UNIVERSITÀ DEGLI STUDI DI TRENTO

Nomadic Communication
AA 2008-2009

Report 1: Measuring throughput of 802.11 b and g protocols

Group N. 2

Michele Segata (138458)
Mattia Avancini (138793)
Chiara Canton (138768)

Abstract

This report contains the analysis of the behaviour of the 802.11 b and g protocols, firstly in a theoretical approach and then in comparison with what happens in reality. Two main experiments will be done: the first in a basic environment, just sending data from a station to another without setting any particular parameter on the stations. The second will analyze the protocols in case of fragmentation.

Contents

1 Introduction

The main purpose of this lab experience is to measure the throughput of the 802.11 b and g protocols in an infrastractured wireless LAN. To do this, we will analyze their behaviour, to compute theoretical results and compare them with the ones obtained by the practical experience. We will use the iperf measurement tool[1] to send data throught a wireless channel and measure its performances. We will also use the program wireshark [2] to monitor what is happening in the network.

The conditions under which we would like to operate are *"the best"*, without collisions and disturbances, but our measurements will take place into a room of the faculty, which is full of access points and stations, and for sure we will deal with that kind of problems. Our goal is to study the ideal behaviour of the protocol, in absence of interferences, and then compare them with the results obtained in a real environment.

Anyway we would like to have as less collisions as possible, so we will set up a monodirectional communication between a station and an access point dedicated to our purposes.

Analysis will be done on both b and g protocols, choosing some of the available speeds. First of all, we will do our measurements in a basic configuration, with all parameters like RTS/CTS or fragmentation thresholds disabled. Then, we will analyze the fragmentation mechanism and see why it is used, how the protocol implements it and what consequences it has on the throughput.

[1] `http://www.noc.ucf.edu/Tools/Iperf/`
[2] `http://www.wireshark.org/`

2 Experiment materials

To do our experiment we need three laptops, an access point and a switch. We use one laptop as a server, to send our data to it. It has a static IP address which is 192.168.10.30 and runs a DHCP server to assign IP addresses to laptops that connects to the AP's network. It runs also a server instance of iperf, which is needed to receive data from clients. To configure it, we need to type three command into the shell. The first is

```
/etc/init.d/networking restart
```

which sets up the network interface. Then we have to start the DHCP server with the following command:

```
/etc/init.d/dhcp3-server restart
```

Finally we run iperf in server mode

```
iperf -s -u
```

The server is connected via ethernet cable (100Mb/s) to the switch, needed for connecting other access points for other groups' experiments. This switch is then connected by ethernet cable (100Mb/s) to the access point, which is a Cisco Aironet 1350 that supports both b and g protocols. As mentioned in the introduction, we are in a infrastructured wireless LAN. Note that the ethernet connection is much more faster than the wireless one, so it shouldn't constitute a bottleneck that affects measurements.

Another laptop is used to run a client instance of iperf, to send data to the server through the access point. First of all we have to connect to it, and the shell command for this purpose is:

```
iwconfig eth1 essid NCG rate 11M
```

This communicate the NIC card to connect to the network called *NCG* and to set transmission speed to 11 Mbps. After that we have to start the DHCP client to obtain an IP address, just typing

```
dhclient eth1
```

Now we are ready to run our iperf tests. Iperf will show us the performance results and will output them into a cvs (comma separated) file that we are going to use to analyze data. A tipical iperf running command could be like this:

```
iperf -c 192.168.10.30 -u -b11M -i 1 -t 30 -yc
```

where -c sets the IP address of the listening server, -u indicates that we want to run an UDP test[3], -b sets the desired network load, -i the interval to collect data speed, -t the length of the test in seconds and -yc indicates the csv formatted output.

The third laptop will be used to run wireshark and monitor the network while doing our tests. To do this, we will put the NIC card into monitor mode, which allows wireshark to pick up every frame passing through the channel, and not only the ones destinated to us. The shell command to put NIC card in monitor mode is

```
iwconfig eth1 mode monitor channel 1
```

This allows interface eth1 to monitor the wireless channel number 1 (2412 MHz). Obviously interface and channel number depends on the laptop and the access point settings.

Wireshark can be used to check whether the network is behaving as expected. For example: if we want to measure the throughput in the case of fragmentation, it is not sufficient to set fragmentation level on the NIC card and on the AP and then run the test. Indeed it could happen that, despite you have set up specific value, from time to time they don't apply as expected. For this reason we prefer to monitor the traffic with Wireshark which allows us to see the content of a 802.11 MAC header and ensure that data, for example, is fragmented. Both laptops have a Intel corporation PRO/Wireless 2200BG NIC card. Figure 1 shows how components are positioned.

All tests will be done with a simmetric configuration (if speed is set to 11Mb/s on the access point, it must be 11Mb/s on the station too).

[3]We will run UDP tests to have a simple flow of data to make analysis easier. In a TCP based communication, we will have to consider also connection and acknoledgement packets, which clearly makes theoretical analysis harder

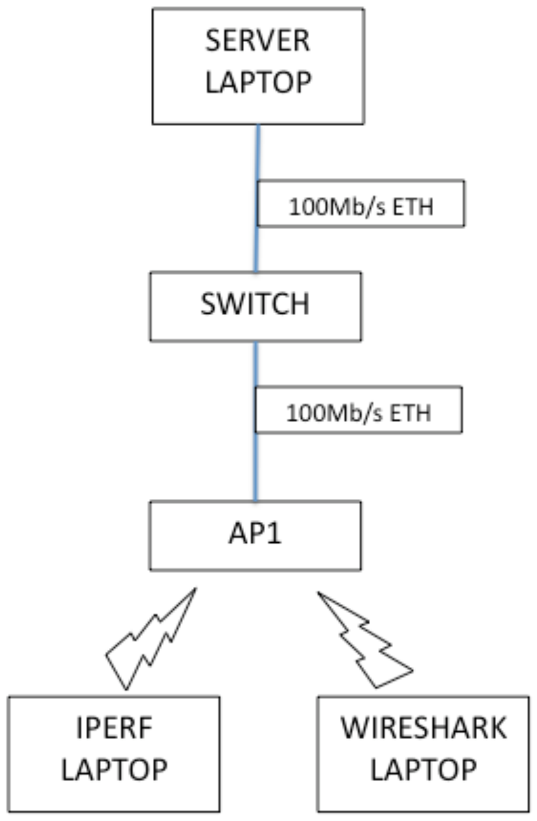


Figure 1: The scheme of the test bed

3 Experiment 1: Basic set up

The first test is done to analyze protocols' behaviour in a basic configuration. We will set RTS/CTS and fragmentation thresholds to the maximum, to disable these mechanisms. We will start iperf to collect our data: in particular, for each analyzed speed, we will run 20 tests of 30 seconds, acquiring data every 1 second. The speeds that we are going to analyze are 1, 5.5, 11 Mb/s of 802.11b and 6, 12, 36, 54 Mb/s of 802.11g. Access point will be configured to require test's speed to the stations that want to join the network[4] and, as mentioned before, tests will be monodirectional, with a simmetric configuration (same speed, RTS and fragmentation threshold on both access point and station).

3.1 Theoretical analysis

To compute theoretical throughput we have to calculate this formula:

$$\text{Throughput } (Mb/s) = \frac{\text{Amount of data } (bits)}{\text{Transmission time } (\mu s)}$$

The amount of data depends on the level we are interested in. The iperf tool, by default, sends UDP data packet of 1470 bytes (application data): this amount can be used to calculate the throughput at the application layer, but to have the value, for example, at IP level, we need to consider also UDP (8 bytes) and IP (20 bytes) headers, so a total amount of 1498 bytes.

To calculate the transmission time we need to have a look at the standard and see its behaviour. Figure 2 shows a general 802.11b/g transmission procedure. Transmission's details change a lot between b and g and we decide to run tests in

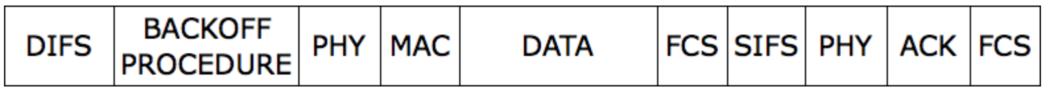


Figure 2: Tipical 802.11b/g transmission

b only and g only mode. It is better to see them separately, starting from b, which is the easiest, and then analysing g which is more complex.

3.2 802.11b

Let's proceed in order and provide details about Figure 2. First of all, we have a DIFS (Distributed coordinator function InterFrame Space), which separates two different transmissions. It is a fixed time and it is calculated as following:

$$T_{DIFS} = T_{SIFS} + 2 \times T_{SLOT} = 10\mu s + 2 \times 20\mu s = 50\mu s$$

[4]CISCO AP lets you to require or to disable every available speed

A SIFS (Short InterFrame Space) is the fixed amount of time that stays between different part of the same transmission, for example a data frame and its acknoledgement.

SLOT time instead, is used in the backoff procedure. 802.11 protocols are based on CSMA/CA (Collision Avoidance), given the nature of the channel, which does not permit collision detection. To avoid, or better, to reduce the number of collisions, each station has a random generator that generates a number which is between 0 and a maximum value (which is called contention window) minus 1. Contention window's size changes based on the result of the transmission. If the transmission complete, its value became CW_{MIN} which is 32, while at every collision, it is doubled till CW_{MAX} (1024) is reached. After this number is generated, a station starts counting down, decrementing this value every SLOT time (20μs) while listening to the channel. When the station reaches 0, it can start transmitting, otherwise if someone else starts sending data before, this value is saved, and countdown continues in the next backoff procedure. In our test bed, we have assumed to have no collisions, so contention window's size never increase from CW_{MIN}. If the random generator is well implemented, we should have an average of countdown values which is 15,5. So the time taken by the backoff procedure is on average

$$T_{BACKOFF} = 15,5 \times T_{SLOT} = 310\mu s$$

The physical header (PHY) is formed by two parts: the first is the PLCP[5] preamble, which is used to synchronize the receiver to the sender. This can be long (144 bits) or short (72 bits) based on the type of transmission adopted. In the basic DSSS[6] mode, the long preamble is used, while in the HR/DSSS[7] mode, the short preamble can be used. In both cases, it is sent at 1Mbps.

The second part of the PHY header is the PLCP header made of 48 bits. It contains four fields:

[5]Physical Layer Convergence Protocol

[6]Direct Sequence Spread Spectrum, used in 1 Mb/s

[7]High Rate Direct Sequence Spread Spectrum, used in 2, 5.5 and 11 Mb/s

- SIGNAL: indicates the speed that shall be used for transmission of the PSDU
- SERVICE: contains some physical parameters, like the modulation method
- LENGTH: means the number of microseconds needed to send the PSDU
- CRC: is the checksum used to protect correctness of the first three fields

In case of DSSS mode it is sent at 1Mbps, in HR/DSSS at 2Mbps. So PHY header times are:

$$T_{PHY_{DSSS}} = \frac{144bits}{1Mbps} + \frac{48bits}{1Mbps} = 192\mu s$$

$$T_{PHY_{HR/DSSS}} = \frac{72bits}{1Mbps} + \frac{48bits}{2Mbps} = 96\mu s$$

Figures 3 and 4 show a graphical interpretation of the long and the short PPDU format.

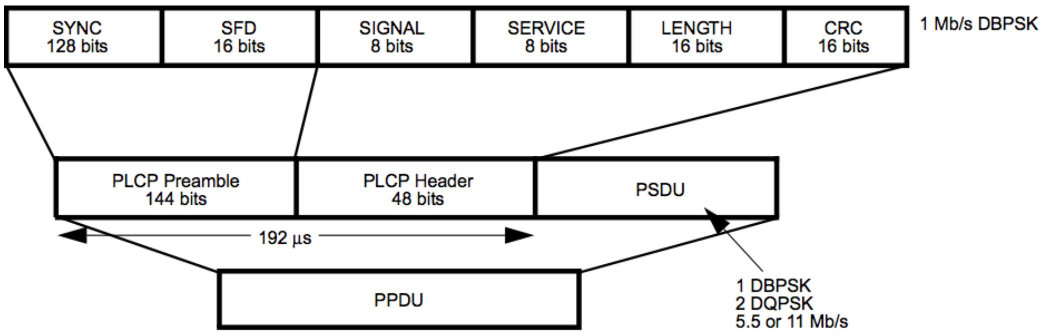


Figure 3: Long PPDU

After that, begins the MAC header, which contains informations like the sender address, the receiver address, the type of frame, etc. Its size is 32 bytes as we can see in Figure 5, but in our case there's no 4th address and no QoS control. It contains:

- Frame control: its content is displayed in Figure 6 and specifies parameters like the type of frame (data, management or control), if there are more fragments in the case of fragmentation, etc. Two interesting fields are ToDS and FromDS, where DS stands for Distribution System. These bits are set

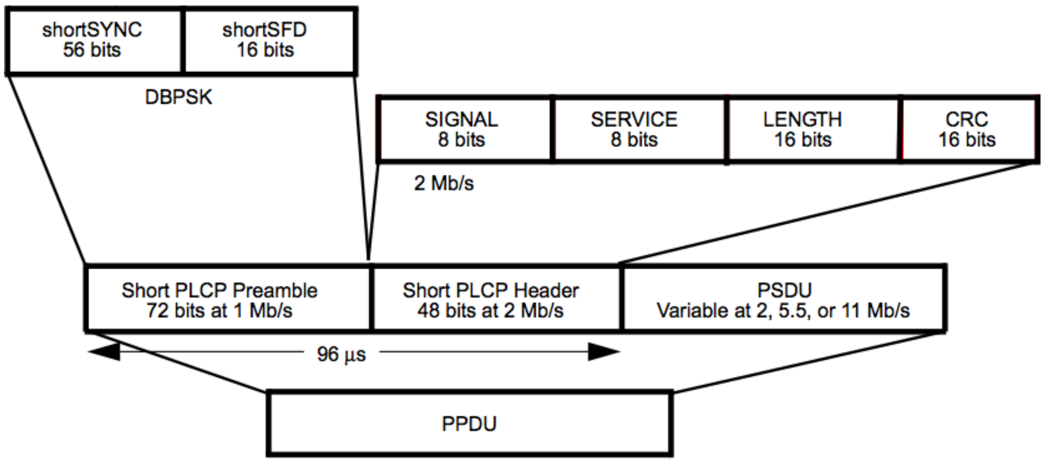


Figure 4: Short PPDU

differently based on the type of transmission we are doing. For example, if a frame is sent from a station and is directed to the server connected via ethernet at the access point, ToDS bit is set to 1 and FromDS to 0. Viceversa, if a frame is directed from the infrastructure to a station, FromDS is set to 1 and ToDS to 0. In an ad-hoc network, where there is no access point and no infrastructure, these bits are set both to 0. When the WLAN is bridged, which means that the access point forward our frames to another access point, both bits are set to 1.

- Duration: in our case, is used to calculate the NAV[8], which represents the time needed by the sender and the receiver to complete the transmission (data + ack). It is used to do the virtual channel sensing and defer medium access by stations that are not using the channel.
- Address 1: the content of the address fields depends on the values of the FromDS and ToDS bits of the Frame control field. We are considering frames going from the station to the server, so FromDS is set to 0 and ToDS to 1. In this case, Address 1 is the MAC address of the access point.
- Address 2: this is the MAC address of the sender station

[8]Network Allocation Vector

- Address 3: contains the MAC address of the station into the distribution system, in our case, server laptop
- Sequence control: this field contains the sequence number of the frame and if it is a fragment, the fragment number

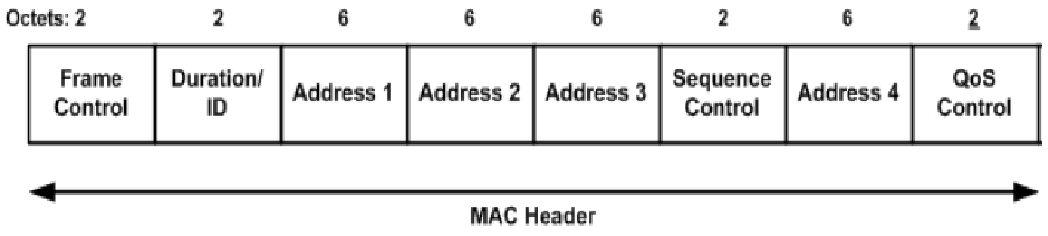


Figure 5: MAC Header

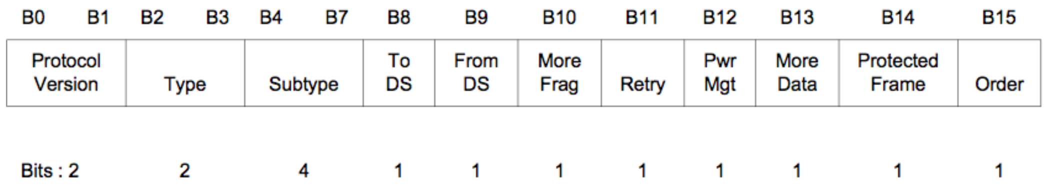


Figure 6: MAC Header frame control

In our case, it is sent at data rate, because it is the only allowed speed by the access point. In other cases, for example when all b speeds are allowed, MAC header must be sent at 1 or 2 Mbps, because all stations should be able to read it, in order to see NAV value. So MAC header time depends on the selected data rate. In case of 11 Mb/s its duration is

$$T_{MAC} = \frac{24 \times 8bits}{11Mb/s} = 17,45\mu s$$

Then starts the transmission of the MSDU, which is the payload for the MAC level. It is made by the size of the application data, plus all the headers of the levels in between and it is sent at data rate. Taking as an example the case of 11 Mb/s and recalling that, by default, iperf sends data packet of 1470 bytes, data duration is:

$$T_{DATA} = \frac{(Data_{APP} + H_{UDP} + H_{IP} + H_{LLC}) \times 8}{DataRate} =$$
$$\frac{(1470 + 8 + 20 + 8) \times 8bits}{11Mb/s} = 1095,27\mu s$$

After data, an FCS[9] which is basically a CRC, is appended. It's made of 4 bytes and it is sent at data rate. In case of 11 Mb/s we have

$$T_{FCS} = \frac{4 \times 8bits}{11Mb/s} = 2,91\mu s$$

Now the transmission for the sender is concluded and, after a SIFS ($10\mu s$), the receiver sends an acknoledgement. Again we have to consider another PHY header like for the data frame and then a MAC header which is only 10 bytes and its sent at data rate[10]. There is no data, but FCS is present and like for data frames is 4 bytes at data rate. So total ACK time in 11 Mb/s is:

$$T_{ACK} = T_{PHY_{HR/DSSS}} + \frac{10 \times 8bits}{11Mb/s} + T_{FCS} = 102,55\mu s$$

Now we can calculate total transmission time for 11 Mb/s summing all the values obtained previously:

$$T_{TRANSMISSION} \simeq 1687,81\mu s$$

So the theoretical throughput for the 11 Mb/s at application level with a payload of 1470 bytes is:

$$Throughput_{APP} = \frac{1470 \times 8bits}{1687,81\mu s} \simeq 6,97Mb/s$$

If we repeat the calculation for all tested speeds and for all levels we obtain the results in table 1.

We have compute also the results with the backoff counter set to 0 because its value is randomly generated, so in case of a bad random generator or if we are particularly lucky, throughput can increase, but it will never be higher than those values.

[9]Frame Check Sequence

[10]Because we have a simmetric configuration with only one speed allowed

Level	**Payload (bytes)**	**Backoff 15,5**			**Backoff 0**		
		Speeds (Mb/s)			**Speeds (Mb/s)**		
		1	5,5	11	1	5,5	11
APPLICATION	1470	0,90	4,18	6,97	0,92	4,70	8,54
UDP	1478	0,90	4,20	7,01	0,92	4,72	8,58
IP	1498	0,91	4,26	7,10	0,93	4,79	8,70
LLC	1506	0,92	4,28	7,14	0,94	4,81	8,74

Table 1: Theoretical average (backoff counter 15,5 and 0) throughputs (Mb/s) for 802.11b

3.3 802.11g

The 802.11g MAC protocol behaviour is the same as the one seen in Figure 2, but with some differences in order to improve performances. First of all, the standard has a new PHY specification which is ERP[11]. It must be backward compatible with DSSS mode defined for 802.11b, but we will analyze the *"non compatible"* mode, because as mentioned before, we will disable all speeds on the AP except the one to be tested. In the standard, it is written that in case of a BSS made only of g clients, a pure ERP-OFDM mode can be used, to increase throughput. OFDM[12] is the modulation technique used in the standard 802.11a to send data at higher speeds, that are 6, 9, 12, 18, 24, 36 and 54 Mbps.

Because of these new specifications, theoretical analysis of physical times changes a lot. First of all, DIFS is shorter, because slot time is decreased to $9\mu s$. So we have

$$T_{DIFS} = T_{SIFS} + 2 \times T_{SLOT} = 28\mu s$$

Because of the shorter slot, even the average time of the backoff procedure decreases significantly:

$$T_{BACKOFF} = 15,5 \times T_{SLOT} = 138,5\mu s$$

[11]Extended Rate PHY

[12]Ortogonal Frequency Division Multiplexing

The PHY header, in pure g mode, changes completely. We don't have long/short preambles anymore. We have now a preamble made of 12 symbols, which takes $16\mu s$ to be sent, as we can see in Figures 7 and 8.

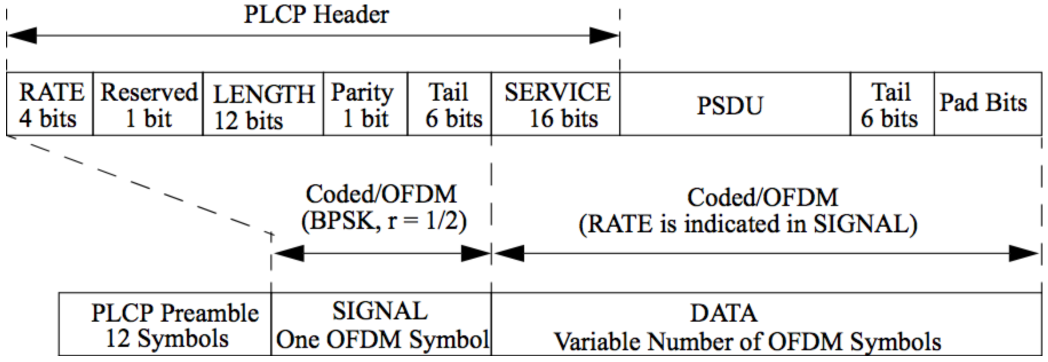


Figure 7: PHY PDU format

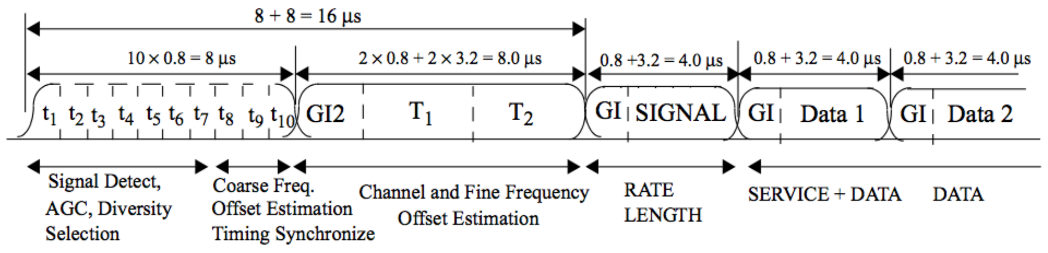

8 + 8 = 16 µs
10 × 0.8 = 8 µs
2 × 0.8 + 2 × 3.2 = 8.0 µs
0.8 +3.2 = 4.0 µs
0.8 + 3.2 = 4.0 µs
0.8 + 3.2 = 4.0 µs
t1 t2 t3 t4 t5 t6 t7 t8 t9 t10
GI2
T1
T2
GI SIGNAL
GI Data 1
GI Data 2
Signal Detect, AGC, Diversity Selection
Coarse Freq. Offset Estimation Timing Synchronize
Channel and Fine Frequency Offset Estimation
RATE LENGTH
SERVICE + DATA
DATA

Figure 8: Preamble duration

After that, we have the PLCP header made of 40 bits. The first 24, are sent at 6 Mbps, so they takes a fixed amount of time of 4 μs. PLCP header contains:

- RATE: indicates the speed at which data will be sent
- RESERVED: for future use
- LENGTH: represents the number of bytes that the MAC layer is requesting the PHY to send
- PARITY: a even parity bit for bits from 0 to 16
- TAIL: these bits are all set to 0

The last 16 bits, called service, contains 7 bits that are all set to 0 and the others are reserved for future use. They are sent at data rate, so for convenience, we will consider them as a part of the PHY data. The total time of the PHY header is

$$T_{PHY_{ERP}} = T_{PREAMBLE} + T_{PLCP} = 20\mu s$$

The PPDU contains service field (16 bits) sent at data rate, then we have the MAC header. As for 802.11b, it is made of 24 bytes transmitted at nominal speed. In case of 54 Mbps, we have

$$T_{SERVICE} = \frac{16bits}{54Mbps} = 0,3\mu s$$

$$T_{MAC} = \frac{24 \times 8bits}{54Mbps} = 3,56\mu s$$

Then we have our payload and the FCS, as for 802.11b, so with a nominal speed of 54 Mbps

$$T_{DATA} = \frac{1506 \times 8bits}{54Mbps} = 223,11\mu s$$

$$T_{FCS} = \frac{4 \times 8bits}{54Mbps} = 0,59\mu s$$

Now, things changes again: as we can see in Figure 7, after MAC frame there are other 6 bits, called tail, that take

$$T_{TAIL} = \frac{6bits}{54Mbps} = 0,11\mu s$$

Then there are pad bits: these are used to make the message a multiple of the N_{DBPS} value, which is the number of data bits per OFDM symbol. To calculate pad length, we refer to the formula 1 which is written on the standard.

$$N_{SYM} = \left\lceil \frac{16 + 8 \times LENGTH + 6}{N_{DBPS}} \right\rceil$$

$$N_{DATA} = N_{SYM} \times N_{DBPS}$$

$$N_{PAD} = N_{DATA} - (16 + 8 \times LENGTH + 6) \tag{1}$$

where $LENGTH$ is the size of the MAC frame, that in our case is made of 1506 bytes of payload, plus MAC header and FCS, so 1534 bytes. 16 and 6 represents service and tail bits. N_{DBPS} can be found in table 2 taken from the standard. For 54 Mbps, N_{DBPS} is 216. So the number of pad bits in our case are:

$$N_{SYM} = \left\lceil \frac{16 + 8 \times 1534 + 6}{216} \right\rceil = 57$$

$$N_{DATA} = 57 \times 216 = 12312$$

$$N_{PAD} = 12312 - (16 + 8 \times 1534 + 6) = 18 bits$$

that in time means

$$T_{PAD} = \frac{18 bits}{54 Mbps} = 0,33 \mu s$$

Then, a supplementary time T_{SIGNAL} of 6μs, called signal extension, is added. Now, sender transmission is over and, after a SIFS, the receiver sends the acknoledgement. Like for a data frame, we have to consider some fixed times (fixed in respect to the data frame):

- PHY header (20μs)
- Service bits (0,3μs)
- Tail bits (0,11μs)
- FCS (0,59μs)
- Signal extension (6μs)

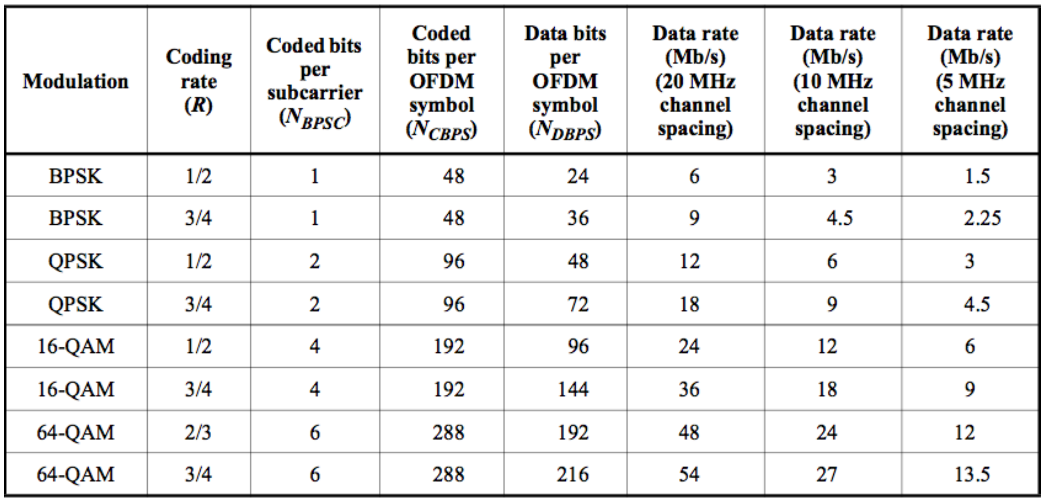
What changes from a data frame is MAC header, that is 10 bytes like in 802.11b, data that is empty, and pad bits that needs to be recalculated. By formula 1 we obtain 82 bits. So total acknoledgement time is

$$T_{ACK} = 27 \mu s + \frac{10 \times 8 + 82 bits}{54 Mbps} = 30 \mu s$$

Finally, we are able to compute theoretical throughput for an application payload of 1470 bytes and a nominal speed of 54 Mbps:

$$Throughput_{APP} = \frac{Data_{APP}}{T_{TRANSMISSION}} = \frac{1470 \times 8 bits}{461,5 \mu s} \simeq 25,48 Mb/s$$

For all measured speeds, we obtain results in table 3.



Modulation	Coding rate (R)	Coded bits per subcarrier (N_{BPSC})	Coded bits per OFDM symbol (N_{CBPS})	Data bits per OFDM symbol (N_{DBPS})	Data rate (Mb/s) (20 MHz channel spacing)	Data rate (Mb/s) (10 MHz channel spacing)	Data rate (Mb/s) (5 MHz channel spacing)
BPSK	1/2	1	48	24	6	3	1.5
BPSK	3/4	1	48	36	9	4.5	2.25
QPSK	1/2	2	96	48	12	6	3
QPSK	3/4	2	96	72	18	9	4.5
16-QAM	1/2	4	192	96	24	12	6
16-QAM	3/4	4	192	144	36	18	9
64-QAM	2/3	6	288	192	48	24	12
64-QAM	3/4	6	288	216	54	27	13.5

Table 2: Modulation dependent parameters

3.4 Considerations about theoretical results

Is interesting to compare now theoretical results obtained in sections 3.2 and 3.3 in terms of efficiency. First of all we can observe in tables 4 and 5 that efficiency is higher at lower speeds. This happens because we have a fixed amount of data that, at lower speeds, requires much more time to be sent, reducing the influence of dead times such as DIFS, backoff procedure or PHY header on throughput.

Now what we can compute is the difference in percentual between nominal and throughput values of similar speeds. For example, the difference between 5,5 and 6 Mbps (nominal speeds) is 8,33% calculated by formula 2.

$$\text{Difference}(\%) = (1 - \frac{5,5Mbps}{6Mbps}) \times 100 = 8,33\% \qquad (2)$$

If we calculate the difference between their respective throughputs, we obtain 18,06%. This means that g protocol is much more performant. We can see this fact even better having a look at 11 and 12 Mbps: the difference between nominal speeds is still 8,33%, but now, the difference in throughput is 24,78%. This is for sure given by the fact that dead times are much more shorter in g than in b, but

Level	**Payload**	**Backoff 15,5**				**Backoff 0**			
		Speeds (Mb/s)				**Speeds (Mb/s)**			
		6	12	36	54	6	12	36	54
APP	1470 B	5,10	9,26	20,36	25,48	5,43	10,41	26,85	36,52
UDP	1478 B	5,13	9,31	20,47	25,62	5,46	10,46	27,00	36,72
IP	1498 B	5,20	9,44	20,75	25,97	5,53	10,61	27,36	37,22
LLC	1506 B	5,23	9,49	20,86	26,11	5,56	10,66	27,51	37,42

Table 3: Theoretical average (backoff counter 15,5 and 0) throughputs (Mb/s) for 802.11g

having this shorter times is possible only in a BSS where no b stations are present. If we use a mixed b/g mode, g clients will loose a lot of performances, because they would have to use a slot time of $20\mu s$ and, in the best case, the short preamble used in HR/DSSS mode. So if we know for sure that our BSS will be made by only g clients, is much better to disable b speeds on the access point.

Speeds (Mb/s)		
1	5,5	11
89,51%	75,99%	63,34%

Table 4: Theoretical efficiency of 802.11b at application level (backoff counter 15,5)

Speeds (Mb/s)			
6	12	36	54
85,01%	77,20%	56,57%	47,19%

Table 5: Theoretical efficiency of 802.11g at application level (backoff counter 15,5)

3.5 Practical results

As specified at the beginning of the section, for each speed we have done 20 tests each of them of 30 seconds, with an output interval of 1 second. We are going to present every speed test with two graphs. The first represents a sort of *"instantaneous throughput"*, in the sense that we have plotted every single second of each test. Note that values are plotted consecutively, but in fact they are not: some value has been discarded and also there is a variable gap between one test and another. Since this gap does not exceed one minute, the conditions between the tests should not change drastically. For these reasons the abscissa is expressed in *"equivalent time"* and not *"time"* and the graphs should anyway provide an idea of the state of the network.

In some cases, interferences disturb almost the entire test, as well shown in 3.5.2. To have anyway a sort of comparison with theoretical values, on the graph two orizontal lines are plotted, which represent theoretical average (backoff 15,5) and maximum (backoff 0) throghputs. Obviously the higher line represents the maximum value while the lower represents the average.

In the second, we have kept separated the 20 tests and for each one we have calculated an average which is plotted in the graph. For every point we have also a confidence interval of the 95% used to have an idea of the reliability of the measurements. For example, having a large confidence interval, means that throughput values were jumping up and down, alternating good and bad moments: these bad moments could be cause of disturbances, unwanted connections to our access point, channel contentions or iperf errors.

Finally there is also a table which shows some common results like average, standard deviation, etc.
The approach followed to clean our data is to discard:

1. values over the theoretical maximum throughput
2. values at the beginning of each test
3. 0 values

We have chosen criteria 1 and 3 because we have interpreted them as iperf's measurement error, the second because sometimes iperf beginning outputs are strange values over the theoretical maximum and often repeated. For example, for 1 Mbps the first values are 999600 bps. This does not always happen, for instance the 54 Mbps test was very good. We have instead decided to keep low values, because they give us an idea of how the network behaves in an environment like the faculty one.

To have an idea of the values we have discarded, simply look at Figures 9 and 10, which show the 1Mbps graph before and after the clean up. It is interesting to notice that high peaks coincide with the beginning of the tests.

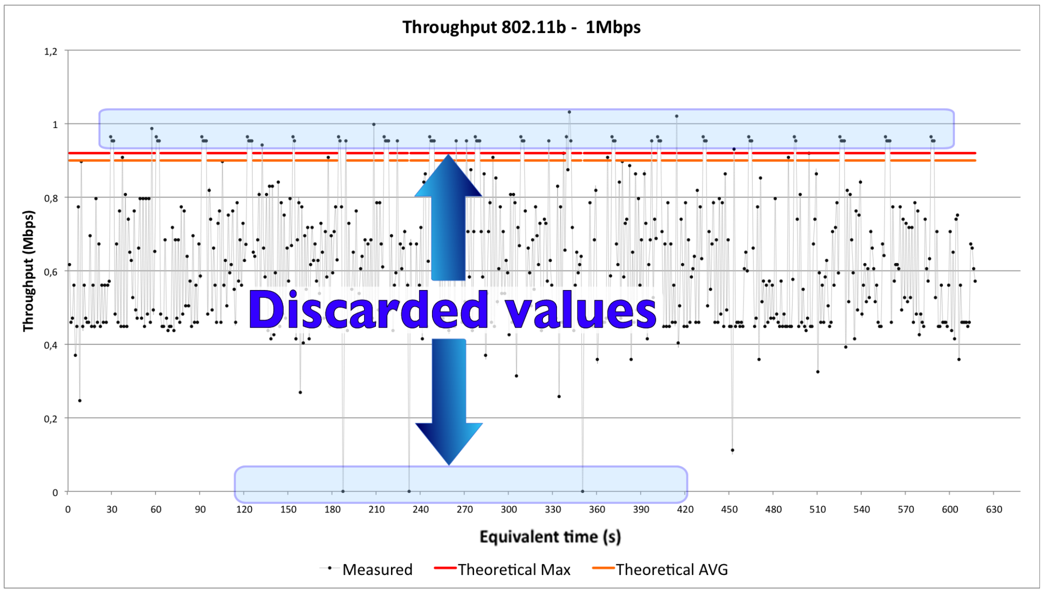


Figure 9: Graph with bad values

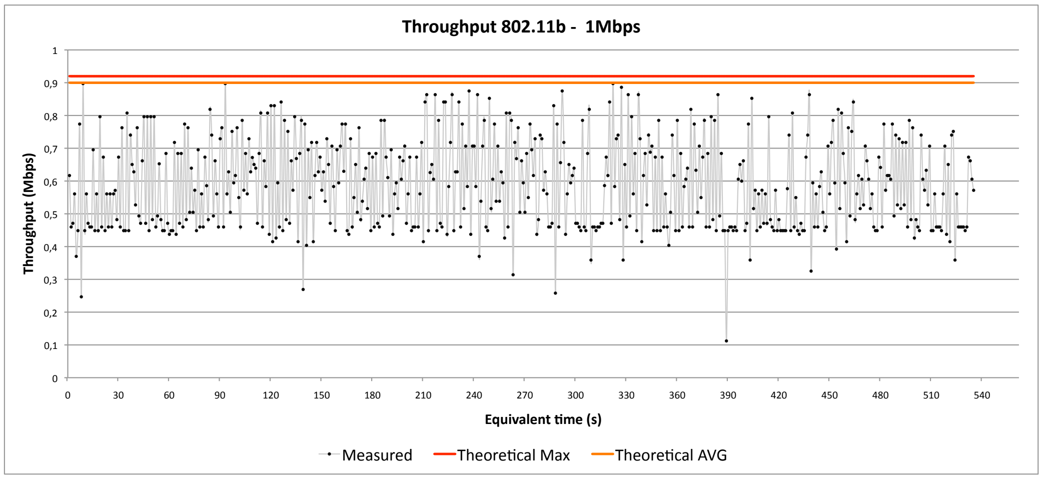


Figure 10: Cleaned graph

Every throughput value presentend in this section is intended at application layer because iperf outputs measured throughputs at this level.
In the next sections we will show an analyze some interesting cases: all other graphs and collected results can be found in the appendix.

3.5.1 802.11b: 1 Mbps

In the instantaneous graph (Figure 11), we can see an alternance of good values, near to the theoretical average, and low values. It is interesting to notice that most of the bad values are between 0,4 and 0,5. Given this fact, we have tried to compute the throughput in case of collision, and so in case of retransmission of the entire PPDU. Given that the time needed to transmit it once is 13138 μs, doubling it we obtain[13]

$$Throughput_{RETR} = \frac{1470 \times 8bits}{13138 \times 2 + (16 \times 20)\mu s} = 0,44Mb/s$$

[13]We have to add 16 $\times$ 20 μs because in case of retransmission, the contention window must be doubled, so it becomes 64. On average we have a backoff value of 32,5, but into the 13138 μs are already counted 15,5 slots, so the remaining are 32,5 - 15,5 = 16 slots

which is roughly between 0,4 and 0,5. So we could suppose that those bad values were caused by collisions and so, by transmission of the entire frame two times. We know that this is a severe supposition because it's like afferming that all frames transmitted in the time slot (roughly 40) were retransmitted. For this reason we have also to consider other possibilities. The values in between, could be justified by the fact that another station, connected to another access point, but on the same channel, has won the contention against our station, so our NIC card has postponed the access to the channel for a time which is described by the duration field (NAV) of the MAC header of the frame sent by the winner station. This time cannot be predicted, because it depends on the amount of data sent by the other station.

Note that these are only suppositions: this particular behaviour can be also a result of the measurements uncertainty of iperf. Indeed, our sampling interval were quite small (1s), so a timing error could count packets belonging to a sample into the successive one. Having a sampling interval of 1 second, and a timing error of 0,5 seconds (which is feasible at application layer) could halve the iperf throughput value.

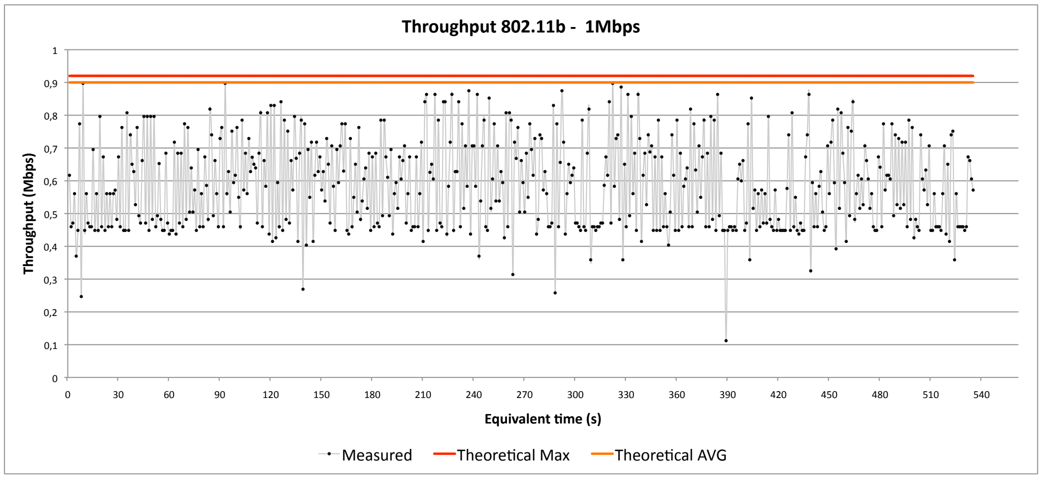


Figure 11: Instantaneous throughput

The second graph (12) shows the averages of the various tests. We can see the problem of the low values by the big confidence intervals caused from the large standard deviations. Note that the last point has a small confidence interval compared to the others: this does not mean that it is more correct, but that measured values were more closed to each other. So, if the average is low compared to the theoretical one and the confidence intervall small, it means that we were in a unlucky situation, with a constant disturbance. Indeed, this is shown in the last part of the graph in Figure 11.

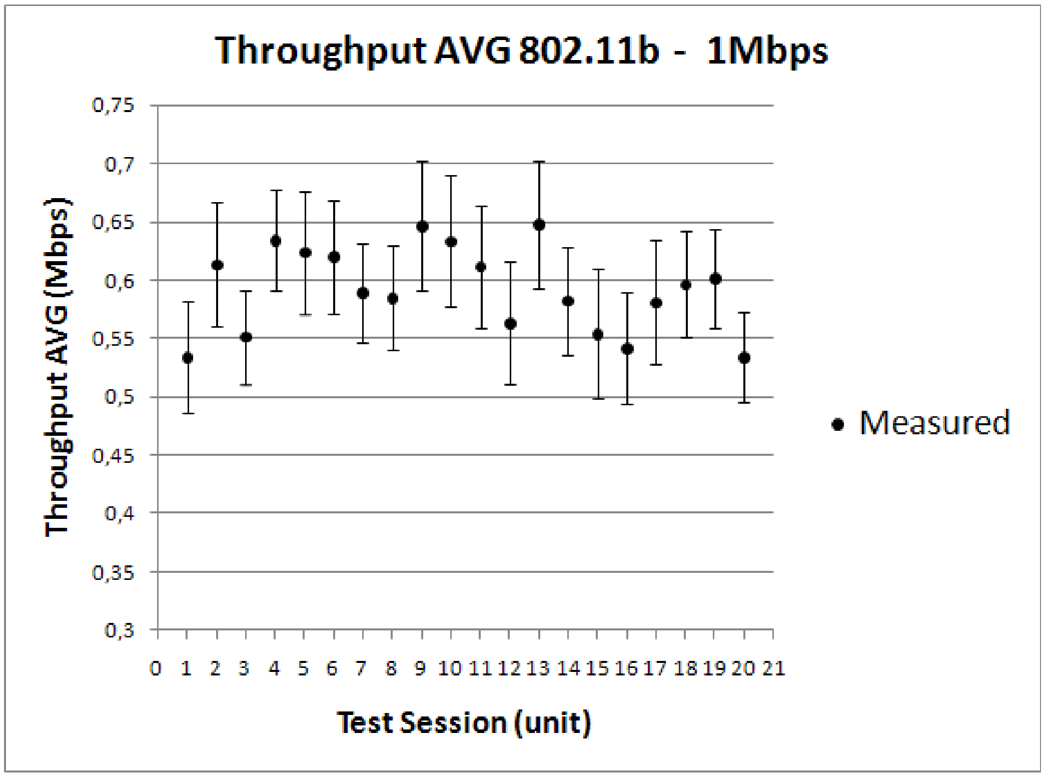


Figure 12: Graph of AVGs of the 1 Mbps tests, with confidence intervals

Table 6 shows the practical results calculated on all values collected by iperf. We can see a very low average value caused by the large number of retransmissions. Indeed, also the minimum value is very low.

Theoretical (AVG)	0.90 Mb/s
Theoretical (MAX)	0.92 Mb/s
Average	0,59 Mb/s
Std. deviation	0,14 Mb/s
Min	0,11 Mb/s
Max	0,90 Mb/s

Table 6: Practical results for 802.11b 1 Mbps

3.5.2 802.11b 5.5 Mbps

This is the worst measurement session. The reason of the majority of bad values in Figure 13 is the particular moment chosen to do our test. Probably there was too much traffic on our channel. This is well shown in Figures 14 and 15. The first shows the great number of access points near to us, and the consequence of this fact is shown in the second picture, where the link quality, which is the overall quality of the link based on various parameters like interference or frame error rate, is quite low.

What is interesting to see is how low values are disposed in Figure 13: note that they are plotted onto lines in corrispondence of the ordinate values 0,5, 0,9, 1,4 and 1,8 roughly. This fact could be, as in 3.5.1, a consequence of the number of retransmissions. Indeed, having a look at wireshark, we were able to find quickly a packet that was retransmitted four times, as shown in Figures 16 and 17. Again, we are conscious that this could be caused by a timing error of iperf.

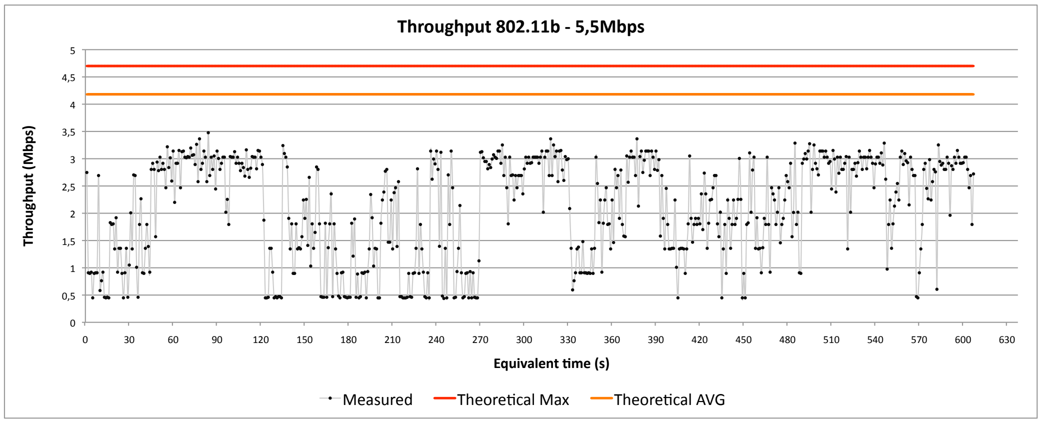
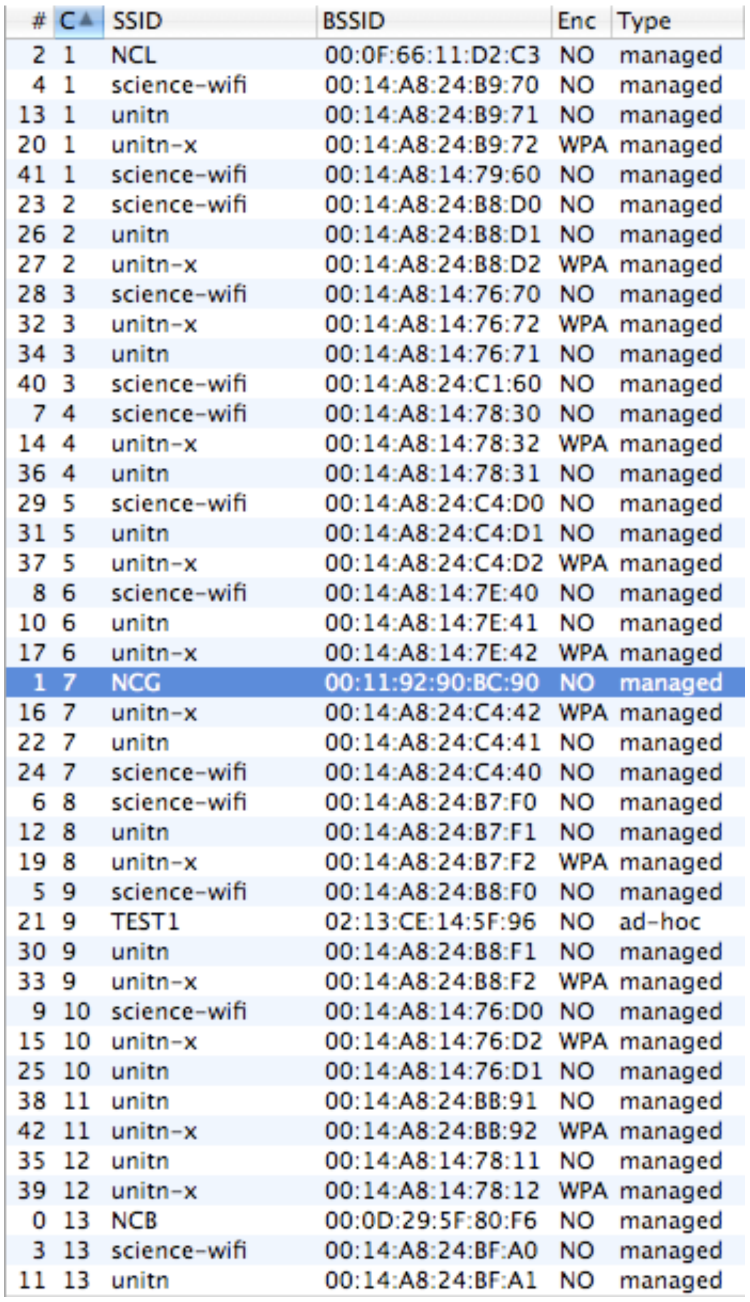
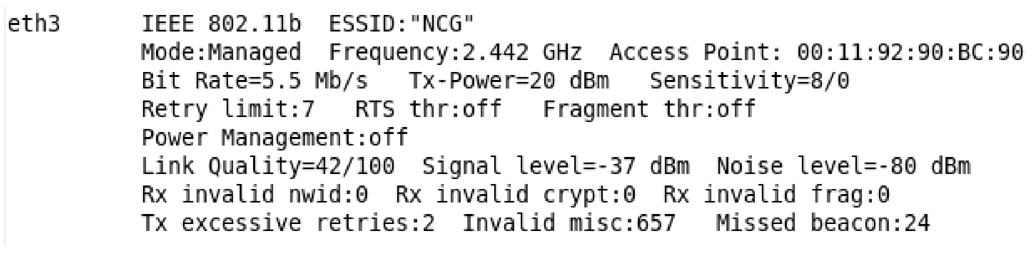


Figure 13: Instantaneous throughput



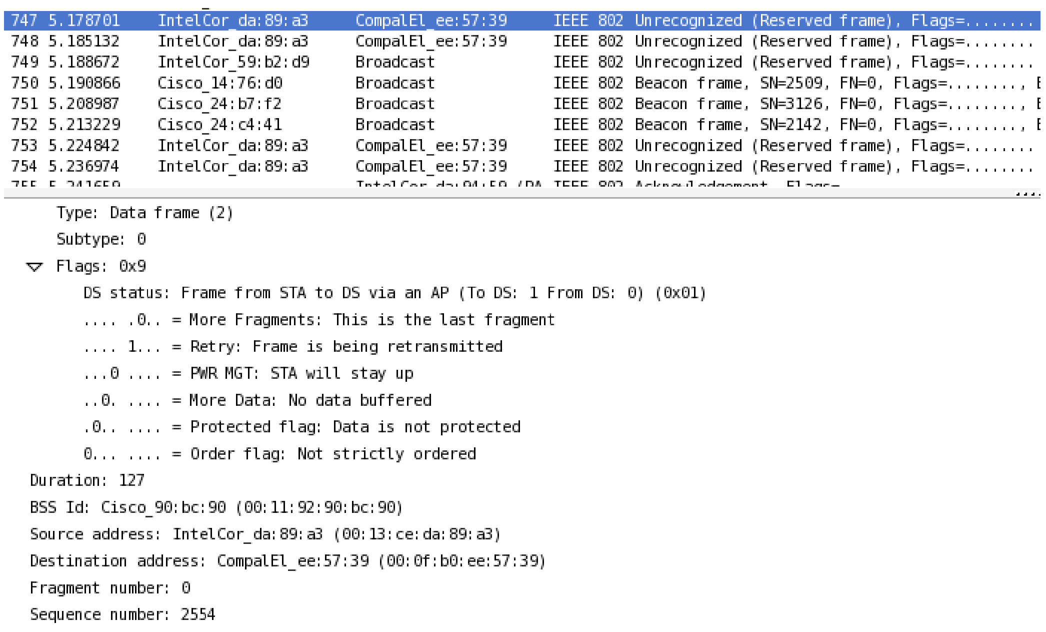
#	C	SSID	BSSID	Enc	Type
2	1	NCL	00:0F:66:11:D2:C3	NO	managed
4	1	science-wifi	00:14:A8:24:B9:70	NO	managed
13	1	unitn	00:14:A8:24:B9:71	NO	managed
20	1	unitn-x	00:14:A8:24:B9:72	WPA	managed
41	1	science-wifi	00:14:A8:14:79:60	NO	managed
23	2	science-wifi	00:14:A8:24:B8:D0	NO	managed
26	2	unitn	00:14:A8:24:B8:D1	NO	managed
27	2	unitn-x	00:14:A8:24:B8:D2	WPA	managed
28	3	science-wifi	00:14:A8:14:76:70	NO	managed
32	3	unitn-x	00:14:A8:14:76:72	WPA	managed
34	3	unitn	00:14:A8:14:76:71	NO	managed
40	3	science-wifi	00:14:A8:24:C1:60	NO	managed
7	4	science-wifi	00:14:A8:14:78:30	NO	managed
14	4	unitn-x	00:14:A8:14:78:32	WPA	managed
36	4	unitn	00:14:A8:14:78:31	NO	managed
29	5	science-wifi	00:14:A8:24:C4:D0	NO	managed
31	5	unitn	00:14:A8:24:C4:D1	NO	managed
37	5	unitn-x	00:14:A8:24:C4:D2	WPA	managed
8	6	science-wifi	00:14:A8:14:7E:40	NO	managed
10	6	unitn	00:14:A8:14:7E:41	NO	managed
17	6	unitn-x	00:14:A8:14:7E:42	WPA	managed
1	7	NCG	00:11:92:90:BC:90	NO	managed
16	7	unitn-x	00:14:A8:24:C4:42	WPA	managed
22	7	unitn	00:14:A8:24:C4:41	NO	managed
24	7	science-wifi	00:14:A8:24:C4:40	NO	managed
6	8	science-wifi	00:14:A8:24:B7:F0	NO	managed
12	8	unitn	00:14:A8:24:B7:F1	NO	managed
19	8	unitn-x	00:14:A8:24:B7:F2	WPA	managed
5	9	science-wifi	00:14:A8:24:B8:F0	NO	managed
21	9	TEST1	02:13:CE:14:5F:96	NO	ad-hoc
30	9	unitn	00:14:A8:24:B8:F1	NO	managed
33	9	unitn-x	00:14:A8:24:B8:F2	WPA	managed
9	10	science-wifi	00:14:A8:14:76:D0	NO	managed
15	10	unitn-x	00:14:A8:14:76:D2	WPA	managed
25	10	unitn	00:14:A8:14:76:D1	NO	managed
38	11	unitn	00:14:A8:24:BB:91	NO	managed
42	11	unitn-x	00:14:A8:24:BB:92	WPA	managed
35	12	unitn	00:14:A8:14:78:11	NO	managed
39	12	unitn-x	00:14:A8:14:78:12	WPA	managed
0	13	NCB	00:0D:29:5F:80:F6	NO	managed
3	13	science-wifi	00:14:A8:24:BF:A0	NO	managed
11	13	unitn	00:14:A8:24:BF:A1	NO	managed

Figure 14: List of access points present during 5.5 Mbps measurements



```
eth3      IEEE 802.11b  ESSID:"NCG"
          Mode:Managed  Frequency:2.442 GHz  Access Point: 00:11:92:90:BC:90
          Bit Rate=5.5 Mb/s   Tx-Power=20 dBm   Sensitivity=8/0
          Retry limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=42/100  Signal level=-37 dBm  Noise level=-80 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:2  Invalid misc:657   Missed beacon:24
```

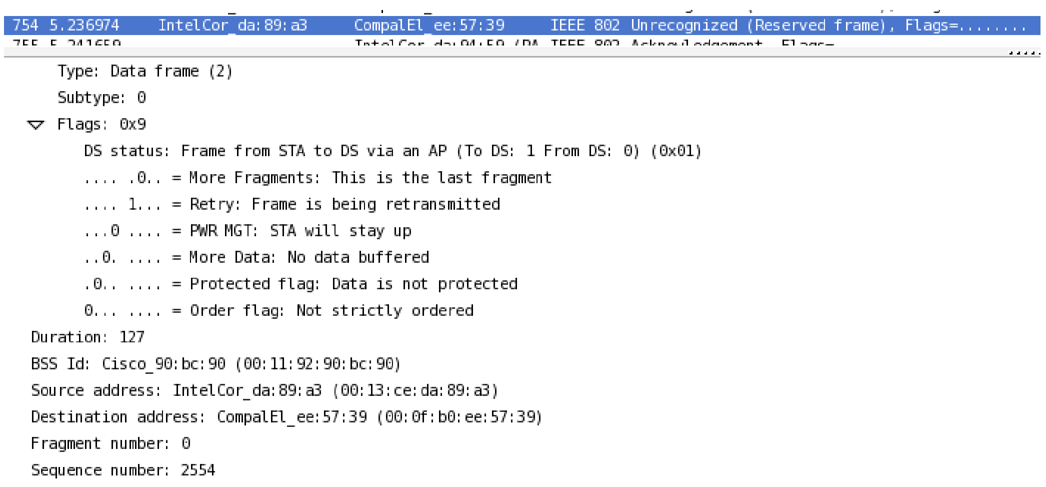
Figure 15: Link quality shown by the NIC card during 5.5 Mbps measurements



```
747 5.178701   IntelCor_da:89:a3    CompalEl_ee:57:39    IEEE 802 Unrecognized (Reserved frame), Flags=........
748 5.185132   IntelCor_da:89:a3    CompalEl_ee:57:39    IEEE 802 Unrecognized (Reserved frame), Flags=........
749 5.188672   IntelCor_59:b2:d9    Broadcast            IEEE 802 Unrecognized (Reserved frame), Flags=........
750 5.190866   Cisco_14:76:d0       Broadcast            IEEE 802 Beacon frame, SN=2509, FN=0, Flags=........, E
751 5.208987   Cisco_24:b7:f2       Broadcast            IEEE 802 Beacon frame, SN=3126, FN=0, Flags=........, E
752 5.213229   Cisco_24:c4:41       Broadcast            IEEE 802 Beacon frame, SN=2142, FN=0, Flags=........, E
753 5.224842   IntelCor_da:89:a3    CompalEl_ee:57:39    IEEE 802 Unrecognized (Reserved frame), Flags=........
754 5.236974   IntelCor_da:89:a3    CompalEl_ee:57:39    IEEE 802 Unrecognized (Reserved frame), Flags=........
755 5.241659                        IntelCor_da:94:59 (RA IEEE 802 Acknowledgement, Flags=

    Type: Data frame (2)
    Subtype: 0
  ▽ Flags: 0x9
      DS status: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x01)
      .... .0.. = More Fragments: This is the last fragment
      .... 1... = Retry: Frame is being retransmitted
      ...0 .... = PWR MGT: STA will stay up
      ..0. .... = More Data: No data buffered
      .0.. .... = Protected flag: Data is not protected
      0... .... = Order flag: Not strictly ordered
  Duration: 127
  BSS Id: Cisco_90:bc:90 (00:11:92:90:bc:90)
  Source address: IntelCor_da:89:a3 (00:13:ce:da:89:a3)
  Destination address: CompalEl_ee:57:39 (00:0f:b0:ee:57:39)
  Fragment number: 0
  Sequence number: 2554
```

Figure 16: Packet with sequence number 2554, at first retransmission



```
754 5.236974     IntelCor_da:89:a3     CompalEl_ee:57:39     IEEE 802 Unrecognized (Reserved frame), Flags=........
      Type: Data frame (2)
      Subtype: 0
   ▽ Flags: 0x9
         DS status: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x01)
         .... .0.. = More Fragments: This is the last fragment
         .... 1... = Retry: Frame is being retransmitted
         ...0 .... = PWR MGT: STA will stay up
         ..0. .... = More Data: No data buffered
         .0.. .... = Protected flag: Data is not protected
         0... .... = Order flag: Not strictly ordered
   Duration: 127
   BSS Id: Cisco_90:bc:90 (00:11:92:90:bc:90)
   Source address: IntelCor_da:89:a3 (00:13:ce:da:89:a3)
   Destination address: CompalEl_ee:57:39 (00:0f:b0:ee:57:39)
   Fragment number: 0
   Sequence number: 2554
```

Figure 17: Packet with sequence number 2554, at forth retransmission

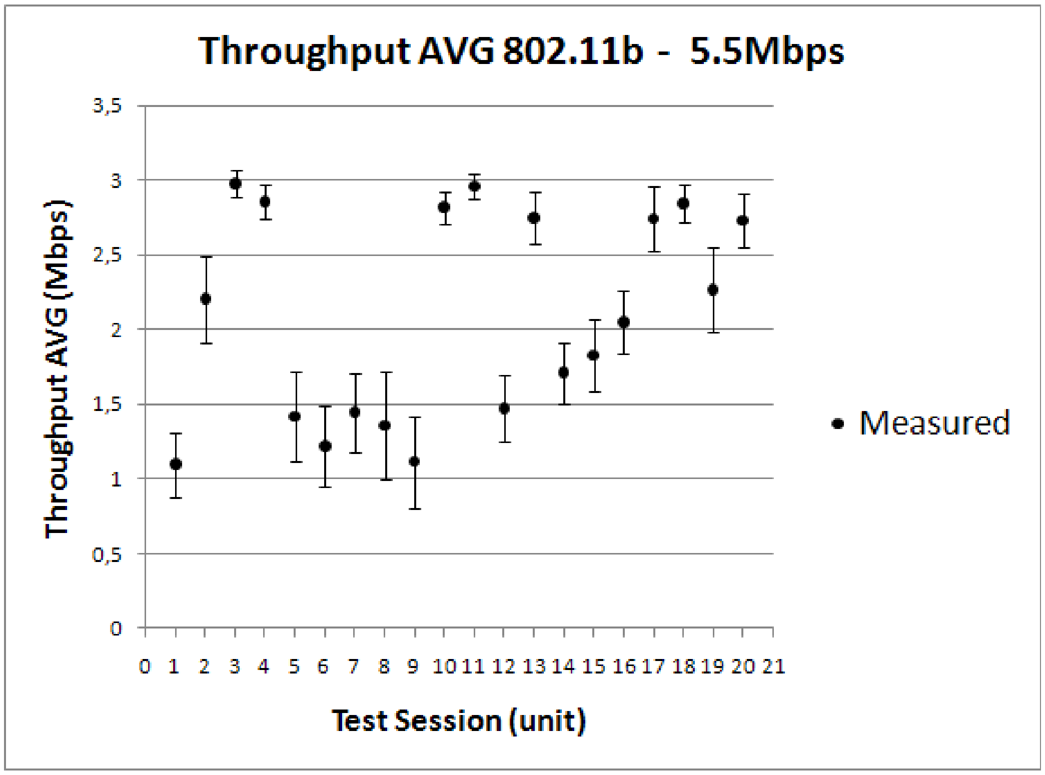


Figure 18: Graph of AVGs of the 5 Mbps tests, with confidence intervals

Theoretical (AVG)	4,18 Mb/s
Theoretical (MAX)	4,70 Mb/s
Average	2,10 Mb/s
Std. deviation	0,93 Mb/s
Min	0,43 Mb/s
Max	3,47 Mb/s

Table 7: Practical results for 802.11b 5.5 Mbps

3.5.3 802.11g 54 Mbps

A good measurement session was the 54 one. As shown in Figure 19, almost all samples are closed to the theoretical throughput except for the last part. We have tried to do this measurement into the room *"Presidio informatico"*, under the escalators. In there we had much less access points interfering, as shown in Figure 20: this fact is reflected on the link quality shown in Figure 21.

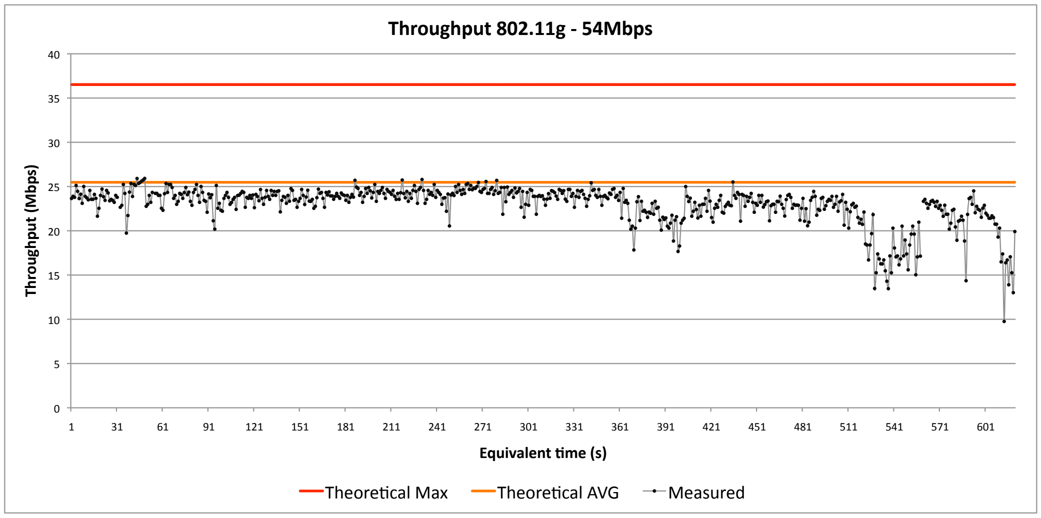
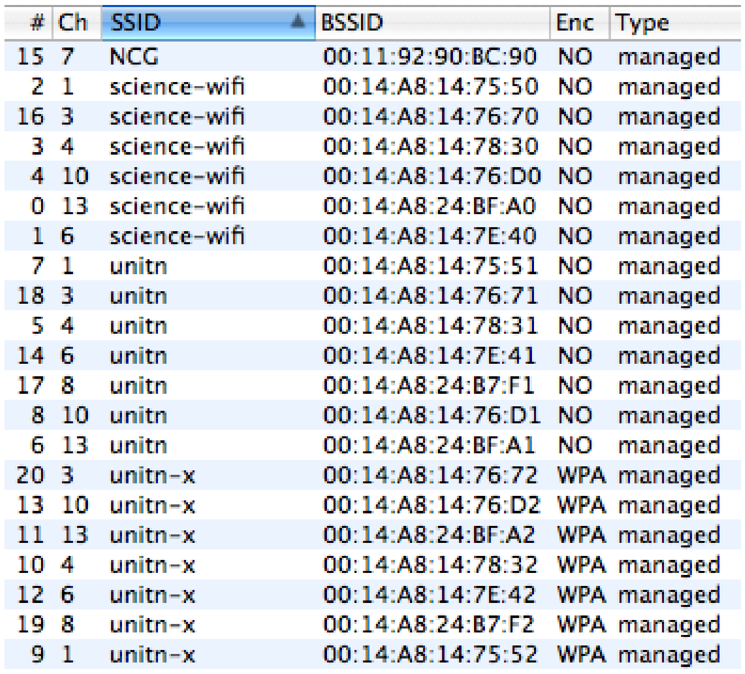
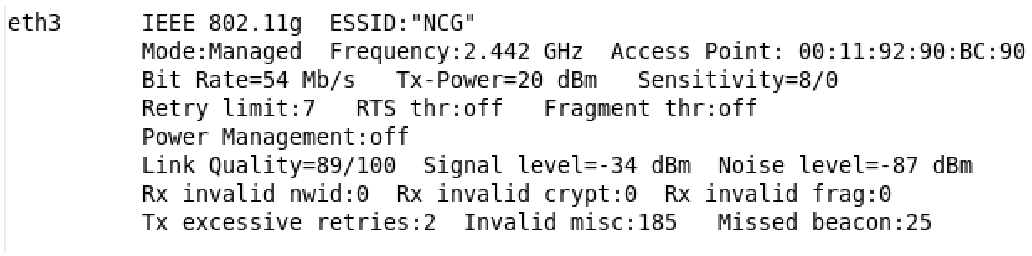


Figure 19: Instantaneous throughput



#	Ch	SSID	BSSID	Enc	Type
15	7	NCG	00:11:92:90:BC:90	NO	managed
2	1	science-wifi	00:14:A8:14:75:50	NO	managed
16	3	science-wifi	00:14:A8:14:76:70	NO	managed
3	4	science-wifi	00:14:A8:14:78:30	NO	managed
4	10	science-wifi	00:14:A8:14:76:D0	NO	managed
0	13	science-wifi	00:14:A8:24:BF:A0	NO	managed
1	6	science-wifi	00:14:A8:14:7E:40	NO	managed
7	1	unitn	00:14:A8:14:75:51	NO	managed
18	3	unitn	00:14:A8:14:76:71	NO	managed
5	4	unitn	00:14:A8:14:78:31	NO	managed
14	6	unitn	00:14:A8:14:7E:41	NO	managed
17	8	unitn	00:14:A8:24:B7:F1	NO	managed
8	10	unitn	00:14:A8:14:76:D1	NO	managed
6	13	unitn	00:14:A8:24:BF:A1	NO	managed
20	3	unitn-x	00:14:A8:14:76:72	WPA	managed
13	10	unitn-x	00:14:A8:14:76:D2	WPA	managed
11	13	unitn-x	00:14:A8:24:BF:A2	WPA	managed
10	4	unitn-x	00:14:A8:14:78:32	WPA	managed
12	6	unitn-x	00:14:A8:14:7E:42	WPA	managed
19	8	unitn-x	00:14:A8:24:B7:F2	WPA	managed
9	1	unitn-x	00:14:A8:14:75:52	WPA	managed

Figure 20: List of access points present during 54 Mbps measurements



```
eth3      IEEE 802.11g  ESSID:"NCG"
          Mode:Managed  Frequency:2.442 GHz  Access Point: 00:11:92:90:BC:90
          Bit Rate=54 Mb/s   Tx-Power=20 dBm   Sensitivity=8/0
          Retry limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=89/100  Signal level=-34 dBm  Noise level=-87 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:2  Invalid misc:185   Missed beacon:25
```

Figure 21: Link quality shown by the NIC card during 54 Mbps measurements

The good results of the test can also be seen in Figure 22, where the first tests have a good average and a small confidence interval.

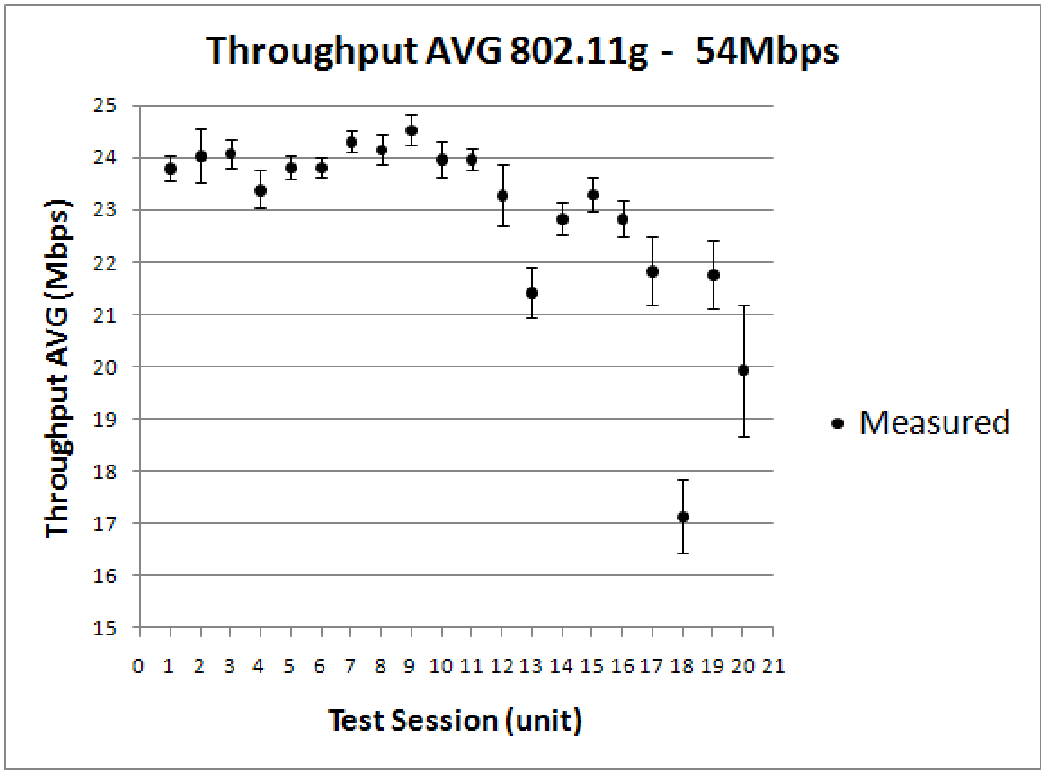


Figure 22: Graph of AVGs of the 54 Mbps tests, with confidence intervals

Theoretical (AVG)	25,48 Mb/s
Theoretical (MAX)	36,52 Mb/s
Average	23,39 Mb/s
Std. deviation	2,23 Mb/s
Min	9,76 Mb/s
Max	25,92 Mb/s

Table 8: Practical results for 802.11g 54 Mbps

4 Experiment 2: Fragmentation

In this experiment, we want to see how much throughput is affected if we start fragmenting our data frames. First of all we need to comprehend why fragmentation is used and how the protocol implements it. To fragment means breaking a data frame into smaller sub pieces. Imagine to have a situation like in Figure 23: we are transmitting a large amount of data in a unique MAC frame. For some reasons, like a station sending packets on a different channel, a disturbance interfere with our signal. The consequence is that the entire frame is lost and needs to be retransmitted. Moreover, our station cannot stop transmitting, because in WLAN is not possible to do carrier sensing and detect collisions. So we have also a big waste of time.

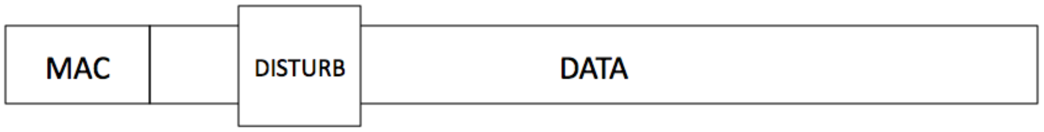


Figure 23: Loss of entire data in case of disturbances

To reduce this problem, imagine to split our data in sub pieces. Consider now Figure 24: in this case, we have divided our data into two fragments and the disturb affects only the first fragment. Each fragment must be acknowledged but the first clearly won't be. So transmitter can stop sending after half of the compromised transmission, saving time. We can also use more fragments, for example three: then it could save even more time.

Notice that we have to be careful while dealing with this mechanism: as we will see, every fragment has its own PHY header, MAC header and ACK frame. This means that the more fragments we have, the more overhead, the less throughput.

In our tests, we want to observe what changes in terms of throughput, using 2, 3 and 4 fragments for the same amount of data we had in the previous test. We will test 1 and 11 Mbps for 802.11b and 12, 36 and 54 Mbps for 802.11g. As before, we will run iperf 20 times for every test, with an output interval of 1 second, again in UDP mode.

Because we want to run the test monodirectionally and with a simmetric configuration, we have to set our parameters not only on the access but also on the NIC card, because the fragmentation threshold[14] is a local parameter, and not agreed by everybody like RTS/CTS. To set fragmentation threshold for example to 1000 bytes just type:

```
iwconfig eth1 frag 1000
```

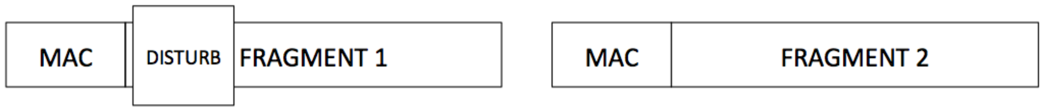


Figure 24: Fragmented MSDU: since the first frame must be acknowledged, sender station can stop immediately. The second fragment is present only to show that data is fragmented: in case of an error in the first fragment, the second won't be sent

4.1 Theoretical analysis

The parameter we tune on the access point or the NIC card is not the number of fragment that we want to obtain, but a threshold that indicates the maximum size of a frame. If a station wants to send a packet which size is higher than that value, it has to split into fragments.
At the beginning, we thought that fragmentation threshold indicates the size of the MSDU so, for example, to calculate the value needed to obtain 4 fragments with a paylod of 1506 bytes we have done:

$$\frac{1506bytes}{4} = 376,5bytes$$

[14]See 4.1 for details about this parameter

and took 380 as fragmentation threshold. But then looking at wireshark, we discovered that the number of fragments was 5. Our calculation was wrong because into the threshold is also counted MAC header. So formula 3 is the correct one.

$$\text{Threshold}(MSDU, NFrags) = \left\lceil \frac{MSDU + (24 \times NFrags)bytes}{NFrags} \right\rceil \tag{3}$$

We need to multiply MAC header for the number of fragments because every fragments has its own.
So the correct threshold to obtain 4 fragments should be:

$$\left\lceil \frac{1506 + 96bytes}{4} \right\rceil = 401bytes$$

But before being sure of this result, we have to remember one thing: the MAC layer appends at the end of every frame the FCS. Are these 4 bytes included or not into the threshold value meant by the access point or the NIC card? The answer is no, and we had verified it setting the threshold to 1530 bytes, so only MSDU size plus MAC header. If the FCS was included into it, we should have observed 2 fragment, but looking at wireshark we saw that there was no fragmentation. For our measurement we have used the following parameters:

Fragments	**Frag threshold (bytes)**
2	1000
3	600
4	420

4.2 802.11b

Calculating theoretical throghput is not so different from the procedures followed in 3.2 and 3.3. Figure 25 which shows how fragmentation works. First of all, there is only one DIFS and only one backoff procedure: after the station wins the contention, it starts transmitting the first frame, which is made of a PHY

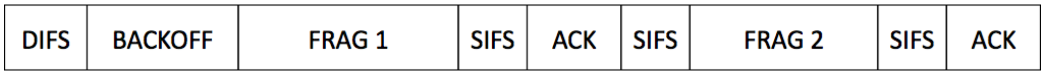


Figure 25: Fragmented transmission

header, a MAC header, the first fragmented MSDU and a FCS. Then as usual, after a SIFS, the receiver sends an ACK to the sender. Now, again after a SIFS, the sender start transmitting the second fragment as done for the first. This procedure is repeated till all fragments have been sent. By having a look picture 25, we can easily see that for calculating theoretical throughput of 802.11b, we just follow the same procedure of 3.2, but now considering:

- 1 DIFS + 1 backoff procedure
- ($2 \times$ N) PHY (N for the fragments, N for the acks)
- N MAC headers
- ($2 \times$ N - 1) SIFS
- ($2 \times$ N) FCS
- N ACK MAC headers
- MSDU that can be considered just as a unique block of data sent at datarate as before

where N is the number of fragments. Now we can easily compute the results in table 9.

4.3 802.11g

For 802.11g we can use the same approach, but now we have just a little complication: pad bits. Pad bits are appended to each fragment so we have to consider also them. If we have N fragments, N - 1 have the same size, which is fragmentation threshold plus FCS, plus SERVICE and TAIL bits, while the last is usually smaller. So we have to calculate two times pad bits, one for the last fragment and one for the others. The procedure is clearly the one already seen in formula 1. Now the first thing to do is to calculate the payloads for these

2 Fragments

Level	Payload (bytes)	Speeds (Mb/s)		
		1	5,5	11
APPLICATION	1470	0,85	3,81	6,09
UDP	1478	0,85	3,83	6,13
IP	1498	0,86	3,88	6,21
LLC	1506	0,87	3,90	6,24

3 Fragments

Level	Payload (bytes)	Speeds (Mb/s)		
		1	5,5	11
APPLICATION	1470	0,80	3,50	5,41
UDP	1478	0,81	3,52	5,44
IP	1498	0,82	3,57	5,52
LLC	1506	0,82	3,59	5,54

4 Fragments

Level	Payload (bytes)	Speeds (Mb/s)		
		1	5,5	11
APPLICATION	1470	0,77	3,24	4,87
UDP	1478	0,77	3,25	4,90
IP	1498	0,78	3,30	4,96
LLC	1506	0,78	3,32	4,99

Table 9: Theoretical average (backoff 15,5) throughputs (Mb/s) for 802.11b, 2, 3 and 4 fragments

fragments, in order to calculate then pad bits. The payload for N - 1 fragments is clearly

$$Payload_{N-1} = Threshold - MAC$$

so if we put the threshold to 1000 bytes, we obtain 976 bytes. Now, if we know the number of fragments, it is easy to compute also the payload for the last fragment, by this formula:

$$Payload_{LAST} = MSDU - (N_{FRAG} - 1) \times Payload_{N-1}$$

With a threshold of 1000 bytes we obtain 2 fragments, so the last has a payload of 530 bytes. By formula 1 we are able to calculate pad bits.

Now, to compute theoretical throughput values, we can use the same approach used for 802.11b, but considering:

- 1 DIFS + 1 backoff procedure
- (2 $\times$ N) PHY (N for the fragments, N for the acks)
- N MAC headers
- (2 $\times$ N - 1) SIFS
- (2 $\times$ N) FCS
- N ACK MAC headers
- MSDU that can be considered just as a unique block of data sent at datarate
- (N - 1) times pad bits for the first N - 1 fragments
- Pad bits of the last fragment
- N times pad bits of the ACK
- (2 $\times$ N) service and tail bits (both for fragments and acks)
- (2 $\times$ N) signal extension

where N is again the number of fragments. Results are shown in tables 10.

4.4 Considerations about theoretical results

By having a look at efficiency in percentual, as done in 3.4, in table 11 we can immediately observe that having fragmentation drastically decrease efficiency:

2 Fragments

Level	**Payload (bytes)**	**Speeds (Mb/s)**		
		12	36	54
APPLICATION	1470	8,56	17,78	21,56
UDP	1478	8,61	17,87	21,68
IP	1498	8,73	18,12	21,97
LLC	1506	8,73	18,21	22,09

3 Fragments

Level	**Payload (bytes)**	**Speeds (Mb/s)**		
		12	36	54
APPLICATION	1470	7,92	15,77	18,68
UDP	1478	7,96	15,86	18,78
IP	1498	8,07	16,08	19,04
LLC	1506	8,11	16,16	19,14

4 Fragments

Level	**Payload (bytes)**	**Speeds (Mb/s)**		
		12	36	54
APPLICATION	1470	7,38	14,18	16,67
UDP	1478	7,42	14,25	16,76
IP	1498	7,52	14,45	16,99
LLC	1506	7,56	14,52	17,08

Table 10: Theoretical average (backoff 15,5) throughputs (Mb/s) for 802.11g, 2, 3 and 4 fragments

this is given by the fact that we have the same amount of data, but more dead times. Trying to calculate percentual differences between b and g, comparing similar speeds (11 and 12 Mb/s), as in 3.4, what we obtain is displayed in table 12.

Fragments	**Speeds (Mb/s)**				
	1	11	12	36	54
1	89,51%	63,34%	77,20%	56,67%	47,19%
2	84,74%	55,38%	71,35%	49,38%	39,92%
3	80,45%	49,20%	65,97%	43,82%	34,60%
4	76,57%	44,26%	61,50%	39,38%	30,87%

Table 11: Theoretical percentual efficiency (backoff 15,5) for fragmented 802.11b/g, with a payload of 1470 bytes

Fragments	**Difference (%)**
1	24,78%
2	28,85%
3	31,64%
4	34,03%

Table 12: Theoretical percententual difference between 11 (802.11b) and 12 (802.11g) Mbps

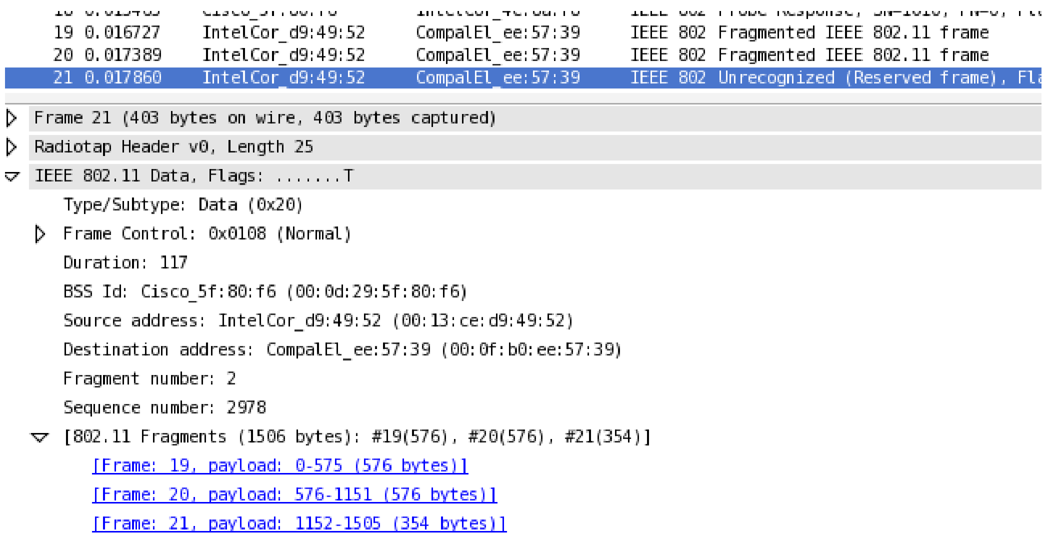
As observed before, 802.11g is much more efficient, and the gap between the two protocols is even more evident with a high number of fragments.

4.5 Practical results

As specified in 4.1, for each speed[15] we have tried to obtain 2, 3 and 4 fragments with the same amount of data. Before starting a test, we wanted to be sure

[15] 1, 11, 12, 36, 54 Mbps

that our packets were really fragmented as expected. To do this we have made use of wireshark that provides the list of fragments for each packet. Figure 26 shows the list of fragments after setting fragmentation threshold to 600 to obtain 3 fragments. We can observe three fragments with a MSDU of 576 bytes for the firts two and 354 bytes for the last. Note that 576 bytes is exactly the fragmentation threshold without 24 bytes of MAC header.



```
 19 0.016727    IntelCor_d9:49:52    CompalEl_ee:57:39    IEEE 802 Fragmented IEEE 802.11 frame
 20 0.017389    IntelCor_d9:49:52    CompalEl_ee:57:39    IEEE 802 Fragmented IEEE 802.11 frame
 21 0.017860    IntelCor_d9:49:52    CompalEl_ee:57:39    IEEE 802 Unrecognized (Reserved frame), Fl

▷ Frame 21 (403 bytes on wire, 403 bytes captured)
▷ Radiotap Header v0, Length 25
▽ IEEE 802.11 Data, Flags: .......T
    Type/Subtype: Data (0x20)
  ▷ Frame Control: 0x0108 (Normal)
    Duration: 117
    BSS Id: Cisco_5f:80:f6 (00:0d:29:5f:80:f6)
    Source address: IntelCor_d9:49:52 (00:13:ce:d9:49:52)
    Destination address: CompalEl_ee:57:39 (00:0f:b0:ee:57:39)
    Fragment number: 2
    Sequence number: 2978
  ▽ [802.11 Fragments (1506 bytes): #19(576), #20(576), #21(354)]
      [Frame: 19, payload: 0-575 (576 bytes)]
      [Frame: 20, payload: 576-1151 (576 bytes)]
      [Frame: 21, payload: 1152-1505 (354 bytes)]
```

Figure 26: A fragmented packet reconstructed by wireshark

4.5.1 802.11b 1 Mbps

Because of the disturbed environment, before starting the tests, we expected to have a better performance with fragmentation, because of the less amount of wasted time in case of collision: at least, we expected a throughput closer to the theoretical one. This kind of behaviour is pretty well shown in Figure 27, where the average throughput with 2 fragments is even more that the one without fragmentation. This difference is smaller for 11 Mbps, shown in Figure 28: in this case the measured average is closer to the theoretical, but we don't have a strong improvement of the performances by comparison with no fragmentation. This could be caused by the fact that dead times, such as SIFS, DIFS and backoff procedure, affect much more throughput at higher speeds in case of retransmission (because the MSDU is the same, but it takes less time to be transmitted). Anyway, we do not have the control of the disturbances, so a variation on the amount of interference could falsify our tests.

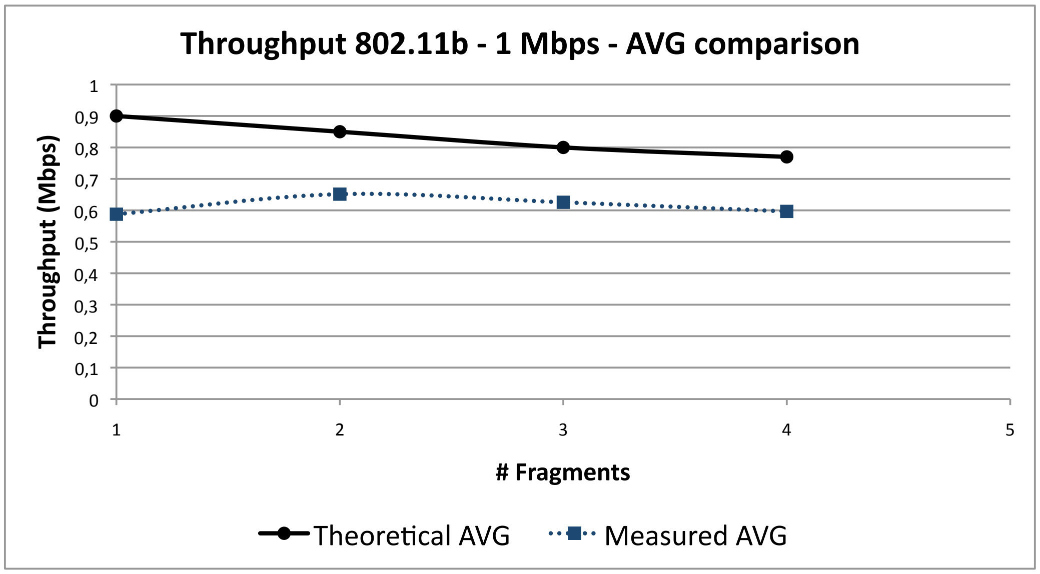


Figure 27: Comparison between theoretical and practical fragmentation throughputs

	2 Fragments	**3 Fragments**	**4 Fragments**
Theoretical (AVG)	0,85 Mb/s	0,82 Mb/s	0,77 Mb/s
Theoretical (MAX)	0,87 Mb/s	0,80 Mb/s	0,78 Mb/s
Average	0,65 Mb/s	0,62 Mb/s	0,60 Mb/s
Std. deviation	0,12 Mb/s	0,11 Mb/s	0,10 Mb/s
Min	0,40 Mb/s	0,40 Mb/s	0,37 Mb/s
Max	0,85 Mb/s	0,81 Mb/s	0,76 Mb/s

Table 13: Practical results for fragmented 802.11b 1 Mbps

4.5.2 802.11b 11 Mbps

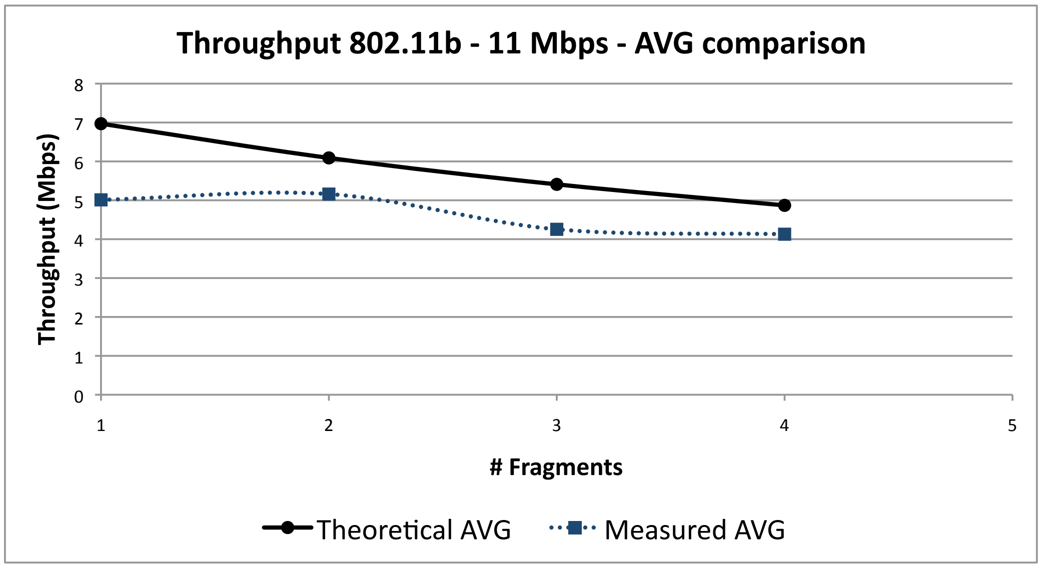


Figure 28: Comparison between theoretical and practical fragmentation throughputs

	2 Fragments	3 Fragments	4 Fragments
Theoretical (AVG)	6,09 Mb/s	5,41 Mb/s	4,87 Mb/s
Theoretical (MAX)	7,26 Mb/s	6,31 Mb/s	5,59 Mb/s
Average	5,16 Mb/s	4,25 Mb/s	4,13 Mb/s
Std. deviation	0,38 Mb/s	0,47 Mb/s	0,20 Mb/s
Min	3,81 Mb/s	3,14 Mb/s	3,59 Mb/s
Max	5,78 Mb/s	5,18 Mb/s	4,60 Mb/s

Table 14: Practical results for fragmented 802.11b 11 Mbps

4.5.3 802.11g 12 Mbps

Figure 29 shows that in this case fragmentation was not working as expected: what is interesting to notice now is that measured throughput is following the same *”trend”*, so probably the test without fragmentation was quite good, so the amount of overhead caused by multiple MAC headers and ACK frames has affected measured throughput.

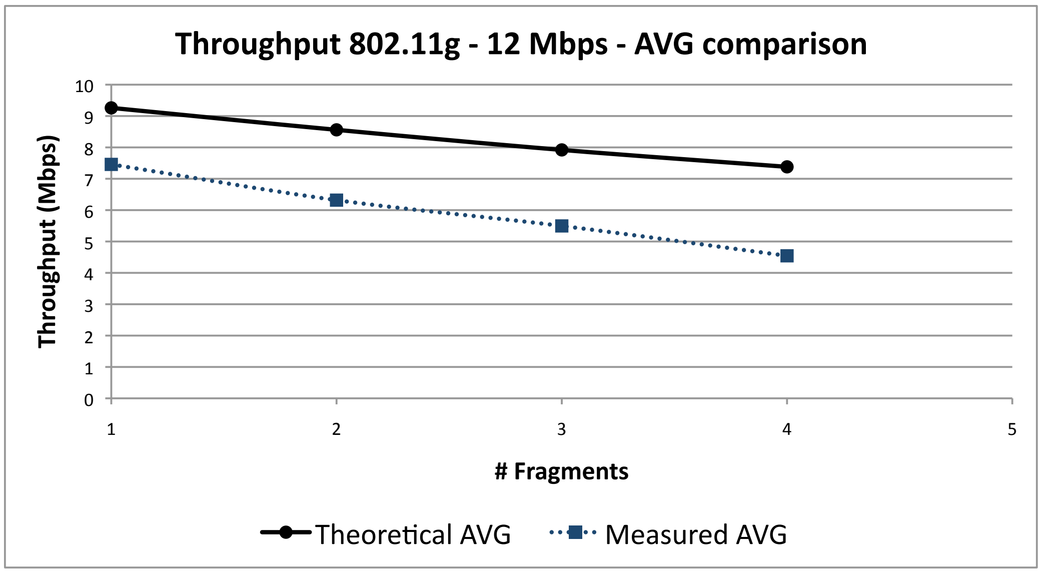


Figure 29: Comparison between theoretical and practical fragmentation throughputs

	2 Fragments	**3 Fragments**	**4 Fragments**
Theoretical (AVG)	8,56 Mb/s	7,92 Mb/s	7,38 Mb/s
Theoretical (MAX)	9,53 Mb/s	8,74 Mb/s	0,09 Mb/s
Average	6,32 Mb/s	5,50 Mb/s	4,54 Mb/s
Std. deviation	1,11 Mb/s	0,95 Mb/s	0,97 Mb/s
Min	0,90 Mb/s	6,66 Mb/s	0,45 Mb/s
Max	7,63 Mb/s	0,35 Mb/s	6,06 Mb/s

Table 15: Practical results for fragmented 802.11g 12 Mbps

5 Conclusions

During this experience, we have studied and tested how the protocols work. We had tested 1, 5,5, 6, 11, 12, 36, 54 Mbps speeds without fragmentation and 1, 11, 12, 36, 54 Mbps with fragmentation. After a careful work on theoretical analysis, we have compared theory and practise, plotting graphs and tables to have a better understanding.

We have faced a lot of problems, starting for the instruments we have used: iperf is not so reliable, and sometimes is really hard to distinguish between good values and iperf errors, making data clean-up quite hard. Also the iperf server is quite instable: sometimes it needs to be restarted, because after a long usage time it starts to output values that are completely wrong. We had some problems with wireshark, while sniffing 802.11g packets: once in a while, it can't detect frames going through the network, making impossible the analysis of their content.

We also were in trouble with our hardware instruments: sometimes, laptops were not able to associate to the access point and the NIC card returned to connect to the wireless network of the faculty.

Finally, variable (and not controllable) disturbances, could have negatively influenced our tests: we wanted some interferences to look at the protocol behaviour, but having a controlled noise should permit a better understanding. Moreover, we had not the time to do all the measurements in the same day, to have a sort of *"fixed"* environment.

In conclusion, the experience was very useful to us to integrate our knowledge, and we would have wanted to widen more aspects, but because of lack of time and low reliability of the instruments, we could not.

References

[1] Wireless LAN medium access control (MAC) and physical layer (PHY) specification, IEEE Standard 802.11^{TM}, June 2007

[2] Matthew Gast, ”802.11 Wireless Networks: The Definitive Guide, 2nd edition”, O’Reilly 2005