University of Trento,
Italy

Laboratory activities for the Nomadic Communications course 2009/2010: Report 1

■	Marco Dalla Torre	136311	marco.dallatorre@studenti.unitn.it	■
■	Simone Raffaele	140950	simone.raffaele@studenti.unitn.it	■

Anno Accademico 2009/2010

Abstract

This report describes the first experimental activity done for the course of Nomadic Communications. We can divide it in three parts: while the first part is introductory, presenting the work and the testbed configuration used to take the measurements, the remaining two parts are about the actual experimental results. In particoular, the second part is about the plain 802.11b and g throughput, and the third part analyzes the throughput in relation with various fragmentation settings. Both these parts will try to approach the problem first from a theoretical point of view and then with the analysis of the actual measured values.

Contents

1 Introduction

The goal of the experimental activity described here is to understand behaviour and capabilities of 802.11b/g devices in a "real world" scenario, and to compare them with theoretical calculations done based on what we learned in class and from the 2007 IEEE 802.11 standard.

1.1 Testbed Setup

All the measurements have been done in the faculty in the rooms 106 and 201, using mainly the CISCA provided equipment. For our group this consisted in:

- two laptops with identical hardware and with Ubuntu Linux installed. More specifically, the two laptops are equipped with the *Intel PRO/Wireless 2200BG* integrated wireless card, of course capable of supporting the b and g communication protocols, and also the *monitor* operational mode.
- a Backtrack Linux live cd[1].
- a Linksys WAP54G access point, which was shared with another group. Two additional access points were used by the other groups for their own measurements.
- an additional laptop, also running Ubuntu Linux, performing as a server for the benchmarking tools and dhcp service. This pc was shared among all the groups.

The system configuration is organized in the following way. The server, with a statically assigned ip address 192.168.10.30, communicates via ethernet connection with the three access points. All the computers involved in the test can then communicate with the server via 802.11b or g wireless connections served by the access points (stations and APs are in the same room, approximately 4/5 meters away). In order to avoid the groups to interfere with each other as much as possible, the access points are configured to use the three non overlapping frequencies allowed by 802.11: the Linksys we use is set on channel 1 (2.412MHz), while the other two are set on channel 6 (2.437Mhz) and 11 (2.462Mhz).
The laptop acting as server has basically only two tasks: running the DHCP service for assigning the testing laptops' IP addresses and running the server side of the network benchmarking tool we used.

On the first testing laptop, the benchmarking tool chose for measuring the connection throughput is netperf[2], with the following command line executed on the client:

```
netperf -H192.168.10.30 -tUDP_STREAM -l30 -fK
```

where:

-H192.168.10.30 indicates the server IP

-tUDP_STREAM indicates to use an UDP connection (default is TCP)

-l30 indicates the length in seconds for each test

-fK sets output unit in KByte/s

[1]homepage: http://www.backtrack-linux.org/
[2]homepage at http://www.netperf.org/netperf/

UDP was chose over TCP mainly because its behaviour is simpler to understand when looking through the packet streams, although less reliable. In fact, the following is what the netperf manual says about it:

> UDP is an unreliable protocol. It is important that you examine the results carefully as the reported send rate can be much higher than the actual receive rate. Great care should be taken when reporting UDP_STREAM test results to make sure they are not misleading. For example, one should always report both send and receive rates together for a UDP_STREAM test. If you are going to report a single number, you should report the receive rate.

We will consider only the performance from the point of view of the receiver: in the experiments reported here, we rarely see cases in which the sent data are different from the received data. As a side note, we can say that this frequently happened in a few experiments done with an ongoing bluetooth communication acting as interference source. However those experiments have not been further investigated, nor they are part of the content of this paper.

To setup the the benchmarking laptop we associate with the `NCL` essid and request an IP address with a dhcp client. Through all the experiments we used channel 1 (2.412GHz). At the beginning and end of the 30 tests series we save the output of the `iwconfig` command to store some statistics. This is the typical output:

```
eth3      IEEE 802.11g  ESSID:"NCL"
          Mode:Managed  Frequency:2.412 GHz  Access Point: 00:0F:66:11:D2:C3
          Bit Rate=1 Mb/s   Tx-Power=20 dBm   Sensitivity=8/0
          Retry limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=72/100  Signal level=-56 dBm  Noise level=-79 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:29   Missed beacon:15
```

One especially interesting is the link quality measurement. It is basically a numeric score based on the quality of a number of parameters, its way of operation being at discretion of the driver developer. By looking at the driver's source code and its comments, the value is recomputed every 3 seconds taking into account the following parameters:

```
* Intel® PRO/Wireless 2200BG Driver
* Link Quality calculation calculate quality based on the following:
* Missed beacon: 100% = 0, 0% = 70% missed
* Rate: 60% = 1Mbs, 100% = Max
* Rx and Tx errors represent a straight % of total Rx/Tx
* RSSI: 100% = > -50,  0% = < -80
* Rx errors: 100% = 0, 0% = 50% missed
*
* The lowest computed quality is used.
```

Now, while one laptop uses netperf to generate and measure network traffic, the second "sniffs" the traveling network packets. The benefit in doing this is that by carefully checking some message exchanges we can confirm that the configuration is what actually intended, rather than uncover anomalies in the environment that might (easily) take place. Unfortunately, it is impractical to capture all the packets exchanged during the tests, given the big quantity of data exchanged within tens of minutes compared to the low hardware capabilities of the machines at our disposal. So we suffer the limitation that, given how we did capture the data, it is not possible for us to review what exactly happened in each experiment in order to explain a peculiar result.
For this activity, an invaluable tool we extensively used is wireshark[3]. It gives us a comfortable interface that allows us to capture the "on wire" packets and examine and interpret their content. This includes MAC and LLC level headers (or even kernel level data like the contents of the radiotap sublayer in Linux), all this data would otherwise be discarded when passing each layer's payload to the upper one. However, while using the tool, we came across a tricky behaviour that should be kept in mind when snooping through the packets. For example, we had an hard time explaining ourselves why the FCS part of the 802.11 MAC header was never present in the packets: it turns out the checksum is always removed by the OS's raw packet capture mechanism. Also the PHY header cannot be captured for similar reasons.

1.2 Access point configuration

For all the test cases described in this document, the access point configuration is always kept to the default settings. All the configuration (that is, data rate and fragmentation threshold) is actually done on the client. This is because it is the client who is sending the data to the server, so setting the fragmentation level on the server would make no difference in our experiments (as the parameter configuration is completely asymmetrical like, for example, the RTS/CTS threshold). In the default configuration the access point is set to use the *mixed mode* behaviour that should allow it to accept both 802.11b and 802.11g NIC cards at the same time. This is not actually a wanted configuration, as we would have preferred to choose from a "b" and "pure g" modes given the rate we wanted to test. Unfortunately, for some reasons it was not possible for the testing laptop to associate to the access point with such configurations, unless the maximum speed allowed (11Mb/s for b, 54Mb/s for g) was used. For this reason we decided not to add any more confusion in the experiments and always leave to the mixed mode on. Moreover, by examining the wireshark captures, it looks like when >11Mb/s rates

[3]homepage: http://www.wireshark.org/

are used the AP behaves like a "pure g" anyway (probably it automatically scales to maximum speed whenever no other slower NIC card is associated).

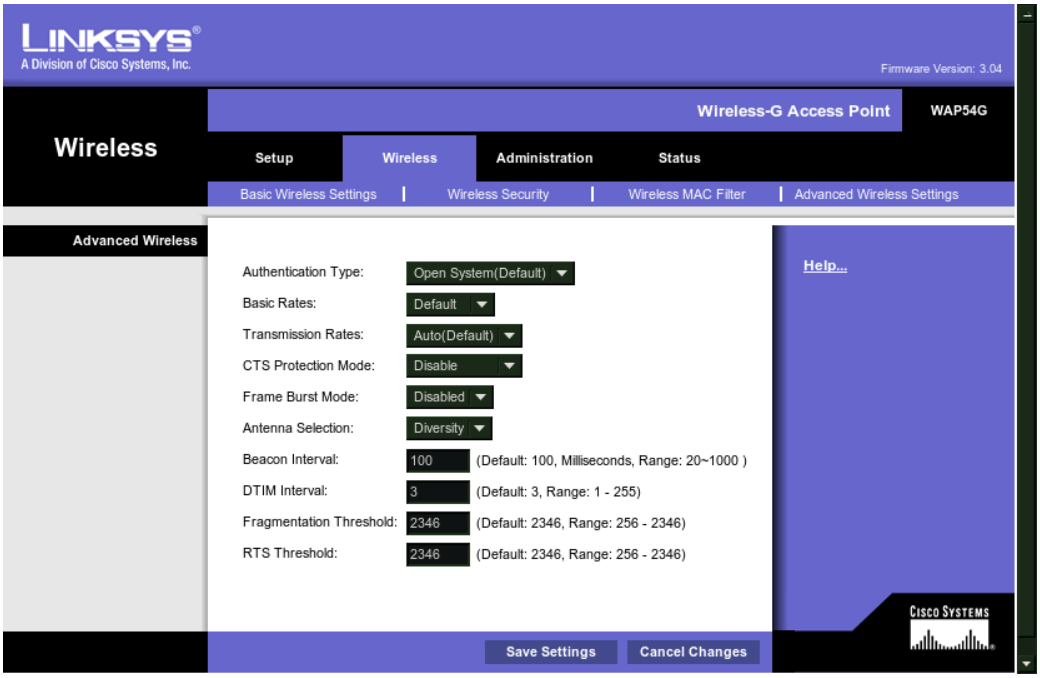


Figure 1: Advanced wireless settings tab for the Linksys AP

In figure 1 we show the `advanced wireless settings` tab of the Linksys access point. This is the configuration used in all the tests described in this report:

Authentication type we set it to `open system`, so that no authentication is required to perform association to the AP.

Basic Rates this parameter has no importance to us since it affects the data rate speeds advertised to the other wireless devices.

Transmission Rates sets the data rate used to send the data. We set it to `auto`, so that it uses the maximum supported rate for each of the connected stations.

CTS Protection Mode This has been left disabled. This option will enable the *request to send*, *clear to send* handshake mechanism when transmitting any data frame to 802.11g devices. This is intended as a protection for g devices that cannot correctly receive the data when many b devices are also present, since they may ignore OFDM signals.

Frame Burst Mode This option is supposed to increase performances in some cases, but it is not clear to us how it actually operates and for this reason it has been kept off.

Antenna Selection selects which antenna is used to transmit the data. The default is Diversity, which should enhance perfomances.

Beacon Interval the beacon sending interval is set to 100ms.

Fragmentation Threshold the fragmentation is set to the maximum segment size, which disables the fragmentation of the packets on the AP side.

RTS Threshold this option will enable the *request to send*, *clear to send*. We kept is set to the maximum segment size, so to disable it on the AP side.

2 Study on speed performances

2.1 Theoretical analysis

Since we want to analize the various experimental result that we have obtained, we have to compare them to theoretical values: these values represents the "maximal" values of throughput that should be encountered in a ideal wireless connection. In general the theoretical values are obtained using the following formula:

$$\frac{Data\ Transmitted\ (bits)}{Duration\ of\ the\ Transmition\ (\mu s)}$$

A general transmission procedure of the 802.11 protocol is shown in figure 2.

Figure 2: 802.11 transmission procedure

Since the details change between b and g mode (and also between b speeds), the following theoretical analysis will be divided in two sections. All the following calculations are created starting from what is specified in [1] (the same goes for the figures).

2.1.1 802.11b

The first part to consider, in this theoretical analysis, is the DIFS (Distributed coordinator function Inter Frame Space). It is composed by:

$$DIFS = SIFS + 2 \times Time\ Slot$$

Since in 802.11b the SIFS (Short InterFrame Space) has a duration of $10\mu s$ and the Time Slot is $20\mu s$ long, the duration of the DIFS is:

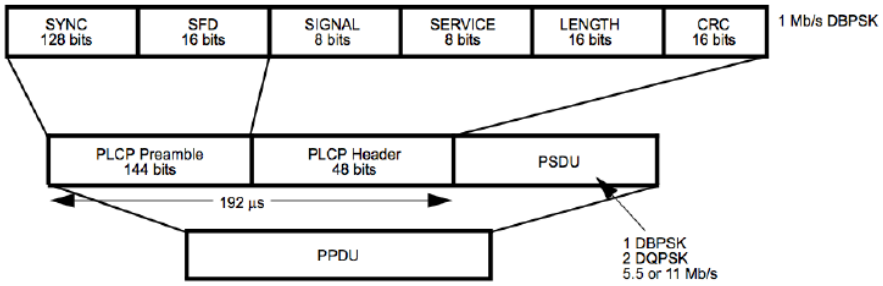
$$T_{DIFS} = T_{SIFS} + 2 \times T_{TimeSlot} = 10\mu s + 2 \times 20\mu s = 50\mu s$$

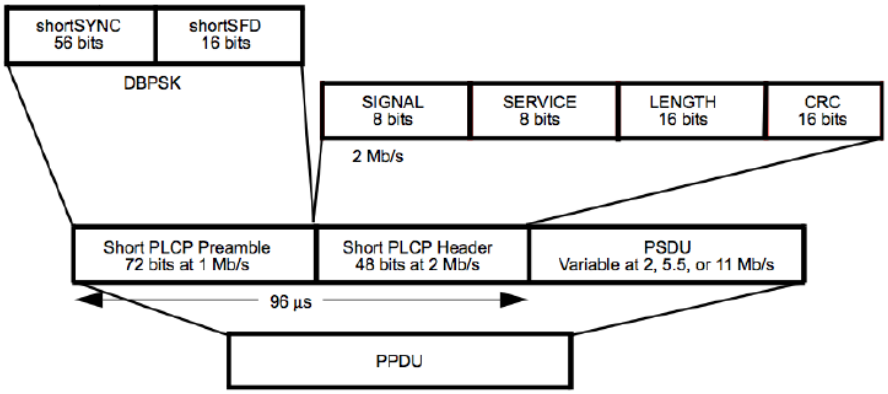
After the DIFS there is the Backoff Procedure. This procedure is used to try to avoid collisions. In our analysis it is simply a time, depending from the Time Slot duration, during which the station wait before transmitting data. The procedure starts from the selection of a random number between 0 and the value of the Contention Window (CW) -1. The Contention Window start from a value of 32 (CW_{MIN}) and it is doubled at every collision (in reality, since it is impossible to detect a collision given the nature of the transmission, the value is doubled if the transmission is incomplete) unless it reaches the limit value of 1024 (CW_{MAX}). The Contention Window's size is decremented only if the transmission is completed succesfully, in which case the value returns 32. The generated number becomes the number of Time Slot that the station waits before starting the transmission. If the random generator is well implemented and we consider an ideal channel (with no collision at all), the random number generated that we should observe will have to be an average of 15.5 ($(0 + 31)/2 = 15.5$). The Backoff Procedure should last for:

$$T_{Backoff} = 15.5 \times T_{TimeSlot} = 15.5 \times 20\mu s = 310\mu s$$

Subsequently there is the Physical Header (PHY): it is divided into two parts, $\mathrm{PLCP}_{Preamble}$ and PLCP_{Header} wich are different in lenght (bits) and transmission rate, based on the adopted transmission mode. The two possible transmission mode, in 802.11b are: DSSS (Direct Sequence Spread Spectrum, used in 1 Mb/s transmissions) and HR/DSSS (High Rate Direct Sequence Spread Spectrum, used in 2, 5.5 and 11 Mb/s). In the DSSS transmission mode the $\mathrm{PLCP}_{Preamble}$ is composed of 144bits and it is sent a 1Mb/s while the PLCP_{Header} is 48bits long always sent at 1Mb/s. In HR/DSSS we have the $\mathrm{PLCP}_{Preamble}$ which is formed by 72bits sent at 1Mb/s while the 48bits of the PLCP_{Header} are sent at 2Mb/s.

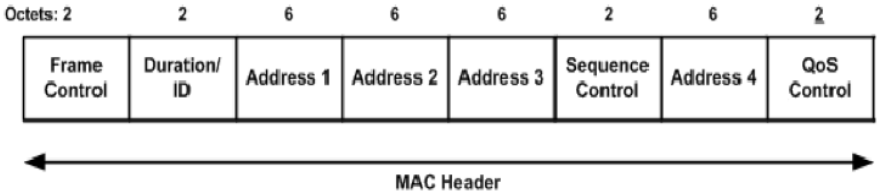
$$T_{PHY_{DSSS}} = \frac{144bits}{1Mb/s} + \frac{48bits}{1Mb/s} = 192\mu s$$





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

After this is to be considered the MAC level Envelope, consisting of the MAC Header and the FCS (Fragment Control Sequence) set after the MSDU (Mac level Service Data Unit). The MAC Header is composed of 24Bytes divided into various fields. The FCS is composed of 4Bytes. Every part of the MAC Envelope is sent at the Data Speed set on the Wireless card of our Laptop: in our case, for 802.11b, we used 1Mb/s, 5.5Mb/s and 11Mb/s:



$$T_{MAC@1Mb/s} = \frac{(24 + 4) \times 8\ bits}{1Mb/s} = 224\mu s$$

$$T_{MAC@5.5Mb/s} = \frac{(24 + 4) \times 8\ bits}{5.5Mb/s} = 40.\overline{72}\mu s$$

$$T_{MAC@11Mb/s} = \frac{(24 + 4) \times 8\ bits}{11Mb/s} = 20.\overline{36}\mu s$$

The MAC MSDU is composed of: LLC Header, IP Header and the data, enveloped at UDP level, created by Netperf. Since Netperf create an UDP Stream of data, the calculus of the transmission time is a bit cumbersome and so left alone. The time spent for the sending of the LLC Header and IP Header is:

$$T_{LLC+IP@1Mb/s} = \frac{(8 + 20) \times 8\ bits}{1Mb/s} = 224\mu s$$

$$T_{LLC+IP@5.5Mb/s} = \frac{(8 + 20) \times 8\ bits}{5.5Mb/s} = 40.\overline{72}\mu s$$

$$T_{LLC+IP@11Mb/s} = \frac{(8 + 20) \times 8\ bits}{11Mb/s} = 20.\overline{36}\mu s$$

For the calculus of the time necessary to transmit the Payload (the Data Unit at IP level) created by Netperf, it is necessary to say that Netperf create an unique UDP packet with a payload of 65507Bytes, then the UDP Header is added (8Bytes). The resultant packet (65515Bytes) is then split in a number of IP packets: a total of 44 packets with a payload of 1480Bytes and a final packet with a payload 395Bytes.

$$Total\ Payload = 44 \times 1480Bytes + 395Bytes = 65515Bytes$$

Since we want to confront the theoretical result with the tests' result, which are relative to the thrughput at the application level, it is more accurate to calculate the theoretical level of thrughput on the entire stream of packets. In our case we have:

$$T_{IPPayload@1Mb/s} = \frac{(44 \times 1480Bytes + 395Bytes) \times 8}{1Mb/s} =$$

$$\frac{65515Bytes \times 8}{1Mb/s} = \frac{524120bits}{1Mb/s} = 524120\mu s$$

$$T_{IPPayload@5.5Mb/s} = \frac{(44 \times 1480Bytes + 395Bytes) \times 8}{5.5Mb/s} =$$

$$\frac{65515Bytes \times 8}{5.5Mb/s} = \frac{524120bits}{5.5Mb/s} = 95294.\overline{54}\mu s$$

$$T_{IPPayload@11Mb/s} = \frac{(44 \times 1480Bytes + 395Bytes) \times 8}{11Mb/s} =$$

$$\frac{65515Bytes \times 8}{11Mb/s} = \frac{524120bits}{11Mb/s} = 47647.\overline{27}\mu s$$

Now the Sender has finished its transmission and the Server send back an Ack. This is done after a SIFS time interval ($10\mu s$). The Ack is composed of:

$$ACK = PHY_{Header} + ShortMAC + FCS$$

As said in precedence the Physical Header is different in dimension and in transmission speed, in relation to the transmission mode used. Also the MAC level of the Ack is different from before: infact it is composed of only 10Bytes. There are no data but the FCS is present and it is composed, as for the DATA frame, of 4Bytes.

$$T_{ACK@1Mb/s} = T_{PHY_{DSSS}} + \frac{(10 + 4)Bytes \times 8}{1Mb/s} = 304\mu s$$

$$T_{ACK@5.5Mb/s} = T_{PHY_{HR/DSSS}} + \frac{(10 + 4)Bytes \times 8}{5.5Mb/s} = 116.\overline{36}\mu s$$

$$T_{ACK@11Mb/s} = T_{PHY_{HR/DSSS}} + \frac{(10 + 4)Bytes \times 8}{11Mb/s} = 106.\overline{18}\mu s$$

Now that we have all the transmission times of the various parts that compose the total transmission, we have only to sum them to have the total transmission time:

$$Transmision\ Time = 45 \times (T_{DIFS} + T_{Backoff} + T_{PHY_{Header}} +$$

$$+ T_{MACEnvelope} + T_{LLC+IP} + T_{SIFS} + T_{ACK}) + T_{IPPayload}$$

(This calculus is done considering the entire stream of 45 packets)

$$Transmission\ Time\ @\ 1Mb/s = 583250\mu s$$

$$Transmission\ Time\ @\ 5.5Mb/s = 125166.\overline{36}\mu s$$

$$Transmission\ Time\ @\ 11Mb/s = 75288.\overline{18}\mu s$$

Now for the calculus of the Thrugput at Application level it is necessary to know the total bits that compose our transmission: like said in precedence we have a total UDP Payload of 65507Bytes

$$65507 \times 8 = 524056\ bits$$

Then the Thrughput at the various speeds is:

$$Thrughput\ @\ 1Mb/s = \frac{524056\ bits}{583250\mu s} \simeq 0.89851\ Mb/s$$

$$Thrughput\ @\ 5.5Mb/s\ =\ \frac{524056\ bits}{125166.\overline{36}\mu s}\ \simeq\ 4.18688\ Mb/s$$

$$Thrughput\ @\ 11Mb/s\ =\ \frac{524056\ bits}{75288.\overline{18}\mu s}\ \simeq\ 6.96622\ Mb/s$$

Table 1 shows a brief recapitulation of what we have done in the previous section:

Trans. Speed	1	5.5	11
Difs	$50\mu s$		
Time Slot	$20\mu s$		
Backoff	$310\mu s$		
PLCP	$192\mu s$	$96\mu s$	
MAC	$224\mu s$	$40.\overline{72}\mu s$	$20.\overline{36}\mu s$
LLC + IP	$224\mu s$	$40.\overline{72}\mu s$	$20.\overline{36}\mu s$
ACK	$304\mu s$	$116.\overline{36}\mu s$	$106.\overline{18}\mu s$
IP Payload	$524120\mu s$	$95294.\overline{54}\mu s$	$47647.\overline{27}\mu s$
Trans. Time	$583250\mu s$	$125166.\overline{36}\mu s$	$75288.\overline{18}\mu s$
Thrughput	$0.89851Mb/s$	$4.18688Mb/s$	$6.96622Mb/s$

Table 1: Times and Throughput at the various speeds considered for the 802.11b standard

2.1.2 802.11g

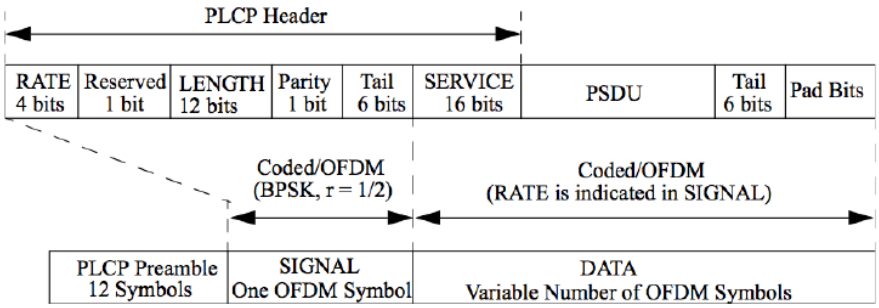
Like in the b case, the first thing to consider is the DIFS. In the g standard, infact, it is different from what analized for the b: the SIFS is slightly longer ($16\mu s$), while the Time Slots are shorter ($9\mu s$)

$$T_{DIFS}\ =\ T_{SIFS}\ +\ 2\ \times\ T_{TimeSlot}\ =\ 16\mu s\ +\ 2\ \times\ 9\mu s\ =\ 34\mu s$$

Like before, just after the DIFS, there is the time taken by the Backoff procedure. Again we are going to consider an "ideal" Backoff procedure. It is interesting to note that, in g mode, the CW_{MIN} is only 16 (so we have an 'average' of 7.5)

$$T_{Backoff}\ =\ 7.5\ \times\ T_{TimeSlot}\ =\ 15.5\ \times\ 9\mu s\ =\ 67.5\mu s$$

Then we have the Physical Envelope. In the g case the transmission mode is the ERP-OFDM (Extended Rate PHY - Ortogonal Frequency Division Multiplexing): the Physical Envelope is composed by a Preamble of 12 Symbols ($16\mu s$), a Header of 24bits, a Service of 16bits, a Tail of 6bits and, finally, a variable number of bits that are called Pad bits. The first two parts are sent at a speed of 6Mb/s, while



the other three are sent at the Data Rate. For simplicity we will consider the Pad bits later.

$$T_{PHY_{ERP-OFDM}@24Mb/s} = 16\mu s + \frac{24\ bits}{6Mb/s} + \frac{16\ +\ 6\ bits}{24Mb/s} = 20.91\overline{6}\mu s$$

$$T_{PHY_{ERP-OFDM}@54Mb/s} = 16\mu s + \frac{24\ bits}{6Mb/s} + \frac{16\ +\ 6\ bits}{54Mb/s} = 20.\overline{407}\mu s$$

The Pad bits depend by the dimension of the OFDM symbol, which in this case is different for different speeds, and by the dimension of the MAC Packets. In our case we have 2 different dimension of MAC pakets: the first is composed of 24Bytes of MAC Header, 8Bytes of LLC Header, 20Bytes of IP Header, 1480Bytes of Data and 4Bytes of FCS, for a total of 1536Bytes, while the second has only 395Bytes of Data, for a total of 451Bytes. In the calculus we have also to consider the Service and Tail bits (a total of 22). At the end we have:

$$LongPacket = 22 + (\ 1536\ \times\ 8\ ) = 12310\ bits$$

$$ShortPacket = 22 + (\ 451\ \times\ 8\ ) = 3630\ bits$$

The formula to calculate the Pad bits is the sequent:

$$N_{Symbols} = \left\lceil \frac{Packet\ bits}{Symbol\ Dimension} \right\rceil$$

$$N_{DATA} = N_{Symbols} \times Symbol\ Dimension$$

$$Pad\ bits = N_{DATA} - Packet\ bits$$

So for our packets we have Padding bits as showed in table 2: As done for the b mode, now we will calculate the necessary time to transmitt MAC Header, FCS, LLC Header and IP Header:

$$T_{MAC+FCS+LLC+IP@24Mb/s} = \frac{(24\ +\ 4\ +\ 8\ +\ 20)\ \times\ 8\ bits}{24Mb/s} = 18.\overline{6}\mu s$$

Speed	Symbol Dimension	Pad Long Packet	Pad Short Packet
24 Mb/s	96 bits	74 bits	18 bits
54 Mb/s	216 bits	2 bits	42 bits

Table 2: Padding bits at the various speeds

$$T_{MAC+FCS+LLC+IP@54Mb/s} = \frac{(24 + 4 + 8 + 20) \times 8\ bits}{54Mb/s} = 8.\overline{296}\mu s$$

Then we have the Stream of packets that compose the IP Payload, as described in precedence for the b mode:

$$T_{IPPayload@24Mb/s} = \frac{(44 \times 1480Bytes + 395Bytes) \times 8}{24Mb/s} =$$

$$\frac{65515Bytes \times 8}{24Mb/s} = \frac{524120bits}{24Mb/s} = 21838.\overline{3}\mu s$$

$$T_{IPPayload@54Mb/s} = \frac{(44 \times 1480Bytes + 395Bytes) \times 8}{54Mb/s} =$$

$$\frac{65515Bytes \times 8}{54Mb/s} = \frac{524120bits}{54Mb/s} = 9705.\overline{925}\mu s$$

And finally, after a DIFS of $16\mu s$, there is the Ack: it is sent at the major commune speed between the mandatory speeds of the g mode. In our case the Ack is always sent at 24Mb/s. Like for the Data packet, it is followed by a variable number of Pad bits: since the 24Mb/s we have a constant of 82 bits.

$$T_{ACK} = T_{PHY_{Preamble}} + T_{PHY_{Header}} + \frac{((10 + 4)Bytes \times 8) + 82}{24Mb/s} =$$

$$= 16\mu s + \frac{24\ bits}{6Mb/s} + \frac{194\ bits}{24Mb/s} = 28.08\overline{3}\mu s$$

So now we can calculate the total transmission times

$$Transmission\ Time\ @\ 24Mb/s = 30307.25\mu s$$

$$Transmission\ Time\ @\ 54Mb/s = 17551.25\mu s$$

and the relatives thrughput

$$Thrughput\ @\ 24Mb/s = \frac{524056\ bits}{30307.25\mu s} \simeq 17.29144\ Mb/s$$

$$Thrughput\ @\ 54Mb/s = \frac{524056\ bits}{17551.25\mu s} \simeq 29.85861\ Mb/s$$

Table 3 shows a brief recapitulation of what we have done in the previous section:

Trans. Speed	24	54
Difs	$34\mu s$	
Time Slot	$9\mu s$	
Backoff	$67.5\mu s$	
$PHY_{ERP-OFDM}$	$20.91\overline{6}\mu s$	$20.\overline{407}\mu s$
MAC + FCS + LLC + IP	$18.\overline{6}\mu s$	$8.\overline{296}\mu s$
ACK + PAD_{ACK}	$28.08\overline{3}\mu s$	
IP Payload	$21838.\overline{3}\mu s$	$9705.\overline{925}\mu s$
Trans. Time	$30307.25\mu s$	$17551.25\mu s$
Thrughput	$17.29144Mb/s$	$29.85861Mb/s$

Table 3: Times and Throughput at the various speeds considered for the 802.11g standard

2.2 Measurements analysis

In the following part of the section we will analyze the data gathered with the "hands on" tests. As already anticipated, these benchmarks have been done using the netperf tool, each being the result of a 30 seconds long test. The rates we used are: 1, 5.5, 11, 24 and 54 Megabit per second. We tried to test the 6Mb/s and the 12Mb/s rates too, but they did fail to correctly function in our testbed for unknown reasons: the behaviour observed is that the iwconfig tool seems to set these two rates as normal but, the moment the first packet is sent from the station, the NIC reverts back to the last "accepted" rate.

In table 4 we show the statistics gathered. It is easy to see than the actual measured values are lower than the expected rates as seen in the theoretical analysis.

Speed Rate	Observed Average	Observed MAX	Observed MIN	Standard Deviation	Conf Interval 90%	Conf Interval 95%
1Mb/s	0.707	0.733	0.666	0.019	0.007	0.008
5.5Mb/s	3.583	3.915	2.949	0.266	0.093	0.111
11Mb/s	5.711	5.981	5.298	0.163	0.057	0.068
24Mb/s	13.887	15.182	13.033	0.493	0.173	0.206
54Mb/s	21.302	23.256	17.675	1.538	0.539	0.643

Table 4: Aggregates of the experiments results. For each speed rate: average of all tests, maximum and minimum resulted value for each test, standard deviation, confidence interval at 90% and 95%

In table 5 we combine the measured data with some of the data resulting from

the theoretical computations. In particoular we can compare for each rate the measured average with the theoretical one and obtain the difference in percentage. This is intended as a measure of the quality of the network conditions during our measurements and not the efficiency of the different protocols, since protocol overhead is always taken into account.

Speed Rate	Theoretical Average	Observed Average	Theoretical vs Observed Difference
1Mb/s	0.898	0.707	21.286%
5.5Mb/s	4.187	3.583	14.417%
11Mb/s	6.966	5.711	18.018%
24Mb/s	17.291	13.887	19.687%
54Mb/s	29.858	21.302	28.657%

Table 5: For each speed rate: maximum theoretical throughput, computed theoretical average, measured average and difference from computed and observed in percentage (100% is the theoretical value)

We can also note that the standard deviation, which measures the variability of the measured values for each test, tends to increase with the speed, as emphasized by figure 3. This tendency is not preserved only in the case when moving from the 5.5Mb/s to 11Mb/s: we believe this is the result of exceptionally high network instability in general, but especially during the 5.5Mb/s measurement.

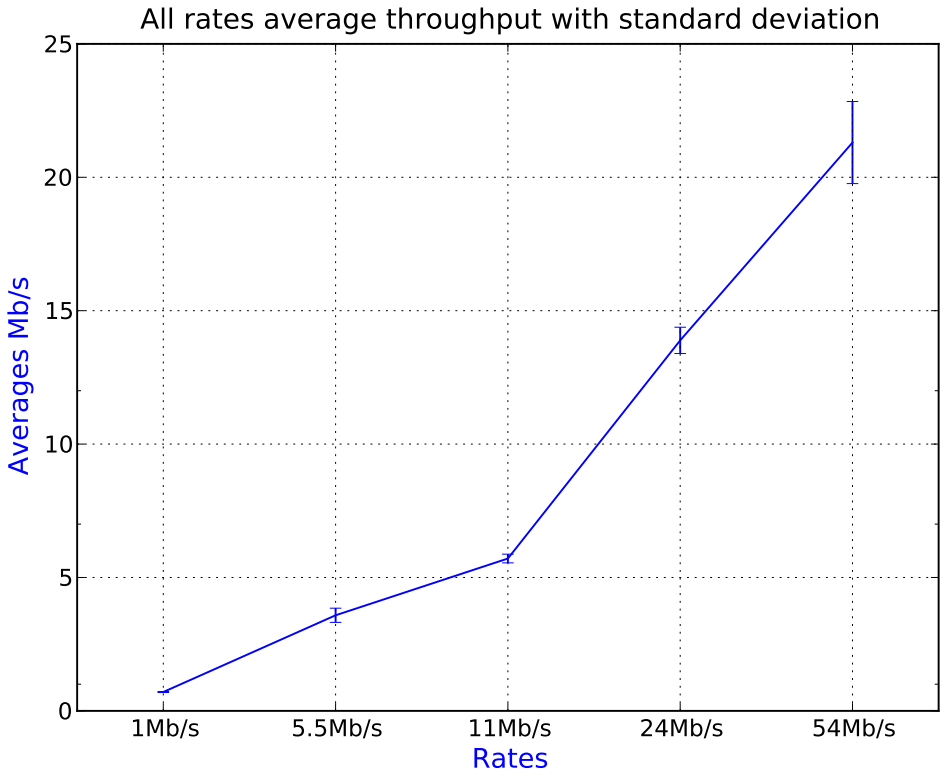


Figure 3: Average of all the tests average throughput for all rates. The error lines represent the standard deviation for each tested rate

2.2.1 802.11b: case 1Mb/s

In figure 4 we show the results for the 1Mb/s experiment. While the resulting standard deviation is low compared to other measurements, these test still suffer from a generically bad network condition. As this cannot be due to excessive distance of the stations from the access points we can guess that this is the result of interferences or channel's contention. In fact we should stress the fact that the faculty's environment is more often than not afflicted by very high wireless traffic. This is also confirmed by the link quality measured by the testing station's drivers. At the beginning of the measurement, *iwconfig* was showing the following output:

```
eth3      IEEE 802.11g  ESSID:"NCL"
          Mode:Managed  Frequency:2.412 GHz  Access Point: 00:0F:66:11:D2:C3
          Bit Rate=1 Mb/s   Tx-Power=20 dBm   Sensitivity=8/0
          Retry limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=72/100  Signal level=-56 dBm  Noise level=-79 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:29   Missed beacon:15
```

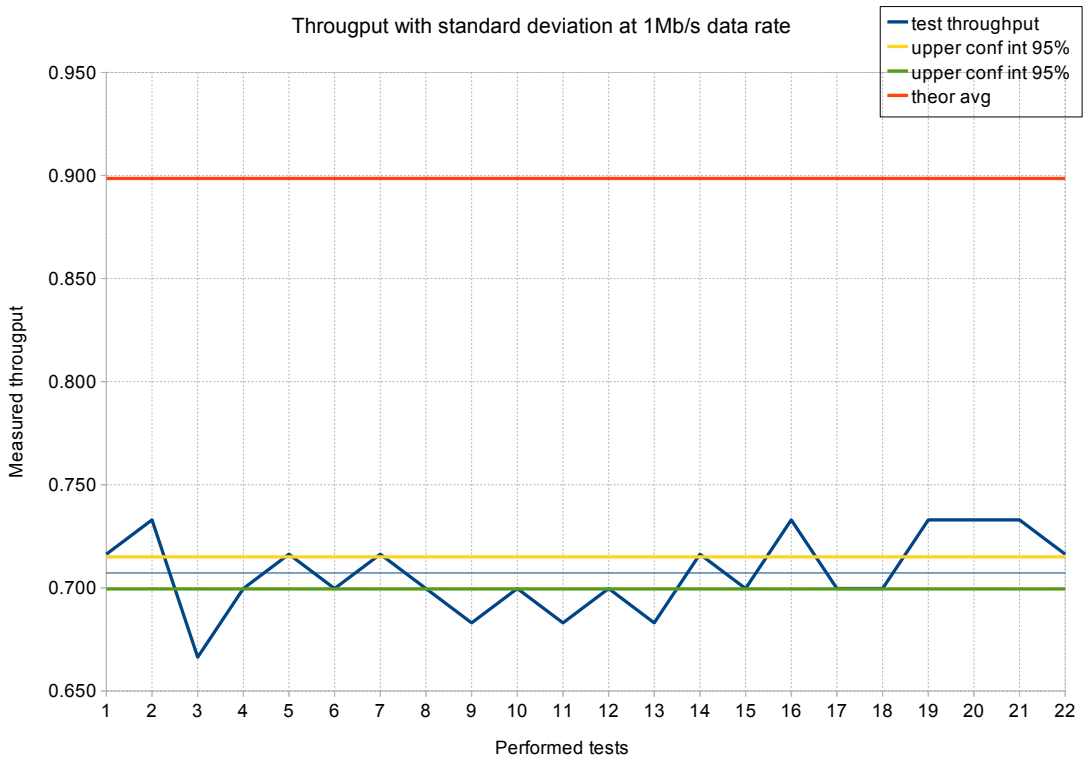


Figure 4: Line chart of the 22 measurements for the 1Mb/s data rate case with mean, upper and lower interval for 95% confidence

Instead, this is the output at the end of the measurement:

```
Link Quality=42/100  Signal level=-31 dBm  Noise level=-83 dBm
Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
Tx excessive retries:0  Invalid misc:29   Missed beacon:71
```

In general, it is not unusual to see link quality greatly varying during brief time intervals.

There are other elements that may suggest that communication on channel 1 was not very smooth during at least a part of the experiment time: by analyzing the packets captured with wireshark in monitor mode, we can extrapolate how many packets "on wire" were belonging to the netperf dialogue, and how many how those were retransmitted. By computing the ratio *packets netperf dialogue/total packets* and the ratio *packets netperf dialogue retransmitted/packets netperf dialogue* we can at least have an idea of how much traffic and how good the quality was during the capture time. How big is the part of experiment being captured depends on the rate of the transmission, since we are pretty much restricted in how many packets the monitoring station can handle. This is both in terms of primary

and secondary memory: for each experiment we captured a number of packets around 100.000, more than means to put the machine under excessive load. The ideal solution would have been to have a program that could gather these simple statistics on the fly or save the packets while discarding the actual content, like *tcpdump*: unfortunately this occurred to us too late.

captured packets on wire	70448
captured dialogue time	728
netperf packets AP $\longleftrightarrow$ station	65555
ratio dialogue/total packets	0.931
netperf retransmitted packets AP $\longleftrightarrow$ station	11991
ratio retransmissions/netperf dialogue	0.183

Table 6: Packet analysis for the 1Mb/s case

In table 6 we analyze the packet for approximatively 12 minutes: as we have 22 tests of 30 seconds each, this is the only case in which we basically cover all the runs. The first ratio tells us that about 93% of the capured packet were in fact part of our benchmark. By looking at the other packets in wireshark, we can tell that they were mostly beacons from various other access points. Having a 93% is in our opinion quite a good value, so we tend to dismiss the heavy channel contention hypothesis. Instead, looking at the second value tells us that we had about 18% of retransmitted packets. In our opinion this is not a good value, and our sensation is confirmed by looking at the other captures (although we have to keep in mind that sampled population are decreasing with the increase of the rates): in all the other cases the highest retransmission value is 6.8%, a much lower value. So at this point the only explanation we can give is that there were some interferences in adjacent channels, or even by non 802.11 devices

2.2.2 802.11b: case 5.5Mb/s

Figure 5's line chart shows the experiments' trend with 5.5Mb/s data rate. This is one of the tests with relatively high deviations from the mean, which is also quite clear from the image.

The last four results are peculiarly low compared to the others: unfortunately, we do not have the tools to further investigate the reasons. Judging from the persistence of the trend (given that each test lasts 30 seconds, the conditions causing it must have been there for two minutes) it is probably due to some other

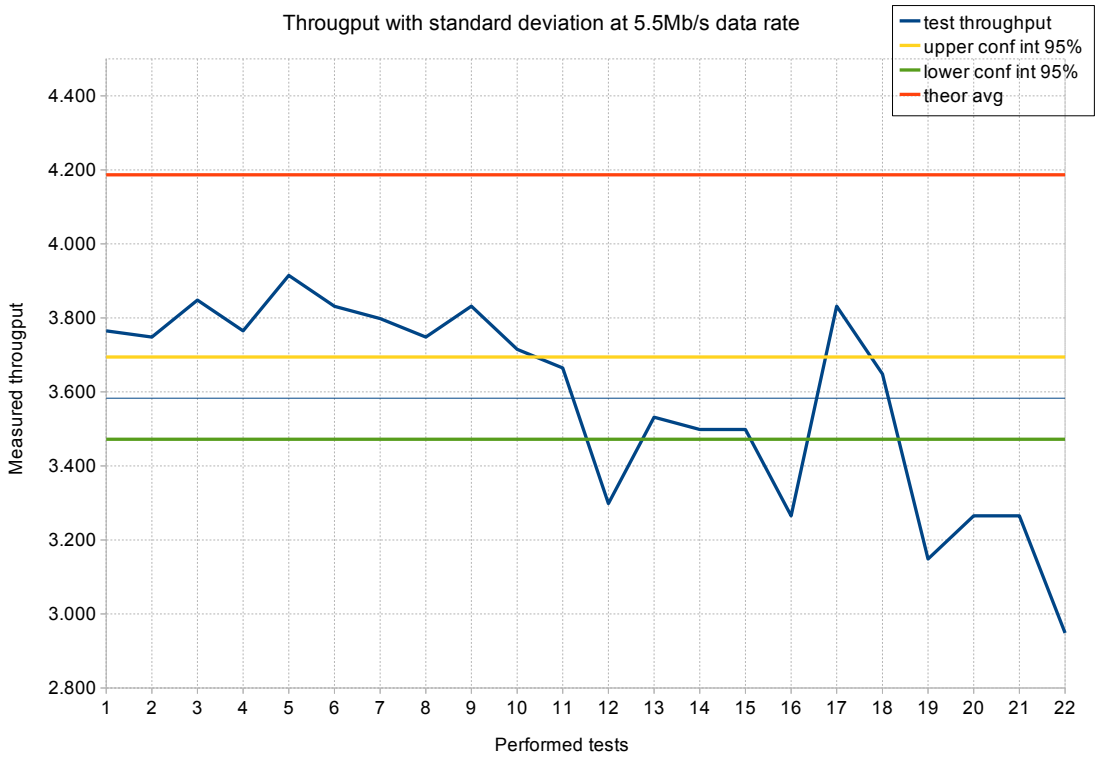


Figure 5: Line chart of the 22 measurements for the 5.5Mb/s rate case with mean and upper and lower interval for 95% confidence

station transmitting on the same channel of in one overlapping it. Especially the first hypothesis seems to be supported by the link quality parameter, outputted right after the end of the last measurement, which is quite high:

```
Link Quality=89/100  Signal level=-35 dBm  Noise level=-84 dBm
Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
Tx excessive retries:0  Invalid misc:29   Missed beacon:27
```

Like in the previous case, in table 8 we show an analysis on the captured packets. With this speed the analysis is less precise, since the portion of captured packets is way less (less than half of the toal experiment). By looking at these values, it would seem like we have good channel conditions, and this could explain the overall good performance, if our "theoretical vs observed" value is compared to other rates. Unfortunately the captured packets are of the first part of the series of tests and not of the last, the one which presents the noticeable "slope".

theoretical average throughput	4.187	
observed minimum throughput	2.949	
observed maximum throughput	3.915	
observed average throughput	3.583	
standard deviation	0.266	
confidence interval 90% and 95%	0.093	0.111
theoretical vs observed difference	14.417%	

Table 7: Statistics gathered for the 5.5Mb/s case. All values are in Mb/s.

captured packets on wire	100428
captured dialogue time	234
netperf packets AP $\longleftrightarrow$ station	98820
ratio dialogue/total packets	0.984
netperf retransmitted packets AP $\longleftrightarrow$ station	2304
ratio retransmissions/netperf dialogue	0.023

Table 8: Packet analysis for the 5.5Mb/s case

2.2.3 802.11b: case 11Mb/s

In figure 6 we show the results for the 11Mb/s experiment. We present again the measured data, summarized for this case in table 9. This is one of the tests with relatively high deviations from the mean, which is also quite clear from the image, by looking at the confidence interval. All in all there is not much left to say. We will not show the analysis done on packets, since is not even two minutes long.

theoretical average throughput	6.966	
observed minimum throughput	5.298	
observed maximum throughput	5.981	
observed average throughput	5.711	
standard deviation	0.163	
confidence interval 90% and 95%	0.057	0.068
theoretical vs observed difference	18.018%	

Table 9: Statistics gathered for the 11Mb/s case. All values are in Mb/s.

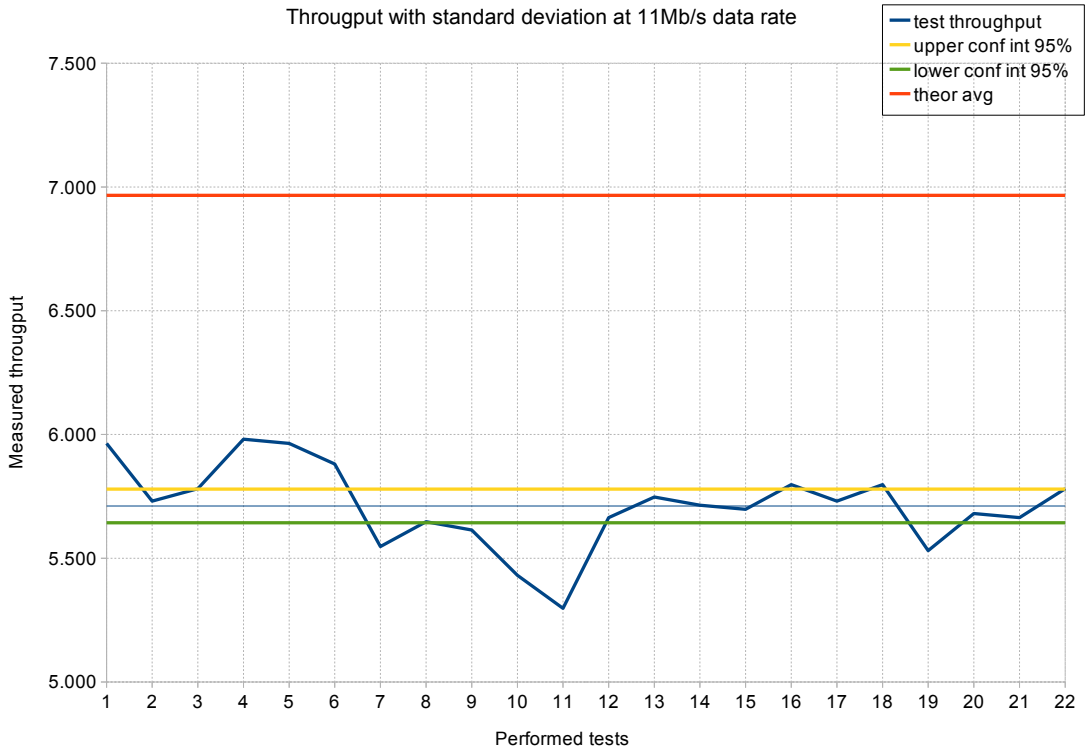


Figure 6: Line chart of the 22 measurements for the 11Mb/s rate case with mean and upper and lower interval for 95% confidence

2.2.4 802.11b: case 24Mb/s and 54Mb/s

These two tests will not be analyzed since we cannot provide any factual data suitable of supporting meaningful explanations. All we can say is that as always the test are plagued by high instability of the measurements. A noteworthy fact is that, if we assume our theoretical value reliable, the spread from observed value and theoretical value is increasing faster than with the other rates (see table 5). In the appendix we provide anyway the charts for these two testing configurations (figures 10 and 11).

3 Study on fragmentation levels

3.1 Theoretical analysis

As done in precedence, to study the performances obtained using Fragmentation, first we have to consider a theoretical analysis. We have chosen to analyze the performances with fragmentation of only two transmission speeds: 5.5Mb/s for the b transmission mode and 24Mb/s for the g. For the fragmentation threshold we have decided to consider 3 different values: 800, 600 and 410 (we have to remember a particular of the stream of packets which is composed of 44 "Long" Packets of 1480Bytes and a "Short" Packet of 395Bytes so we have that these 3 fragmentation values split the Long Packet of our UDP Stream in, respectively, 2, 3 and 4 fragments, while only the 410 threshold interact with the Short Packet, splitting it in two). So for the various fragmentation thresholds we have a number of fragments that is:

- for the Fragmentation Threshold of 800, the number of fragment is:
  $N_{Frag@800} = 2 \times Long\ Packet\ (44) + Short\ Packet = 89$
  the number of fragment per packet is: 2 $for\ each\ Long\ Packet$ and 1 $for\ each\ Short\ Packet$
- for the Fragmentation Threshold of 600, the number of fragment is:
  $N_{Frag@600} = 3 \times Long\ Packet\ (44) + Short\ Packet = 133$
  the number of fragment per packet is: 3 $for\ each\ Long\ Packet$ and 1 $for\ each\ Short\ Packet$
- for the Fragmentation Threshold of 410, the number of fragment is:
  $N_{Frag@410} = 4 \times Long\ Packet\ (44) + 2 \times Short\ Packet = 178$
  the number of fragment per packet is: 4 $for\ each\ Long\ Packet$ and 2 $for\ each\ Short\ Packet$

Figure 7: 802.11 fragmentation procedure

3.1.1 5.5Mb/s

For the calculus of the Throughput, we will follow the figure 7, utilizing the previously calculated times of transmission (see Sec2.1.1), with the following considerations: each of the varius parts will be counted in respect with their frequences. This means that, in general, we will count each part of the stream as follow:

- $DIFS \times N_{Packet}$
- $Backoff \times N_{Packet}$
- $PhysicalHeader\ (DataFragment) \times N_{Frag}$
- $MACHeader\ (DataFragment) \times N_{Frag}$
- $FCS\ (DataFragment) \times N_{Frag}$
- $LLC \times N_{Frag}$
- $IP \times N_{Frag}$
- $SIFS \times (Long\ Packet \times ((2 \times N_{FragLongPacket}) - 1) + Short\ Packet \times ((2 \times N_{FragShortPacket}) - 1))$
- $PhysicalHeader\ (ACK) \times N_{Frag}$
- $MACHeader\ (ACK) \times N_{Frag}$
- $FCS\ (ACK) \times N_{Frag}$
- the complete stream of packet with the UDP Header (65515 Bytes), as analized in precedence

where N_{Packet} is the number of packets (45) and N_{Frag} is, instead, the number of fragments (the number changes for each fragmentation threshold) that compose our stream ($N_{FragLongPacket}$ and $N_{FragShortPacket}$ are, respectivelly, the number of fragments into wich are divided the Long and Short packets).
Table 10 shows the various frequencies for each part of the stream in relation at the different fragmentation threshold:

	Frequencies				
Fragmentation Threshold	DIFS	Backof	Pyh + MAC + FCS + LLC + IP Data	SIFS	Phy + MAC + FCS ACK
800	45	45	89	133	89
600	45	45	133	221	133
410	45	45	178	311	178

Table 10: Frequencies of the various part that composes the UDP stream at each of the fragmentation threshold

Given the frequencies of the various parts of the stream is easy, with the data

calculated in Sec2.1.1, to obtain the transmission times, and then the throughputs relatives to the fragmentation threshold:

$$\begin{aligned} Transmission\ Time\ @\ 800 &= 138974.\overline{36}\mu s \\ Transmission\ Time\ @\ 600 &= 152782.\overline{36}\mu s \\ Transmission\ Time\ @\ 410 &= 166904.\overline{18}\mu s \\ Thrughput\ @\ 800 &= \frac{524056\ bits}{138974.\overline{36}\mu s} \simeq 3.77088\ Mb/s \\ Thrughput\ @\ 600 &= \frac{524056\ bits}{152782.\overline{36}\mu s} \simeq 3.43008\ Mb/s \\ Thrughput\ @\ 410 &= \frac{524056\ bits}{166904.\overline{18}\mu s} \simeq 3.13986\ Mb/s \end{aligned}$$

3.1.2 24Mb/s

As done for the 5.5Mb/s, now we will calculate the theoretical throughput for the g mode at speed 24Mb/s. Again each part of the stream will be calculated in relation to its frequency: the same frequencies calculated for the 5.5Mb/s case are valid also for the 24Mb/s case (10) with the adding of Services, Tail and Padding bits for each fragment. While Services and Tail bits are fixed (22 bits in total), we have to recalculate Pad bits for each fragment (the Pad bits of the ACK are always 82).
Table 11 shows the dimensions of the various fragments, their frequencies and the dimension of their Pad bits:

	Fragmentation 800			Fragmentation 600			Fragmentation 410		
Fragments (Bytes)	776	732	423	576	356	423	348	356	423
Frequencies	44	44	1	88	44	1	133	44	1
Pad bits	74	42	18	42	74	18	42	74	18

Table 11: Dimensions of the padding bits for each fragment encountered during the analysis

To better comprehend table 11 consider the following: at fragmentation threshold 800, the Long packet is divided into 2 fragments of respectively 776 and 732 Bytes, while the Short packet is left untouched (432 Bytes); at 600 the Long packet is divided into 3 fragments of respectively 576, 576 and 356 Bytes, while the Short packet is left, again, untouched; at 410 the Long packet is divided into 4 fragments

of respectively 384, 384, 384 and 356 Bytes, while the Short packet is divided into 2 fragments of respectively 384 and 39 Bytes.
Using frequencies found in Sec3.1.1, transmission times for the singles parts as in Sec2.1.2 and the Pad bits as in table 11, is possible to calculate the transmissions times and relatives throughput for each of the fragmentation threshold:

$$Transmission\ Time\ @\ 800\ =\ 34769.85\overline{3}\mu s$$

$$Transmission\ Time\ @\ 600\ =\ 39231.91\overline{6}\mu s$$

$$Transmission\ Time\ @\ 410\ =\ 43795.\overline{6}\mu s$$

$$Thrughput\ @\ 800\ =\ \frac{524056\ bits}{34769.85\overline{3}\mu s}\ \simeq\ 15.07225\ Mb/s$$

$$Thrughput\ @\ 600\ =\ \frac{524056\ bits}{39231.91\overline{6}\mu s}\ \simeq\ 13.35790\ Mb/s$$

$$Thrughput\ @\ 410\ =\ \frac{524056\ bits}{43795.\overline{6}\mu s}\ \simeq\ 11.96593\ Mb/s$$

3.2 Measurements analysis

Fragmentation should help in situations where the transmission is disrupted by short "impulses" of rumor, allowing in this case for the retransmission of only part of the original packet. Conversely, it should slightly degrade performances in basically all other cases. In our situation a high amount of interference is dominating the test environment, such that the fragmentation's result show a noticeable degrade in performances, increasing with the number of fragments.

3.2.1 802.11**b** - 5.5**Mb/s**

Table 12 shows some data obtained with the various tests done with different fragmentation thresholds, while figure 8 show the comparison between theoretical and measured averages at the various fragmentation threshold:

	Frag 800	Frag 600	Frag 410
Average (Theor)	3.77088	3.43008	3.13986
Average (Measu)	2.84796	2.08088	2.01980
Efficiency (Measu)	75.525%	60.666%	64.328%
Standard Deviation	0.18669	0.33669	0.25487
Min (Measu)	2.44891	1.64922	1.26609
Max (Measu)	3.13195	2.68211	2.33227

Table 12: Parameters taken from theoretical and measured data, relative to the fragmentation threshold at 5.5Mb/s datarate

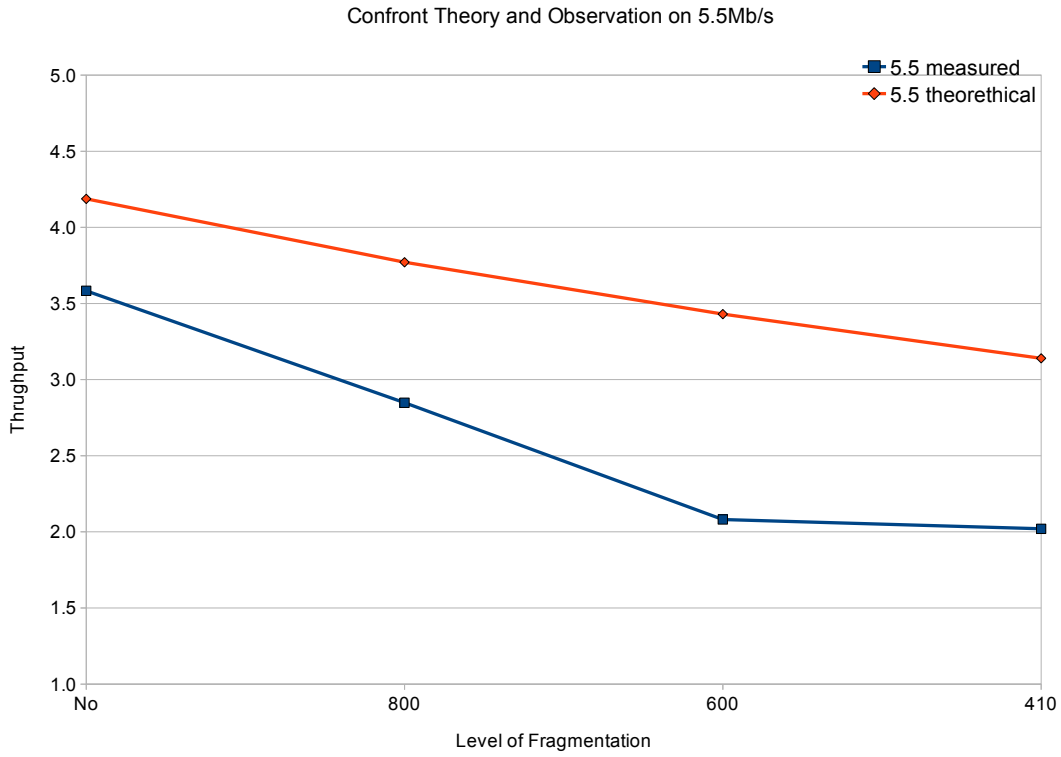


Figure 8: comparison of theoretical and measured throughput relative to the fragmentation threshold for 5.5Mb/s datarate

3.2.2 802.11**g - 24Mb/s**

Analizing the data for the 24Mb/s case, we can observe that some tests are lower than the others due to significant performance losses. In the attempt to have a fair idea about the performance impact of fragmentation we removed these discordant entries. Table 13 shows tests data both raw (including all the actual measured entries) and refined (after removing the lower values), Table 14 shows statistics over the data and figure 9 shows the difference between theoretical and measured averages, also in both the case of raw and refined data:

Frag 800		Frag 600		Frag 410	
Raw	Refined	Raw	Refined	Raw	Refined
11.811	11.811	10.045	10.045	2.016	0
11.157	11.157	9.775	9.775	9.826	9.826
12.590	12.590	9.843	9.843	3.224	0
9.365	0	10.287	10.287	4.060	0
9.519	0	10.542	10.542	5.357	0
8.598	0	9.877	9.877	9.638	9.638
9.826	0	9.843	9.843	9.860	9.860
11.498	11.498	10.406	10.406	10.116	10.116
11.344	11.344	1.482	0	9.672	9.672
11.208	11.208	10.798	10.798	9.672	9.672
12.010	12.010	10.611	10.611	9.724	9.724
11.122	11.122	10.747	10.747	10.031	10.031
11.890	11.890	10.730	10.730	9.553	9.553
11.532	11.532	10.474	10.474	9.536	9.536
11.788	11.788	10.338	10.338	10.082	10.082
11.430	11.430	5.049	0	9.604	9.604
12.248	12.248	7.950	0	9.792	9.792
10.952	10.952	10.645	10.645	9.280	9.280
11.890	11.890	6.772	0	8.768	8.768
11.242	11.242	7.642	0	9.347	9.347
11.805	11.805	10.696	10.696	8.888	8.888
9.877	0	10.406	10.406	8.683	8.683

Table 13: Measured througput in Mb/s for the tests with fragmentation at 24Mb/s

	Frag 800		Frag 600		Frag 410	
	Raw	Refined	Raw	Refined	Raw	Refined
Avg (T)	15.07225		13.35790		11.96593	
Avg (M)	11.12276	11.61856	9.33272	10.35668	8.48770	9.55960
Effic (M)	73.796%	77.086%	69.867%	77.532%	70.932%	79.890%
Std. Dev.	1.01213	0.42621	2.19092	0.34309	2.36192	0.41161
Min (M)	8.59768	10.95192	1.84240	9.77488	2.01578	8.68304
Max (M)	12.58960		10.79832		10.11600	

Table 14: Parameters taken from theoretical, measured and a refined subset of the measured data, relative to the fragmentation threshold at 24Mb/s datarate

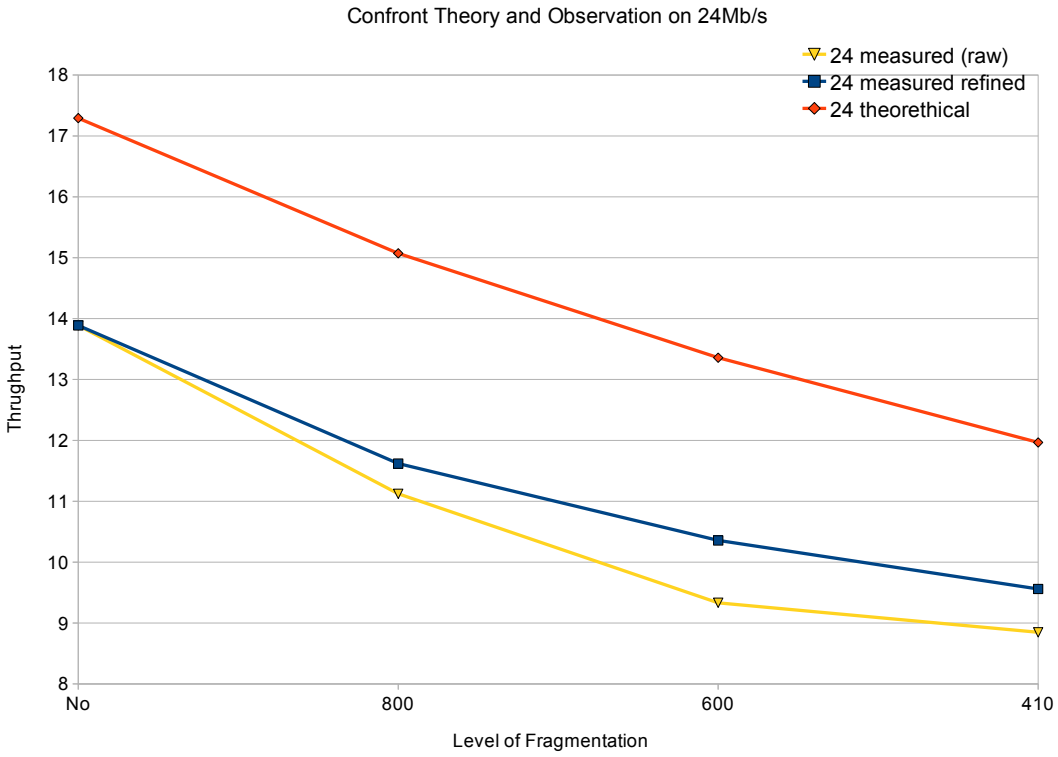


Figure 9: comparison of theoretical, measured and refined throughput in relation to the fragmentation threshold for speed 24Mb/s

4 Conclusions

During this laboratory experience we examined the protocol behaviour by "playing" with speed rates and fragmentation. Unfortunately, by looking at the experiments results, it is clear that doing a thorough analysis is very difficult given the condition of the network in the faculty and even the great complexity of the protocol. Nonetheless we can still try to draw some conclusions on some general properties on how the protocol works in a real world deployment. First of all it is immediate to notice that, every time we increase the data rate, we assist to a corresponding increase in the delta between the actual and nominal data throughput. This is immediate by taking the 54Mb/s datarate case, where the actual output is just about half that value. This can be explained with the fact that the protocol needs to maintain compatibility with all (compulsory) data rates, which translates to inter frame spaces and data rate speeds for some packets (or even part of them) sent at a common maximum for every standard. To add more complexity, we can point out that the maximum one can reach varies given the network conditions, like for example the NIC cards capabilities of the other stations connected to the same BSSID, in addition to the amount of traffic and other interferences.

Things can get even more problematic when taking into consideration the observed throughput. The measurements done are very inconsistent but, if taken on average, always significantly under the theoretic average we computed. Given this inconsistency it is difficult to judge whether our theoretical values are too high or the channel were simply too much disturbed during the tests. One thing that seems to stand out is that, in general, the gap tends to increase with the used data rate (if we do not take into account the 1Mb/s case), topping at 28.657% in the 54Mb/s case.

In the process of preparing the report, we did rely on the wireshark tool in order to try explain some of the results we obtained. Although it is an excellent tool, during this time we noticed that it needs too much resources to handle the huge quantity of data that we had to examine, forcing us to take small samples instead. For these reasons it occurred to us that we should have used a tool that could handle higher quantity of packets, also by discarding most of the payload, like tcpdump. Unfortunately we could not repeat the experiments at that point.

References

[1] P 802.11; draft wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Technical report, New York, 2007.

A Additional Material

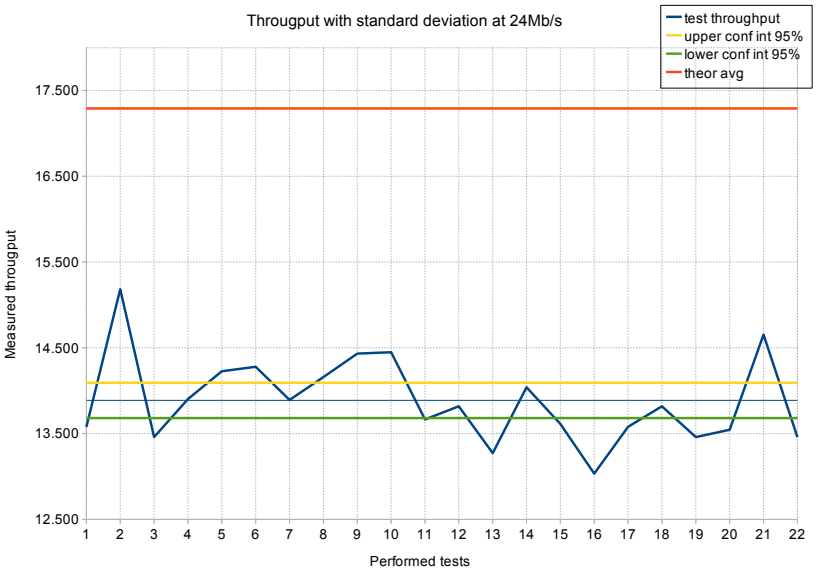


Figure 10: Line chart of the 22 measurements for the 24Mb/s

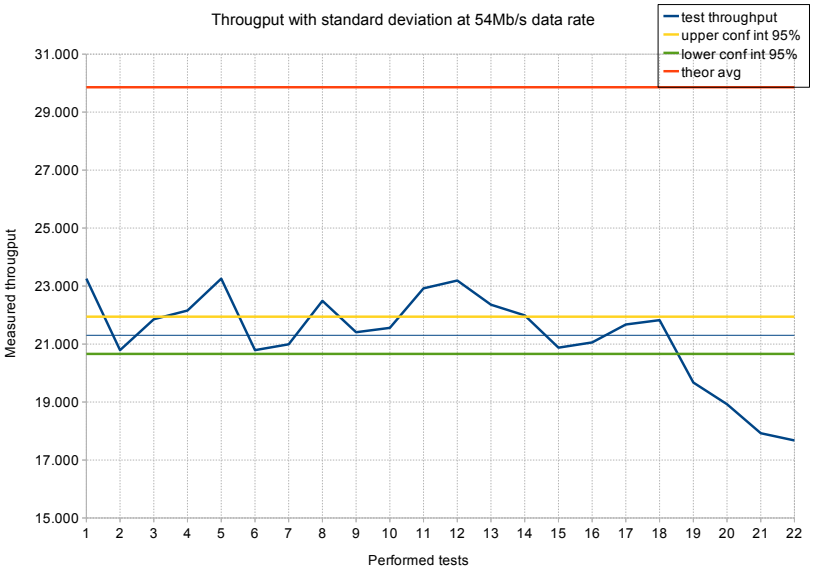


Figure 11: Line chart of the 22 measurements for the 54Mb/s