD. Juan López Rubio

CO-DIRECTOR: MA. Marc Antoni Bajet Mena

DATE: September 27, 2013

Títol : Inserció dels factors de medi i mobilitat a xarxes emulades 802.11

Autor: Alberto Martínez Illán

Director: PhD. Juan López Rubio

Co-director: MA. Marc Antoni Bajet Mena

Data: 27 de setembre de 2013

Resum

Wmediumd és una aplicació basada en el llenguatge de programació C i que va ser desemvolupada per una consultora d'Estats Units anomenada Cozybit.

Wmediumd va ser creada per dur a terme emulació del medi sense fils a xarxes emulades Linux. Aquesta aplicació permet als programadors de controladors 802.11 crear un entorn de desenvolupament/prova amb un sol ordinador, estalviant temps i equips.

La versió actual d'aquesta aplicació només emula el comportament del medi mitjançant la pèrdua de paquets basant-se en probabilitats, sense patrons de mobilitat entre ràdios emulades i sense tindre en compte les interferències del medi.

És interessant per Cozybit i per altres desenvolupadors crear una extensió de Wmediumd introduint el comportament del medi i de mobilitat de les ràdios emulades. D'aquesta manera es millorarien els test-beds de 802.11 per als desenvolupadors de controladors.

Title : Medium and mobility behaviour insertion for 802.11 emulated networks

Author: Alberto Martínez Illán

Director: PhD. Juan López Rubio

Co-director: MA. Marc Antoni Bajet Mena

Date: September 27, 2013

Overview

Wmediumd is an application based on the C programming language and was developed by a United States company called Cozybit.

Wmediumd was created to perform emulation of the wireless environment on emulated networks created on Linux OS. This application allows programmers of 802.11 drivers to create an environment for development/testing with a single computer, saving time and hardware.

Current version of this application only emulates the behaviour of the medium creating frame losses depending on probabilities, but not using mobility patterns between emulated radios and also without taking into account the interferences the environment has.

It is interesting for Cozybit and other developers to create an extension of Wmediumd to introduce environmental behaviour and mobility on the emulated radios. Introducing this extension will improve the test-beds for the 802.11 driver developers.

A mis padres Enrique y Gisela por su apoyo constante.

A María porque siempre está a mi lado dispuesta a ayudarme.

A todo el sistema educativo por permitirme tener unos estudios.

A los compañeros y amigos que me han ayudado, en especial
a Juan y Javier López y a Marc Fernandez.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Acronym	**W**hat (it) **S**tands **F**or
API	**A**pplication **P**latform **I**nterface
CRC	**C**yclic **R**edundancy **C**heck
CSMA/CA	**C**arrier **S**ense **M**ultiple **A**ccess With **C**ollision **A**voidance
DSL	**D**igital **S**ubscriber **L**ine
EIRP	**E**quivalent **I**sotropically **R**adiated **P**ower
FP	**F**ading **P**robability
FIC	**F**ading **I**ntensity **C**onstant
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IP	**I**nternet **P**rotocol
IPC	**I**nter **P**rocess **C**ommunication
LPC	**L**oss **P**robability **C**onstant
MAC	**M**edia **A**ccess **C**ontrol
MGEN	**M**ulti **G**enerator
NIC	**N**etwork **I**nterface **C**ard
OS	**O**perative **S**ystem
PS	**P**ower **S**ave
PC	**P**ersonal **C**omputer
PCAP	**P**acket **C**apture
PID	**P**rocess **I**dentifier
SNR	**S**ignal to **N**oise **R**atio
SSID	**S**ervice **S**et **I**dentification
SVN	**S**ub**v**ersio**n**
TCP	**T**ransmission **C**ontrol **P**rotocol
UML	**U**nified **M**odeling **L**anguage
UTS	**U**nix **T**ime **S**haring
WiFi	**W**ireless **F**idelity
WLAN	**W**ireless **L**ocal **A**rea **N**etwork

INTRODUCTION

With the increase of the process capacity on mobile devices, the decrease of their costs and the current investments on digital communications, 802.11 networks starts to get relevance in many scenarios of our society.

Some of these scenarios are:

- Mobile communications.
- Military communications.
- Energetic meters now being deployed on residences.
- Internet access in poor countries.
- ...

IEEE 802.11 is a set of protocols used for wireless local area networks (WLAN) and nowadays it's the most used protocol.

The wireless networks have an exponentially growth during lasts years as seen in Table 2 which shows the global WLAN chips shipments.

Table 2: WLAN global shipments [18].

Year	WLAN chip shipments (Millions of units)
2009	100
2010	400
2011	800
2012	1300
2013	1700
2014	2200

Cozybit is a consultant company from United States located in San Francisco, it's costumers include companies like: Google, Qualcomm, Sony, Samsung, etc.. The working areas of the company are: distributed networks, wireless communications, the development of the Linux kernel and embedded devices. This company created the first open source 802.11s [1] implementation.

[1] IEEE 802.11s is an IEEE 802.11 amendment for mesh networking, defining how wireless devices can interconnect to create a WLAN mesh network, which may be used for static topologies and ad hoc networks [23].

In order to improve the 802.11 driver a real scenario can be deployed, but this is highly resource/time consuming. Using the Linux tools can be created an emulated environment ready to test a driver as it would be done with real radios.

The problem is that created links will just forward all frames from one interface to the others (as it would be done with a wireless medium), but with the difference that real wireless medium has interferences, multipath and other problems. These factors create transmission errors while reading the frames that the wireless network interface receives, thing that doesn't happen without a wireless medium emulator.

For that purpose Cozybit created and application called Wmediumd that communicates it self with the kernel to create probabilistic errors and to emulate with this behaviour the wireless medium. The problem of that is that the wireless medium has different errors behaviour, in some way, related to the mobility of the stations, which is not taken into account in the current behaviour of Wmediumd.

This project will study and perform an extension of Wmediumd to add capabilities to emulate medium and mobility behaviour to it.

CHAPTER 1. OBJECTIVES

Current 802.11 development environment can be improved. This is the aim of this project which will extend the wireless medium emulator Wmediumd to integrate on it the medium and the mobility behaviour.

To do that next objective list is considered:

- Learn how Linux namespaces work in order to insulate operative system resources.
- Understand how radio network interfaces are emulated on Linux. Load and test the specific kernel module that performs emulation, this module is *mac80211_hwsim*.
- To test network namespaces with emulated radio network interfaces and configure an test a topology.
- Understand and test current wireless emulator Wmediumd. This is a relevant point, because the knowledge acquired on current solution will help to find where the extension code has to be placed.
- Study the source code of Wmediumd and see how the execution flows and how the simulator has communication with the kernel in order to produce or not transmission errors.
- Study and search characterization models of wireless medium depending on interferences, mobility and multipath effects.
- Create and code studied behaviours in the emulator extension.
- Insert and test the mobility behaviour in Wmediumd to test the expected behaviour. Dumping PCAP [1] files from the interfaces and reading them with Wireshark in order to check the frames communication.
- Extract conclusions of the implemented extension and its fidelity.
- Perform a study of possible environmental impact associated to this project.

This objectives follow the normal procedures on the software development process and will guarantee a good study and deployment of this extension.

[1] In the field of computer network administration, pcap (packet capture) consists of an application programming interface (API) for capturing network traffic. Pcap also support saving captured packets to a file, and reading files containing saved packets [3].

CHAPTER 2. NAMESPACES, HARDWARE SIMULATION AND OLD WMEDIUMD

Chapter 2 will explain the main concepts on hardware simulation networks for Linux devices. To do that this chapter will first introduce main scenario resources to then integrate them into the used development environment. Using this approach the reader will understand main aspects before start to naming them in the rest of this document.

2.1. Linux Namespaces

In Linux there are many kernel resources that are globally shared, one of them is the network interfaces. In order to perform the normal operations with a Linux OS this is fine, but Kernel developers may want to perform testing on their network interface driver improvements in an easy way. Namespaces solve that bringing the Linux capability to isolate the network stack. This is similar to the chroot jails that some applications use.

Linux implements six different types of namespaces, the relevant for this project is the network namespace:

- User namespaces (kernel v. > 2.6.23). Isolate the user and group ID number spaces. A process's user and group IDs can be different inside and outside a user namespace.
- Network namespaces (kernel v. > 2.6.24). Isolated network resources. Each network space can have its own (virtual) network device and its own applications that bind to the namespace port number space; suitable routing rules in the host system can direct network packets to the network device associated with a specific container. For example, it is possible to have multiple web servers on the same host, with each server bound to port 80 in its network namespace.
- UTS namespaces (kernel v. > 2.6.19). Isolate two system identifiers: *nodename* and *domainname* returned by the *uname()* system call.
- Mount namespaces (kernel v. > 2.6.19). Isolate the filesystem mount points. One use is to emulate the chroot jails.
- IPC namespaces (kernel v. > 2.6.19). Isolate interprocess communication (IPC) resources

- PID namespaces (kernel v. > 2.6.24). Isolate the process ID number space. In other words, processes in different PID namespaces can have the same PID.

Figure 2.1 shows a Linux OS network namespaces scheme where each network space is isolated from the other (taking into account that here doesn't exist real links or radio interfaces):

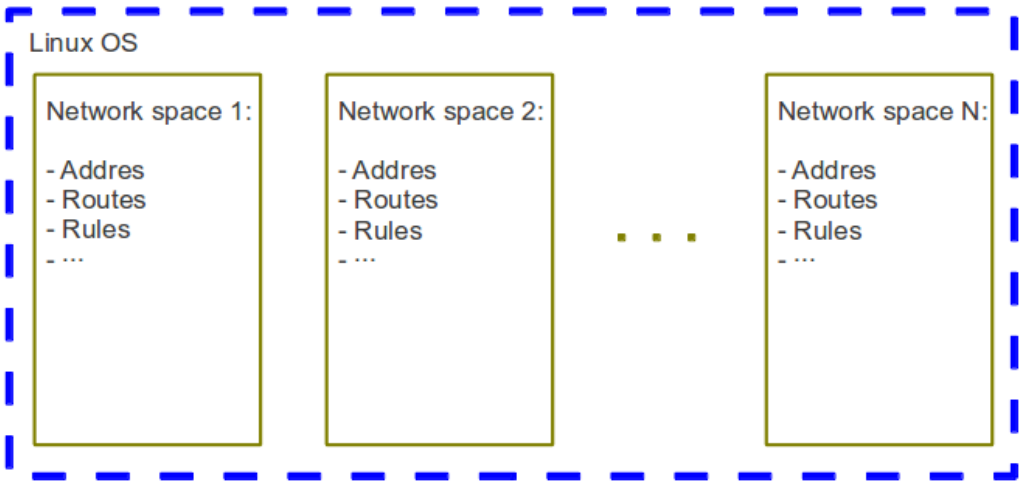

Figure 2.1: Linux OS network namespaces.

Using namespaces is possible to create several network spaces independents ones from others. In this way can be created many network spaces with independent network stacks. [25] [26].

2.2. Hardware simulation with mac80211_hwsim

Having isolated network namespaces is not enough, is necessary to create network interfaces and a medium to connect them. This is provided by the mac80211_hwsim kernel module which is a software emulator of 802.11 radios for mac80211 [1].

Mac80211_hwsim allow to simulate any number of IEEE 802.11 radios and allow to test most of the functionality for mac80211 framework and the network namespace tools in a way that matches very closely with the real world usage on WLAN hardware. Mac80211 framework just see mac80211_hwsim as another hardware drivers so there is no previous setting of the mac80211 framework in order to work with mac80211_hwsim in testing environments.

[1] Mac80211 is the framework that is mostly used today to write drivers for SoftMAC wireless devices, the more used at this times. SoftMAC devices permit a finer control of the hardware, it allows for 802.11 frame management to be done by software, for both parsing and generation of 802.11 wireless frames.

The principal achievement of mac80211_hwsim is to create an easy test-bed scenario for the wireless driver developers, because the simulated radios doesn't have the hardware resources needs that real hardware implies. With it, it's very simple to create a scenario and to recreate it when its need, it allows to full automate the scenario creation using shell scripts. And as is always emulated by software there is no possibility of infringement of regulatory country rules or real interference creation that would disturb other communications.

The software works tracking the channel of each virtual radio and copying all transmitted frames to each one of the currently enabled virtual radios that have set the same channel for transmitting. The real encryption layer is also used in here to allow complete testing as would be done with real interfaces.

Figure 2.2 shows how the mac802_hwsim works.

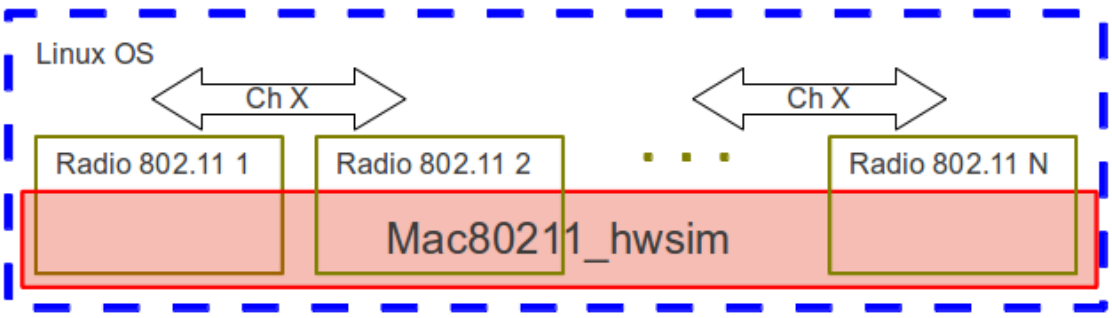

Figure 2.2: Mac80211_hwsim kernel module.

The mac802_hwsim has also an argument that is used when the module is loaded, this argument is the number of radios to simulate (by default 2). This allows the configuration of very large scale tests (hundreds of stations) or simple ones (e.g., a single access point with a station) [4].

2.3. Resource integration for development environment creation

This section will explain how all the theory explained in previous sections of this chapter are integrated and configured together to have the development environment.

The scenario has next actors launched each one of them with a terminal on same PC:

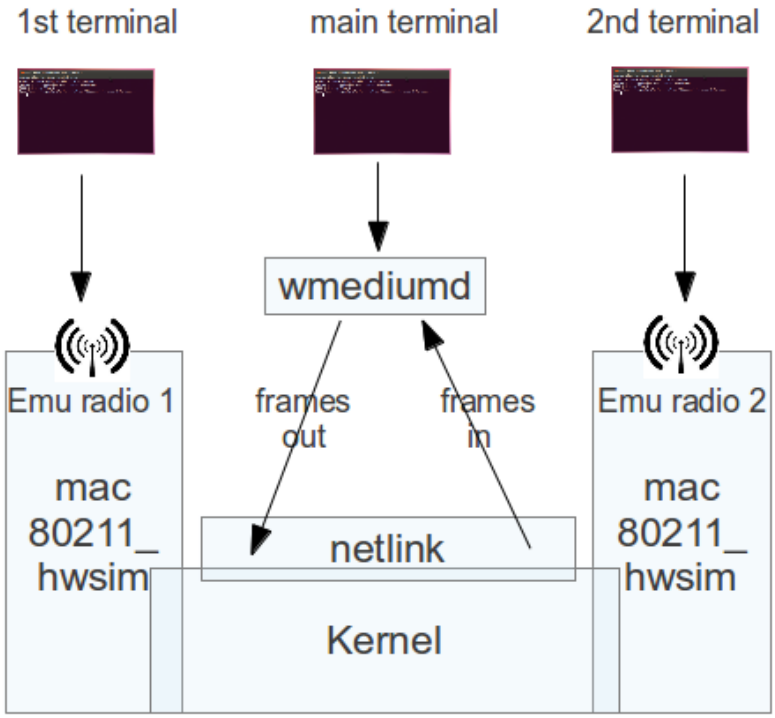

Figure 2.3: Scenario actors.

Each terminal runs a different Bash[2] script that has a menu to trigger actions. The actions will be performed in different shells to be able to create different network namespaces which will be assigned to it's correspondent shell. A command launched in a terminal will affect only to its network namespace. To each network namespace will be assigned its emulated radio.

[2]Bash is a Unix shell that can run commands directly over it or using scripts that can be manually or automatically launched.

In next Figure it can be appreciated the menus to trigger actions on the terminals:

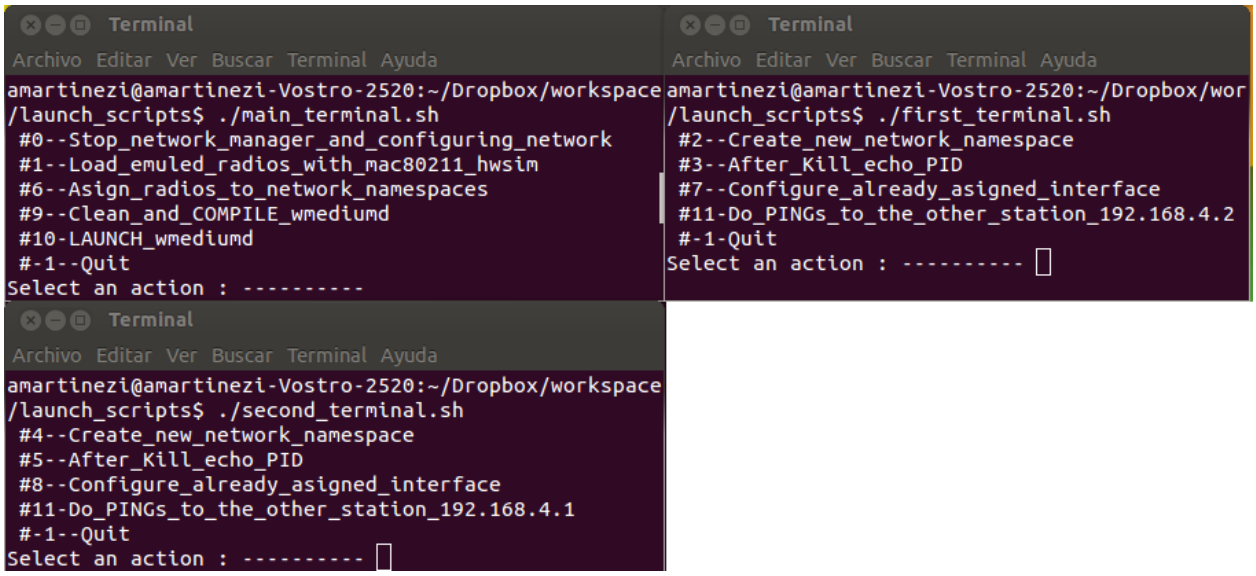

Figure 2.4: Terminal menus. Left-top terminal is main terminal, right-top terminal is 1st terminal and down-left it's the 2nd terminal.

In the options is possible to see numbers that follow a sequence from 0 to 11 jumping from one terminal to the others. This is done because the procedure needs this order to assign resources from main terminal and main namespace to the others and data information obtained from one terminal is need into the others to can configure the scenario.

2.3.1. Mac80211_hwsim and Wmediumd settings: main terminal

In next subsections follow the enumeration order even if you have to jump from one subsection to the other.

Main terminal performs next actions:

0. Stop network manager and configuring internet network (in this case to our wiFi DSL default gateway)

```
sudo /etc/init.d/network-manager stop
sudo ifconfig wlan0 up
sudo iwconfig wlan0 essid $WIFI_SSID
sleep 3
sudo iwconfig wlan0 essid $WIFI_SSID key $WIFI_KEY
sleep 3
sudo iwconfig wlan0 essid $WIFI_SSID key $WIFI_KEY
sudo dhclient wlan0
```

```
sleep 3
ping -c 5 8.8.8.8

```

1. Load emuled radios with mac80211_hwsim. Module is removed if it's already configured to load a newly one with two emulated radios. Then (3) a search of the name of this newly virtual radios.

```
sudo rmmod mac80211_hwsim
sudo modprobe mac80211_hwsim radios=2
sudo find /sys/kernel/debug/ieee80211 -name hwsim | cut -d/ -f 6 | sort

```

6. Assign radios to network namespaces using obtained PID. The script will ask the number of physics interface and the PIDs of both network namespaces (1,2,4), it will perform the assignation (5,6).

```
read -e -p 'Number of first phy interface: ' -a phy1
read -e -p 'PID of first network namespace: ' -a pid1
phy2=$(($phy1 + 1))
read -e -p 'PID of second network namespace: ' -a pid2
sudo iw phy phy$phy1 set netns $pid1
sudo iw phy phy$phy2 set netns $pid2

```

9. Clean and compile wmediumd source code.

```
echo 'START CLEAN'
make clean -C $PATH_WMEDIUMD
echo 'END CLEAN'
echo '============================================================'
echo 'START COMPILE'
make -C $PATH_WMEDIUMD
echo 'END COMPILE'

```

10. Launch wmediumd loading the probability configuration file.

```
sudo $PATH_WMEDIUMD/wmediumd -c $PATH_WMEDIUMD/test.cfg

```

2.3.2. Network namespace 1: 1st terminal

2. Create new network namespace. Script will die but it's need the new PID of the namespace so it's need to call again the script using next command ./first_terminal.sh

```
sudo unshare -n bash

```

3. Echo the PID of the namespace

```
echo '##########'
echo $$
echo '##########'

```

7. Configure already assigned virtual radio (3). Assign IP to the interface (5) and set a link to join a network (6,7). Start pinging the other node (10).

```
read -e -p 'Number of first phy interface: ' -a phy1
sudo iwconfig
sudo iw phy phy$phy1 interface add mesh1 type mesh
sudo ifconfig
sudo ip address add dev mesh1 192.168.4.1/24
sudo ip link set mesh1 up
sudo iw dev mesh1 mesh join bazooka
sudo ifconfig lo up
sudo ifconfig
sudo ping -c 20 192.168.4.2
sudo ifconfig

```

11. Doing pings to 192.168.4.2 to force some traffic into the network.

```
ping 192.168.4.2

```

2.3.3. Network namespace 2: 2nd terminal

4. Create new network namespace. Script will day call it again using next step. ./second_terminal.sh

```
sudo unshare -n bash

```

5. Show PID of the namespace

```
echo '##########'
echo $$
echo '##########'

```

8. Configure already assigned virtual radio (3). Assign IP to the interface (4) and set a link to join a network (5,6). Start pinging the other node (8).

```
read -e -p 'Number of second phy interface: ' -a phy2
iwconfig
sudo iw phy phy$phy2 interface add mesh2 type mesh
sudo ip address add dev mesh2 192.168.4.2/24
sudo ip link set mesh2 up
sudo iw dev mesh2 mesh join bazooka
sudo ifconfig lo up
sudo ping -c 10 192.168.4.1
sudo ifconfig

```

11. Doing pings to 192.168.4.1 to force some traffic into the network.

```
ping 192.168.4.1

```

With all this settings the development environment is configured and ready to perform the extension and the suitable testing of it. The scripts allow to rapidly configure all environment in exactly same manner always, which will help to reproduce the results independently on when or where they are performed. Compilation of the source code and simple testing traffic generation is also included.

2.4. Old Wmediumd inspection

The problem that has mac80211_hwsim is that it doesn't implement medium emulation, it just forwards frames from one radio to others into same channel without interferences. So there is a perfect communication that it really doesn't exist into the the real wireless medium.

To solve that the company Cozybit implemented a Wireless medium emulator called Wmediumd. This software handle frames sent from the kernel space to the user space. It basically load a probabilistic losses behaviour which will apply to each frame coming from mac80211_hwsim radios [5].

In this section Wmediumd is dissected to know how it works and operates processing frames coming from the kernel via Netlink[3].

2.4.1. Communications with the Kernel via Netlink

To start it is need an explanation of how an user-space application as Wmediumd can have communication with the kernel-space. This is a need because all network stack and the radio emulations are performed on kernel-space and is necessary to make a kind of "*man in middle*" to get the frames from sender radio, create errors or not into these frames and send them back or not to the receiver.

Netlink was designed to transfer network information between the user space processes and the Linux kernel-space.

Figure 2.5 is a scheme of the Netlink architecture.

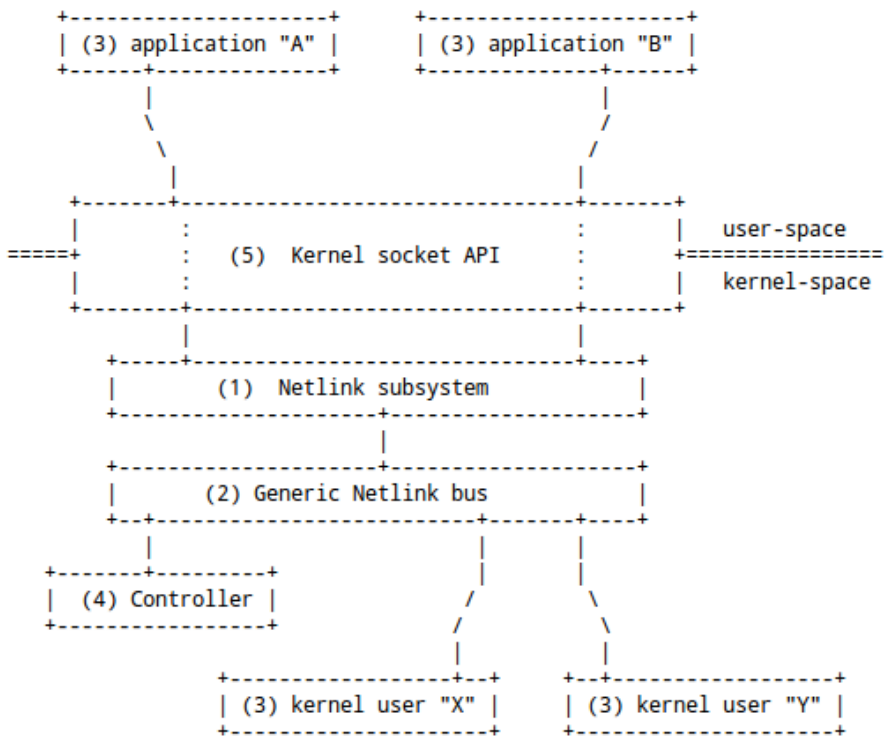

Figure 2.5: Netlink communication scheme.

[3]Netlink is a socket family used for Inter Process Communication(IPC) between the kernel and user space processes, as well as between user processes (e.g. Unix domain sockets) or a mixture of both types[6].

Next enumeration explain each part of Figure 2.5:

1. Netlink subsystem transport layer.
2. Bus implemented inside the kernel, but which is available to userspace using the socket API and inside the kernel with the normal Netlink API.
3. Netlink users.
4. Netlink controller that performs the communication functions and necessary calls.
5. The kernel socket API which attends the applications calls triggering the controller functions.

Netlink consists of a standard socket-based interface for user space processes and a kernel API for kernel modules. It's a more flexible alternative to the ioctl communication system [6][7].

2.4.2. Wmediumd application folders and files

This subsection will explain Wmediumd folder and main file structure.

- rawsocket/ : Raw socket based ping-pong application to test stability and wmediumd.
  - client.c : Client socket app source code.
  - server.c : Server socket app source code.
- wmediumd/ : Application to emulate wireless medium.
  - config.c : Functions that are related with load a configuration file or write a sample configuration one.
  - config.h : Headers file from config.h
  - ieee80211.h : Headers file from ieee80211.h
  - mac_address.c : Parser from string to mac address struct.
  - mac_address.h : Headers file from mac_address.h
  - probability.c : Probability related functions.
  - probability.h : Headers file from probability.h
  - wmediumd.c : Main Wmediumd file with the core functionality.

    - wmediumd.h : Headers file from wmediumd.h

- wconfig/ : Java app to create configuration files for wmediumd.
    - images/ : Application images folder.
    - lib/ : Java libraries.
    - src/ : Source code java files.
    - manifest/ : Java manifest folder.
    - wconfig.jar : Java app to execute

This file structure is relevant because on next subsection the reader will be able to comprehend the flow and the file and code location where an extension will be placed in order to add the mobility behaviour.

2.4.3. Wmediumd general application flow with UML activity diagram

This subsection will explain the flow of function calls Wmediumd has, to perform that the next Figure is used:

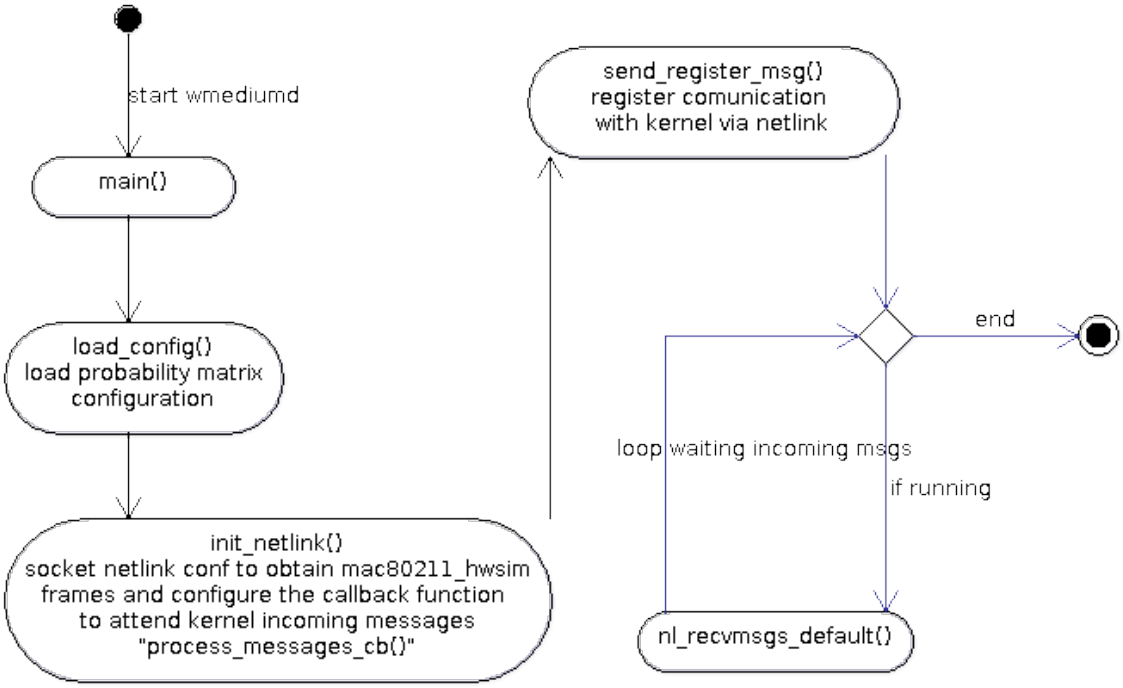

Figure 2.6: Wmediumd program start calls flow.

1. wmediumd.c - main(): function wich trigger the load_config() function.
2. config.c - load_config(): this function loads the configuration file with the associated probability matrix which depends on the frames source, destiny and the link speed.
3. wmediumd.c - init_netlink(): after the load_config() this function configures the netlink framework to obtain the mac80211_hwsim frames. It also configure the callback function to attend the kernel messages which is process_messages_cb().
4. wmediumd.c - send_register_msg(): register the communication with kernel via netlink.
5. wmediumd.c - nl_recvmsgs_default(): listen incoming messages in loop while *running* integer variable remains on 1.

Figure 2.7 has the second part of the process where the callback receives the frames from the Kernel.

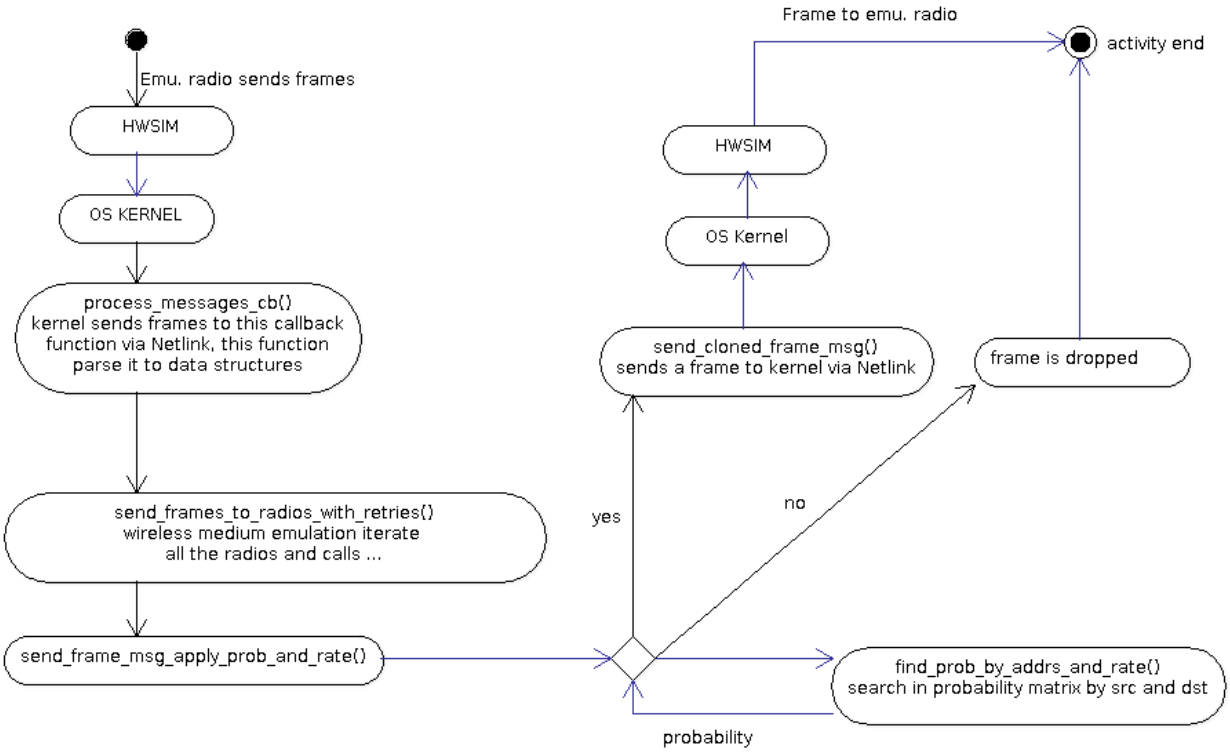

Figure 2.7: Wmediumd callback function flow.

6. wmediumd.c - process_messages_cb(): callback function that receives messages with frames from the kernel via Netlink, this function parse them to data structures to then call send_frames_with_retries().

7. wmediumd.c - send_frames_with_retries(): It iterates all the radios and try to send a copy of the frame to each interface. To do that it calls a function that will apply the probabilistic behaviour which is send_frame_msg_apply_prob_and_rate().

8. wmediumd.c - send_frame_msg_apply_prob_and_rate() : Send or not a frame to a radio depending on the probabilistic behaviour defined in the probability matrix. If yes, send_cloned_frame_msg() is called.

9. wmediumd.c - send_cloned_frame_msg() : Send a frame to the kernel via established Netlink communication.

These functions are located mainly into the wmediumd.c file but there are specific functions or data structures that are called from these functions that are defined in the other files.

Taking into account the described flow is possible to determine where the extension location can reside. The function will create errors depending on the position that the stations have, this will increase the probability of error depending on the distance between the sender and receiver.

The function will be called inside the "*send_frame_msg_apply_prob_and_rate()*" function where the transmission error is introduced or not.

CHAPTER 3. MEDIUM CHARACTERIZATION AND MOBILITY

On the Chapter 2 the development scenario and it's elements is introduced, on it, it's also studied how Wmediumd works and where to place the mobility behaviour inside the code.

This chapter will face how the mobility factor have to be characterized. To do that a study of how the mobility affect to the frame error probability will be performed.

3.1. Medium

As it is known the wireless medium has many interferences. The parameter that objectively measures the problem of interferences is the relation of the power transmitted signal and the interferences, this is the SNR[1].

3.1.1. Interferences

The possible interference signals in the scenario are the external ones, this is because the stations from the scenario are 802.11 which use CSMA/CA [2] that will avoid collisions.

The problem is that the interference power is very dependent on the scenario, it's very different to transmit in a city than in the middle of a desert. And there are empiric models that deals with this medium changes, one example is the *Okomura-Hata* model. The problem of this model is that it only takes into account frequencies bellow 1.5GHz, not used into 802.11 which use 2.4, 3.6, 5 and 60 GHz frequency bands. The *COST 231 Walfish-Ikegami* model it also models urban environments but as the *Okomura* or the *Hata* it can just be used for frequency carriers bellow 2GHz.

For that reason maybe a good approach to simulate the medium would be introducing a constant coefficient applied depending on the level of interference to convert the medium model in a more or less interfering.

[1] Signal-to-noise ratio (often abbreviated SNR or S/N) is a measure used in science and engineering that compares the level of a desired signal to the level of background noise. $\mathrm{SNR} = \frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}}$ [9]

[2] Carrier Sense Multiple Access with Collision Avoidance, in computer networking, is a network multiple access method in which carrier sensing is used, but nodes attempt to avoid collisions by transmitting only when the channel is sensed to be "idle" [10] .

3.1.2. Fading

Fading its also known in wireless transmission world. Most of the times it's related to the movement of stations or objects inside the communication scenario. This phenomenon usually happens when two signal paths create a deviation on signal time arrivals to receiver, this can result in either constructive or destructive interference. When paths addition result in a destructive interference, signal power seen by the receiver antenna tends to zero and the communication is interrupted while the phenomenon occurs.

This phenomenon is usually modelled as a random process and using some intensity, for that reason a random variable applied or not a constant fading intensity value in our scenario would be a good approach.

3.2. Mobility

The other factor that will modify the SNR will be the mobility. This mobility will increase or decrease the distance between the emitter and the receiver.

There are some equations that can help to model that behaviour.

3.2.1. Link budget with Free Space Transmission Equation

Equation 3.1 is the *Free Space Transmission Equation* included in a *Link Budget*[3] and it shows the power received in W. The equation takes the transmitted power by the emitter, then it applies the antenna gain of the emitter, the antenna gain of the receiver and it discounts the loss of power associated to the distance and the wavelength used (carrier frequency).

$$Pr[W] = P_T[W] \cdot G_E[W] \cdot G_R[W] \cdot \frac{1}{\left(\frac{4 \cdot \Pi \cdot d[m]}{\lambda}\right)^2} \tag{3.1}$$

The equation can be expressed in logarithmic scale using dB and dBm (as an addition of terms) and in linear (as multiplications), expressing it in linear will allow an easy insulation of terms.

[3] A link budget is the accounting of all of the gains and losses from the transmitter, through the medium to the receiver in a telecommunication system [13].

3.2.2. Maximum radio link distance

Knowing the lower power (Pr_{nom}) that a 802.11 radio needs to can demodulate a incoming signal, the gain of the receiver antenna (G_R), the transmitted power (P_T) and the gain of transmission antenna (G_T) it can be obtained the maximum distance that the communication can have insulating from the equation of the distance $d_{max}[m]$. The transmitted power and the gain of transmission antenna are the $EIRP$ [4], depending the country law it would have to be taken into account if the simulation wants to be done following EIRP country maximum values.

$$d_{max}[m] = \frac{|\sqrt{\frac{P_T \cdot G_T \cdot G_R}{Pr_{min}}}|}{\frac{4 \cdot \Pi}{\lambda}} \tag{3.2}$$

The more sensitivity the receiver antenna has, the radio link will be able to have a bigger distance from antenna to antenna, same happens with the transmitted power P_T or the gain of both antennas G_T and G_R , carrier frequency also changes the result.

3.2.3. Loss probability

All distances bigger than d_{max} will make a frame loss probability of 100% and the ones lower than it a lower loss probability.

For that, a step of $\frac{d_{current}[m]}{d_{max}[m]}$ will be done getting a loss probability Equation:

$$Loss\ \ probability\ \ [per\ \ unit] = \frac{d_{current}[m]}{d_{max}[m]} \tag{3.3}$$

In that manner it can be model the $Loss\ \ probability\ \ [per\ \ unit]$ using the *Free Space model* linked with a *Link Budged*.

[4] EIRP, Equivalent isotropically radiated power is the amount of power that a theoretical isotropic antenna would emit to produce the peak power density observed in the direction of maximum antenna gain [12]. $EIRP = P_T[W] \cdot G_E[W]$

3.3. Mobility and medium emulation coupling

In order to couple mobility and medium models the Equation 3.3 will be tuned to introduce the parameters explained in Section 3.1.:

- Constant coefficient applied depending on the level of interference to convert our medium model in a more or less interfering one. Addition of loss probability with a constant parameter can model that, this is the meaning of LPC .
- Factor to introduce probabilistic Fadings with a defined intensity. It's modelled with two constants, the fading probability (FP) and the Fading Intensity Constant that would bee also model with a loss probability constant (FIC).

  FP - is a function that takes values [0, 1] depending on the fading probability enabling or disabling the Fading effect.

This Equation would be:

$$Loss\ \ probability\ \ [per\ \ unit] = \frac{d_{current}[m]}{d_{max}[m]} + LPC + FP(fading\ \ probability) \cdot FIC \tag{3.4}$$

In that way mobility and medium can be emulated with tuning capabilities. Results can be bigger than 1 but then a loss probability of 100% will be apply for that frame.

3.4. Model adjustments

The theoretical model explained in Section 3.3. has a deviation with the real behaviour. This is because the loss probability is assigned linearly depending the relation of d_{max} and $d_{current}$, but the real behaviour applies less grow of attenuation on first meters compared with the other distances. For that reason the loss probability should be applied following the same rule.

Next Figure represents the linear path loss that a radio link has depending the distance that follows a power law (blue color line) and an approximated linear model (purple color lines).

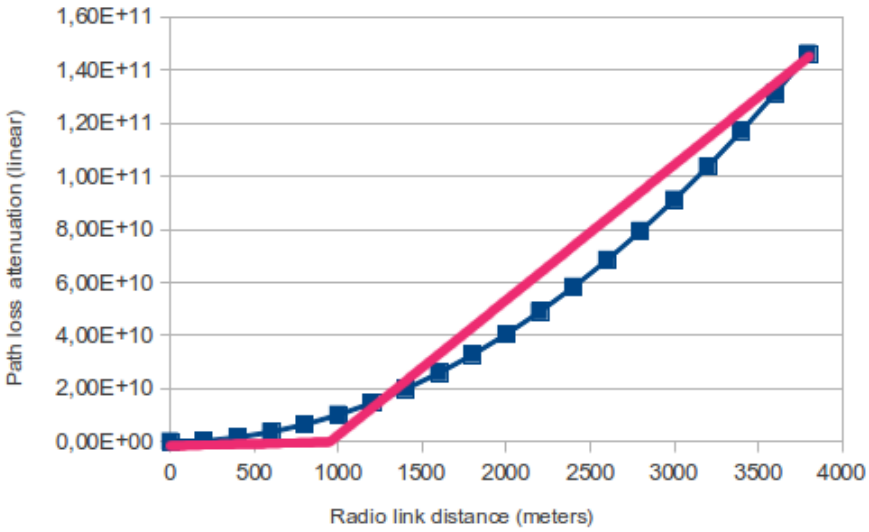

Figure 3.1: Path loss depending radio link distance with a 2,4GHz carrier.

Figure 3.1 has been calculated with standard radio link parameters and a carrier frequency of 2.4GHz. Figure 3.2 has same parameters and scale but a uses a carrier frequency of 5GHz.

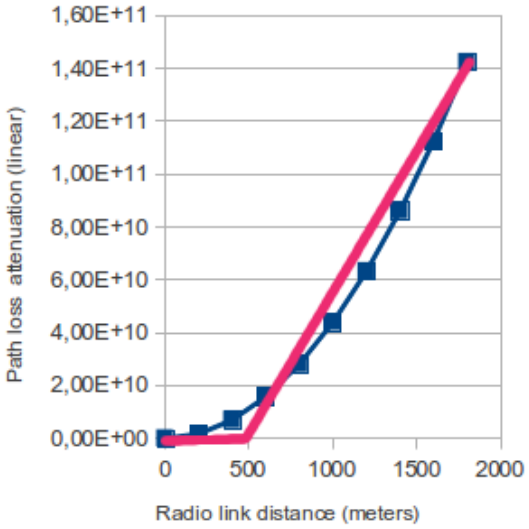

Figure 3.2: Path loss depending radio link distance with a 5GHz carrier.

If the regression equations from power law models of both Figures (3.1 and 3.2) are extracted it will be obtained:

$$\begin{aligned} PL_{2,4GHz} &= 10120,47 \cdot X^2 \\ PL_{5GHz} &= 43925,66 \cdot X^2 \end{aligned} \tag{3.5}$$

It can be appreciated that depending on the carrier frequency the coefficient that multiplies the dependent term changes, so it's difficult to model the loss probability in same way because a new regression equation would have to be calculated each time. The linear approach may be the solution if it's applied with two functions as it's shown in Figures 3.1 and 3.2 (purple color lines).

3.4.1. Model to apply

Having the previous sections into account it can be obtained a final theoretical model that will be the one coded and inserted in Wmediumd. This model is reflected in next step defined Equation:

$$Loss\ \ prob.(d_{current}) = \begin{cases} LPC + FP() \cdot FIC & \text{if } d_{current} \leq \frac{d_{current}}{4} \\ \frac{1}{d_{max} - (\frac{d_{max}}{4})} \cdot d_{current} + LPC + FP() \cdot FIC & \text{if } \frac{d_{current}}{4} < d_{current} \end{cases} \tag{3.6}$$

Once the theoretical model has been created the coding stage can start, next Chapter will explain that process.

CHAPTER 4. WMEDIUMD SIMULATOR EXTENSION

On Chapter 2 it is explained the Wmediumd working environment and flow, it has been found where the new code could be applied to recalculate the loss probabilities depending on medium and mobility, this function was "*send_frame_msg_apply_prob_and_rate()*".

In Chapter 3 it have also been search how to model theoretically the medium and the mobility behaviours in radio link communications.

This chapter will explain how the mobility and medium emulation will be introduced into the Wmediumd simulator explaining main extension topics: configuration data process, data structures and extension flow with created functions.

4.1. Configuration file

Configuration is done independently on the old-way the emulator has worked. This means that Wmediumd has two ways of being loaded: directly with probabilities, and with medium/mobility behaviour. In that form the user will be able to load the extension in old direct probability behaviour or in the new one.

The extension with mobility has next parameters which are loaded to configure the extension and the emulator:

```
interference_tunner = 100;     //10% of interference power, is modeled as a loss
    probability

fading_probability = 200;      //20% of fading appearance probability
fading_intensity = 900;        //90% of fading power, is modeled as a loss
   probability

carrier_frequency = 2.4E+9;        //2,4GHz
transmit_power = 50.0E-3;          //50mW Tx used power
transmit_gain = 2.0;               //3dB Tx antenna gain
receiver_gain = 2.0;               //3dB Rx antenna gain
receiver_min_power = 1E-12;        //-90dBm reciver sensitivity

ifaces_with_mobility: {
    count_ids = 2;
    ids = ["42:00:00:00:00:00", "42:00:00:00:01:00"];

    count_positions_time = 3;
```

```
    positions_time_container = (
        ["1|0|0", "5|500|500", "10|2500|2500"],
        ["1|0|0", "5|0|0", "10|0|0", "15|0|0"]
    );
};
```

4.2. Data structures

New data structures need to be created to introduce the configuration file scenario parameters.

Next structure has a position and time of one radio.

```
struct position_time {
  float time;
  int x;
  int y;
};
```

This data structure defines a radio which has it's MAC address a counter and an array of positions that this radio will have.

```
struct radio_mobility {
  struct mac_address mac;
  int count_positions;
  struct position_time positions[100];
};
```

Next structure has the mobility settings which include the maximum distance ($dmax$) possible to communicate the radio stations with the configured carrier frequency, etc. It also has the array of radios that exist on the scenario.

```
struct mobility_medium_cfg {
  double dmax;
  int interference_tunner;
  int fading_probability;
  int fading_intensity;
  int count_ids;
  struct radio_mobility radios[100];
};
```

4.3. Extension flow with a UML activity diagram

Next Figure shows the mobility extension flow with main implemented functions in a UML activity diagram [14].

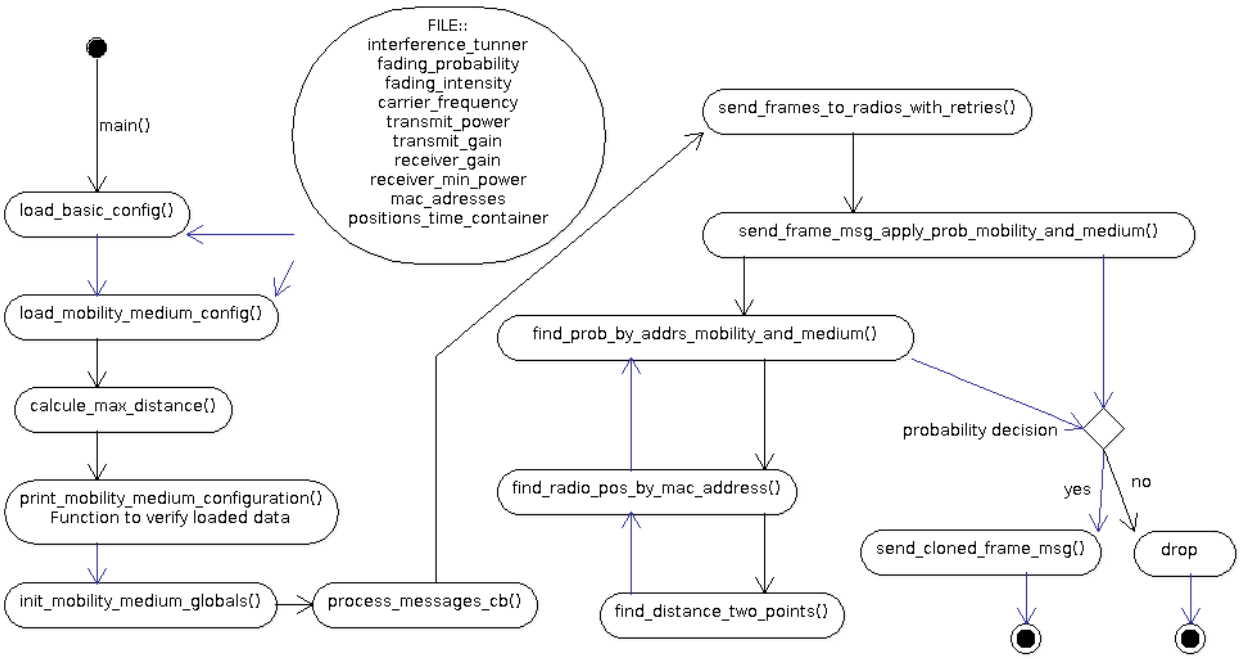

Figure 4.1: UML activity diagram.

Next Subsection will explain each function and the location of extension coding.

4.3.1. Functions, affected and new files

Next list will show which files need to be modified or created to be able to install on them the extension functionalities. The listed functions are all new and created to support the medium/mobility behaviour:

- **wmediumd**/: Main application folder.
  - **config.c** - affected: File where the configuration parameters are read from the scenario configuration file. Some calculations related to configure the scenario are calculated here.
    - *int load_mobility_medium_config(const char *file)*: This function reads *libconfig* [15] formatted files and reads all the medium, mobility and stations scenario configuration. It also have debug information and checks if configuration file is correct.

- *int load_basic_config(const char *file)*: Function to load application required data from mobility and medium configuration file.
- *void calcule_max_distance(double carr_freq, double trans_pow, double trans_gain, double rec_gain, double rec_min_pow, double *dmax)*: Theoretical maximum distance in meters calculated with a link budged including free space equation.
- *void print_mobility_medium_configuration()*: This function prints mobility and medium configurations in user screen.

○ **globals_mobility_medium.h** - new: This file has the definition of global variables used to store the medium and the mobility settings and relevant data used across all application.

○ **probability.c** - affected: File with the necessary coding to calculate loss probability depending on mobility of the stations.

- *int find_prob_by_addrs_mobility_and_medium(struct mac_address *src, struct mac_address *dst)*: Returns the loss probability for a given radio link, mobility and medium.
- *int find_radio_pos_by_mac_address(struct mac_address *addr)*: Returns the position inside the radios array given a mac_address.
- *int find_distance_two_points(int x1, int y1, int x2, int y2)*: Calculates the distance between two points on a 2D plane.

○ **wmedium.c**: Main file of wmediumd.

- *int send_frame_msg_apply_prob_mobility_and_medium(struct mac_address *src, struct mac_address *dst, char *data, int data_len, int rate_idx)*: Function that sends back or not the frame to the kernel.
- *void init_mobility_medium_globals()*: Initialized the mobility and medium extension need variables.
- *int main(int argc, char* argv[])*: Main application function. It has been modified to introduce the new case on switch statement to load the mobility behaviour.

○ **launch_scripts**/ - new : folder where test scripts reside.

- *main_terminal.sh* - new: Script to configure operative system with mac80211_hwsim.
- *first_terminal.sh* - new: Script to configure first network namespace.
- *second_terminal.sh* - new: Script to configure second network namespace.

CHAPTER 5. WMEDIUMD SIMULATOR EXTENSION TEST WITH 802.11S

Wmediumd wireless medium simulation can be test with any 802.11 variants. In this project it's done with 802.11s because the tests begin following some examples of networks spaces that have been done with this protocol. Cozybit also has tests using 802.11s so maybe useful for them to compare results with this protocol than with other.

5.1. Relevant aspects

Some aspects will have to be taken into account because some times they will condition/-modify the results:

- The dump files have 802.11s frames which can not be parse it in Wireshark normal installation. This problem could be solved compiling the Wireshrak from the development *SVN* repository because the newest version has the 802.11s dissector installed on it. The procedure to be able to install it can be found in Appendix D.
- Wmediumd configures the kernel module Mac80211_hwsim to send all frames to him. This means that all frames are send to the medium emulator, even the communication frames with the beacon intervals or other communication control aspects will be send to Wmediumd. Then, the drop of frames would be applied to any type of frame that will arrive to the emulator, which is the expected behaviour. But this has to be taken into account because less TCP packets will be dropped than the expected ones. This is because the drop will take into account the communication control plane frames also and the percentage of this frames arriving to the Wmediumd will depend on the traffic the emulated interfaces will handle.
- Another problem is that after capturing packets into the monitor attached to each interface the same frames are represented in both files running Wmediumd with loss probability. This can be because the frames are send to Wmediumd by the kernel before they have been send to the emitter emulated radios. And then both emulated radios receive the same packets. No diagram of that process have been found so it's difficult to see the exact process without being a Linux Network specialist. In any case it is sure that this is the behaviour because the packets present in both dumps are increasing/decreasing in same order of probability increase/decrease of loss probability.

For that reason it only will be used one monitor in receiver interface. The packet loss statistics presented by the Wmediumd app will also give the relevant data of processed frames.

5.2. Scenario

It have been set one network monitor on the receiver interface from the second network space, and then it have been capturing from there with *tcpdump* [1] application.

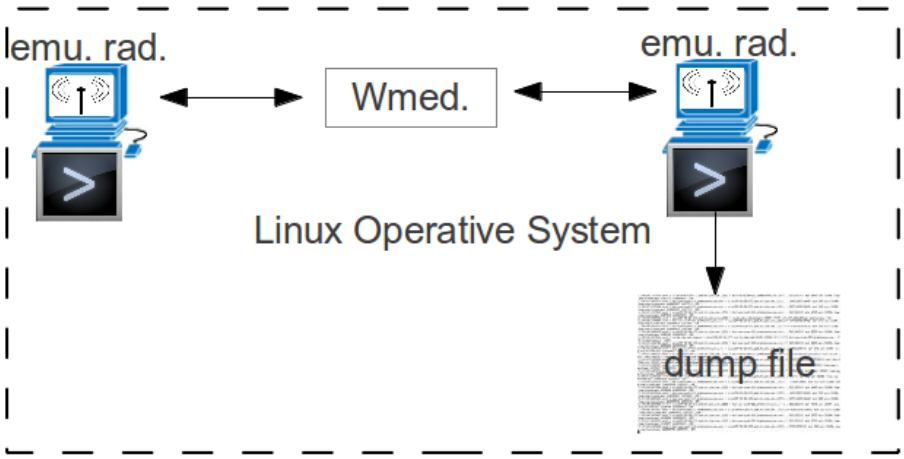

Figure 5.1: Scenario test.

In that manner it's more easy to identify the frames timing and to analyse the behaviour of the communication.

5.3. Results

Test have been perform to validate the modelled behaviour and to check the accuracy the medium emulation has. The application will be check using some stages: static calculations, dynamic calculations and dumped results.

[1] Tcpdump is a packet analyser application with traffic capturing capabilities.

5.3.1. Static calculations

This subsection will check if the application calculate the main static parameter, this is the maximum distance (d_{max}) that a radio link will have taking into account communication parameters. This parameter is one of the more important of the application because all probabilities are calculated with it.

Let's define the input parameters:

- Carrier frequency = 2,4 GHz.
- Transmit power = 50 mW.
- Transmitter gain = 3 dB (2 in linear).
- Receiver gain = 3 dB (2 in linear) .
- Receiver sensitivity = -90 dBm (1E-12 W).

Using the Equation 3.2 it can be obtained the theoretical result of d_{max} that will be used to perform dynamic calculations, then it will be compared with the one that application provides.

$$d_{max}[m] = \frac{|\sqrt{\frac{P_T \cdot G_T \cdot G_R}{Pr_{min}}}|}{\frac{4 \cdot \Pi}{\lambda}} = \frac{|\sqrt{\frac{50E-3 \cdot 2 \cdot 2}{1E-12}}|}{\frac{4 \cdot \Pi}{\frac{299792458}{2,4E9}}} = 4445.438 meters \quad (5.1)$$

Next Figure shows how the application outputs the parameters and it result.

```
Launch wmediumd with mobility
Input mobility and medium configuration file: /home/amartinezi/Dropb
#_if = 2
A[0]:42:00:00:00:00:00
A[1]:42:00:00:00:01:00
=================DEBUG INFO====================
carrier_frequency: 2400000000.000000 [Hz]
transmit_power: 0.050000 [Watts]
transmit_gain: 2.000000 [Linear]
receiver_gain: 2.000000 [Linear]
receiver_min_power: 0.000000000001000 [Watts]
================================================
      PRINT MOBILITI MEDIUM CONFIGURATION
================================================

dmax              =  4445.432464 [meters](theoretical max distance)
```

Figure 5.2: d_{max} Wmediumd result.

It can be verified that the coding of the Equation 3.2 it's correct.

5.3.2. Dynamic calculations

This calculations will be calculated each time a frame arrives to Wmediumd and depending on the execution time because the stations will simulate movement while the time changes.

Now some other parameters will be defined that will have to be taken into account on dynamic calculations.

- Interference intensity = 10%
- Fading probability = 20%
- Fading intensity = 90% (of fading intensity power, it is modelled as a loss probability)
- First radio mobility info (time|x|y) [seconds|meters|meters] = "1|0|0", "5|500|500", "10|2500|2500", "15|3500|3500", "20|4500|4500".
- Second radio mobility info (time|x|y) [seconds|meters|meters] = "1|0|0", "5|0|0", "10|0|0", "15|0|0", "20|0|0".

Configured this parameters on the application it can be obtained from Wmediumd extension shell output parameters seen on next Figure:

```
framesss [20] length:75
DEBUG INFO. POSITIONS: p1=(0,0) p2=(500,500) EXEC_TIME: 10sec. LAST_ARR_INDEX: 1
 DISTANCE_BETWEEN_TX_RX: 707
PROBABILITIES. random_value 43 | distance_prob 0 (0%) | loss_probability 1000 (1
00%) | fading 900| dcurrent 707 | dmax 4445
Dropped
received: 20 tried: 20 sent: 13 acked: 0flags: 2
framesss [21] length:75
DEBUG INFO. POSITIONS: p1=(2500,2500) p2=(0,0) EXEC_TIME: 11sec. LAST_ARR_INDEX:
 2 DISTANCE_BETWEEN_TX_RX: 3535
PROBABILITIES. random_value 374 | distance_prob 726 (72%) | loss_probability 826
 (82%) | fading 0| dcurrent 3535 | dmax 4445
Dropped
received: 21 tried: 21 sent: 13 acked: 0flags: 2
framesss [22] length:75
DEBUG INFO. POSITIONS: p1=(0,0) p2=(2500,2500) EXEC_TIME: 11sec. LAST_ARR_INDEX:
 2 DISTANCE_BETWEEN_TX_RX: 3535
PROBABILITIES. random_value 920 | distance_prob 726 (72%) | loss_probability 826
 (82%) | fading 0| dcurrent 3535 | dmax 4445
Dropped
received: 22 tried: 22 sent: 13 acked: 0flags: 2
```

Figure 5.3: Wmediumd extension shell output.

With this data it can be constructed Table 5.1 that will have to accomplish the Equation 3.6 to be consistent.

Table 5.1: Time ranges losses depending mobility and interferences.

Time	Position X	Position Y	Distance between rad.	Loss prob distance	Interference	Loss probability
1	0	0	0	0	10	10
2	0	0	0	0	10	10
3	0	0	0	0	10	10
4	0	0	0	0	10	10
5	0	0	0	0	10	10
6	500	500	707	0	10	10
7	500	500	707	0	10	10
8	500	500	707	0	10	10
9	500	500	707	0	10	10
10	500	500	707	0	10	10
11	2500	2500	3535	72	10	82
12	2500	2500	3535	72	10	82
13	2500	2500	3535	72	10	82
14	2500	2500	3535	72	10	82
15	2500	2500	3535	72	10	82
16	3500	3500	4949	115	10	125
17	3500	3500	4949	115	10	125
18	3500	3500	4949	115	10	125
19	3500	3500	4949	115	10	125
20	3500	3500	4949	115	10	125
21	4500	4500	6363	157	10	167
22	4500	4500	6363	157	10	167
23	4500	4500	6363	157	10	167
24	4500	4500	6363	157	10	167
25	4500	4500	6363	157	10	167

The obtained values are following the Equation 3.6, these values have been compared with the ones from a calculus sheet successfully. The addition of probabilistic fading is also added but it was not introduced into the table because frames arrive faster than a frame per second and the table shows non consecutive frames data, so introducing fading would not show the set fading probability. In any way, fading is add depending of it's intensity (a percentage value) and the apparition probability percentage.

5.3.3. Dumped results

Lets check a TCP connection over this scenario. For that *MGEN* will be used as traffic generator from one network namespace to the other one.

For doing that a Server socket listening will be used and a Client starting a TCP session with the server. For doing that *MGEN* is configured as follows:

- Server input file: *server_mgen.mgn*

```
1 LISTEN TCP 4002
```

- Client input file: *client_mgen.mgn*

```
# time | udp-tcp |src port | dst ip/port |traff type [packet rate packet
    size in bytes]
#100*1000*8 = 0,8Mbit/s
0.0 ON  1 TCP SRC 4001 DST 192.168.4.2/4002 PERIODIC [100 1000]
```

MGEN will be instantiated in each network namespace shell and load as input file it's correspondent configuration file [16]. On client side it will be set next TCP flow characteristics following file comments. These are:

- TCP traffic.
- From client port 4001.
- To port 4002.
- To IP 192.168.4.2.
- Packet rate 100 packet/second.
- Size of packet 1000 bytes.

Some of the captured frames are displayed with Wireshark to analyse if the behaviour is the expected one. Figure 5.4 shows the capture of it:

Time	Source	Destination	Protocol	Length	Info
1.028091	42:00:00:00:01:00	Broadcast	802.11	78	Action, SN=954, FN=0, Flags=.....
1.028261	42:00:00:00:01:00	42:00:00:00:00:00	802.11	72	Action, SN=955, FN=0, Flags=.....
1.028347	192.168.4.2	192.168.4.1	TCP	119	pxc-spvr-ft > newoak [SYN, ACK] Seq
1.028401	192.168.4.2	192.168.4.1	TCP	119	[TCP Out-Of-Order] pxc-spvr-ft > ne
1.028408	42:00:00:00:00:00	42:00:00:00:01:00	802.11	77	Action, SN=801, FN=0, Flags=.....
1.028445	192.168.4.1	192.168.4.2	TCP	124	[TCP Spurious Retransmission] newoa
1.028502	192.168.4.1	192.168.4.2	TCP	116	newoak > pxc-spvr-ft [ACK] Seq=1 Ac
1.028565	42:00:00:00:01:00	Broadcast	802.11	56	Action, SN=0, FN=0, Flags=.....
1.028572	192.168.4.1	192.168.4.2	TCP	128	[TCP Dup ACK 17#1] newoak > pxc-spv
1.028859	192.168.4.2	192.168.4.1	TCP	119	[TCP Out-Of-Order] pxc-spvr-ft > ne
1.028883	192.168.4.1	192.168.4.2	TCP	128	[TCP Dup ACK 17#2] newoak > pxc-spv
1.091893	42:00:00:00:01:00	Broadcast	802.11	140	Beacon frame, SN=0, FN=0, Flags=...
2.055904	42:00:00:00:00:00	Broadcast	802.11	145	Beacon frame, SN=0, FN=0, Flags=...
2.119939	42:00:00:00:01:00	Broadcast	802.11	140	Beacon frame, SN=0, FN=0, Flags=...
3.147923	42:00:00:00:01:00	Broadcast	802.11	140	Beacon frame, SN=0, FN=0, Flags=...
4.035921	42:00:00:00:00:00	Broadcast	802.11	83	Action, SN=802, FN=0, Flags=.....
4.036050	42:00:00:00:01:00	42:00:00:00:00:00	802.11	72	Action, SN=956, FN=0, Flags=.....
4.036140	192.168.4.1	192.168.4.2	TCP	1564	[TCP Previous segment not captured]

Figure 5.4: Wireshark loaded monitor dump file.

Using the configuration parameters from Subsections 5.3.1. and 5.3.2. it's seen how TCP finds missing transmission packets and order to client retransmissions of them. Timing of

1 second has probability losses of 10% due to medium constant interferences so at the MGEN transmission rate of 100 packets/second, 10 packets will be drop per second. This 10 packet loss per second causes the retransmissions.

This is the expected results for a TCP connection with 10% of losses.

CHAPTER 6. MATERIAL AND METHODS

This chapter refers to the tools used to perform the initial objectives. On it hardware and software resources will be list to permit others emulate the results. Also some relevant aspects that can affect the results and related to hardware will be explain in this section.

6.1. Hardware

In order to perform all tests and deployments the only used equipment was a 2013 laptop Dell Vostro 2520 with next characteristic list:

- Processor : Intel Celeron processor 1000M (2M Cache, 1.8 GHz)
  - 2M Cache
  - 1.8 GHz
  - 64-bit Instruction Set
- Memory : 2048MB (1x2GB) 1600MHz DDR3 Dual Channel
- Wireless NIC: Dell Wireless 1704 802.11b/g/n, Bluetooth v4.0+LE

Wmediumd could have to process a large amount of frames per time unit depending on the transmission rate between configured radios. The processing time is present and changes depending on machine. The hardware capabilities will change frame delay, this has to be taken into account when choosing the hardware.

6.2. Software

Next software was used to perform the tests and deployments:

- Linux based operating system Ubuntu 12.04 LTS i386 (32 bits).
- Mac80211_hwsim kernel module has to be loaded (*lsmod* have to list it, if its present and not active, it can be loaded with *a2enmod mac80211_hwsim*, then *lsmod* will show it).

- Used kernel: 3.4.0 generic.
- Wireshark development version: Version 1.11.0 (SVN Rev 50893 from /trunk)
- Bash scripting was used to deploy the test-bed.

CHAPTER 7. ENVIRONMENTAL IMPACT

The environmental impact associated to this project is very difficult to quantify. This extension represents an improvement into the 802.11 development environment and it will be used to make it better.

If it's supposed an improvement of the 802.11 power-save mode using this extension, this improvement would save charging cycles of mobile devices and would be partially caused by this project.

7.1. Mobile devices shipments evolution

In this section it will be studied the volume of mobile devices that can exist in the society in a near future. Table 7.1 shows the real evolution that mobile devices shipments had from 2009 to 2013 in millions of units [20].

Table 7.1: Mobile devices evolution.

Mobile device type ⟍ Year	**2009**	**2010**	**2011**	**2012**	**2013**
Smartphones	175	240	310	390	500
Desktops	136	146	152	157	159
Notebooks	135	164	189	210	232
Netbooks	34	36	29	26	27
Tablets	0	16	55	85	102
Total	**480**	**602**	**735**	**868**	**1020**

Taking into account the real data a growing Equation 7.1 can be obtained, that formula can be used to create an expectation of how many mobile devices will be shipped in 2020. If a 2 years of lifetime for each mobile device are supposed it can be obtained approximately the number of mobile devices that will exist in near future.

$$f(x) = 134,6x - 269939,6 \tag{7.1}$$

Next Figure (7.1) shows real and expected values till 2020, where blue line is the known values and the purple one is the expected values maintaining the actual growths.

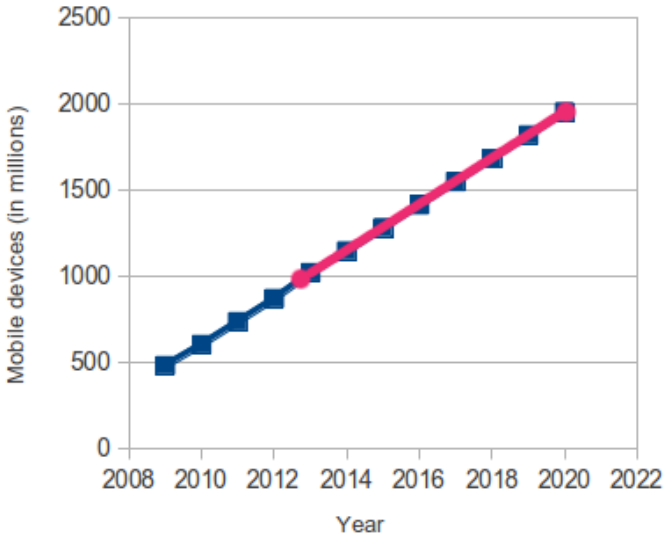

Figure 7.1: Mobile devices evolution.

Maintaining actual growth of mobile device shipments in 2020 approximately 4000 millions of them will exist. Supposing that the 70% of these devices will use 802.11 protocols for communication purposes, will exist 2800 millions of devices that will be improved if the 802.11 power consumption is also improved.

7.2. Power saving

These 2800 millions of terminals will use less energy so they will need less charging cycles.

Next Figure (7.2) shows the power consumption of 802.11 networks in power-save mode (PS).

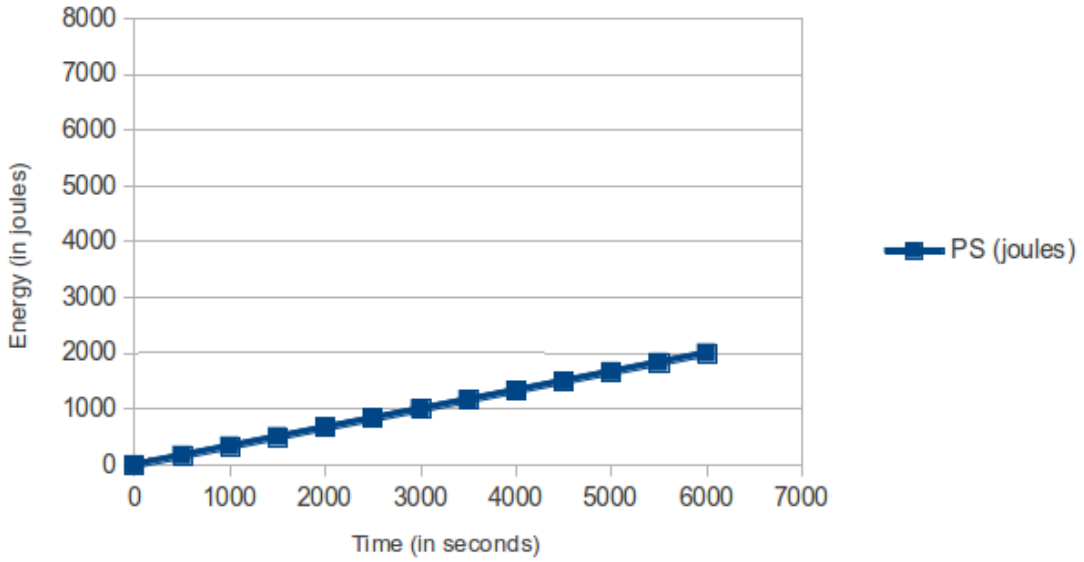

Figure 7.2: PS energy consumption over time.

The growing factor is the power in [watt] that each network interface will use, on **PS it's 0,33 watts**. Assuming the use of 802.11 network interfaces is 25% of device power on time and supposing 24 hours powered devices next results will be obtained:

$$\begin{aligned} \text{PS consume after a year} &= 0,33 \cdot 2800E6 \cdot 365 \cdot 24 \cdot 0,25 = \textbf{2023,56E9 Wh} \\ \text{PS}_{optimized} \text{ consume after a year} &= 0,20 \cdot 2800E6 \cdot 365 \cdot 24 \cdot 0,25 = \textbf{1226,40E9 Wh} \end{aligned} \tag{7.2}$$

As seen there is a relevant reduction of total consumed energy between the two modes, this would be a positive impact related to this project.

7.3. Mobile devices promotion and resources

But improving the 802.11 developing environment will end with better 802.11 specifications, which will stimulate the consumers to buy new devices. This renewal process will also require resources in order to build them and to recycle the replaced ones.

The 802.11 protocol family improvements maybe encourage some developers to implement new applications using this protocol.

For example: an application to transfer photos between phones at hight speed. In some way, this application can be very cool and excite some consumers to change their devices to have this new features.

This effect will increase the mobile phones renewal ratio and increase the demand of natural materials(such as coltan and tantaline) and energy in order to satisfy this demand. This will increase the waste generation of old devices that will have to be processed using also energy.

As discussed it's very difficult to quantify the environmental impact of this project because it has possible positive and negative aspects which are difficult to predict. In any case this could be possible environmental impacts of this project.

CHAPTER 8. CONCLUSIONS

This project contributes to create a better 802.11 development environment for driver developers introducing medium and mobility behaviour in their testing scenarios. This is a better approach than the existent one because it's more similar to real world wireless conditions than simply apply probabilistic frame errors.

The implemented extension allow developers to simulate movement on emulated radio interfaces and to test their drivers. The coding is also very tunable and allows users to set the most commonly used variables on wireless communications as: antenna gains, receiver sensitivity, transmission power, carrier frequencies, constant interferences and probabilistic fadings.

Main objective is accomplished, the extension works providing mobility and medium emulation following the defined theoretical model. This has been checked with calculus sheets using directly the model equations and checking the program outputs (see 5.3.). The environmental impact study was difficult to perform because is difficult to predict if project positive or negative aspects will dominate on future.

One interesting thing is that this project touch relevant aspects related to state of the art Linux Kernel resources management as:

- Hardware simulation with emulation of network radios with mac80211_hwsim kernel module.
- Insulating completely network stacks using network name spaces and assigning OS resources to it.
- Inter process communications between user space and Kernel using Netlink API calls.

Other aspects as changing Linux Kernels images or how creating/install Wireshark dissectors are also important. Advanced C programming techniques where used, for example; comparing data structures directly from memory with pointers and the structure block sizes in bytes or using the libconfig to read well formed data structures stored in files.

Some problems introduced delays in project stages and some times they arrived to stop the project. Wmediumd had no documentation, for running it where required weeks because the found scripts did not work and all theory of network namespaces and mac80211_hwsim had to be learn before switch on the old Wmediumd. The emulator couldn't be compiled

with kernel versions after v.3.4 and no one know that, for that reason was required to ask a C expert why the coding was not running, the expert found that some macro definitions where not included in newest Linux Kernels. In any case the problems where solved and the project was successfully deployed.

This project represents a base that can be optimized or extended to add new functionalities, for example to improve the process time delays. This delays could be improved working with pointers to frame structures on memory, actually Wmediumd works with frame copies. Other thing that could be nice to force bit errors on frames, to then, deliver them to the radios, current extension just drops the frames. Also a good thing to introduce would be a bursty losses behaviour.

To conclude, just say that it would be good that with current improvements and this documentation some one would use Wmediumd extended version. ;)

BIBLIOGRAPHY

[1] Namespaces in operation, part 1: namespaces overview. [25 July 2013]
http://lwn.net/Articles/531114/

[2] Linux Namespaces at Arista. [25 July 2013]
https://eos.aristanetworks.com/2011/06/linux-namespaces-at-arista/

[3] Pcap. [22 July 2013]
http://en.wikipedia.org/wiki/Pcap

[4] Mac80211_hwsim. [16 August 2013]
http://wireless.kernel.org/en/users/Drivers/mac80211_hwsim

[5] Wmediumd. [16 August 2013]
https://github.com/cozybit/wmediumd

[6] Netlink. [16 August 2013]
http://en.wikipedia.org/wiki/Netlink

[7] Generic Netlink HOW-TO. [16 August 2013]
http://lwn.net/Articles/208755/

[8] Cyclic redundancy check. [30 July 2013]
http://en.wikipedia.org/wiki/Cyclic_redundancy_check

[9] Signal-to-noise ratio. [17 August 2013]
http://en.wikipedia.org/wiki/Signal-to-noise_ratio

[10] Carrier sense multiple access with collision avoidance. [18 August 2013]
http://en.wikipedia.org/wiki/Carrier_sense_multiple_access_with_collision_avoidance

[11] Wireless Propagation. [18 August 2013]
http://www.eecis.udel.edu/ bohacek/Classes/SensorNets/WirelessProp.ppt

[12] Equivalent isotropically radiated power. [17 August 2013]
http://en.wikipedia.org/wiki/Equivalent_isotropically_radiated_power

[13] Link budget. [19 August 2013]
http://en.wikipedia.org/wiki/Link_budget

[14] Activity diagram. [24 August 2013]
http://en.wikipedia.org/wiki/Activity_diagram

[15] Libconfig manual. [24 August 2013]
www.hyperrealm.com/libconfig/libconfig_manual.html

[16] MGEN User's and Reference Guide Version 5.0. [25 August 2013]
http://pf.itd.nrl.navy.mil/mgen/mgen.html

[17] Gartner Says Worldwide PC, Tablet and Mobile Phone Shipments. [10 September 2013]
http://www.gartner.com/newsroom/id/2525515

[18] Wi-Fi Chipset Shipments to Double in 2011. [10 September 2013]
http://www.isuppli.com/mobile-and-wireless-communications/marketwatch/pages/wi-fi-chipset-shipments-to-double-in-2011.aspx

[19] 802.11 Power Save Mode. [21 August 2013]
http://www.cs.cornell.edu/people/ranveer/multinet/multinet/With_Power_Save_Mode.html

[20] Tablet Demand and Disruption - Morgan Stanley . [21 August 2013]
www.morganstanley.com/views/perspectives/tablets_demand.pdf

[21] Smartphone. [21 August 2013]
http://en.wikipedia.org/wiki/Smartphone

[22] Wireless-Lan mesh (cronismo). [30 July 2013]
http://www.goo.gl/83r6Ri?Wlan=87392dss_index

[23] IEEE 802.11s. [30 July 2013]
http://en.wikipedia.org/wiki/IEEE_802.11s

[24] Can I install Linux kernel 3.4 in Ubuntu and Kubuntu 12.04?. [13 July 2013]
http://askubuntu.com/questions/142192/can-i-install-linux-kernel-3-4-in-ubuntu-and-kubuntu-12-04

[25] Namespaces in operation, part 1: namespaces overview. [1 September 2013]
http://lwn.net/Articles/531114/

[26] Linux Namespaces at Arista. [1 September 2013]
https://eos.aristanetworks.com/2011/06/linux-namespaces-at-arista/

[27] Build Wireshark. [30 August 2013]
http://www.wireshark.org/docs/wsdg_html_chunked/ChSrcBuildFirstTime.html

[28] Template for an external Wireshark dissector Plugin. [30 August 2013]
https://gitorious.org/wiresharkdissectortemplate/pages/Home

[29] Adding a basic dissector. [30 August 2013]
http://www.wireshark.org/docs/wsdg_html_chunked/ChDissectAdd.html

[30] Build Wireshark. [30 August 2013]
http://www.wireshark.org/docs/wsdg_html_chunked/ChSrcBuildFirstTime.html

[31] Template for an external Wireshark dissector Plugin. [30 August 2013]
https://gitorious.org/wiresharkdissectortemplate/pages/Home

[32] Adding a basic dissector. [30 August 2013]
http://www.wireshark.org/docs/wsdg_html_chunked/ChDissectAdd.html

APPENDICES

APPENDIX A. SETTING A SPECIFIC KERNEL IN UBUNTU

This Appendix will show how to set and boot a precompiled Kernel version of Ubuntu Linux OS.

To use a new kernel as-is it's only need to download and install a image.deb package that corresponds to the machine architecture.

Ubuntu kernel images can be found in: http://kernel.ubuntu.com/ kernel-ppa/mainline/

Open a terminal and move to the directory where the downloaded Kernel is.

```
$ cd ~/Descargas/
```

Then use *dpkg* command to install the packages: linux-headers, linux-headers-generic and linux-image-generic.

For linux-headers (is not architecture specific):

```
$ sudo dpkg -i linux-headers-3.4.0-030400_3.4.0-030400.201205210521_all.deb
```

For linux-headers-generic (architecture specific):

```
$ sudo dpkg -i linux-headers-3.4.0-030400-generic_3.4.0-030400.201205210521
    _i386.deb
```

For linux-image-generic (architecture specific):

```
$ sudo dpkg -i linux-image-3.4.0-030400-generic_3.4.0-030400.201205210521_i386.
    deb
```

If any warnings or errors are seen while installing then try installing module-init-tools first, and try again. Restart the system, choose the kernel to boot on select box of Grub and check the kernel version after booting [24].

```
$ uname -a
```

APPENDIX B. LINUX NAMESPACES WITH 802.11

One of the Linux Namespaces is the Network space. Next steps will show the process of configuring separate network spaces to insulate two emulated radio interfaces:

1. On father terminal type:

```
rmmod mac80211_hwsim
modprobe mac80211_hwsim radios=2
find /sys/kernel/debug/ieee80211 -name hwsim | cut -d/ -f 6 | sort
iwconfig

```

2. On children terminal 1 type:

```
unshare -n bash
echo $$

```

3. On children terminal 2 type:

```
unshare -n bash
echo $$

```

4. On father terminal type:

```
iw phy phy9 set netns 2766
iw phy phy10 set netns 2781

```

5. On children terminal 1 type:

```
iwconfig
iw phy phy9 interface add mesh9 type mesh
iwconfig
ifconfig
ip address add dev mesh9 192.168.4.9/24
ip link set mesh9 up
iw dev mesh9 mesh join bazooka
ping 192.168.4.10
ifconfig
ping 192.168.4.10
ifconfig lo up
ifconfig
```

```
13 ping 192.168.4.9
14
15
```

6. On children terminal 2 type:

```
iwconfig
iw phy phy10 interface add mesh10 type mesh
ip address add dev mesh10 192.168.4.10/24
ip link set mesh10 up
iw dev mesh10 mesh join bazooka
ping 192.168.4.9
ping 192.168.4.10
ping 192.168.4.9
ifconfig lo up
ifconfig
ping 192.168.4.10

```

APPENDIX C. MAIN EXTENSION FUNCTIONS

This Appendix will show some of the most relevant functions this extension has created.

C.1. calcule_max_distance()

Theoretical maximum distance in meters calculated with a link budged including free space equation.

```
void calcule_max_distance(double carr_freq, double trans_pow, double trans_gain
    ,
    double rec_gain, double rec_min_pow, double *dmax) {

  double lambda, pi = 3.14159265358979323846, c = 299792458.0, first, second,
      third;

  lambda = c / carr_freq;

  first = (trans_pow * trans_gain * rec_gain) / rec_min_pow;
  second = abs(pow(first,1.0/2.0));
  third = ((double) 4.0 * pi) / lambda;

  *dmax = second / third;
}
```

C.2. find_distance_two_points()

Calculates the distance between two points on a 2D plane.

```
int find_distance_two_points(int x1, int y1, int x2, int y2) {

  return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));

}
```

C.3. find_radio_pos_by_mac_address()

Returns the position inside the radios array given a mac_address.

```
int find_radio_pos_by_mac_address(struct mac_address *addr) {
```

```
2
3   int i = 0;
4
5   while (memcmp(&mob_med_cfg.radios[i].mac, addr, sizeof(struct mac_address))
6       != 0 && i < mob_med_cfg.count_ids) {
7     i++;
8   }
9
10  return i;
11 }
```

C.4. find_prob_by_addrs_mobility_and_medium()

Returns the loss probability for a given radio link, mobility and medium. If an error occurs returns -1.

```
1
2 int find_prob_by_addrs_mobility_and_medium(struct mac_address *src,
3     struct mac_address *dst) {
4
5   //Probabilities are defined per 1000 units to perform them using integers
6   int random_value = rand() % 1000 + 1, distance_prob = 0, loss_probability =
7       0, fading = 0, src_radio_pos, dst_radio_pos, time_pos;
8   unsigned long int execution_time;
9
10  struct timeval current_time;
11  gettimeofday(&current_time, NULL);
12
13  execution_time = current_time.tv_sec - start_execution_timestamp;
14
15  src_radio_pos = find_radio_pos_by_mac_address(src);
16  dst_radio_pos = find_radio_pos_by_mac_address(dst);
17  //Both stations need to have same time line definition, for that we can use
      or src or dst to take the time.
18  if (execution_time
19      > (int) mob_med_cfg.radios[src_radio_pos].positions[(last_def_position
20          + 1)].time
21      && last_def_position
22          < (mob_med_cfg.radios[src_radio_pos].count_positions - 1)) {
23    last_def_position++;
24    dcurrent =
25        find_distance_two_points(
26            mob_med_cfg.radios[src_radio_pos].positions[last_def_position].x,
27            mob_med_cfg.radios[src_radio_pos].positions[last_def_position].y,
```

```
28              mob_med_cfg.radios[dst_radio_pos].positions[last_def_position].x,
29              mob_med_cfg.radios[dst_radio_pos].positions[last_def_position].y);
      /*meters*/
30    }
31
32    if (dcurrent <= mob_med_cfg.dmax / 4) {
33      distance_prob = 0;
34    } else {
35      distance_prob = (((float) dcurrent - (float) mob_med_cfg.dmax / 4.0)
36          / ((float) mob_med_cfg.dmax - ((float) mob_med_cfg.dmax / 4.0)))
37          * 1000; //1000 is a scale factor
38    }
39
40    if (random_value < mob_med_cfg.fading_probability) {
41      fading = mob_med_cfg.fading_intensity;
42    } else {
43      fading = 0;
44    }
45
46    loss_probability = distance_prob + mob_med_cfg.interference_tunner + fading;
47
48    if (debug == 1) {
49      printf(
50          "DEBUG INFO. POSITIONS: p1=(%d,%d) p2=(%d,%d) EXEC_TIME: %dsec.
      LAST_ARR_INDEX: %d DISTANCE_BETWEEN_TX_RX: %d\n",
51          mob_med_cfg.radios[src_radio_pos].positions[last_def_position].x,
52          mob_med_cfg.radios[src_radio_pos].positions[last_def_position].y,
53          mob_med_cfg.radios[dst_radio_pos].positions[last_def_position].x,
54          mob_med_cfg.radios[dst_radio_pos].positions[last_def_position].y,
55          execution_time, last_def_position, dcurrent);
56
57      printf(
58          "PROBABILITIES. random_value %d | distance_prob %d (%d%) |
      loss_probability %d (%d%) | fading %d| dcurrent %d | dmax %d \n",
59          random_value, distance_prob, (distance_prob / 10),
60          loss_probability, (loss_probability / 10), fading, dcurrent,
61          (int) mob_med_cfg.dmax);
62    }
63
64    return loss_probability;
65  }
```

APPENDIX D. SETTING WIRESHARK NETWORK ANALYSER

In order to be able to check the behaviour of the implementations mesh network traffic it's used. This traffic is captured and processed by a network analyser, in that case Wireshark will be the used one.

D.1. Compile Wireshark from sources

Wireshark uses a plugin type called dissector to parse the captured data into fields for the graphic interface, it usually uses one dissector per protocol. Standard Wireshark installations doesn't have the 802.11s dissector ready to use. Cozibit recommends in their documentation to install Wireshark from sources in order to get the state of the art Wireshark code and the mesh dissector on it. This is the procedure that will be used.

For this Wireshark has to be compiled following next procedure:

1. Install dependencies:

```
aptitude install bison flex libgtk-3-dev  subversion libpcap-dev
aptitude install build-essential automake autoconf libtool

```

2. Getting the Wireshark source code from the official Subversion repository:

```
svn co http://anonsvn.wireshark.org/wireshark/trunk/ wireshark

```

3. Run the autogen.sh script at the top-level Wireshark directory to configure the build directory.

```
cd wireshark
./autogen.sh
./configure
make

```

4. Now you will find the binary file to run Wireshark on the root directory of the source code , you can call it from the shell to run it [30].

```
./wireshark

```

D.2. Installing a dissector

This section will explain how to install a dissector. This may be useful if you want to extend some dissector or to install the 802.11s dissector into a standard Wireshark installation.

The following shows how to install one dissector:

1. Download a template of the dissector:

```
git clone git://gitorious.org/wireshark-dissector-template/wireshark-
    dissector-template.git wireshark-dissector-template
cd wireshark-dissector-template

```

   That template contains next files:

   - Makefile.am - This is the UNIX/Linux makefile template
   - Makefile.common - This contains the file names of this plugin
   - Makefile.nmake - This contains the Wireshark plugin makefile for Windows
   - moduleinfo.h - This contains plugin version info
   - moduleinfo.nmake - This contains DLL version info for Windows
   - packet-foo.c - This is the dissector source
   - plugin.rc.in - This contains the DLL resource template for Windows

2. To use the template just change all occurrences of foo by your own plugin name, and rename packet-foo.c by packet-yourpluginname.c

3. Building and installing the dissector:

```
mkdir build
cd build
cmake ..
make
make install

```

Now the dissector must be installed on the user home (*/home/user/.wireshark/plugins/*) folder and loaded when this user makes the call to Wireshark [31].

If a new dissector has to be created the procedure to do it will be found in the Bibliographic Element [32].