First Lab Report

Dandrea Silvio, Facchini Christian, Ferrari Rudi
{*silvio.dandrea, c.facchini, rudi.ferrari*}*@studenti.unitn.it*

April 16, 2007

Abstract

IEEE Standard 802.11 defines a medium access control and physical layer specifications for so-called wireless local area networks (WLANs) while IEEE Standard 802.11b is a supplement which introduced a higher-speed physical layer extension. As most of nowadays WLANs are based on that standard, a 802.11b compliant equipment has been chosen to conduct the experiments. Only a few particular aspects of the standard are investigated, namely, request to send / clear to send (RTS/CTS) mechanism, fragmentation threshold and transmission speed.

1 Introduction

We have chosen to use the abbreviations and acronyms present in the standard. For particularly important terms we report the whole transcription and the acronym or the abbreviation enclosed in brackets.

2 Test bench

In this session we will illustrate the characteristics of the environment where we have studied the behavior of the access point (AP), the software and hardware used for tests and their configurations. The devices used for this laboratory are an AP, a computer, playing the role of the server, and some computers, acting as clients, where the number of which was depending on the type of tests. The connection among the devices was as in Figure 1.

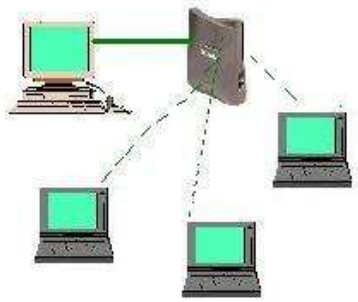

Figure 1: Connection between Server, Access Point and Clients.

2.1 Environment

The tests have been carried out in a environment that was not perfect for this type of measurements; in fact the tests have been made in places with interference and noise.

In the room where we have carried out the tests, there were interferences due to the presence of other APs, signals emitted from the several clients that were connected to the wireless network of the Faculty, and there were also the interferences caused by people moving back and forth.

It is worth emphasizing that all the tests have not been performed in the same conditions. As we have had to move in different rooms every testing day, the environment has changed as well. Each test has ben run several times (from ten to twenty times), in order to remove randomness as much as possible. However, our attempt has been to roughly maintain the same distances among the devices. The distances are summarized in Table 1.

Devices	Distance
S - AP	[1-2] m
C1 - C2	[2-4] m
AP - C1	[2-5] m
AP - C2	[2-5] m

Table 1: Table of distances.

In the table we denote the Server, Access Point and the several Client with S, AP and C1/C2 in order. For a better vision of the distances we can observe Figure 2.

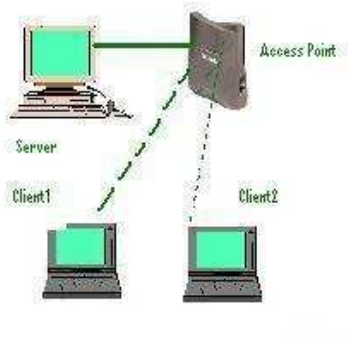

Figure 2: Distances between devices.

Moreover, it is important to observe that the interference due to the presence of APs should not have weighted in heavy way on the tests. As a matter of fact, the frequencies were in the same range but the channel used for our experiments and the one of the wireless network of Faculty were sufficiently distant.

Finally, it is necessary to specify that the movement of the people in the measurement environment, may have caused light variations on the measures, which, by the way, are far from being considered imposing.

2.2 Hardware

In this section we will describe the hardware used in the experiments. We will illustrate the devices used as server(s), as client(s) and the AP.

PC Server

The personal computer used as server has these characteristics:

PROC	AMD Sempron 3300+ 1.60GHz
RAM	1024 MB
O.S.	Windows Xp Home 32 bit
NIC	Realtek rtl8169/8110 Gigabit Ethernet
IPERF	Version 2.0.2 with Java Interface

Table 2: Characteristics of PC Server 1.

This devices has been utilized for testing the transmission speed. In order to test the fragmentation and RTS thresholds, we have made use of another PC: PC Server 2. This choice has been made because for our tests we had need of a device that was not busy from other groups.

PROC	AMD Turion 64x2 Mobile
RAM	1024 MB
O.S.	Windows Xp Home 32 bit
NIC	Realtek rtl-8169/8110 Gigabit Ethernet
IPERF	Version 2.0.2 with java interface

Table 3: Characteristics of PC Server 2.

PC Client

PC Client 1, whose specifics are summarized in Table 4, has being used for testing the transmission speed.

PROC	Intel Pentium IV 2.8GHz
RAM	512 MB
O.S.	Debian - Kernel 2.4.27
Wifi card	Nortek Wireless LAN W-11 PCMCIA
IPERF	Version 2.0.2 (03 May 2005) pthreads

Table 4: Characteristics of PC Client 1.

When we have had to use two different clients, as in the RTS/CTS test, the other one is PC Client 2; its characteristics are written in Table 5.

PROC	Intel Pentium Mobile 1.6 GHz
RAM	512 MB
O.S.	Linux Ubuntu 6.06 LTS 32 bit
NIC	Intel ipw-2100
IPERF	Version 2.0.2 (03 May 2005) pthreads

Table 5: Characteristics of PC Client 2.

In Section 3.2 we will refer to ourselves as "group 1" (PC Server 1 and PC Client 1) and to our colleagues Davide Molteni and Marco Azzoni

as "group 2" (PC Server 2 and PC Client 2). For all the other tests only "group 1" was involved.

Access Point

The access point used for our tests is Cisco AP 1200 Series. It is a mid-high range AP that implements the IEEE Standard 802.11b.

2.3 Software

In this session we will present the characteristics of the softwares used to set up the AP and to observe the behavior of the connection between client and the AP itself. These softwares are the AP firmware for configurating the Cisco AP 1200 Series, Iperf for the performance measurement of the network with a graphical interface named Jperf, MatLab for processing the data files and computing some calculations, such as variance and mean. At last, we have used Wireshark software for catching the flow of packets in the network and to analyze the correct operation.

AP firmware

This is downloadable from the Cisco web site at the page `http://tools.cisco.com/support/downloads/go/MDFTree.x?butype=wireless`. Different parameters can be set, more than the ones a low range AP would let set: authentication, the maximum number of customers, role in a wireless network, transmission speed, power, RTS/CTS, fragmentation, etc, just to mention some of them. The version used is 12.3(8)JA.

Iperf and Jperf

Iperf is widely used for the performance analysis of a network. With such a tool, it is possible to carry out different measurements, depending on the transport level protocol that is being used. For example with TCP, the bandwidth and the throughput are reported, while with UDP, it allows to measure the jitter and the packet loss. It is possible to use this tool both as client and server. The version of Iperf used is 2.0.2.

Jperf is a graphical interface developed in Java, really useful to those who do not have familiarity with the command line interpreter. Additionally, it is also possible to use Jperf to make bandwidth plots.

3 Tests

3.1 Transmission speed

The 2.4 GHz channel is affected by multipath fading, which causes attenuations and amplifications; moreover it is timevarying. Four different transmission speeds are hence defined by the standard, i.e. 1, 2, 5.5, and 11 Mbps: they are dinamically chosen depending on the channel conditions.

The MAC header is sent at a fixed speed. IEEE Standard 802.11b defines two kinds of physical layer convergence protocol (PLCP) PPDU format: a long one (mandatory) and a short one (optional).

Figure 3 shows both the PLCP PPDU formats: we see that the preamble and the header are sent at 1 and 2 Mbps respectively. The total time needed for this transmission is 96 µs.

$$\begin{aligned} Total_{time} &= \left[\left(\frac{n.\ of\ bit}{vel.}\right) + \left(\frac{n.\ of\ bit}{vel.}\right)\right] \\ &= \left[\left(\frac{72}{1 \times 10^6}\right) + \left(\frac{48}{2 \times 10^6}\right)\right] \\ &= 96\mu s; \end{aligned}$$

When using the long format, both preamble and header are transmitted at 1 Mbps: transmitting them will take 192 µs:

$$\begin{aligned} Total_{time} &= \left[\left(\frac{n.\ of\ bit}{vel.}\right) + \left(\frac{n.\ of\ bit}{vel.}\right)\right] \\ &= \left[\left(\frac{144}{1 \times 10^6}\right) + \left(\frac{48}{1 \times 10^6}\right)\right] \\ &= 192\mu s; \end{aligned}$$

For every packet that is sent, these two frame (ShortPLCP and PLCP) have always the same speed, even if the client is able to transmit at the maximum velocity i.e. 11 Mbps. All the other frames are transmitted, depending on the speed at which every device succeeds in using.

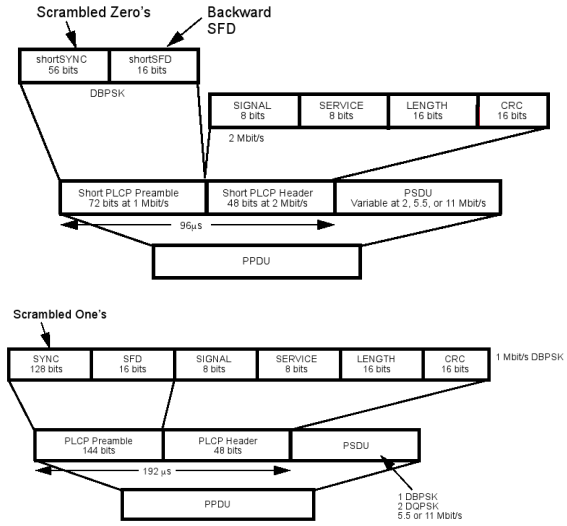

Figure 3: Above, the short PLCP PPDU format, and below, the long one.

Our tests have been made on both TCP and UDP, thus we have changed the transmission speed. The tests have been run ten times, each one of them lasting twenty seconds. From the data collected by Iperf, we have calculated average speed and the average amount of data transmitted.

Tests with TCP. For every test we have calculated the values shown in Table 6. In the last two columns we can note the average data transmitted with and without the header. In the third column we can observe the average data transmit obtained from Jperf, while in the fourth column we have the same data obtained from follow equation:

$$DATA \quad = \quad \left[\left(\frac{Avg\ Speed}{8\ bit}\right) * 20\ s\right]$$

We multiply for 20 second because our test is based on 20 second of transmission from client to AP.

Speed (Mbps)	Avg Speed (Kbps)	W/o Header (Kbytes)	W/ Header (Kbytes)
1	626.6	1544	1566
2	1252.2	3075	3130
5.5	2942.6	7211	7356
11	4196.6	10267	10491

Table 6: Speed, Average Speed and Average Data Tx, TCP being used.

In Figure 4 it is possible to observe the results of our tests for the four speeds 1, 2, 5.5 and 11 Mbps.

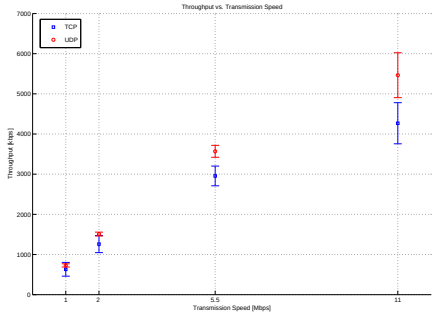

Figure 4: Throughput as a function of the transmission speed in TCP and UDP.

Tests with UDP. The same tests have been carried out with UDP protocol. From such tests we expected that without the overhead peculiar to TCP and the acknowkedgment mechanism, UDP is faster and more efficient than TCP for determinate measurements. In the following table are summarized the results of tests carried out with UDP.

Speed (Mbps)	Avg Speed (Kbps)	W/o Header (Kbytes)	W/ Header (Kbytes)
1	727.8	1766	1819
2	1508.8	3670	3772
5.5	3569.2	8580	8923
11	5464.7	13022	13661

Table 7: Speed, Average Speed and Average Data Tx, UDP being used.

We can observe these results in the next plot and so compare them with the results previously shown.

At the beginning of this tests, we thought that the speed of transmission with TCP was smaller than the speed of transmission with UDP. Now we can observe that this is correct, as shown in the tables and the plots.

Analysis

We have tried to analyze the UDP data rates, as such a communication is far simpler to model than a TCP one.

Our hypotheses:

- Every UDP PDU is 1470 bytes long (we actually have forced it in Iperf)
- No packet is lost
- Backoff duration has been analyzed in a stochastic way
- Long PLPC PPDU format has been used

A single UDP packet transmission consist of the following actions:

- A stations waits for a DIFS, then performs the backoff procedure, and then transmits the data
- the AP answers after a SIFS with an ACK

Some calculations follow:

- The SIFS is defined to be 10 µs
- The DIFS is equal to one SIFS plus two slots duration, so 10 µs $+2 \times 20$ µs $= 50$ µs
- The backoff procedure (BO) means picking a random value from 0 to the dimension of the contention window minus 1, where every number has the same probability to occurr (i.e. it is a uniformly distributed random variable); that will be the number of slots to be waited before a transmission is allowed to take place $E[contention\_window] = 15.5$ slots[1] $= 15.5 \times 20$ µs $= 310$ µs
- The UDP packet is 1500 bytes long while the ACK is 14 bytes; their duration depends on the transmission speed: 192 µs $+(1500$ *or* $14)/speed$

The durations for every different speed are written in Table 8 (if not explicited, values are to be intended expressed in µs).

TS (Mbps)	11	5.5	2	1
DIFS	50	50	50	50
BO	310	310	310	310
DATA	1283	2374	6192	12192
SIFS	10	10	10	10
ACK	202	212	248	304
TOT	1855	2956	6810	12866
TPS (s^{-1})	539	338	147	77
ADR (Mbps)	6.3	4	1.7	0.9

Table 8

[1]IEEE Standard 802.11 defines the contention window to be 32 slots if the first transmission is going to be made.

TS stands for "transmission speed", TPS means "transactions per second" and ADR means "achieved data rate".

What we have just computed validates what reported in Figure 4.

We have also computed some sort of utilization factor, that is the ratio between the transmission speed used and the data rate achieved. It's no wonder that the higher the transmission speed, the lower the utilization factor.

These factors are (from the minimum to the maximum transmission speed): 90%, 85%, 72% and 57%.

This is due mainly to the interframe spaces, which tend to weigh more and more as the transmission speed gets higher.

Even PLPC preamble and header damage the throughput: let us assume they are sent at the maximum available speed. If it were so, DATA and ACK would take 1108 and 27 µs, in order. Thus the whole transaction would need 1505 µs and that way there would be 664 transactions per second, carrying approximately 7.8 Mb per second. The utilization factor would reach 71%.

3.2 RTS threshold

Carrier sense multiple access with collision avoidance (CSMA/CA) works perfectly if the medium is really shared, i.e. when everybody hears everybody else. Unfortunately, this assumption cannot always be made and that is when RTS/CTS handshake comes into play.

RTS/CTS handshake is an exchange of messages in order to reserve the channel; basically, the station willing to start a communication with another station sends a RTS frame: that way it declares it wants to reserve the channel. The recipient will reply with a CTS frame, allowing the other station to initiate the communication.

As stated in the standard, this procedure aims at solving the hidden terminal problem (HTP), which is depicted in Figure 5(a). Stations A and C are not able to sense each other, so if they transmit to the same station collisions occurr.

Although not expressly debated in the standard, this mechanism solves the exposed terminal problem (ETP), too. We have such a problem whenever a node is prevented from talking to another station, because it thinks the destination is busy

with someone else. This situation is sketched in Figure 5(b).

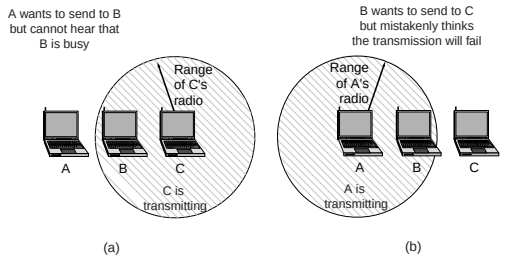

Figure 5: (a) Hidden terminal problem. (b) Exposed terminal problem.

Broadly speaking, RTS/CTS helps by reducing the waste of resources in case of collision, as RTS and CTS frames are simple MAC layer headers. Additionally the station sending the CTS frame to the node allowed to start the communication, makes every other station aware that the channel is reserved for a certain amount of time. These stations will wake up only after that amount of time, to wit, when the channel will be released.

RTS/CTS handshake is turned on on a threshold basis: the handshake is required if the MPDU that is going to be sent is larger than the beforehand chosen threshold. If the MPDU is smaller, it is sent immediately.

The RTS threshold is expressed in bytes. Values within the range $[0 - 2347]$ are permitted: the lower bound means that the RTS/CTS handshake will be used for every MPDU, while values "larger than the maximum MSDU length shall indicate that all MPDUs shall be delivered without RTS/CTS exchange".

Due to physical limitations we have not been able to reproduce neither HTP nor ETP scenarios; as a consequence we have not taken any measure about improvements by RTS/CTS in those situations. We have decided instead to inspect how throughput varied as the RTS threshold was changed in an Infrastructured BSS, where all the nodes (two stations and one AP) see each other.

Since RTS/CTS mechanism adds a non-negligible overhead, we expected to have a degradation in the total throughput (in inverse ratio to the threshold value). The fragmentation threshold has been turned off.

We have changed the threshold in increments of 2347/10, thus getting ten steps. We have run each test twenty times, making each one of them last 10 seconds.

The output of this experiment is summarized in Figure 6: "group 1" is represented by the continuous line, "group 2" by the dashed line, and the total throughput by the dotted one.

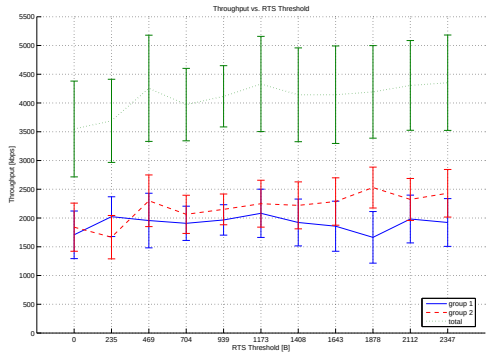

Figure 6: Throughput as a function of the RTS threshold.

Threshold (Byte)	Grp 1 thr (kbps)	Grp 2 thr (kbps)	Tot thr (kbps)
0	1707.88	1840.40	3548.28
235	2023.20	1666.75	3689.96
469	1955.85	2299.68	4255.53
704	1907.78	2064.09	3971.87
939	1966.47	2150.05	4116.57
1173	2081.82	2249.05	4330.88
1408	1922.37	2219.91	4142.28
1643	1857.58	2286.55	4144.14
1878	1663.50	2529.75	4193.26
2112	1982.77	2322.98	4305.74
2347	1922.37	2430.37	4352.74

Table 9: Data results from tests on RTS threshold.

As it can be seen, throughput of "group 1" keeps almost constant while throughput of "group 2" increases a little. This is confirmed by the *polyfit* Matlab tool which "finds the coefficients of a polynomial $P(X)$ of degree N that fits the data, $P(X(I)) \approx Y(I)$, in a least-squares sense".

We have set N to 1, so to fit the data with a line. No surprise to find a slope whose value is near to 0 (-0.004, for the sake of precision). As for the other group's throughput, we have a slope of about 0.26, indicating the throughput gets actually bigger. Consequently, the whole throughput increases as well.

Actually, what comes out is not exactly what

we expected to see: we thought to see an increase (if not for single throughputs, at least for the sum of them) till the threshold would have reached the value of 1643 bytes. Then the curve should have converged to an approximately constant value.

Accordingly to what we stated before, a station shall start the handshaking mechanism if the MPDU to be sent is higher than the threshold; the maximum segment size (MSS) for a TCP segment should be a little less than the maximum transfer unit (MTU). Even if IEEE Standard 802.11 defines that the MTU shall be 2304 bytes, most of the network interface cards (NICs) are (on purpose) not aware and simply do not let this value to be set. On top of that, often, an AP is directly connected to an Ethernet-based distribution system (DS): this is one of the reasons, if not of the only one, why the MTU on 802.11 devices is set to 1500 bytes, and finally this is the reason, for which we believed to find a somewhat constant value. But this turned out not to be verified.

We have analyzed the communication by means of Wireshark, a software protocol analyzer, and we have found that the RTS/CTS mechanism was triggered even when the threshold was set to the maximum value. Since we have no access to specific technical documentation for the NICs used, we can only suppose that those NICs are *compatible* with other 802.11 devices, yet not *compliant* to the standard.

This theory may also confirm the more aggressive behavior of "group 2" NIC, whose throughput gets higher at the expense of the other group's throughput.

Analysis

We have tried to dig out an upper bound for this tests.

First of all, these are some assumptions we have made in order to simplify calculations:

- Transmission speed is supposed to be always 11 Mbps
- RTS/CTS handshake is done only by the clients (i.e. RTS threshold is quite high)
- Neither the setup nor the teardown of the TCP connection are considered
- Every TCP PDU is 1460 bytes long
- No packet is lost
- Backoff duration is analyzed in a stochastic way
- Long PLCP PPDU format is used
- Stations and AP never collide (well, it's way too optimistic but it simplifies a lot all the calculations)

According to these assumptions, we have tried to figure out if the results we have found (the threshold being between 1408 and 1643) actually make some sense.

The transmission of a single TCP segment goes like this:

- a station, A, waits for a DIFS, just to be sure the channel is effectively idle and then, after the backoff procedure, sends a RTS
- the recipient, say B, waits for a SIFS and sends a CTS back to A
- A waits for a SIFS, too, and sends the data
- after a SIFS, an ACK is sent back by B

A TCP ACK has to follow, so here is the procedure:

- The DIFS has to expire, then the backoff procedure has to be done and eventually B sends the data
- A waits for the SIFS to end and transmits the ACK

Here some calculations (please refer to Section 3.1 for entries which have been already computed):

- The SIFS is defined to be 10 µs
- The DIFS lasts 50 µs
- The backoff procedure (BO) lasts, on average, 310 µs
- A RTS frame is 20 bytes long and is transmitted at 1 Mbps. Since we have to take into account the long PLCP, the whole transmission will last: $20 \times 8/1E6 + 192$ µs $= 352$ µs

- CTS frames are sent, like RTS ones, at 1 Mbps, but it is 14 bytes long; thus the duration will be: $14 \times 8/1E6 + 192\ \mu s = 304\ \mu s$
- The ACK frame is as long as the CTS one and it is transmitted at the maximum available speed; hence $14 \times 8/11E6 + 192\ \mu s \approx 202\ \mu s$
- The frame that contains the TCP data is 1500 bytes long, so it lasts: $1500 \times 8/11E6 + 192\ \mu s \approx 1283\ \mu s$
- The frame that contains the TCP ACK is supposed to be 74 bytes (20 from the TCP header + 20 from the IP header + 34 from the MAC header); therefore we have $74 \times 8/11E6 + 192\ \mu s \approx 246\ \mu s$

The whole amount of time needed by the transmission of the is summarized in Table 10

	TCP data	TCP ack
DIFS	50	50
BO	310	310
RTS	352	-
SIFS	10	-
CTS	304	-
SIFS	10	-
DATA	1283	246
SIFS	10	10
ACK	202	202
TOT	2531	818

Table 10

So a complete TCP transmission of 1460 bytes will last 2531+818 $\mu s = 3349\ \mu s$. Thus if there was only a station transmitting to an AP, there would be approximately 300 transactions, corresponding to a rate of 3.5 Mbps, more or less.

As we have supposed that stations never collide (that is, we consider each one of the two nodes transmitting half the rate we found) the throughput per group should be about 1.75 Mbps: the result we have found fits quite good the throughput of "group 1".

Maybe the other group's throughput can explained by considering the short PLPC PPDU format instead: in this case we find that a complete TCP transmission would take 2773 μs. Then, the approximate number of transactions per second would be 360. As a consequence, the data rate would become 4.2 Mbps, and the data rate per group would be 2.1 Mbps, which could make sense.

3.3 Fragmentation threshold

Very large frames may reduce transmission reliability, e.g. an error in a large frame wastes more resources than an error in a smaller frame: actually the time taken by retransmission would be greater in the first case. Normally small frames are not widely used unless the channel is really noisy, because too much overhead would occurr.

In order to overcome this problem, IEEE Standard 802.11 defined the fragmentation process. During the transmission of the data, the frame is subdivided in smaller frames by means of one fragmentation threshold: MSDUs larger than the established value are to be splitted, otherwise are directly sent.

Each fragment is sent as it were a standalone frame: hence a ACK must follow each fragment.

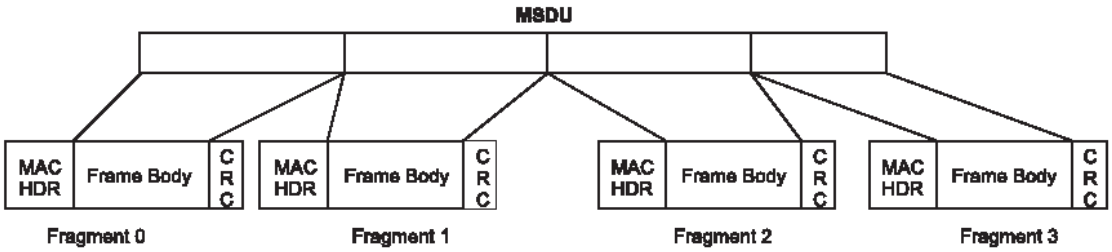

Figure 7: Fragmented frame.

Threshold (Byte)	Max thr. (Kbps)	Avg thr. (Kbps)	Std dev.
256	5308	4314.43	552.40
465	5243	4332.65	628.66
674	5308	4110.24	715.70
883	4981	4250.38	430.44
1092	5046	4168.08	555.26
1301	4915	4163.68	511.32
1510	5374	4196.87	516.39
1719	5112	4309.70	472.56
1928	4981	4000.96	572.99
2137	5177	3410.76	772.26
2346	4981	3829.50	630.36

Table 11: Data results from tests on fragmentation threshold.

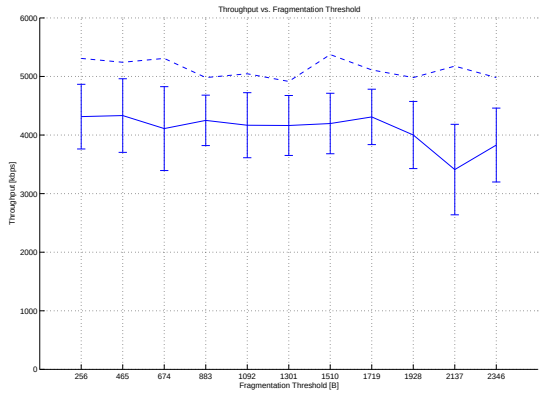

Figure 8: Throughput as a function of the fragmentation threshold.

Moreover if a station wins a contention, all the fragments forming the original frame may be sent within that dialogue (the same observation also applies to a RTS/CTS based contention). Each fragment consists of a MAC header, frame check sequence (FCS) and a fragment number indicating its ordered position within the original frame. The process of fragmentation is depicted in Figure 7.

The fragmentation threshold has a maximum value of 2346 bytes and a minimum value of 256. We have divided that range in eleven intervals, hence changing the threshold in increments of 209 bytes. We have performed ten times each test, letting it run for twenty seconds. The RTS threshold had a value of 2347 Byte (maximum value) i.e. it was not active. The protocol we have used is TCP.

The throughput is shown in Figure 8. As it can be seen, the throughput keeps almost constant around a certain value, exception made for the final points, where throughput gets a little worse.

We expected to have a bell-shaped chart: if fragmentation threshold were too low, too much overhead would be generated. While if the threshold is elevated, the decrease of throughput is caused by distorsion introduced by the channel.

By using again the *polyfit* tool, we have tried to fit the data with a second order polynomial: what we have found is that a maximum value appears around the threshold value of 674 bytes. This may indicate the conditions of the channel were very adverse.