



Escola d'Enginyeria de Telecomunicació i
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: Evaluation and monitoring of wireless community networks

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Sergi Madonar Soria

DIRECTOR: Roc Meseguer Pallarès

DATE: July 1st 2013

Títol: Avaluació i monitorització de xarxes inalàmbriques comunitàries

Autor: Sergi Madonar Soria

Director: Roc Meseguer Pallarès

Data: 1 de juliol de 2013

Resum

En les darreres dècades s'han produït molts canvis en l'àmbit de les telecomunicacions, gran part d'aquest canvi ha estat propiciat per l'aparició d'Internet, la "xarxa de xarxes" capaç de connectar usuaris a gairebé qualsevol punt del món.

Inicialment totes les infraestructures eren cablejades, però en els darrers anys ha anat avançant l'electrònica permetent reduir la mida dels dispositius terminals. Això ha afavorit i facilitat la mobilitat d'usuaris amb els seus terminals, pel que les tecnologies inalàmbriques han evolucionat conjuntament per oferir-los connectivitat.

Diversos projectes per tot el món han aparegut amb l'objectiu de desplegar xarxes inalàmbriques obertes creant comunitats d'usuaris, empreses i entitats; l'exemple més proper a Catalunya és la fundació Guifi-net.

Degut a la dificultat d'administrar aquest tipus de xarxes ja que són completament obertes i qualsevol usuari s'hi pot afegir lliurement, l'objectiu d'aquest projecte és proporcionar orientació i eines per tal de fer-ho. Altres factors com l'entorn poden alterar el correcte funcionament de la xarxa degut al medi inalàmbric, això en complica encara més la gestió.

Amb totes les eines i mesures descrites dins d'aquest projecte, un administrador d'una nova xarxa inalàmbrica comunitària seria capaç d'avaluar l'estat d'aquesta i, així, mantenir i millorar el seu rendiment.

Title: Evaluation and monitoring of wireless community networks

Author: Sergi Madonar Soria

Director: Roc Meseguer Pallarès

Date: July, 1 st 2013

Overview

In the last decades there have been many changes in the telecommunications sector, much of this change has been brought by the creation of the Internet, the "network of networks" that can connect users to almost any point in the world.

Initially all infrastructures were wired, but in recent years electronics has advanced allowing to reduce the size of the terminal devices. This has encouraged and facilitated the users' mobility with their terminals, so wireless technologies have evolved together to offer connectivity.

Several projects around the world have emerged with the aim of deploying open wireless networks composed by user communities, companies and organizations; the closest example in Catalonia is the Guifi-net foundation.

Due to the difficulty of managing such networks because they are completely open and anyone can freely join, the aim of this project is to provide guidance and tools to do so. Other factors such as the environment can alter the proper functioning of the network due to the wireless medium, it further complicates the management.

With all the tools and measures described within this project, a new wireless network administrator of a community would be able to evaluate its status and maintain and improve their performance.

INDEX

INTRODUCTION	1
CHAPTER 1. WIRELESS COMMUNITY NETWORKS	3
CHAPTER 2. WCN NODES AND TOOLS	5
2.1. OpenWrt	5
2.2. snmpd.....	5
2.3. ntop.....	7
2.3.1. nProbe	7
2.4. bandwidthd	7
2.5. nmap	8
2.5.1. Zenmap.....	8
2.6. LinSSID.....	9
CHAPTER 3. LIST OF MEASUREMENTS	11
3.1. Active links. Links used by routes in the network.....	11
3.2. All links that receive at least one beacon in a measurement interval	13
3.3. Total amount of data transferred by clients	13
3.4. Traffic shape	14
3.5. Latency and Jitter.....	16
3.6. Signal parameters	17
3.7. Discover the network devices.....	18
3.8. Device information and configuration	20
3.8.1. lspci.....	20
3.8.2. modinfo.....	22
3.8.3. hwinfo	23
3.9. Transfer rates	24
3.10. Application classes.....	25
3.11. Frame Error Rate (FER) and Deliver probability	26
3.12. Network topology	27
3.13. List of available wireless networks	28
CHAPTER 4. USING THE TOOLS	29

4.1. Where.....	29
4.2. What.....	31
4.3. When.....	31
4.4. How	33
CHAPTER 5. CONCLUSIONS.....	35
REFERENCES.....	37
ANNEXES	39
1. Active links script source code.....	39
2. Link that receive a beacon receive script source code.....	41
3. Amount of traffic sent by clients script source code	43
4. Latency and jitter script source code.....	45
5. Deliver probability and Frame Error Rate script source code	47

LIST OF FIGURES

Fig. 1 Wireless mesh network topology example	3
Fig. 2 Guifi.net project logo.....	3
Fig. 3 Links and nodes representation over Google Maps	4
Fig. 4 OpenWrt logo	5
Fig. 5 MIB structure example.....	6
Fig. 6 ntop logo.....	7
Fig. 7 bandwidthd logo	8
Fig. 8 nmap logo.....	8
Fig. 9 Zenmap interface	9
Fig. 10 LinSSID logo	9
Fig. 11 Output of the route links script.....	11
Fig. 12 List of localhost interfaces and routes	12
Fig. 13 Output of the active links	13
Fig. 14 Output of the amount of data script	13
Fig. 15 Traffic graph from the “Bandwidthd” tool	14
Fig. 16 ntop traffic graph in different time periods	15
Fig. 17 ntop traffic graph separated by protocols	15
Fig. 18 Ping command example to Google website	16
Fig. 19 Output of the latency and jitter script	16
Fig. 20 Output of the iwconfig command	17
Fig. 21 Devices discovered by nmap in the network	18
Fig. 22 Devices and their open ports discovered in the network	19
Fig. 23 lspci output for the wireless device	20
Fig. 24 lspci wireless driver information.....	21

Fig. 25 modinfo output for the wireless kernel module	22
Fig. 26 hwinfo output for the wireless interface	23
Fig. 27 Main screen of the “iftop” command	24
Fig. 28 Configuration screen of the “iftop” interface.....	24
Fig. 29 ntop and nProbe output view for application classification	25
Fig. 30 iw command output for the wireless interface.....	26
Fig. 31 Output of the FER and delivery probability script	26
Fig. 32 Network topology graphic from Zenmap software	27
Fig. 33 Evaluation of the near WiFi networks over time	28
Fig. 34 Evaluation of the near WiFi networks by channels	28
Fig. 35 Wireless Community Network example schema	29

INTRODUCTION

Internet is the biggest and more heterogeneous network, formed by uncountable personal computers, connecting devices, servers... connecting thousands of millions of people around the world.

From its origins fifty years ago, Internet has evolved significantly in terms of usage and performance. Several infrastructures have been built in order to widely offer access to this network and nowadays it's seen as a basic service for the citizen; a service that is billed to the user by the telecommunication operators in function of its performance.

The access to the Internet has been wired through static desktop computers; but in the last years the wireless connections have spread their presence due to the increasingly users' mobility with their laptops, tablets and smart phones. However the backbone links which connect the major networks are still wired because of their significantly higher capacity compared with the wireless ones.

Due to the facility and the lower cost of installing a wireless mesh network¹, this solution has been well accepted in some regions and environments. A wireless mesh community network can be seen as a connectivity solution for towns far from the cities or a performance improvement for users who have a poor quality wired connection.

Many projects have been created worldwide with the common goal to deploy an open wireless network available for the citizens. These are very interesting projects because they promote the cooperation between individuals, companies and different kind of entities converging into a common goal. Also they are usually closely related to the open source initiative².

This project is aligned with this initiative with the aim to help the administrators of this wireless mesh networks. The inherent wireless part of the networks makes their topology constantly changing; their links unreliable and their performance oscillate through time depending also on environmental factors.

Due to all of these network characteristics, it's difficult to manage a network of this kind. In this project a set of measurements and tools are defined to obtain feedback about the status of the network, so being able to detect and maybe correct problems in it.

A wireless community network has no any single administrator (person or organization), it belongs to the community and any person in it can take an active role in the network management. It would be a nonsense trying to design a centralized management system for this kind of network, so it's not the objective of the project.

¹ http://en.wikipedia.org/wiki/Wireless_mesh_network

² <http://opensource.org/>

Smaller communities can be connected to the community network and they usually have a technical responsible for this link establishment and aftercare. This person or organization would require certain tools to evaluate the status of this connection and the users' behavior inside this new piece of the whole network.

The main objective of this project is to provide a set of tools to this entity in order to evaluate the piece of the network that administrates. The evaluation zone will be defined from the terminal user devices until the entrance point to the community network which is usually a gateway owned by the organization or user that joined it.

All the set of tools can be very useful to determine the behavior of the piece of network that is administrated. The administrator could obtain traffic statistics, hardware information of the devices connected in the network, discover the network topology, some characteristics of the environment... All these measurements would help the administrator to improve the performance of the network by the detection of possible malfunctions in it.

It will be described how to obtain each measurement and what conclusions can be extracted from its possible results.

CHAPTER 1. WIRELESS COMMUNITY NETWORKS

In wireless community networks (WCN) there isn't a single traditional telecommunications operator behind, usually the users are who share their Internet connections. The only constraint to access to the network is to have Wi-Fi equipment and use it as a node of the network too, so other traffic flows from different users can be routed through it.

An open IEEE 802.11³ standard wireless mesh network is an easy way to share bandwidth among several users in a certain community. Instead of the classical station bases, are used fixed wireless nodes. In a mesh network topology there are usually two kind of different links between nodes: point-to-multipoint links with omnidirectional antennas and point-to-point links using unidirectional antennas.

Any mobile device with a WiFi interface could be connected to this kind of network (with the administrator permission) and also fixed ones directly wired connected to the routing devices as it can be seen in the figure.

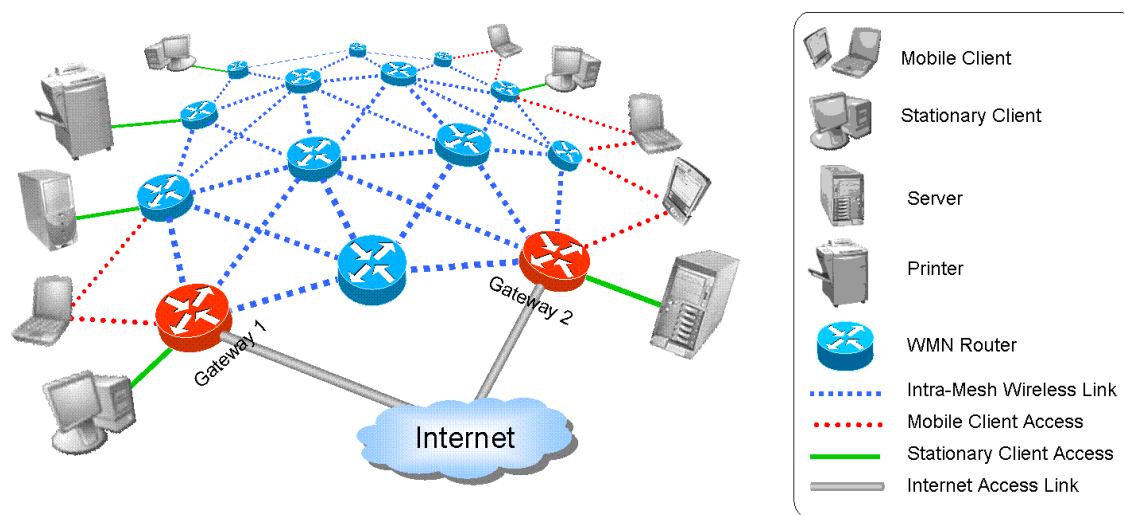


Fig. 1 Wireless mesh network topology example

The Guifi.net⁴ project is the most representative example and the biggest open wireless community network in Catalonia. The network is built by its users so they are also its owners, who can be individuals, companies and public administrations.

guifi.net

Fig. 2 Guifi.net project logo

³ <http://standards.ieee.org/about/get/802/802.11.html>

⁴ <http://guifi.net/>

It's an open network because all the configuration files are published to allow to anyone to improve and maintain it. There isn't any central entity that applies restrictions like contents or bandwidth bounding, which is the main limitation, directly related with the users' fees in the traditional operators' networks.

The Guifi.net foundation was created in 2008 and in the mid-2013 there are more than 32.000 nodes. In the following map it can be seen all the links and nodes represented over the Catalonia region.



Fig. 3 Links and nodes representation over Google Maps

CHAPTER 2. WCN NODES AND TOOLS

This project is aimed to be used majorly on routing devices with open source operating systems, but also there are software or measurements that have to be done in the users' computers.

In this chapter are defined how the device should be configured to obtain all the measurements defined in chapter 3. It's specified the operating system and a set of software programs that need a special installation, usually more complicated that just install the package.

2.1. OpenWrt

OpenWrt⁵ is the chosen embedded open source operating system because allows a full custom configuration to the administrator, working as an almost complete Linux command interface with a package manager to upgrade it.

It's a freeware software and the operating system image can be downloaded from the website; it's an empty image, so there isn't any pre-configuration, all must be done by the administrator.

It's the trade-off of this solution, the owner can enjoy a full customization but, in the other hand, even the simplest configuration must be done. Also there's a lot of documentation and forums in the Internet devoted to this solution and the possible issues.

This operating system will be installed in the wireless community network nodes allowing to install all the required software and to execute the scripts to evaluate the network status.



Fig. 4 OpenWrt logo

2.2. snmpd

The SNMP⁶ (Simple Network Management Protocol) is an application protocol that stores at the device databases called MIBs (Management Information Base). MIBs are focused on a singular topic; for example there's a MIB devoted to the network interfaces, other to the BGP⁷ routing protocol...

⁵ <https://openwrt.org/>

⁶ http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

⁷ https://en.wikipedia.org/wiki/Border_Gateway_Protocol

The `snmpd`⁸ is the service that manages all of these databases in the system and manages its access from remote systems. The MIBs are hierarchical databases composed by objects identified with a unique identifier called OID.

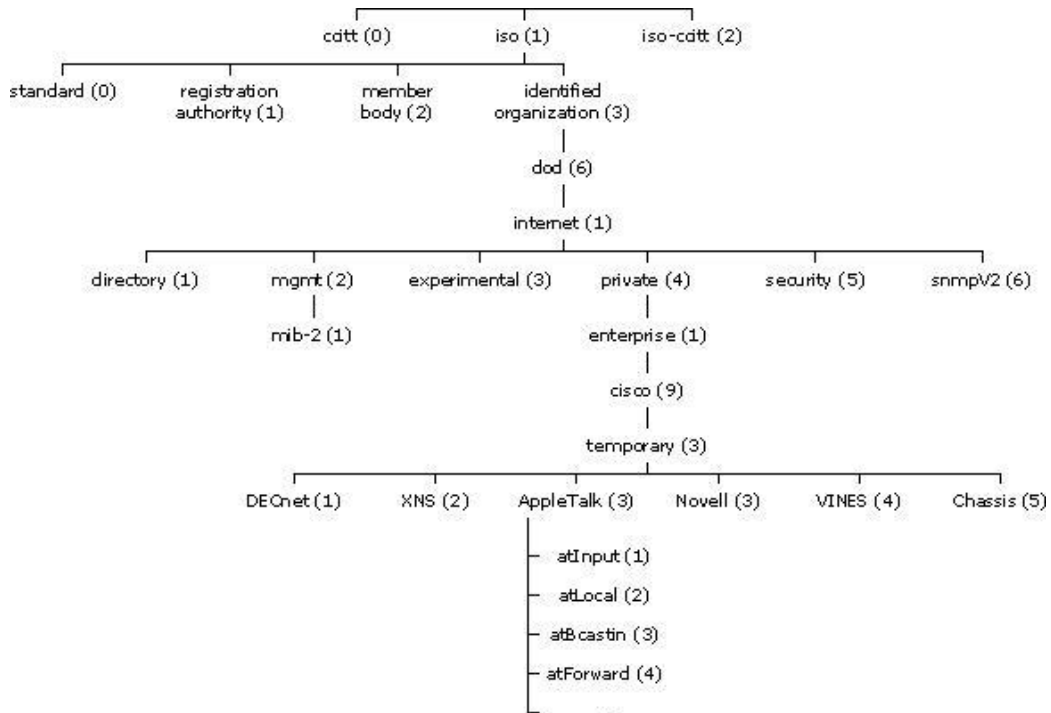


Fig. 5 MIB structure example

Required packages:

- `snmpd`
- `snmp-mibs-downloader`

The first package installs the SNMP daemon to obtain all the service in the device but it's empty at the beginning, there aren't any MIBs. The second one downloads all the supported MIBs and installs them; there are more than 340 available at this moment.

The real-time data obtained from the node via the SNMP service is used in some measurements specified in chapter 3 like the link status.

⁸ <http://www.net-snmp.org/>

2.3. ntop

ntop⁹ is a network traffic analyzer based on libpcap¹⁰ and has a web-based interface, so it's needed also a web server to work with it. It allows classifying traffic by protocols and other criteria and allows storing the captured packets.

Required packages:

- Apache2¹¹
- Ntop



Fig. 6 ntop logo

2.3.1. nProbe

nProbe¹² is an extension for the ntop traffic analyzer, its main functionality is to classify traffic also by layer 7 information such as the used application, like Facebook or Mozilla Firefox.

The ntop software is used to plot the node traffic shape and the nProbe allows classifying it with layer 7 parameters.

2.4. bandwidthd

This is a web based software with the purpose to plot into graphs the bandwidth usage. Bandwidthd¹³ captures the host traffic using the libpcap library [X] and stores it into SQL databases, so a part of the main package it's necessary to install before the web server (Apache) and the SQL (PostgreSQL¹⁴ in this case) daemon to make it work.

Required packages:

- Postgresql
- Apache2
- Bandwidthd

⁹ <http://www.ntop.org/>

¹⁰ <http://www.tcpdump.org/>

¹¹ <http://httpd.apache.org/>

¹² <http://www.ntop.org/products/nprobe/>

¹³ <http://bandwidthd.sourceforge.net/>

¹⁴ <http://www.postgresql.org/>

After all these packages have been installed, the default access to the website will be at “http://[HOST_ADDRESS]/bandwidthd” where [HOST_ADDRESS] is the IP address of the device where the software has been installed.

The bandwidthd software is used to view the traffic shape that passes through the wireless network node.



Fig. 7 bandwidthd logo

2.5. nmap

Nmap¹⁵ is an open source tool used to tasks related to network discovery and security auditing. It allows not only discovering which devices are online in the network, also it's able to scan all the available open ports and which service is running through it.

Many other features can be performed by this software but the mentioned before are the most useful ones for this project purpose.



Fig. 8 nmap logo

2.5.1. Zenmap

Zenmap¹⁶ is the graphical interface for the nmap software, making easier to use the command line features. Also the scan results can be saved and viewed later.

¹⁵ <http://nmap.org/>

¹⁶ <http://nmap.org/zenmap/>

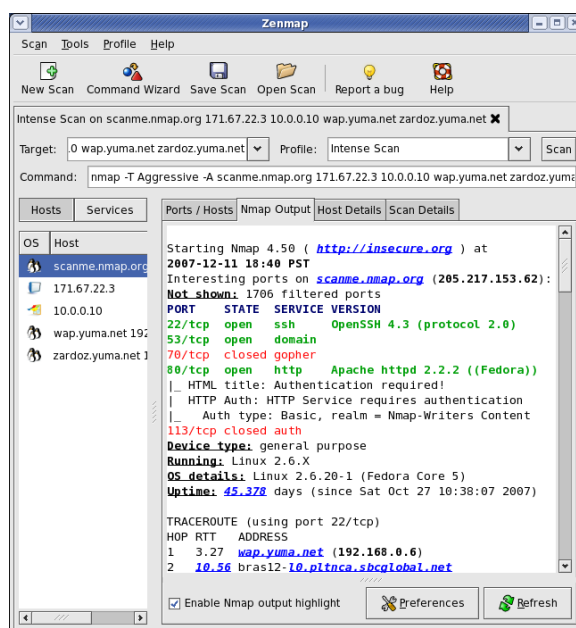


Fig. 9 Zenmap interface

The nmap software is used in this project to discover online network devices in the network and to list their open ports with security purposes. The Zenmap interface allows the administrator to plot the discovered network devices in a graphical and more intuitive way.

2.6. LinSSID

LinSSID¹⁷ is an open source program equivalent to the inSSIDer¹⁸ software for Windows and can be downloaded easily with the Linux package manager. Its functionality is to scan all the near WiFi networks and display it to a graphical interface.

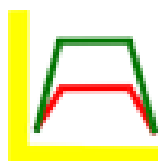


Fig. 10 LinSSID logo

The LinSSID software will be used to detect all the WiFi networks available near the host and to analyze their signal power and operating channel.

¹⁷ <http://sourceforge.net/projects/linssid/>

¹⁸ <http://www.metageek.net/products/inssider/>

CHAPTER 3. LIST OF MEASUREMENTS

The main objective of this chapter is to define a set of useful measurements and tools for a wireless mesh network administrator to evaluate, maintain and improve it. This list of measurements based in different software solutions is the best approximation because there isn't any open source software that covers all the measurements described below.

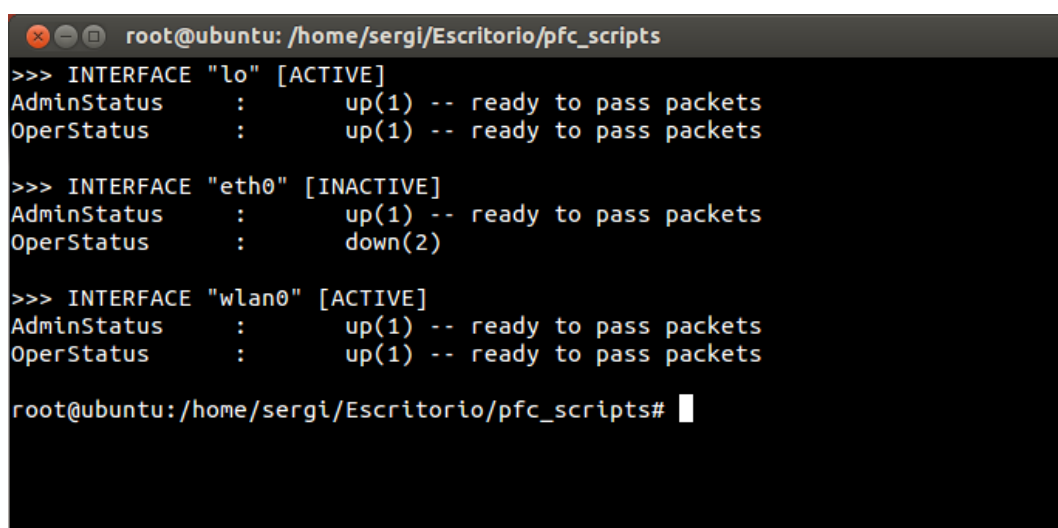
For each measurement it will be described its purpose, the output obtained and the deductions that can be extracted from them.

3.1. Active links. Links used by routes in the network

Have all the interfaces information can be very useful to detect if any interface is down for any reason. The status of the router interfaces can be determined by the properties of the SNMP object ifEntry inside the IF-MIB¹⁹:

- ifAdminstatus: integer value that indicates the status of the interface, three different values, testing (3), down (2) and up (1).
- ifOperStatus: Seven different values for the current operational state of the interface. Up(1) ready to pass packets, down(2), testing(3) in some test mode, unknown(4) status can't be determined for some reason, dormant(5), notPresent(6) some component is missing, lowerLayerDown(7) down due to state of lower-layer interface(s).

Only if both integer values are 1, it can be deduced that a link is active. So the script used in this case gets via SNMP these couple of values for each interface, shows them grouped by interface like in the following picture.:



```
root@ubuntu: /home/sergi/Escritorio/pfc_scripts
>>> INTERFACE "lo" [ACTIVE]
AdminStatus      :      up(1) -- ready to pass packets
OperStatus       :      up(1) -- ready to pass packets

>>> INTERFACE "eth0" [INACTIVE]
AdminStatus      :      up(1) -- ready to pass packets
OperStatus       :      down(2)

>>> INTERFACE "wlan0" [ACTIVE]
AdminStatus      :      up(1) -- ready to pass packets
OperStatus       :      up(1) -- ready to pass packets

root@ubuntu: /home/sergi/Escritorio/pfc_scripts#
```

Fig. 11 Output of the route links script

¹⁹ <http://net-snmp.sourceforge.net/docs/mibs/interfaces.html>

Other way to list all the available interfaces together with the host routes is to execute the “nmap --iflist” command:

```
root@ubuntu: /home/sergi/Escritorio
root@ubuntu: /home/sergi/Escritorio# nmap --iflist

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-26 17:37 CEST
*****INTERFACES*****
DEV (SHORT) IP/MASK          TYPE  UP MTU  MAC
eth0 (eth0) 192.168.1.133/24      ethernet up 1500 F4:6D:04:B7:14:7F
eth0 (eth0) fe80::f66d:4ff:feb7:147f/64 ethernet up 1500 F4:6D:04:B7:14:7F
lo (lo) 127.0.0.1/8          loopback up 65536
lo (lo) ::1/128           loopback up 65536
wlan0 (wlan0) 192.168.1.134/24      ethernet up 1500 E0:B9:A5:5F:90:AF
wlan0 (wlan0) fe80::e2b9:a5ff:fe5f:90af/64 ethernet up 1500 E0:B9:A5:5F:90:AF

*****ROUTES*****
DST/MASK  DEV  GATEWAY
192.168.1.0/24 eth0
192.168.1.0/24 eth0
169.254.0.0/16 eth0
0.0.0.0/0    eth0 192.168.1.1

root@ubuntu: /home/sergi/Escritorio#
```

Fig. 12 List of localhost interfaces and routes

3.2. All links that receive at least one beacon in a measurement interval

Having this information can be useful to determine low quality or unused links, which don't process any data due to the lack of users or the poor quality of the medium.

To detect if an interface has received any packet, the `ifInOctets` property can be obtained for each interface (`ifEntry`) and after the desired period of time, measure it again and compare the values. All the data can be found inside the IF-MIB. Then the active interfaces will be the ones where the property has changed.

```
root@ubuntu: /home/sergi/Escritorio/pfc_scripts
The interface will be considered ACTIVE if it have received at least
one packet in 5 seconds
>>> INTERFACE "lo"      -->  ACTIVE
>>> INTERFACE "eth0"   -->  INACTIVE
>>> INTERFACE "wlan0"  -->  ACTIVE
root@ubuntu: /home/sergi/Escritorio/pfc_scripts#
```

Fig. 13 Output of the active links

3.3. Total amount of data transferred by clients

Having the total volume of data transferred in a given time period can be used to determine the network load, for example. So the total amount of data transferred through each interface can be a key factor to correct punctual problems like bottlenecks in the network, which produce packet losses and a bad performance.

The amount of data transferred of each interface can be measured in the field `ifOutOctets` from all the `ifEntry` objects in the IF-MIB, then after a time period, it's measured again. Adding all the difference between the two measurements in the output bytes, the total data transmitted in the network during this period is obtained. It could be measured also the input packets but only input or output must be measured, otherwise the data is counted double.

```
root@ubuntu: /home/sergi/Escritorio/pfc_scripts
Amount of data transferred in 10 seconds
>>> INTERFACE "lo"      :      1.112 KB
>>> INTERFACE "eth0"   :           0 B
>>> INTERFACE "wlan0"  :     541.324 KB
root@ubuntu: /home/sergi/Escritorio/pfc_scripts#
```

Fig. 14 Output of the amount of data script

3.4. Traffic shape

Measuring the total amount of data transmitted for a given period in different instants during the day, the traffic shape can be obtained. Also it can be done for different periods like a week or a month.

For this measurement the bandwidthd and the ntop softwares can be used. Both show a graph of the network traffic; it can be seen for different periods of time.

The traffic can also be separated depending on the protocol, making easier to measure and quantify the amount of traffic compared with the others.

Having this graphical view of the network load over time have a different purpose that the previous measurement, the total amount of data transferred by clients. The previous measurement is an average estimation of the total data transferred in a certain period, but in this case the traffic bursts are taken into account, so the peaks can be detected and located in time. The peaks can be predicted like the increment of traffic at the evening when the users arrive home after work or at the weekends. Having this information, the administrator can detect when are more used the network resources and if there's necessary any upgrade.

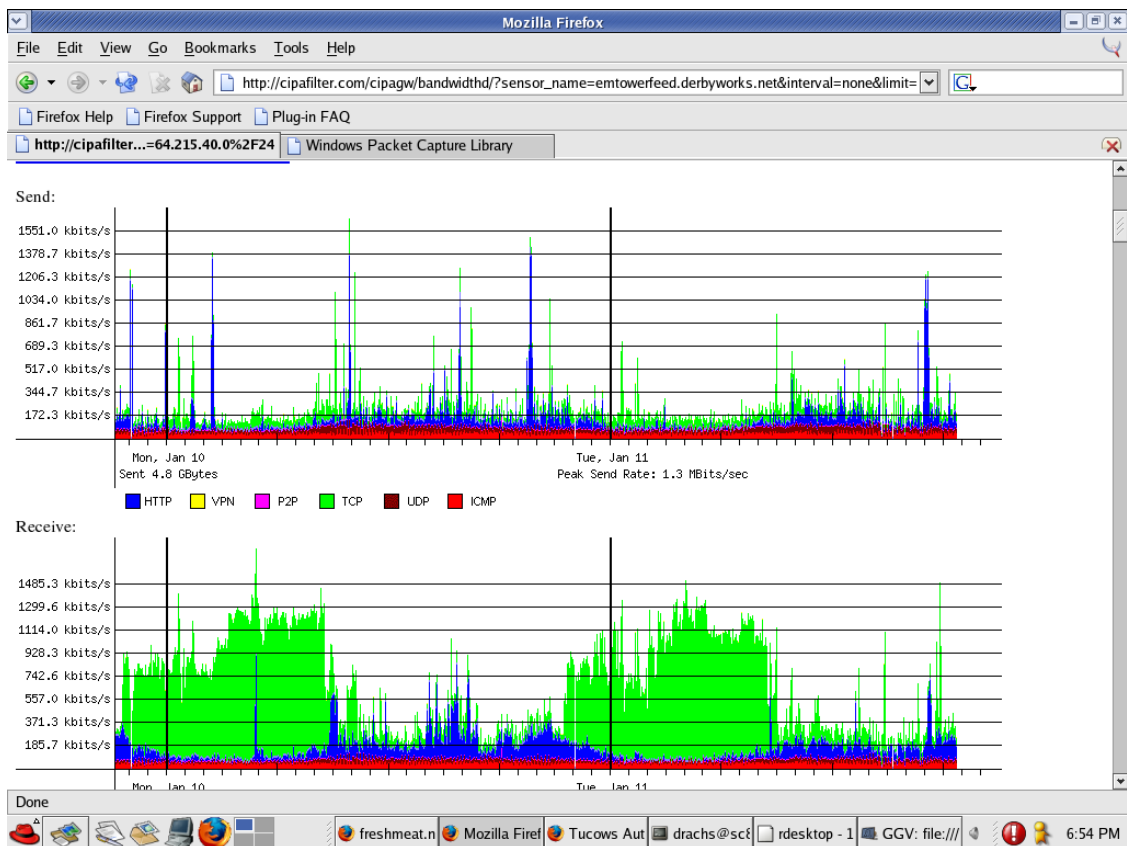


Fig. 15 Traffic graph from the “Bandwidthd” tool

Network Load Statistics

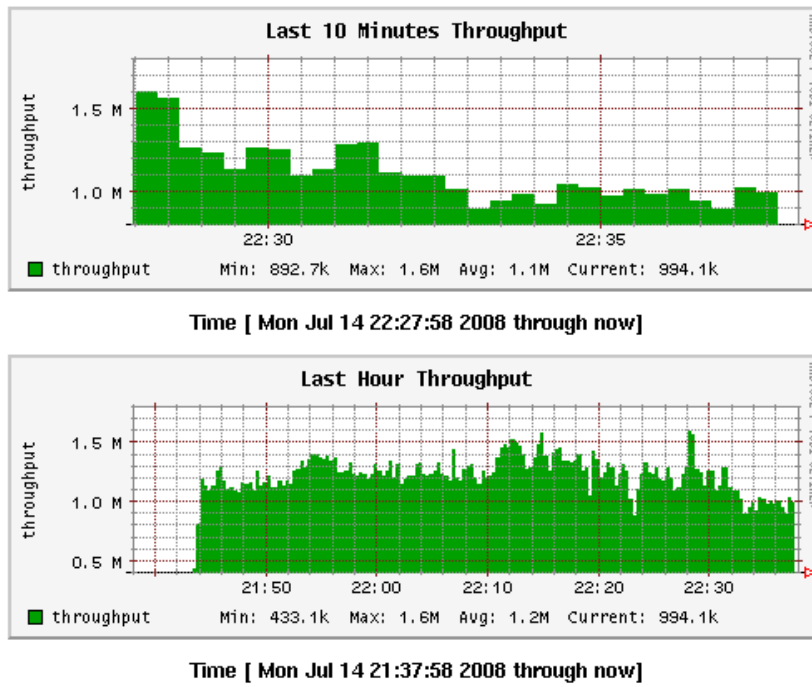


Fig. 16 ntop traffic graph in different time periods

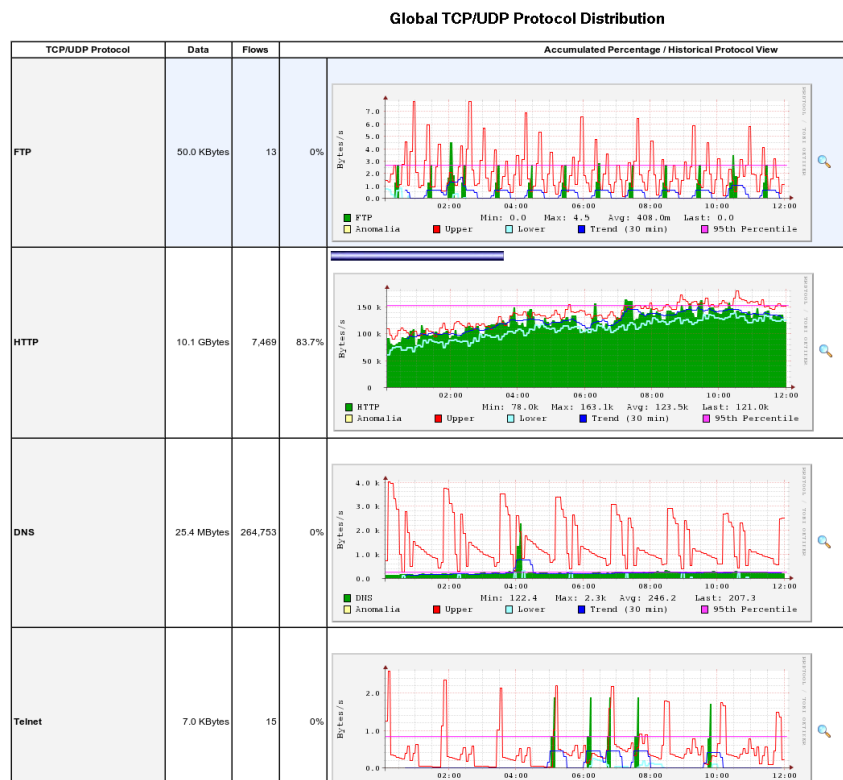


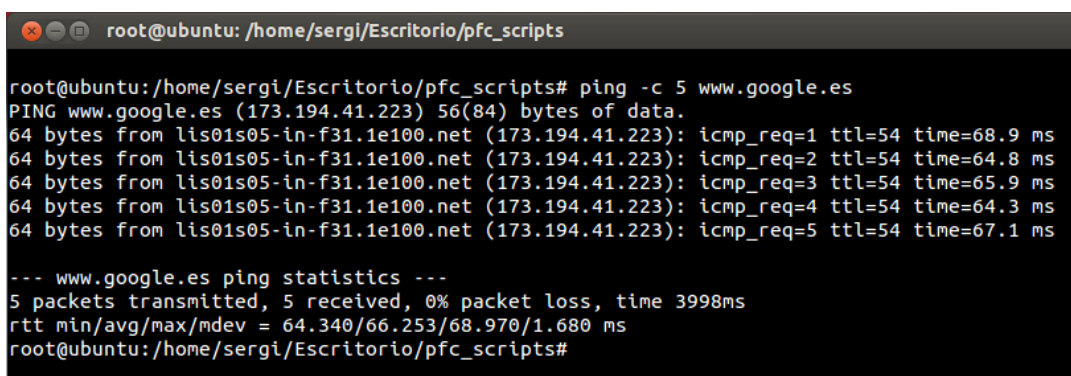
Fig. 17 ntop traffic graph separated by protocols

3.5. Latency and Jitter

The latency can be defined as the time that a packet reaches the other side of the communication or the time that lasts the packet to arrive again to the origin. Usually and in this case, the second approach is taken into account because the measurement equipment only has to be allocated in one network equipment. The jitter is the variance of the different latency measurements obtained.

The latency to a concrete device can be used to determine how congested the network between the two points is. The jitter can be used to deduce the status of the physical medium of transmission, if it's high, significant changes over time in it can be deduced. For a single latency measurement it would be enough making a ping to the desired destination.

It can be seen below the output of a PING²⁰ command, in this case to Google website. The latency is the parameter called "time" and it can be extracted and processed easily. Also at the end of the command the minimum, maximum, average and maximum deviation values can be found and there's no need to calculate them.

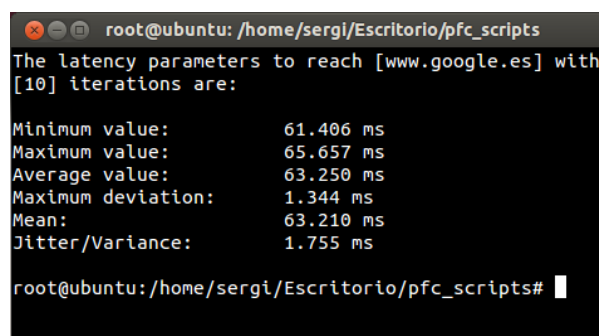


```
root@ubuntu: /home/sergi/Escritorio/pfc_scripts
root@ubuntu: /home/sergi/Escritorio/pfc_scripts# ping -c 5 www.google.es
PING www.google.es (173.194.41.223) 56(84) bytes of data:
64 bytes from lis01s05-in-f31.1e100.net (173.194.41.223): icmp_req=1 ttl=54 time=68.9 ms
64 bytes from lis01s05-in-f31.1e100.net (173.194.41.223): icmp_req=2 ttl=54 time=64.8 ms
64 bytes from lis01s05-in-f31.1e100.net (173.194.41.223): icmp_req=3 ttl=54 time=65.9 ms
64 bytes from lis01s05-in-f31.1e100.net (173.194.41.223): icmp_req=4 ttl=54 time=64.3 ms
64 bytes from lis01s05-in-f31.1e100.net (173.194.41.223): icmp_req=5 ttl=54 time=67.1 ms

--- www.google.es ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 64.340/66.253/68.970/1.680 ms
root@ubuntu: /home/sergi/Escritorio/pfc_scripts#
```

Fig. 18 Ping command example to Google website

An example output of the script communicating with Google website and 10 different iterations could be:



```
root@ubuntu: /home/sergi/Escritorio/pfc_scripts
The latency parameters to reach [www.google.es] with
[10] iterations are:

Minimum value:      61.406 ms
Maximum value:     65.657 ms
Average value:     63.250 ms
Maximum deviation: 1.344 ms
Mean:              63.210 ms
Jitter/Variance:   1.755 ms

root@ubuntu: /home/sergi/Escritorio/pfc_scripts#
```

Fig. 19 Output of the latency and jitter script

²⁰ [http://en.wikipedia.org/wiki/Ping_\(networking_utility\)](http://en.wikipedia.org/wiki/Ping_(networking_utility))

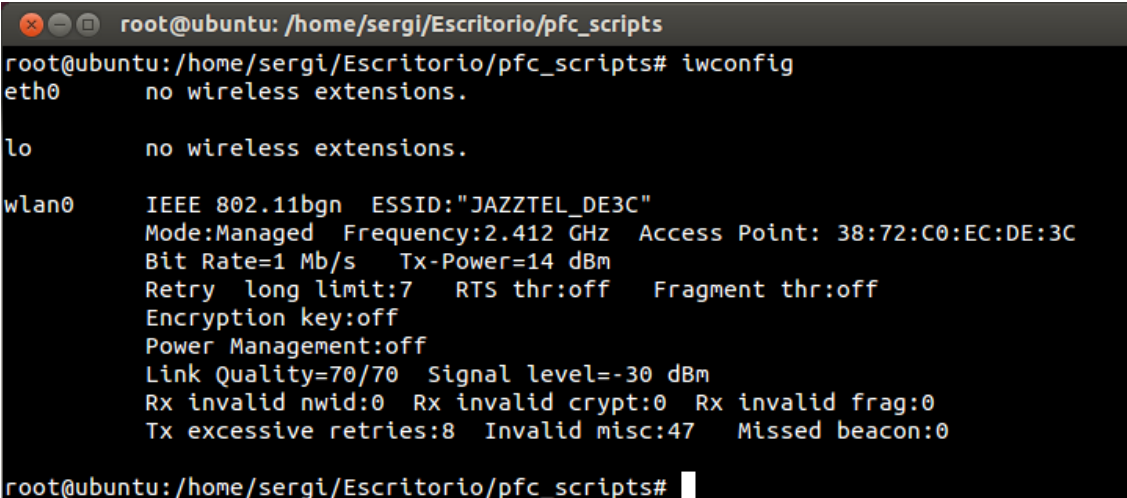
3.6. Signal parameters

It can be interesting to measure the link quality to guess the performance that the user could experiment. A bad physical link quality would derive directly in a poor connection increasing the transmission errors and a throughput decrease.

The medium interferences can't be detected with the network routers because the hardware devices are not designed to evaluate or measure them. On the other hand, the routers can measure the signal power from their own interfaces.

A set of wireless parameters can be observed for each wireless interface using the "iwconfig"²¹ command, the more interesting for an administrator:

- Name of the network where it's connected (ESSID).
- Frequency.
- Bit rate.
- Transmission power.
- MAC address of the access point.
- Link quality.
- Signal level.



```
root@ubuntu: /home/sergi/Escritorio/pfc_scripts
root@ubuntu:/home/sergi/Escritorio/pfc_scripts# iwconfig
eth0      no wireless extensions.

lo        no wireless extensions.

wlan0     IEEE 802.11bgn  ESSID:"JAZZTEL_DE3C"
          Mode:Managed  Frequency:2.412 GHz  Access Point: 38:72:C0:EC:DE:3C
          Bit Rate=1 Mb/s   Tx-Power=14 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=70/70  Signal level=-30 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:8  Invalid misc:47  Missed beacon:0

root@ubuntu:/home/sergi/Escritorio/pfc_scripts#
```

Fig. 20 Output of the iwconfig command

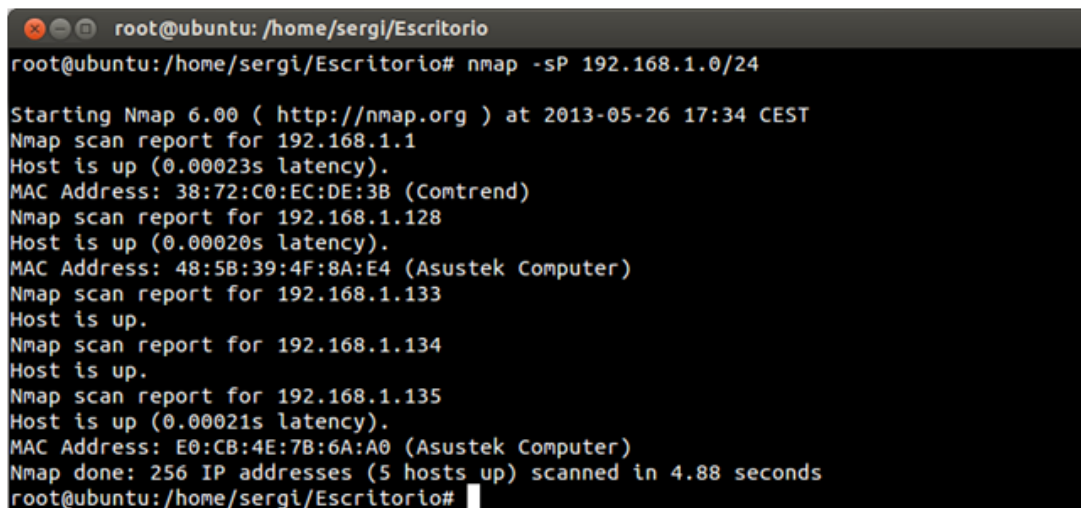
²¹ http://www.linuxcommand.org/man_pages/iwconfig8.html

3.7. Discover the network devices

The amount of hosts and connection devices is a very important factor in terms of congestion and scalability of the network, it's not the same to have 10 devices transmitting in the network that 100. The nmap software was created for this purpose, but only a couple of measurements among the large list of available ones have sense in this context. It can be used also to detect intruders that are connected to the network without any authorization.

First of all a host discovery by "Ping" command (ping scan) can be used. It's a very simple mechanism to detect the network devices, it consists in send a ping request to every possible IP address in the network. If the address sends a response, it will be considered as an online device. The main problem here is that some devices could have the ICMP²² requests like ping blocked for security reasons so these devices wouldn't be detected.

The command to execute this scan in the network is "nmap -sP [NETWORK_IP/MASK]".

A terminal window showing the execution of an nmap ping scan on the network 192.168.1.0/24. The output lists five discovered hosts with their MAC addresses and manufacturers: 192.168.1.1 (Comtrend), 192.168.1.128 (Asustek Computer), 192.168.1.133, 192.168.1.134, and 192.168.1.135 (Asustek Computer). The scan completed in 4.88 seconds.

```
root@ubuntu: /home/sergi/Escritorio
root@ubuntu:/home/sergi/Escritorio# nmap -sP 192.168.1.0/24

Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-26 17:34 CEST
Nmap scan report for 192.168.1.1
Host is up (0.00023s latency).
MAC Address: 38:72:C0:EC:DE:3B (Comtrend)
Nmap scan report for 192.168.1.128
Host is up (0.00020s latency).
MAC Address: 48:5B:39:4F:8A:E4 (Asustek Computer)
Nmap scan report for 192.168.1.133
Host is up.
Nmap scan report for 192.168.1.134
Host is up.
Nmap scan report for 192.168.1.135
Host is up (0.00021s latency).
MAC Address: E0:CB:4E:7B:6A:A0 (Asustek Computer)
Nmap done: 256 IP addresses (5 hosts up) scanned in 4.88 seconds
root@ubuntu:/home/sergi/Escritorio#
```

Fig. 21 Devices discovered by nmap in the network

²² https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol

The previous scan could be very simple and lack of extra information a part of the existence of a device that answers pings behind the address. Also the open ports and the service which is devoted to could be very useful information in terms of security, for example.

So the command to obtain also the open ports for each device is “nmap -T5 [NETWORK_IP/MASK]”.

```
root@ubuntu: /home/sergi/Escritorio
root@ubuntu: /home/sergi/Escritorio# nmap -T5 192.168.1.0/24
Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-26 17:39 CEST
Nmap scan report for 192.168.1.1
Host is up (0.0028s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
139/tcp   open  netbios-ssn
5431/tcp  open  park-agent
MAC Address: 38:72:C0:EC:DE:3B (Comtrend)

Nmap scan report for 192.168.1.128
Host is up (0.00020s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
2869/tcp  open  icslap
MAC Address: 48:5B:39:4F:8A:E4 (Asustek Computer)

Nmap scan report for 192.168.1.133
Host is up (0.0000060s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
80/tcp    open  http
3000/tcp  open  ppp

Nmap scan report for 192.168.1.134
Host is up (0.0000050s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
80/tcp    open  http
3000/tcp  open  ppp

Nmap scan report for 192.168.1.135
Host is up (0.00015s latency).
All 1000 scanned ports on 192.168.1.135 are filtered
MAC Address: E0:CB:4E:7B:6A:A0 (Asustek Computer)

Nmap done: 256 IP addresses (5 hosts up) scanned in 18.38 seconds
root@ubuntu: /home/sergi/Escritorio#
```

Fig. 22 Devices and their open ports discovered in the network

3.8. Device information and configuration

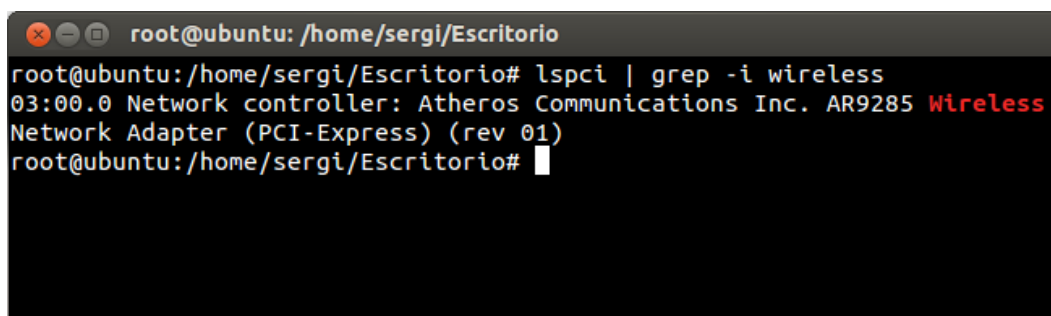
The network administrator may require deep knowledge about the network adapter configuration, not only about layer 2 and 3, more about physical layer. For discover its capabilities, performance configuration or driver updates all this information can be taken into account.

UNIX operating systems have already a set of commands to access to all the hardware components in the system.

3.8.1. lspci

`lspci`²³ command is used to get all the information about the PCI²⁴ buses and the devices. There are several arguments to obtain different information about the hardware selected.

First the wireless device must be identified, it's only necessary to list all the PCI components and search for the correct one. To reduce the output and make it simpler, showed information has been filtered by the word "wireless" so only the wireless device appears.



```
root@ubuntu: /home/sergi/Escritorio
root@ubuntu:/home/sergi/Escritorio# lspci | grep -i wireless
03:00.0 Network controller: Atheros Communications Inc. AR9285 Wireless
Network Adapter (PCI-Express) (rev 01)
root@ubuntu:/home/sergi/Escritorio#
```

Fig. 23 lspci output for the wireless device

In the previous output, the device identifier can be seen at the beginning (03:00.0) and with it, more information about its driver can be obtained.

²³ http://www.linuxcommand.org/man_pages/lspci8.html

²⁴ http://en.wikipedia.org/wiki/Conventional_PCI

```

root@ubuntu: /home/sergi/Escritorio

root@ubuntu:/home/sergi/Escritorio# lspci -vv -s 03:00.0
03:00.0 Network controller: Atheros Communications Inc. AR9285 Wireless Network Adapter (PCI-Express) (rev 01)
Subsystem: AzureWave AW-NE785 / AW-NE785H 802.11bgn Wireless Full or Half-size Mini PCIe Card
Physical Slot: 1
Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
Latency: 0, Cache Line Size: 64 bytes
Interrupt: pin A routed to IRQ 17
Region 0: Memory at d2a00000 (64-bit, non-prefetchable) [size=64K]
Capabilities: [40] Power Management version 3
Flags: PMEClk- DSI- D1+ D2- AuxCurrent=375mA PME(D0+,D1+,D2-,D3hot+,D3cold+)
Status: D0 NoSoftRst- PME-Enable- DSel=0 DScale=0 PME-
Capabilities: [50] MSI: Enable- Count=1/1 Maskable- 64bit-
Address: 00000000 Data: 0000
Capabilities: [60] Express (v2) Legacy Endpoint, MSI 00
DevCap: MaxPayload 128 bytes, PhantFunc 0, Latency L0s <512ns, L1 <64us
ExtTag- AttnBtn- AttnInd- PwrInd- RBE+ FlrReset-
DevCtl: Report errors: Correctable- Non-Fatal- Fatal- Unsupported-
RxdOrd+ ExtTag- PhantFunc- AuxPwr- NoSnoop-
MaxPayload 128 bytes, MaxReadReq 512 bytes
DevSta: CorrErr+ UncorrErr- FatalErr- UnsuppReq+ AuxPwr+ TransPend-
LnkCap: Port #0, Speed 2.5GT/s, Width x1, ASPM L0s L1, Latency L0 <512ns, L1 <64us
ClockPM- Surprise- LLActRep- BwNot-
LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- Retrain- CommClk+
ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
LnkSta: Speed 2.5GT/s, Width x1, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
DevCap2: Completion Timeout: Not Supported, TimeoutDis+
DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-
LnkCtl2: Target Link Speed: 2.5GT/s, EnterCompliance- SpeedDis-, Selectable De-emphasis: -6dB
Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
Compliance De-emphasis: -6dB
LnkSta2: Current De-emphasis Level: -6dB, EqualizationComplete-, EqualizationPhase1-
EqualizationPhase2-, EqualizationPhase3-, LinkEqualizationRequest-
Capabilities: [100 v1] Advanced Error Reporting
UESSta: DLP- SDES- TLP- FCP- CplmtTO- CplmtAbrt- UnxCplmt- RxOF- MalfTLP- ECRC- UnsupReq- ACSViol-
UEMSk: DLP- SDES- TLP- FCP- CplmtTO- CplmtAbrt- UnxCplmt- RxOF- MalfTLP- ECRC- UnsupReq- ACSViol-
UESvrt: DLP+ SDES+ TLP- FCP+ CplmtTO- CplmtAbrt- UnxCplmt- RxOF+ MalfTLP+ ECRC- UnsupReq- ACSViol-
CESSta: RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
CEMSk: RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
AERCap: First Error Pointer: 00, GenCap+ CGenEn- ChkCap+ ChkEn-
Capabilities: [140 v1] Virtual Channel
Caps: LPEVC=0 RefClk=100ns PATEntryBits=1
Arb: Fixed- WRR32- WRR64- WRR128-
Ctrl: ArbSelect=Fixed
Status: InProgress-
VC0: Caps: PATOffset=00 MaxTimeSlots=1 RejSnoopTrans-
Arb: Fixed- WRR32- WRR64- WRR128- TWRR128- WRR256-
Ctrl: Enable+ ID=0 ArbSelect=Fixed TC/VC=ff
Status: NegoPending- InProgress-
Capabilities: [160 v1] Device Serial Number 00-15-17-ff-ff-24-14-12
Capabilities: [170 v1] Power Budgeting <?>
Kernel driver in use: ath9k

root@ubuntu:/home/sergi/Escritorio#

```

Fig. 24 lspci wireless driver information

3.8.2. modinfo

The `modinfo`²⁵ command allows obtaining information about kernel modules²⁶. In the previous `lspci` measurement, in the last line, the wireless module can be obtained as the “Kernel driver in use” value, “ath9k” in this example.

```

root@ubuntu: /home/sergi/Escritorio
root@ubuntu:/home/sergi/Escritorio# modinfo ath9k
filename:          /lib/modules/3.8.0-22-generic/kernel/drivers/net/wireless/ath/ath9k/ath9k.ko
license:          Dual BSD/GPL
description:      Support for Atheros 802.11n wireless LAN cards.
author:           Atheros Communications
srcversion:       998C2D0E2F8B67646D6C273
alias:            platform:qca955x_wmac
alias:            platform:ar934x_wmac
alias:            platform:ar933x_wmac
alias:            platform:ath9k
alias:            pci:v0000168Cd00000036sv*sd*bc*sc*i*
alias:            pci:v0000168Cd00000037sv*sd*bc*sc*i*
alias:            pci:v0000168Cd00000034sv*sd*bc*sc*i*
alias:            pci:v0000168Cd00000033sv*sd*bc*sc*i*
alias:            pci:v0000168Cd00000032sv*sd*bc*sc*i*
alias:            pci:v0000168Cd00000030sv*sd*bc*sc*i*
alias:            pci:v0000168Cd0000002Esv*sd*bc*sc*i*
alias:            pci:v0000168Cd0000002Dsv*sd*bc*sc*i*
alias:            pci:v0000168Cd0000002Csv*sd*bc*sc*i*
alias:            pci:v0000168Cd0000002Bsv*sd*bc*sc*i*
alias:            pci:v0000168Cd0000002Asv*sd*bc*sc*i*
alias:            pci:v0000168Cd00000029sv*sd*bc*sc*i*
alias:            pci:v0000168Cd00000027sv*sd*bc*sc*i*
alias:            pci:v0000168Cd00000024sv*sd*bc*sc*i*
alias:            pci:v0000168Cd00000023sv*sd*bc*sc*i*
depends:           ath9k_hw,ath9k_common,mac80211,ath,cfg80211
intree:           Y
vermagic:         3.8.0-22-generic SMP mod_unload modversions
parm:             debug:Debugging mask (uint)
parm:             nohwcrypt:Disable hardware encryption (int)
parm:             blink:Enable LED blink on activity (int)
parm:             btcoex_enable:Enable wifi-BT coexistence (int)
parm:             enable_diversity:Enable Antenna diversity for AR9565 (int)
root@ubuntu:/home/sergi/Escritorio#

```

Fig. 25 `modinfo` output for the wireless kernel module

²⁵ http://www.linuxcommand.org/man_pages/modinfo8.html

²⁶ https://en.wikipedia.org/wiki/Loadable_kernel_module

3.8.3. hwinfo

The `hwinfo`²⁷ is not included by default in the UNIX operating systems so this package must be installed a part. The functionality of this command is to display a very complete set of hardware parameters.

```
root@ubuntu: /home/sergi
root@ubuntu:/home/sergi# hwinfo --wlan
> hal.1: read hal dataprocess 7128: arguments to dbus_move_error() were incorrect, assertion "(dest) == NULL || !dbus_error_is_set ((dest))" failed in file ../../dbus/dbus-errors.c line 282.
This is normally a bug in some application using the D-Bus library.
libhal.c 3483 : Error unsubscribing to signals, error=The name org.freedesktop.Hal was not provided by any .service files
17: PCI 300.0: 0282 WLAN controller
[Created at pci.318]
Unique ID: y9sn.apYioKQ9666
Parent ID: qTvu.tbAxFwydWRA
SysFS ID: /devices/pci0000:00/0000:00:1c.1/0000:03:00.0
SysFS BusID: 0000:03:00.0
Hardware Class: network
Model: "Atheros WLAN controller"
Vendor: pci 0x168c "Atheros Communications Inc."
Device: pci 0x002b
SubVendor: pci 0x1a3b
SubDevice: pci 0x1089
Revision: 0x01
Driver: "ath9k"
Driver Modules: "ath9k", "ath9k"
Device File: wlan0
Features: WLAN
Memory Range: 0xd2a00000-0xd2a0ffff (rw,non-prefetchable)
IRQ: 17 (no events)
HW Address: e0:b9:a5:5f:90:af
Link detected: yes
WLAN channels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14
WLAN frequencies: 2.412 2.417 2.422 2.427 2.432 2.437 2.442 2.447 2.452 2.457 2.462 2.467 2.472 2.484
WLAN encryption modes: WEP40 WEP104 TKIP CCMP
WLAN authentication modes: open sharedkey wpa-psk wpa-eap
Module Alias: "pci:v0000168Cd0000002Bsv00001A3Bsd00001089bc02sc80i00"
Driver Info #0:
  Driver Status: ath9k is active
  Driver Activation Cmd: "modprobe ath9k"
  Config Status: cfg=new, avail=yes, need=no, active=unknown
  Attached to: #7 (PCI bridge)
root@ubuntu:/home/sergi#
```

Fig. 26 hwinfo output for the wireless interface

²⁷ <http://www.hwinfo.com/>

3.9. Transfer rates

Also if there's a need to go deeper in information about the users' traffic, it can be used the "iftop"²⁸ command, a real-time traffic analyzer. It shows in a table the current bandwidth usage by pair of hosts in a single interface.

This information can be useful for the administrator to detect in real time which are the most relevant connections (in terms of bandwidth consumption) passing through the device.

```

root@ubuntu: /home/sergi/Escritorio/pfc_scripts
┌──────────┬──────────┬──────────┬──────────┬──────────┐
│          │ 195kb    │ 391kb    │ 586kb    │ 781kb    │ 977kb    │
├──────────┴──────────┴──────────┴──────────┴──────────┘
ubuntu.local:mdns => 224.0.0.251:mdns      284b  538b  421b
                  <=                    0b   0b   0b
ubuntu.local      => 66.1.216.87.static.jazz 0b   466b  255b
                  <=                    0b   0b   0b
ubuntu.local:32566 => 65.1.216.87.stati:domain 0b   58b   19b
                  <=                    0b  131b  44b
ubuntu.local:49916 => 66.1.216.87.stati:domain 0b   60b   40b
                  <=                    0b  120b  40b
ubuntu.local:63061 => 66.1.216.87.stati:domain 0b   60b   40b
                  <=                    0b  120b  40b
ubuntu.local:49916 => 65.1.216.87.stati:domain 0b   60b   20b
                  <=                    0b  120b  40b
ubuntu.local:6397  => 65.1.216.87.stati:domain 0b   60b   20b
                  <=                    0b  120b  40b
ubuntu.local:29861 => 65.1.216.87.stati:domain 0b   60b   20b
                  <=                    0b  120b  40b
ubuntu.local:63061 => 65.1.216.87.stati:domain 0b   60b   20b
                  <=                    0b  120b  40b
ubuntu.local:27993 => 65.1.216.87.stati:domain 0b   58b   19b
                  <=                    0b  115b  38b
├──────────┴──────────┴──────────┴──────────┴──────────┘
TX:          cum:  57,5kB  peak:  76,2rates:  572b  2,35kb  15,3kb
RX:          cum: 196kB   peak:  310kb  444b  2,64kb  52,2kb
TOTAL:       cum: 253kB   peak:  387kb  0,99kb 4,98kb  67,5kb

```

Fig. 27 Main screen of the "iftop" command

```

root@ubuntu: /home/sergi/Escritorio/pfc_scripts
┌──────────┬──────────┬──────────┬──────────┬──────────┐
│          │ 195kb    │ 391kb    │ 586kb    │ 781kb    │ 977kb    │
├──────────┴──────────┴──────────┴──────────┴──────────┘
Host display:
n - toggle DNS host resolution
s - toggle show source host
d - toggle show destination host
t - cycle line display mode

Port display:
N - toggle service resolution
S - toggle show source port
D - toggle show destination port
p - toggle port display

Sorting:
1/2/3 - sort by 1st/2nd/3rd column
< - sort by source name
> - sort by dest name
o - freeze current order

iftop, version 1.0pre2

General:
P - pause display
h - toggle this help display
b - toggle bar graph display
B - cycle bar graph average
T - toggle cumulative line totals
j/k - scroll display
f - edit filter code
l - set screen filter
L - lin/log scales
! - shell command
q - quit

```

```

TX:          cum:  59,7kB  peak:  63,1rates:  0b   683b  4,99kb
RX:          cum: 197kB   peak:  180kb  0b   695b  11,0kb
TOTAL:       cum: 257kB   peak:  243kb  0b   1,35kb  16,0kb

```

Fig. 28 Configuration screen of the "iftop" interface

²⁸ <http://www.ex-parrot.com/pdw/iftop/>

3.10. Application classes

In order to define the clients' behavior and improve the network features according to its type of usage, can be very interesting to determine which the data flow types are.

The ntop software with the nProbe update, the traffic captured at the device can be more effectively identified and not only by the port, also by the layer 7 application as it can be seen in the picture below.

Actually there are 150 applications identified like Facebook, Twitter, Skype, Whatsapp, WindowsUpdate, Grooveshark, Warcraft 3... So the administrator is able to detect for example if many users connect to social networks and can take the decision to forbid its access. On the other hand, it could be seen that some applications have a great presence in the network and a configuration adaptation could be applied to benefit them.

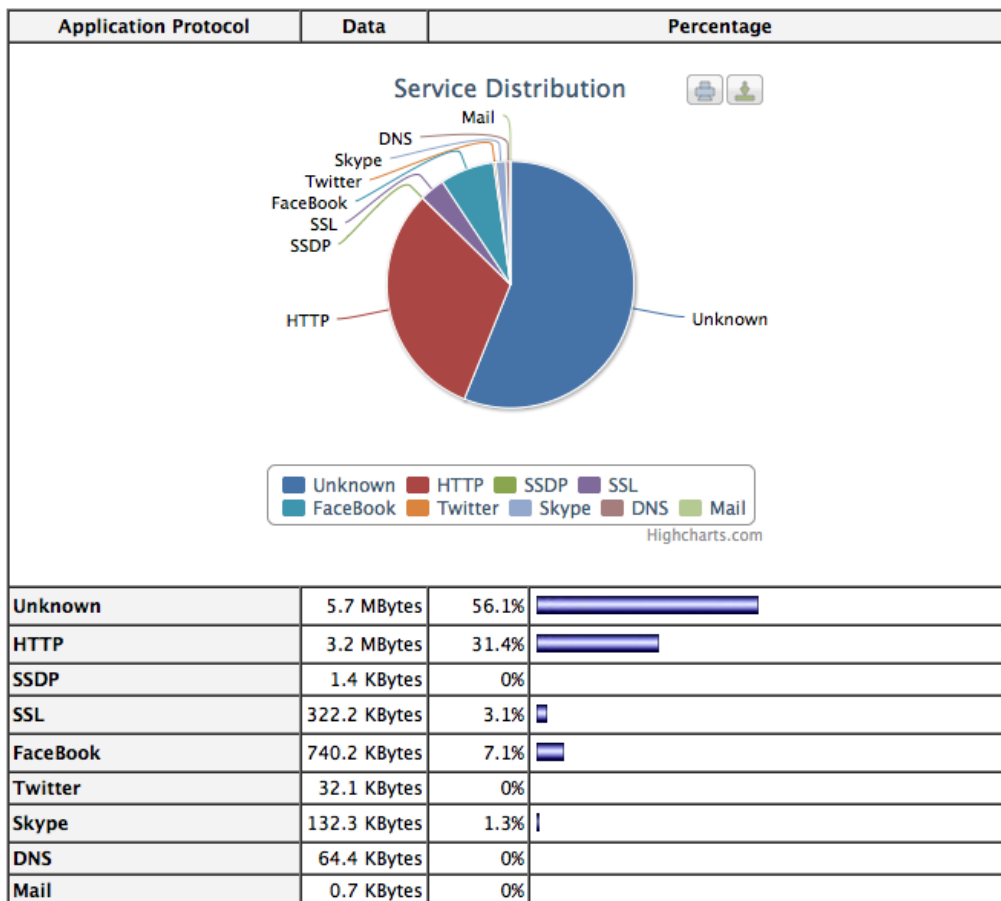


Fig. 29 ntop and nProbe output view for application classification

3.11. Frame Error Rate (FER) and Deliver probability

From these measurements the administrator can deduce information like how congested is the network or if any link has a bad performance due to environmental conditions. These are only deductions that can be made by knowing in advance the error ratio in the link, but it's not a direct consequence.

With the "iw"²⁹ command, the administrator can obtain all the parameters to calculate the Frame Error Ratio (FER) and the Deliver probability.

```

root@ubuntu: /home/sergi
root@ubuntu: /home/sergi# iw dev wlan0 station dump
Station 38:72:c0:ec:de:3c (on wlan0)
  inactive time: 38696 ms
  rx bytes: 3381810
  rx packets: 34258
  tx bytes: 74865
  tx packets: 1022
  tx retries: 307
  tx failed: 71
  signal: -34 dBm
  signal avg: -34 dBm
  tx bitrate: 1.0 MBit/s
  authorized: yes
  authenticated: yes
  preamble: long
  WMM/WME: yes
  MFP: no
  TDLS peer: no
root@ubuntu: /home/sergi#

```

Fig. 30 iw command output for the wireless interface

The following formulas explain how to calculate both parameters, there's only need to get the packets transmitted, the failed ones and the bit rate:

$$Deliver_probability[\%] = \left[1 - \frac{Tx_failed}{Tx_total}\right] * 100$$

$$Frame_Error_Rate[bps] = Bit_rate * \frac{Tx_failed}{Tx_total}$$

```

root@ubuntu: /home/sergi/Escritorio/pfc_scripts
Data for the [wlan0] interface:
MAC address: 38:72:c0:ec:de:3c
Transmitted bytes: 1747952 B
Transmitted packets: 16443 packets
Transmitted failed packets: 2022 packets
Bit Rate: 1 Mbps

Deliver probability: 87.70%
FER: 122.97 Kbps
root@ubuntu: /home/sergi/Escritorio/pfc_scripts#

```

Fig. 31 Output of the FER and delivery probability script

²⁹ <http://wireless.kernel.org/en/users/Documentation/iw>

3.12. Network topology

In a wireless community network where it grows with no structure and usually in a decentralized way, it can be very interesting to know how it's the topology in a given instant. Also the network won't preserve the same topology due to the lack of reliability in the wireless links and the facility to establish new ones.

The Zenmap software which is based in nmap, software used below in other measurements to scan the network looking for connected devices, allows collecting all the nmap information and plotting it into a more intuitive graph. From this graphical view, it can be deduced which devices are terminals connected to the access points and the links established between devices.

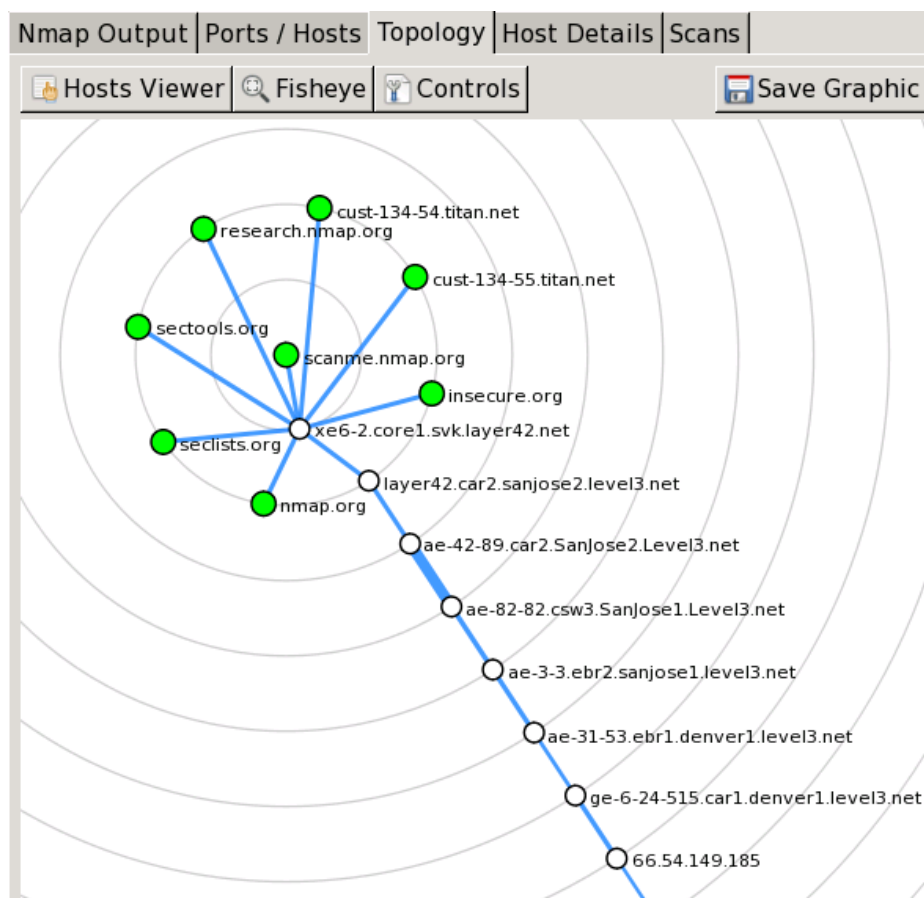


Fig. 32 Network topology graphic from Zenmap software

3.13. List of available wireless networks

Using the LinSSID software the administrator can scan the available WiFi networks and check if there could be interferences by other networks transmitting in the same channel or the adjacent ones, for example.

The interference detection with this tool is limited to the ones coming from other networks, other sources wouldn't be detected.

Basically there are two modes of networks visualization, one seeing the power variation over time and the other one over the channels available. It can be useful to detect other WiFi networks transmitting in the same or near channels, so the administrator could change the transmission channel to avoid the interferences.

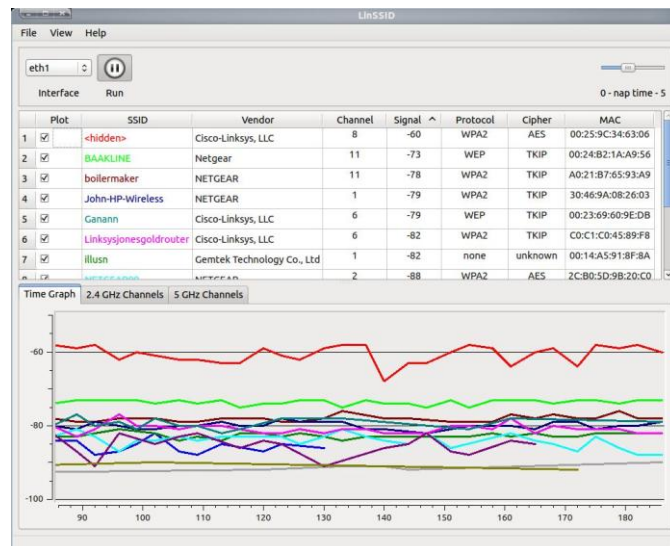


Fig. 33 Evaluation of the near WiFi networks over time

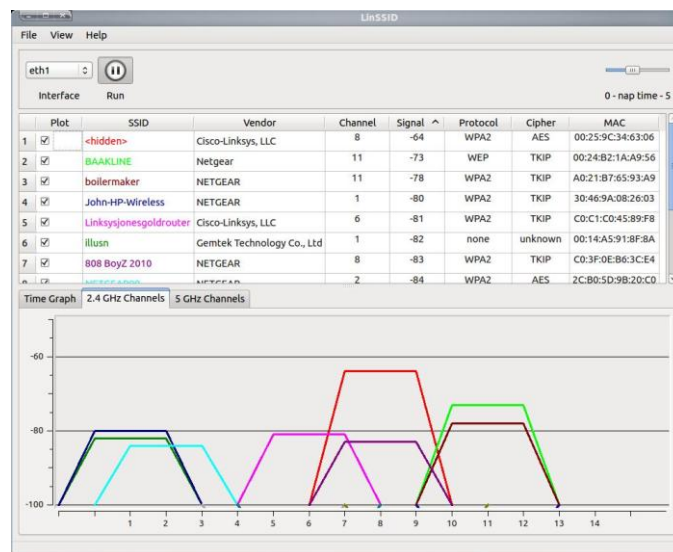


Fig. 34 Evaluation of the near WiFi networks by channels

CHAPTER 4. USING THE TOOLS

To create a wireless community network is not the purpose of this project; it's to define methodologies, measurements and tools to administrate a part of it in an efficient way. It has been defined also that the lack of stability in terms of performance and topology in this kind of networks which adds an extra difficulty to its management.

Before offering network connectivity to the users, it's necessary to configure all the routing protocols allowing the internal and external communications. This feature is performed by the internal and external routing protocols; usually OSPF³⁰ and BGP³¹ are implemented. At this point, the users are able to access to the network and have full connectivity to Internet and inside the community network itself.

In this chapter is explained the way to implement and use all the measurements defined in the previous chapter. It must be explained how to deploy this monitoring toolkit because if it's bad implemented, the results can become meaningless or incorrect.

4.1. Where

A wireless mesh community network is usually composed by other networks, so it's decentralized. There isn't any central device in charge of the management of the whole network and, in addition, all of these networks usually have different administrators and most probably would share their administration to other users.

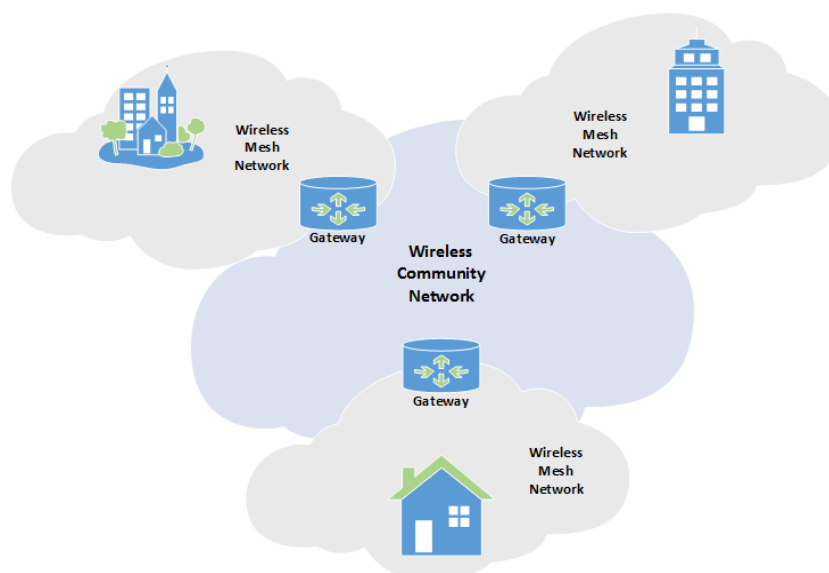


Fig. 35 Wireless Community Network example schema

³⁰ http://en.wikipedia.org/wiki/Open_Shortest_Path_First

³¹ https://en.wikipedia.org/wiki/Border_Gateway_Protocol

Taking all this facts into account, let's focus on one of these wireless mesh networks that are part in the whole community network. The networks obtain connectivity to other networks, thus establishing the community, through gateway devices in charge of redirect the mesh network traffic in both directions.

A part of the gateway acting as a border to the external world for the network, there could be more routing devices (routers) working internally and redirecting the traffic inside the network (before reaching the gateway).

Depending on the administrator needs and on the network topology, it could be interesting to configure certain devices a part from the gateway with the OpenWrt operating system in order to configure all the tools and exploit all the advantages and the potential of the device. The gateway is a key point to configure because it's where all the traffic created by the network users passes through.

So the choice of the desired devices to monitor depends strictly on the administrator of the network and its topology; but it's almost mandatory to select the gateway for the reasons explained above.

There are a couple of measurements that can't be installed in the OpenWrt operating system because they require a graphical user interface. The mentioned measurements are:

- Network topology: This measurement requires installing and launching the Zenmap software, which could be defined as a graphical extension for the nmap.
- List of available networks: To detect the available wireless networks it's needed the LinSSID software, that is also a software with a graphical interface for the user.

These measurements should be performed in a device different from the OpenWrt routing devices or the gateway. Both could be installed in personal computers but maybe the LinSSID software would be more useful in a laptop device, allowing the administrator to go to any desired point and check the available wireless networks.

4.2. What

Having all the selected devices to monitor, it has to be decided what measurements want the administrator to perform depending on the needs and in which device make sense to be implemented.

Knowing that the gateway is the traffic concentration point of all the network, all the measurements that involve all the users must be implemented in this device. They can be also implemented in other routing device but it must be taken into account that the results would be incomplete because not all the users would be taken into account. The recommended measurements applied in the gateway are:

- Total amount of data transferred by clients: The volume of data would be more accurate if it's measured in the gateway because it passes all through this point, in both directions.
- Traffic shape: The software is measuring the bandwidth usage in background, but most probably it's important to measure the whole network usage instead of only a restricted network zone.
- Application classes: Like in the traffic shape, the applications used by the users should be measured in the gateway where all the traffic passes through, else maybe some traffic fluxes couldn't be monitored.
- Transfer rates: The data could be sent by different paths inside the network but it will pass through the gateway for sure.

The other measurements described in the project have not such a direct dependency on the users as the previous ones; so they don't have any restriction of where should be performed, can be done in the gateway, in other routing devices or in both places.

4.3. When

Once the desired tools are installed in the chosen devices, the next step is to know when or how often a measurement has sense to be done. For example, it's pointless to check the status of a link based on a measurement done one week ago; it should be checked in real time.

The measurements that must be allocated in the gateway are based on statistics that are stored periodically so there's no need to execute any command to make the measurement; the administrator only has to view the evolution of this saved data.

Some measurements are designed with the purpose of detect errors in the network:

- **Active links:** A set of users could notice that there's no connection and the wireless network disappeared, a reason could be a malfunctioning in the device causing the wireless interface to become down. Also a non-operative link could cause the isolation of a whole zone, that could be of different sizes, leaving several users without connectivity.
- **All links that receive at least one beacon in a measurement interval:** Executing this measurement, links with low usage or none could be detected, this could indicate a problem in the environment or that there aren't any users using this access. Taking this into account, the administrator should verify if the link is working properly and if there are users connected to it.
- **Latency and Jitter:** A bad performance in real time applications could be caused by a high latency and jitter; and this could be caused by network congestion. The solution to this problem could be to increment the network bandwidth capacity or balance the network load applying routing policies.
- **Frame Error Rate (FER) and Deliver Probability:** Having these two factors elevated would cause connection problems so probably the user would contact the administrator requesting a solution. The main possible cause for this case could be the existence of interferences coming from the environment, which are very difficult to eliminate for the administrator because usually they become from external sources.

Other measurements don't detect errors in the network, are thought to obtain information and maybe provide an opportunity to the administrator to prevent them:

- **Traffic shape:** The bandwidth used by the users measured over time can be very useful for the administrator to detect problems of capacity in the network if the graphic becomes saturated. Other use for this tool is to define user behaviors in the traffic demand for a given period in order to have a better resource management.
- **Signal parameters:** These are physical measurements like the signal power, it could be used to detect any error in the device hardware, the direct cause could be the loss of connectivity or the disappearance of the network from the user device.
- **Discover the network devices:** In order to count the amount of users connected to the network, this measurement can be very useful. If the number of devices increases significantly, the bandwidth available will decrease significantly.
- **Device information and configuration:** For more precise tuning configuration, it can be necessary for the administrator to obtain hardware and driver information about the devices. Some

incompatibilities could be solved and also upgrades to obtain a better performance.

- **Transfer rates:** For a punctual network traffic overload, the administrator could execute this real-time measurement and maybe discover if there's any connection that is consuming the major part of the bandwidth. The source inside the network could be identified and perhaps ban it forbidding its access to the network.
- **Application classes:** The administrator would like to obtain information about the layer 7 applications used in the network to take some actions. For example, some file exchange software should be blocked to avoid high bandwidth consumption from a few users.
- **Network topology:** Wireless networks have the advantage of deploying new network links, but it can become also an inconvenience if the network starts to grow without any control. In the whole network it's almost impossible to control its growth, but in the piece of network controlled by the administrator maybe could be possible. Detecting new routing devices or access points without the administrator's permission could be one of the goals to obtain and, on the other hand, to check that all the devices are working properly in the network.
- **List of available wireless networks:** A bad connection could be caused by the interference caused by near wireless networks. There are a few channels available and maybe all could become used in places like apartment buildings where the population density is higher. This kind of interference is hard to eliminate because it would suppose to turn off another wireless owned by other user, but the source of the problem can be detected at least.

4.4. How

It has been defined where the tools should be placed, what measurements have more sense to monitor depending on the device location and also the periodicity or the instant when the measurements must be done, the last issue to define is how to perform the different measurements.

There are three kinds of methods to perform the measurements described in the previous chapters:

- **Secure Shell (SSH)³²:** All the command line and script-based measurements should be executed using this protocol. This technique allows the administrator to access remotely to the devices safely, the main drawback of this mechanism is the lack of graphical interface but it's not necessary for this group of measurements.

³² http://en.wikipedia.org/wiki/Secure_Shell

- **Web:** The traffic shape and the application usage are measurements that a service is making periodically and is stored in the device. In order to access to the data, the same service provides a web server so the administrator only should have to put the IP address of the remote device in a web browser.
- **Laptop graphical interface:** The measurements based on the Zenmap and LinSSID software must be done in devices with graphical interfaces, so the administrator would need a Linux operating system installed in a laptop device, for example.

CHAPTER 5. CONCLUSIONS

The main goal of this project is to advise wireless community networks administrators giving them a set of tools and measurements helping in their management tasks. A brief study of wireless mesh networks behavior has been necessary to define which measurements are interesting to perform.

In order to support open initiatives, in this project all the tools and solutions are open source without any software cost to the possible users.

This project provides a set of tools and measurements but they don't cover all the needs for a wireless network. There are other sets of measurements like more user-oriented like the session duration, the number of active clients, the oscillations in network associations and mobility aspects.

Other group of possible measurements to take into account could be the routing ones, it's very important to maintain configured the routing protocols of the network, they could suppose a great difference in terms of network performance.

Possible improvements could be done in terms of group all the measurements in a single software package; a unique software devoted to wireless networks management. It could simplify the installation and configuration tasks and be added also directly to the OpenWrt operating system.

The final result of the project has been positive increasing the initial knowledge due to the necessity to go deeper in UNIX operating systems, discovering new functionalities and also investigating further in WiFi technologies.

REFERENCES

- [1] Wireless mesh networks page on Wikipedia - http://en.wikipedia.org/wiki/Wireless_mesh_network
- [2] Open Source Initiative website - <http://opensource.org/>
- [3] IEEE 802.11 web page - <http://standards.ieee.org/about/get/802/802.11.html>
- [4] Guifi.net project website - <http://guifi.net/>
- [5] OpenWrt project website - <https://openwrt.org/>
- [6] SNMP page on Wikipedia - http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol
- [7] BGP page on Wikipedia - https://en.wikipedia.org/wiki/Border_Gateway_Protocol
- [8] snmpd software official page - <http://www.net-snmp.org/>
- [9] Ntop project website - <http://www.ntop.org/>
- [10] Tcpdump and Libpcap website - <http://www.tcpdump.org>
- [11] Apache web server official website - <http://httpd.apache.org/>
- [12] nProbe project website - <http://www.ntop.org/products/nprobe/>
- [13] Bandwidthd project website - <http://bandwidthd.sourceforge.net/>
- [14] PostgreSQL project website - <http://www.postgresql.org/>
- [15] Nmap project website - <http://nmap.org/>
- [16] Zenmap project website - <http://nmap.org/zenmap/>
- [17] LinSSID project website - <http://sourceforge.net/projects/linssid/>
- [18] inSSIDer project website - <http://www.metageek.net/products/inssider/>
- [19] IF-MIB definition page - <http://netsnmp.sourceforge.net/docs/mibs/interfaces.html>
- [20] Ping page on Wikipedia - [http://en.wikipedia.org/wiki/Ping_\(networking_utility\)](http://en.wikipedia.org/wiki/Ping_(networking_utility))
- [21] iwconfig command help page - http://www.linuxcommand.org/man_pages/iwconfig8.html
- [22] ICMP page on Wikipedia - https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol
- [23] lspci command help page - http://www.linuxcommand.org/man_pages/lspci8.html
- [24] PCI bus page on Wikipedia - http://en.wikipedia.org/wiki/Conventional_PCI

- [25] modinfo command help page - http://www.linuxcommand.org/man_pages/modinfo8.html
- [26] Kernel module page on Wikipedia - https://en.wikipedia.org/wiki/Loadable_kernel_module
- [27] hwinfo project website - <http://www.hwinfo.com/>
- [28] lftop project website - <http://www.ex-parrot.com/pdw/iftop/>
- [29] iw command help page - <http://wireless.kernel.org/en/users/Documentation/iw>
- [30] OSPF routing protocol page on Wikipedia - http://en.wikipedia.org/wiki/Open_Shortest_Path_First
- [31] BGP routing protocol page on Wikipedia - https://en.wikipedia.org/wiki/Border_Gateway_Protocol
- [32] Secure Shell page on Wikipedia - http://en.wikipedia.org/wiki/Secure_Shell
- [33] Peer-to-peer page on Wikipedia - <http://en.wikipedia.org/wiki/Peer-to-peer>

ANNEXES

1. Active links script source code

```
#!/usr/bin/perl

#####
# LIST OF OIDs USED
#
# ifTable          1.3.6.1.2.1.2
# ifNumber         1.3.6.1.2.1.2.1.0
# ifDescr          1.3.6.1.2.1.2.2.1.2.[INTERFACE-INDEX]
# ifAdminStatus    1.3.6.1.2.1.2.2.1.7.[INTERFACE-INDEX]
# ifOperStatus     1.3.6.1.2.1.2.2.1.8.[INTERFACE-INDEX]
#####

use Switch;

my $address = "localhost";

# We obtain the number of interfaces via SNMP
my $line_numInt = qx/snmpget -v 2c -c public $address 1.3.6.1.2.1.2.1.0 ./;
my @aux_array = split(" ", $line_numInt);
my $numInt = @aux_array[1];
#print "Number of interfaces : $numInt\n";

my $i;
my @ifDescr;
my @adminStatus;
my @operStatus;
my @active;

# We get all the SNMP information needed for each interface
for ($i=1; $i<=$numInt; $i++) {
    my $line_descr = qx/snmpget -v 2c -c public $address
1.3.6.1.2.1.2.2.1.2.$i ./;
    my $line_admin = qx/snmpget -v 2c -c public $address
1.3.6.1.2.1.2.2.1.7.$i ./;
    my $line_oper = qx/snmpget -v 2c -c public $address
1.3.6.1.2.1.2.2.1.8.$i ./;

    my @aux_array = split("", $line_descr);
    @ifDescr[$i] = @aux_array[1];
    my @aux_array = split(':', $line_admin);
    @adminStatus[$i] = @aux_array[1] + 0;
    my @aux_array = split(':', $line_oper);
    @operStatus[$i] = @aux_array[1] + 0;
}
```

```

if ((@adminStatus[$i]==1) && (@operStatus[$i]==1)) {
    @active[$i] = "ACTIVE";
} else {
    @active[$i] = "INACTIVE";
}

# Switch to add more information to the AdminStatus of the interface
switch (@adminStatus[$i]) {
    case(1) { @adminStatus[$i] = "up(1) -- ready to pass packets"; }
    case(2) { @adminStatus[$i] = "down(2)"; }
    case(3) { @adminStatus[$i] = "testing(3) -- in some test mode"; }
    else { @adminStatus[$i] = "error"; }
}

# Switch to add more information to the OperStatus of the interface
switch (@operStatus[$i]) {
    case(1) { @operStatus[$i] = "up(1) -- ready to pass packets"; }
    case(2) { @operStatus[$i] = "down(2)"; }
    case(3) { @operStatus[$i] = "testing(3) -- in some test mode"; }
    case(4) { @operStatus[$i] = "unknown(4) -- status can not be
determined for some reason"; }
    case(5) { @operStatus[$i] = "dormant(5)"; }
    case(6) { @operStatus[$i] = "not present(6) -- some component is
missing"; }
    case(7) { @operStatus[$i] = "lowerLayerDown(7) -- down due to
state of lower-layer interface(s)"; }
    else { @adminStatus[$i] = "error"; }
}
}

clear_screen();

# We print on the screen all the information obtained above
for ($i=1;$i<=$numInt;$i++) {
    print ">>> INTERFACE \"@ifDescr[$i]\" [@active[$i]\n";
    print "AdminStatus : @adminStatus[$i]\n";
    print "OperStatus : @operStatus[$i]\n\n";
}

# Some auxiliar functions
sub clear_screen {
    system("clear");
}

```


2. Link that receive a beacon receive script source code

```
#!/usr/bin/perl

# All links that receive at least one beacon in a measurement interval

#####
# LIST OF OIDs USED
#
# ifTable          1.3.6.1.2.1.2
# ifNumber         1.3.6.1.2.1.2.1.0
# ifDescr          1.3.6.1.2.1.2.2.1.2.[INTERFACE-INDEX]
# ifInOctets       1.3.6.1.2.1.2.2.1.10.[INTERFACE-INDEX]
#####

use Switch;

my $address = "localhost";
my $interval = 5; # In seconds

# We obtain the number of bytes transferred from the device
my $line_numInt = qx/snmpget -v 2c -c public $address 1.3.6.1.2.1.2.1.0 ./;
my @aux_array = split(" ", $line_numInt);
my $numInt = @aux_array[1];
#print "Number of interfaces : $numInt\n";

my $i;
my @ifDescr;
my @beginBytes;
my @endBytes;

# We get all the SNMP information needed for each interface
for ($i=1; $i<=$numInt; $i++) {
    my $line_descr = qx/snmpget -v 2c -c public $address
1.3.6.1.2.1.2.2.1.2.$i ./;
    my $line_inBytes = qx/snmpget -v 2c -c public $address
1.3.6.1.2.1.2.2.1.10.$i ./;

    my @aux_array = split("", $line_descr);
    @ifDescr[$i] = @aux_array[1];
    my @aux_array = split(':', $line_inBytes);
    @beginBytes[$i] = @aux_array[1] + 0;
}

# We wait the interval specified to obtain again the information
sleep($interval);
```

```
# We get again all the SNMP information needed for each interface
for ($i=1;$i<=$numInt;$i++) {
    my $line_inBytes = qx/snmpget -v 2c -c public $address
    1.3.6.1.2.1.2.2.1.10.$i ./;

    my @aux_array = split(':', $line_inBytes);
    @endBytes[$i] = @aux_array[1] + 0;
}

clear_screen();

# We print on the screen all the information obtained above
print "The interface will be considered ACTIVE if it have received at least one
packet in $interval seconds\n";
for ($i=1;$i<=$numInt;$i++) {
    # print "@beginBytes[$i] - @endBytes[$i]\n";
    if (@beginBytes[$i]==@endBytes[$i]) {
        print ">>> INTERFACE \"@ifDescr[$i]\" --> INACTIVE\n";
    } else {

        print ">>> INTERFACE \"@ifDescr[$i]\" --> ACTIVE\n";
    }
}

# Some auxiliar functions
sub clear_screen {
    system("clear");
}
```

3. Amount of traffic sent by clients script source code

```
#!/usr/bin/perl

# Amount of traffic sent by clients in a given time period

#####
# LIST OF OIDs USED
#
# ifTable          1.3.6.1.2.1.2
# ifNumber         1.3.6.1.2.1.2.1.0
# ifDescr          1.3.6.1.2.1.2.2.1.2.[INTERFACE-INDEX]
# ifOutOctets      1.3.6.1.2.1.2.2.1.16.[INTERFACE-INDEX]
#####

use Switch;

my $address = "localhost";
my $interval = 10; # In seconds

# We obtain the number of bytes transferred from the device
my $line_numInt = qx/snmpget -v 2c -c public $address 1.3.6.1.2.1.2.1.0 ./;
my @aux_array = split(" ", $line_numInt);
my $numInt = @aux_array[1];
#print "Number of interfaces : $numInt\n";

my $i;
my @ifDescr;
my @beginBytes;
my @endBytes;

# We get all the SNMP information needed for each interface
for ($i=1; $i<=$numInt; $i++) {
    my $line_descr = qx/snmpget -v 2c -c public $address
1.3.6.1.2.1.2.2.1.2.$i ./;
    my $line_outBytes = qx/snmpget -v 2c -c public $address
1.3.6.1.2.1.2.2.1.16.$i ./;

    my @aux_array = split("", $line_descr);
    @ifDescr[$i] = @aux_array[1];
    my @aux_array = split(':', $line_outBytes);
    @beginBytes[$i] = @aux_array[1] + 0;
}

# We wait the interval specified to obtain again the information
sleep($interval);
```

```

# We get again all the SNMP information needed for each interface
for ($i=1;$i<=$numInt;$i++) {
    my $line_outBytes = qx/snmpget -v 2c -c public $address
1.3.6.1.2.1.2.2.1.16.$i ./;

    my @aux_array = split(':', $line_outBytes);
    @endBytes[$i] = @aux_array[1] + 0;
}

clear_screen();

# We print on the screen all the information obtained above
print "Amount of data transferred in $interval seconds\n";
for ($i=1;$i<=$numInt;$i++) {
    $amountB = @endBytes[$i] - @beginBytes[$i];
    #print "BEGIN      @beginBytes[$i]\n";
    #print "END        @endBytes[$i]\n";
    #print "AMOUNT    $amountB\n";
    if ($amountB > 1073741824) {
        $amountB = $amountB / 1073741824;
        $amountB = sprintf "%.3f", $amountB;
        $text = "$amountB GB";
    } elseif ($amountB > 1048576) {
        $amountB = $amountB / 1048576;
        $amountB = sprintf "%.3f", $amountB;
        $text = "$amountB MB";
    } elseif ($amountB > 1024) {
        $amountB = $amountB / 1024;
        $amountB = sprintf "%.3f", $amountB;
        $text = "$amountB KB";
    } else {
        $text = "$amountB B";
    }

    print ">>> INTERFACE \"@ifDescr[$i]\" :      $text\n";
}

# Some auxiliar functions
sub clear_screen {
    system("clear");
}

```

4. Latency and jitter script source code

```
#!/usr/bin/perl

# Latency and Jitter

# We execute the PING command and store the output to a temporal file
my $file = "output.tmp";
my $num_pings = 10;
my $address = "www.google.es";
system("ping -c $num_pings $address > $file");
#system("cat " . $file);

# We read the output file line by line
my $first_line = 0;
my @times;
open my $text, $file or die "Could not open $file: $!";

while( my $line = <$text> ) {
    $last_line = $line;

    if (($first_line > 0) && ($first_line <= $num_pings)) {
        #print "$line\n";
        my @first_array = split(' ', $line);
        #print @first_array[7];
        my @second_array = split('=', @first_array[7]);
        #print "@second_array[1]\n";
        push(@times, @second_array[1]);
        #print "@second_array[1]\n";
    }
    $first_line++;
}

# We close the output file
close $text;

#print "$last_line\n";
my @array = split(' ', $last_line);
my @values = split('/', @array[3]);
my $min = @values[0];
my $avg = @values[1];
my $max = @values[2];
my $mdev = @values[3];
my $mean = mean (\@times);
$mean = sprintf "%.3f", $mean;
my $variance = variance (\@times);
$variance = sprintf "%.3f", $variance;
```

```
# We print the calculated information
clear_screen();
print "The latency parameters to reach [$address] with [$num_pings] iterations
are:\n\n";
print "Minimum value:           $min ms\n";
print "Maximum value:           $max ms\n";
print "Average value:            $avg ms\n";
print "Maximum deviation:        $mdev ms\n";
print "Mean:                     $mean ms\n";
print "Jitter/Variance:          $variance ms\n\n";

# We delete the output file
system("rm $file");

# Some auxiliar functions
sub sum {
    my ($arrayref) = @_ ;
    my $result;
    foreach (@$arrayref) { $result+= $_; }
    return $result;
}

sub mean {
    my ($arrayref) = @_ ;
    my $result;
    foreach (@$arrayref) { $result += $_ }
    return $result / @$arrayref;
}

sub variance {
    return (sum [ map { ($_ - $mean)**2 } @{$_[0]} ] ) / @{$_[0]};
}

sub clear_screen {
    system("clear");
}
```

5. Deliver probability and Frame Error Rate script source code

```
#!/usr/bin/perl

# Deliver Probability and Frame Error Rate

# COMMAND --> iw dev wlan0 station dump

# DP = 1 - ( tx_failed / tx_packets )

# MAL --> REVISAR
# FER = ( tx_failed / tx_packets ) * Ttx_packet
# Estimation Ttx_packet = L/BW

use Switch;

my $interface = "wlan0";
my $file = "output.tmp";

# We obtain the output of the iw command
system("iw dev $interface station dump > $file");

open my $text, $file or die "Could not open $file: $!";

my $num_line = 1;
my @data;

while( my $line = <$text> ) {
    # We delete the tabs
    $line =~ s/ //g;

    switch ($num_line) {
        case(1) {
            # MAC address
            my @aux_array = split(' ', $line);
            @data[0] = @aux_array[1];
        }
        case(5) {
            # TX Bytes
            my @aux_array = split(':', $line);
            @data[1] = @aux_array[1]+0;
        }
        case(6) {
            # TX packets
            my @aux_array = split(':', $line);
            @data[2] = @aux_array[1]+0;
        }
    }
}
```

```

        case(8) {
            # TX failed packets
            my @aux_array = split(':', $line);
            @data[3] = @aux_array[1]+0;
        }
        case(11) {
            # Bit rate
            print "$line\n";
            my @aux_array = split(':', $line);
            my @aux_array2 = split(' ', @aux_array[1]);
            @data[4] = @aux_array2[0]+0;
        }
    }

    #print $line;
    $num_line++;
}

# We delete the temporal output file
system("rm output.tmp");

clear_screen();

# We process and display the results
print("Data for the [$interface] interface:\n\n");
print("MAC address:           @data[0]\n");
print("Transmitted bytes:       @data[1] B\n");
print("Transmitted packets:      @data[2] packets\n");
print("Transmitted failed packets: @data[3] packets\n");
print("Bit Rate:                  @data[4] Mbps\n\n");

# Deliver probability
my $DP = 100*(1-(@data[3]/@data[2]));
$DP = sprintf "%.2f", $DP;
print("Deliver probability:      $DP%\n");

# Frame error rate
my $FER = @data[4]*1000*(@data[3]/@data[2]);
$FER = sprintf "%.2f", $FER;
print("FER:                        $FER Kbps\n");

# Some auxiliar functions
sub clear_screen {
    system("clear");
}

```