uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Afsaneh Sattari

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Systems Science)

GRADE / DEGREE

Systems Science

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

An Experimental Study of Wireless LANs

TITRE DE LA THÈSE / TITLE OF THESIS

Professor D. Makrakis

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Professor N.D. Georganas

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Professor Daniel Amyot

Professor Azzedine Boukerche

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

An Experimental Study of wireless LANs

Afsaneh Sattari

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the MSc degree of Systems Science

Systems Science

University of Ottawa

Your file Votre référence
ISBN: 978-0-494-25833-0
Our file Notre référence
ISBN: 978-0-494-25833-0

Abstract

Wireless equipment based on IEEE 802.11 standard has been increasingly successful during the recent years. The popularity of wireless LAN hardware, and increasing use of different applications has brought challenges and competition among different vendors of IEEE 802.11 standards. Traditionally, such equipment has mainly been used indoors to allow users to wirelessly connect to the Internet or to a LAN.

In this thesis we evaluate wireless equipment of the 802.11b standard. We consider typical scenarios and also consider typical transport protocols (i.e. TCP and UDP) in order to observe the performance of the system in real settings.

Although the performance of a system mainly depends on the following metrics: throughput, packet loss, delay and jitter, in this thesis we take into consideration a large number of metrics of performance in order to obtain a better understanding of the way diverse factors impact performance in real wireless LANs. Indeed, we perform a large set of tests and measurements with different hardware. This let us obtain a more complete view of the performance of the system. Such parameters are important in real networks since its interrelation affect the quality of service perceived by various services carried by the network.

We observe that real throughput of WLANs is sensibly lower than the raw amount usually advertised by the manufacturers. From the obtained results we conclude that in addition to the lack of efficiency of the MAC layer, this situation is also due to the combination of other factors that are detailed in the thesis. We found that one of the most important factor affecting the performance is a corner in the signal path between the sender and the receiver, which significantly affects the performance. We also describe some different hardware and software tools that are useful to carry out the tasks required for a performance study in a wireless LAN.

Page iii-viii missing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Goal of the Thesis

It is undisputable that we live at the time of a big change in computer networking; everything is becoming wireless now. Wireless networking is a hot industry nowadays. Wireless technology has been successful because it enables people to communicate regardless of location. Wireless technology applied to Internet applications such as those based on the IEEE 802.11 have been very successful. Recently, the Wireless Ethernet Compatibility Alliance (WECA) has been pushing its Wi-Fi (wireless fidelity) technology certification program. In recent years, a number of experimental work has been conducted, primarily focusing on understanding the behavior of the wireless networks in different environments, understand and resolve interoperability issues between wireless networks and applications, as well as quantify their performance. In order to know what applications can be considered for 802.11-based equipments in these situations, a thorough understanding of transmission protocols, delays, packet loss, and the effect of the distance and topology is needed. Therefore, it is the goal of this thesis to contribute to this understanding.

1.2 Literature review

In recent years a number of studies on wireless channel performance have appeared in the literature. There are simulation (e.g. [CHI-1]), analytical (e.g. [CHII-10]) and experimental (e.g. [CHI-2]) studies. A performance evaluation of ad hoc wireless networks is presented in [CHI-2]. That paper examines the impact of varying packet size, beaconing interval, and route hop count on communication throughput, end-to-end delay, and packet loss. In [CHI-3] three techniques for composite performance and availability analysis is discussed in detail. They use queuing models to analyze wireless communication networks.

In [CHI-4] the performance of a real campus area network is studied. In order to carry out the tests, the authors used three performance-monitoring software tools: CWINS wireless benchmarking tools, Harris LAN Evolution Software and WaveLan Diagnostic software. Performance measurements were carried out adjusting several parameters: received power, walls and floors separating two radio interfaces and finally interfering traffic. In [CHI-5] the authors present a comprehensive study on TCP and UDP behavior over a WLAN taking into account radio hardware, device drivers, and network protocols. [CHI-6] presents performance measurements carried out on a real MAN in order to measure the actual throughput. They also compared 802.11b with the wired network. In [CHI-7], the authors discuss the problems that arise when the TCP/IP protocol suite is used to provide Internet connectivity over existing and emerging wireless links. They describe the characteristics and performance limitations of various existing and emerging wireless systems. The authors explain how these characteristics adversely interact with TCP mechanisms. They also discuss the causes of these problems and give examples of their

magnitude. They evaluated various TCP performance enhancements with focus on the transport and the link layers solutions. [CHI-8] presents studies on the capabilities of the IEEE 802.11. The metric for the performance in this work is throughput, and the factors that are adjusted in the experiments are SNR, file size (large file size and small file size), number of simultaneous users and the direction of the file transfer (from mobile station to mobile station and from mobile station to wired). Three wireless laptops, a wireless and wired desktop and an access point are used for the test. They observed that the height and positions of the antennas are very crucial in the communications quality. When there is an obstacle like a human body on the line of sight between the antennas, the SNR level falls dramatically. They also observed that mobile-to- mobile throughput is higher than mobile to wired throughput and proved that simultaneous number of users has the highest effect on the throughput. In addition, the SNR level has effect on the throughput. In [CHI-9] they tested a real heterogeneous mobile environment and presented a performance evaluation from the application point of view for a wide range of parameters. They presented a complete evaluation, from the application point of view, of heterogeneous systems in terms of a wide range of QoS parameters. Measured parameters were obtained for different packet sizes: this way, they determined the optimal packet size for each "service condition". They experienced better TCP performance than UDP performance when the packet size is under the 512 bytes. It is important to note that this work can be extended with a more complete analysis in order to understand what is the relationship between measured performance at application/transport layer and modeled/measured performance at physical/data link layer.

In this thesis we test how diverse factors impact the performance. Whereas previous works studied mostly the wireless channel performance, we take into account the effect of several factors like operating systems, accuracy of software and hardware tools, processor speeds and wireless devices.

1.3 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 introduces the IEEE 802.11 standard and some of its extensions. Chapter 3 presents the transport protocols evaluated in this thesis and brief information about problems related to wireless transmissions using the TCP protocol. Chapter 4 provides information about the equipment we used and the testing setup. It also reports the problems and solutions encountered from our experience in the use of 802.11 devices and different versions of the Linux operating system. We also address some problems derived from the signal strength. Chapter 5 presents the testing methodology with results and discussions. Finally, chapter 6 presents the conclusions.

1.4 Thesis Contributions

In this thesis we study how diverse factors affect network performance in real wireless LANs. These factors may be of two kinds. On one hand, network capacity is decreased by evident factors such as the overhead introduced by network protocols. In this case it is feasible to estimate the impact of such elements since they are related to predictable

issues such as header lengths and the timing relations between transmitters and receivers. On the other hand, in the case of wireless networks, other factors have to be taken into consideration such as building layout, construction materials, RF interference from nearby networks, mobility, antenna gain, receiver sensitivity, Doppler shift, etc. These factors may interact all at once making it difficult to analyze a wireless system. It is possible to find some works that address one or several of these issues, although the possible combinations are just too many and also new standards emerge quite rapidly. Therefore, there is lack of studies, which help to understand how several of these elements interact in order to impact performance in the networks we use today. This thesis is one step in that direction. In this thesis we study how data transmission in wireless networks is affected by the interaction of several factors present in real situations. From the obtained results we observed that in addition to the lack of efficiency of the MAC layer, there is a combination of several other factors, which are sometimes neglected and have a negative effect on performance. For instance, we found that one important factor affecting the performance is the presence of a corner in the signal path between the sender and the receiver. We also describe some different hardware and software tools that are useful to carry out the tasks required for a performance study in a wireless LAN. Summarizing, with the observations presented in this work we contribute with a description of how some factors interact as to affect network performance of a WLAN.

Chapter 2

Wireless LANs overview

In this chapter we describe the IEEE 802.11 standard [CHII-1] and some of its extensions. We focus on the type of services provided by the standard as well as relevant design considerations that guided its design.

2.1 Wireless transmission

Over the past several years, communications have become increasingly wireless. As a result, wireless technologies are encroaching on the traditional realm of "fixed" or "wired" networks. The most successful wireless networking technology this far has been 802.11. Like all networks, wireless networks transmit data over a transmission medium, which in this case is the air. Source data is transformed into electromagnetic waves that are propagated through the air thus carrying data. Such carrier signals usually are infrared light and radio waves. Radio waves can pass through most office obstructions and offer a wider coverage range than infrared light, which is easily blocked by walls, and some other office constructions. Most of the 802.11 products in the market use the radio wave physical layer [CHII-2].

2.1.1 Radio spectrum

Wireless devices operate in a certain frequency range. In order to prevent overlapping uses of the spectrum, in each country frequencies are allocated in bands by a regulatory institution. As an example of this, Table 2-1 lists some common uses of frequency bands in the U.S. ISM bands are generally license-free, provided that the wireless devices generate low-power transmissions. Wireless LAN standards usually operate in these bands. For instance, it can be mentioned that 802.11b devices operate in the S-band ISM.

Band	Frequency range
UHF ISM	902-928 MHz
S-Band	2-4 GHz
S-Band ISM	2.4-2.5 GHz
C-Band	4-8 GHz
C-Band sattelite downlink	3.7-4.2 GHz
C-Band radar(weather)	5.25-5.925 GHz
C-Band ISM	5.725-5.875 GHz
C-Band sattelite uplink	5.925-6.425 GHz
X-Band	8-12 GHz
X-Band Radar (police/weathe	8.5-10.55 GHz
Ku-Band	12-18 GHz
Ku-Band Radar(police)	13.4-14 GHz
	15.7-17.7 GHz

Table 2.1: Common US. Frequency bands [CHII-2]

2.1.2 Advantages and limitations of wireless communications

Wireless communications offer a great deal of advantages. For instance, they offer a less invasive method to provide network coverage where it becomes difficult to install a cabling system such as in historic sites. They also enable new services due to the fact that wireless stations let users become mobile. Such advantages have strongly motivated the fast adoption of novel networking technologies based on wireless communications.

However, not all components of an enterprise network can be wireless. Servers and the other data center equipment, which access data usually need to be connected to a wired medium such as fiber optics, which, under the current state of the technology, are a more reliable, and faster transmission media. Wireless-network hardware is currently slower than wired hardware. In order to properly control errors due to the unreliability of the wireless-medium, wireless-network standards have to apply strong methods of error correction / detection and also slow down data transmission when the signal quality decreases in the received frames. Security is also a critical concern in wireless networks. Network transmissions are available for everyone within the same range of transmission with the appropriate antenna, whereas in the wired network, the signals stay in the wires and can be protected by physical-access control.

2.2 Overview of IEEE802.11 wireless local area network

Wireless LANs (WLAN) have become increasingly popular due to factors like low cost and flexibility; therefore several wireless technologies have been used mainly for data transmission. IEEE802.11 standard has been a great success among all of them now. 802.11 was first standardized by IEEE in 1997, and then revised in 1999. The standard specifications are focused on a medium access control (MAC) layer, and a physical (PHY) layer for wireless connectivity in an ad-hoc or infrastructure network, see Figure 2.1. The MAC is a set of rules to determine how to access the medium and send data, but the details of transmission and reception are in the PHY.

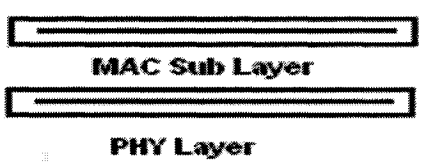


Figure 2.1: Part of 802.11 OSI Model

802.11 is a member of the 802 family, which is a series of specifications for local area networks (LAN) technologies. Figure 2.2 shows the IEEE 802 family and its relationship to the OSI model.

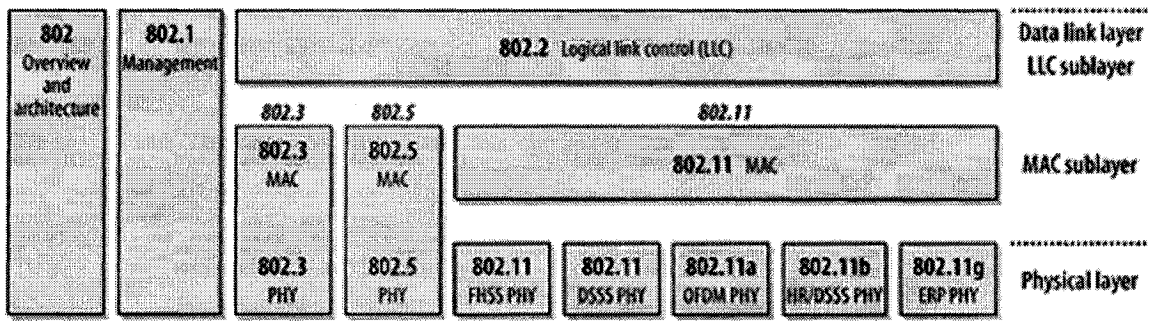


Figure 2.2: The IEEE 802.11 family and its relationship with OSI model

Some of the major differences of 802.11 standard from Ethernet is the use of wireless medium instead of wired for transmission, its inability to detect collision on its own (it requires that the receiving station acknowledges receipt of the packet), servicing connections of different speeds), experiencing the "hidden terminal" phenomenon. Individual specifications in the 802 series identify the number of its family. For example 802.3 is the specification for a carrier Sense Multiple Access network with Collision Detection (CSMA/CD) [CHII-2], 802.5 is the Token Ring specification [CHII-2]. Other specifications describe other parts of the 802 protocol stacks. 802.2 specifies a common Logical Link Control (LLC) sub-layer, which can be used by any lower-layer LAN technology. Management features for 802 are specified in 802.1. Among 802.1's, many provisions are bridging (802.1d) and virtual LANs, or WLANs (802.1q) [CHII-2]. 802.11 is just another link layer that can use the 802.2/LLC encapsulation. The base 802.11 specification includes the 802.11 MAC and two physical layers: a Frequency

Hoping Spread Spectrum (FHSS) physical layer and a High-Rate Direct-Sequence Spread Spectrum layer (HR/DSSS). The IEEE 802.11 MAC, compared to 802.2 MAC specifications, is more complex. A number of task groups have been made to add functionality and increase performance of 802.11. (See Appendix A)

A WLAN may either consist of stations (STAs) running in ad-hoc mode, or it may consist of STAs and access points (AP) in infrastructure mode; the duty of the AP is to maintain access to a wired LAN and distribution services like association within the WLAN.

In this thesis, the focus is on the 802.11, and most of the performed tests have been conducted using the 802.11b version of the standard, which allows for a data rate up to 11 Mbps. Higher bit rates, such as in 802.11g or 802.11a, are achieved by using more advanced digital modulation schemes. It is important to clarify that the mentioned data rate is "raw" bit rate. Since some of the 802.11 family WLANs operate in the unlicensed 2.4 GHz band, they might get interference from Bluetooth, microwave ovens and certain cordless phones.

2.2.1 The 802.11 Medium Access Control

The MAC sub-layer of 802.11 is responsible for providing fair access to the shared medium, and reliable data transfer. The main purpose of the MAC layer is assist the transmitting stations avoid performing simultaneous transmissions, which when occurring, (most probably) result in packet collision In the MAC functionality, an Inter Frame Space (IFS) time period is defined, which regulates access to the shared medium. A time period equivalent to an IFS period has to pass between the transmission of each frame. Depending on the type of the frame, IFS takes different values, a matter we will

discuss later in this chapter. Transmission of a frame should be finished before the transmission of another one starts. When the MAC layer receives a service data unit (SDU), (a frame received from an upper layer, also called MAC-SDU (MSDU)) it encapsulates the frame by adding a header and a checksum before passing it down to the physical (PHY) layer as a MAC protocol data unit (MPDU). There are three different frame types: management frames, control frames, and data frames. Figure 2.3 shows the general frame format of 802.11.

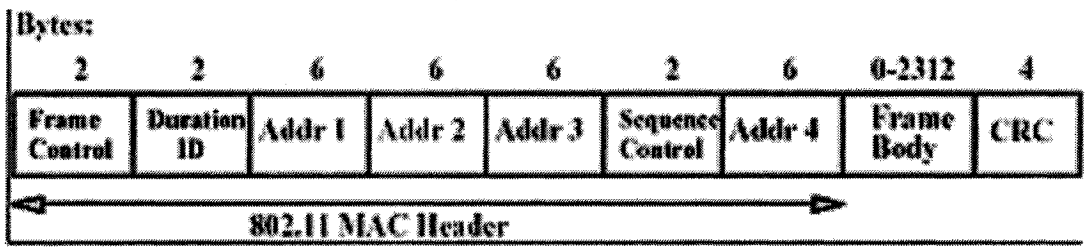


Figure 2.3: General 802.11Frame Format [CHII-2]

Access to the wireless medium is controlled by coordination functions. The medium access control is Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) [CHII-2], which is performed by the Distributed Coordination Function (CDF). This access scheme is mandatory, and if contention-free service is required, it will be provided by a Point Coordination Function (PCF), which is optional and is built on top of DCF. As we mentioned, DCF is based on the CSMA/CA. The design of the protocol is based on the wish to reduce, as much as possible, the occurrence of packet collisions, occurring in the medium. A step to this direction is that nodes are checking whether the medium is busy or not before entering the process of sending a frame. This mechanism is described in the following section. Collision detection cannot be used with this standard because it is not possible for a transmitting station to detect the occurrence of collision. This is

because during transmission, the "electronics" of its receiver are overwhelmed by the transmitted signal (due to the proximity of receiver and transmitter), which tends to be several orders of magnitude stronger than any signal reaching the receiver, produced by another, simultaneously transmitting node. Moreover, because of the hidden terminal problem, transmission and collisions could happen and not be detected by all stations in the network. A short explanation of the "hidden terminal" phenomenon, is as follows.

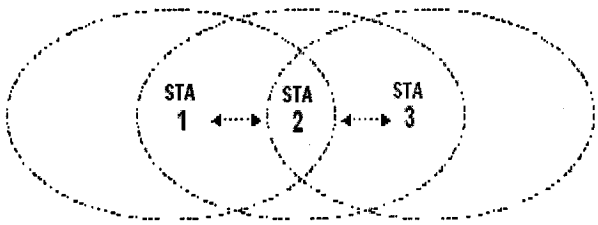


Figure 2.4: Hidden Station Problem

Consider the case shown in Figure 2.4. Among the three stations, station 2 (STA 2), is withing the coverage area of both, STA 1 and STA 3. However, STA 1 and STA 3 aren't within the others coverage area, they are thus unable to hear each other (in this case because of the distance between them). It is evident that, transmission by one of them (towards STA 2) will be unnoticed by the other, which could start its own transmission (towards STA 2 or another station) while the earlier transmission is still in process. Since STA 2 is in the joint part of the coverage areas, it will experience collision.

Distributed Coordination Function (DCF)

The distributed coordination function, consists of a basic access mode as well as an optional RTS/CTS access mode. In the basic access mode the station senses the channel

to see whether the channel is busy, i.e. another station is transmitting, before it starts its transmission. If the medium is free for at least a time interval equal to a DCF inter-frame space, (DIFS), then it may start the transmission immediately. The value of DIFS depends on the standard, for example according to [CHII-3] the value of DIFS for 802.11b is 50 μs, for 802.11a is 34 μs, and for 802.11g is 28 μs. This value helps the stations to determine or update their network allocation vector (NAV); therefore, they know how long they should defer their access to the channel. When NAV becomes zero, the station must wait for the channel to become idle, 802.11 refers to this wait as an access deferral. If access is deferred, the station waits for the medium to become idle for a DIFS time period, and then starts (or continues) the exponential back off algorithm (EBA). In order to decrease the probability of packet collisions occurring, the EBA contains uses a random back off (BO) time durations, which is calculated as equation (2.1)

BO=back off value * Slot Time (2.1)

The value of Slot Time (ST) depends on the standard. For example, it is 20 μs for the 802.11b, it is 9 μs, for the 802.11a 9 μs and for 802.11g[1] 9 μs. The Random Backoff_Counter (rbc) is selected randomly from a [0, CW] interval, using a uniform distribution, where CW is the contention window that is derived as (2.2), it is initially set to CW_MIN

$$CW = 2^n - 1 \quad (2.2)$$

[1] This applies when all stations are 802.11g compliant [CHII-3]. If there are stations that can only operate according to 802.11b standard, the value corresponding to 802.11b (i.e. 20 μs) applies.

Where n controls the contention window size. This value will be doubled when a transmission fails; still it reaches to its maximum value. In other words, if the transmission results in a collision, CW will be doubled and the rbc will be randomly selected using the new interval 0 to CW (where CW=2*CW_MIN). If there is a collision again, CW will be doubled again. If there is more collisions this procedure will continue until CW reaches CW_MAX. The maximum size the contention window is allowed to grow depends on the used physical layer. For example, the Direct Sequence physical layer limits the contention window to 1023 transmission slots. The contention window remains at its maximum size until it can be reset; it is reset when frames are successfully transmitted, or when the packet is discarded after a limited number of unsuccessful tries. [CHII-2].

The EBA also decreases the random back off time when the channel is idle. And finally, when the random back off time reaches zero, the station can immediately send its frame. The station can also use virtual carrier sensing methods, by sending a request-to-send (RTS) packet to the receiver and waiting for a clear-to-send (CTS) packet. The RTS packet broadcasts the duration of the intended transmission; that duration is also transmitted in the CTS response, which reduces the possibility of hidden terminal problems. After the frame transmission completes, the receiver sends back an acknowledgment packet (ACK). Once RTS (and/or CTS) is detected, other stations update their NAV.

If the previous frame was received without errors, the medium must be free for at least DIFS. If the previous transmission occurs with errors, the medium must be free for the

amount of Extended inter-frame space (EIFS). (See Figure 2.5 and 2.6 for both basic access and RTS/CTS method).

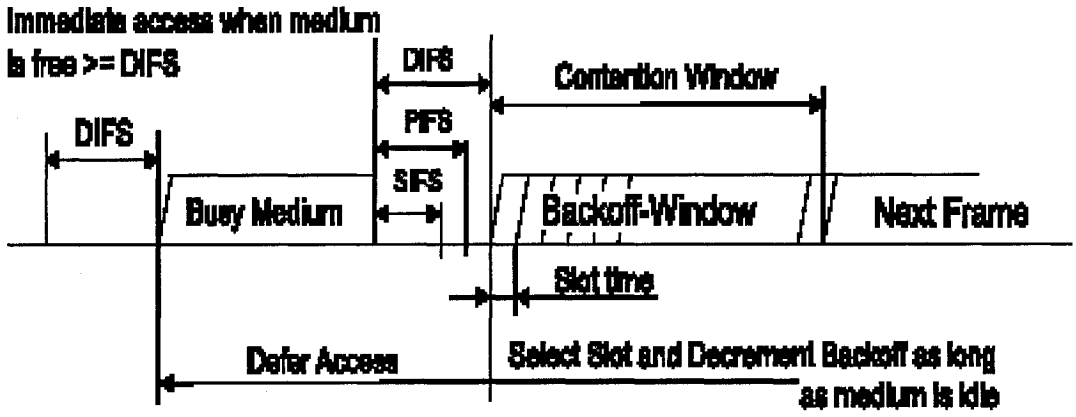


Figure 2.5: DCF basic access method [CHII-12]

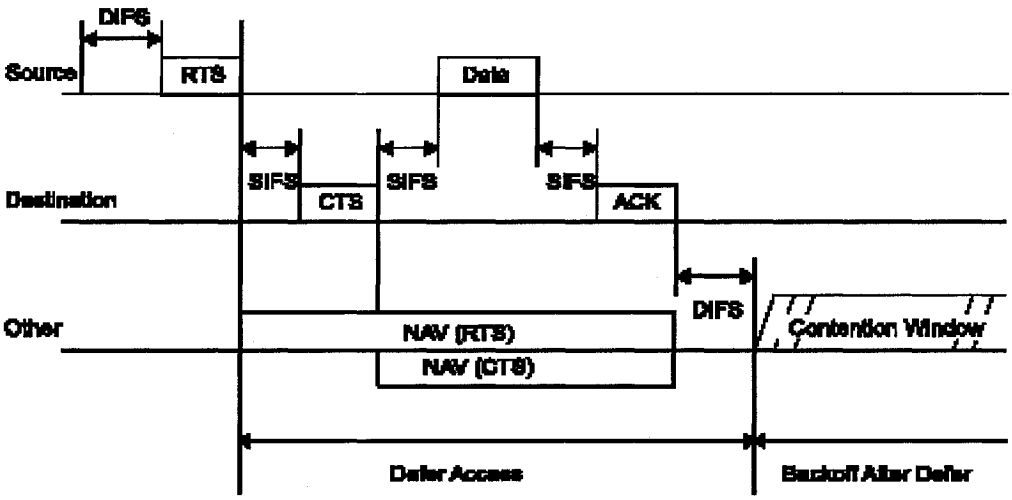


Figure 2.6: CTS/RTS [CHII-12]

2.2.2 Physical Layer Overview

The IEEE 802.11 physical layer combines several technologies of radio frequency transmissions. It splits the PHY into two generic components: the Physical Layer Convergence Procedure (PLCP), to map the MAC frames onto the medium, and a

Physical Medium Dependent (PMD) system to transmit those frames. The PLCP adds a number of fields to the frame as it is transmitted "in the air." See Figure 2.7

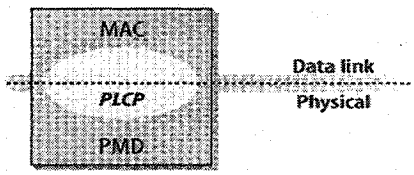


Figure 2.7: PHY Sub layers [CHII-2]

In the first revision of 802.11 in 1997, the three following physical layers were standardized:

- Frequency-hopping (FH) spread-spectrum radio PHY
- Direct-sequence (DS) spread-spectrum radio PHY
- Infrared light (IR) PHY, which is not widely used

Later, in 1999, two more physical layers on radio technology were included:

- 802.11a: Orthogonal Frequency Division Multiplexing (OFDM) PHY
- 802.11b: High-Rate Direct Sequence (HR/DS or HR/DSSS)

The current version of the 802.11 standard specifies three physical layers in the 2.4-GHz ISM band.

- FH PHY: A low-rate, frequency-hopping layer
- DS PHY: A low-rate, direct-sequence layer
- HR/DSS PHY: A high-rate, direct-sequence layer added by 802.11b. The 802.11b PHY is known as the A high-rate, direct-sequence, abbreviated HR/DS or HR/DSSS.

- 802.11a is based on orthogonal frequency division multiplexing (OFDM)[CHII2] It operates in the 5-GHz frequency band in US.
- The 802.11g standard is known as 802.11b- extended, for increasing data throughput. This standard uses the same 2.4-GHz ISM frequency band like 802.11b, but it also employs the OFDM modulation technique, established by 802.11g's committee in 2001. In a part of the following section we will discuss in more detail the performance of these three standards.

2.3 Theoretical and simulation based performance evaluation of 802.11

There are several theoretical and simulation-based performance evaluations of IEEE802.11 [CHII-7] [CHII-8] [CHII-9] [CHII-10] [CHII-11]. These works include MAC layer theoretical throughput with or without CTS/RTS, simulation results, and DCF for asynchronous data traffic. As stated earlier, the speed of 802.11 is defined in terms of available bit rate, not taking encapsulation of data, collisions in the wireless media or processing delay in the wireless equipment into account. In order to calculate the theoretical throughput, it is necessary to make some simplifying assumptions. For example, in [CHII-11], the only type of MAC frames considered are data and ACK frames, and none of them are subject to corruption by interference or packet collisions. Normally fragmentation is not an issue, and is the case if maximum sized Ethernet frames (1500 bytes long) are used. Other important parameters are timing parameters, preamble and data rates, which vary between different extensions of the standard. For the IEEE 802.11b standard, the transmission rate speed can be set as 1M, 2M, 5.5M and 11M.

2.4 Comparison of 802.11

Cisco Systems in [CHII-4] and [CHII-5] have done a precise study in 802.11b and compared it with 802.11a and 802.11g, in order to help the user identify the technology best suited to his/her specific requirements. Their results are summarized in Table 2.2

	802.11b	802.11a	802.11g
Likely uses	Provides overlays to wired Networks, enables mobility within office environment, provides networks where wires don't or can't exist, and enables outdoor bridging over very long distances (25 miles)	Provides overlays to wired Networks, especially for high-bandwidth applications (CAD, voice, video, and so on), and servers in Greenfield environments, where wired or wireless access in not currently deployed	Serves as an upgrade to 802.11b networks or as an alternative to the 802.11a frequency band. Its performance will likely be similar to 802.11a
Pros	Mature, reasonably priced technology (enterprice product prices and dropping) that provides very adequate throughput and very good range	Much improved throughput over short distance; four times the numberof non overlappingchannels; and less frequency band interference	Backward compatibility with, and range comparable to, 802.11b systems, but with improved throughput
Cons	Provides lower throughput of all wirelss technologies (11 Mbps) and has only three networking channels	More expensive and less mature, and incompatible with 802.11b technology. Has reduced range (mileage varies); FCC restrictions on antennas within each band	No products at this time (likely in 2003). Same channel limitations as 802.11b

Table 2.2: Comparison of IEEE 802.11 Standards. Based on [CHII]

Table 2.2 is based on the previous reference but dated information was not included so that the information shown in the table can be considered still valid. . In more recent documents, for example [CHII-6] and [CHII-3], the authors discuss several issues related

with the performance, interoperability and adoption of new wireless standards. The addressed subject is how to decide between 802.11g or a. As we know, both 802.11a and 802.11g variants specify a bandwidth of 54 MHz, while the 802.11b specifies 11MHz. According to the material presented in those papers, although 802.11g is significantly faster than 802.11b, once an 802.11b station associates to an 802.11g network, the throughput drops dramatically, because protection must be enabled. Thus, it its backward compatibility with 802.11b, which was recognized as one of its advantages, produces a serious loss in throughput. Modern standards also offer more radio channels for easier layout of high-density deployments.

Chapter 3

Transport Layer Protocols

This chapter is a background review of the most commonly used transport layer protocols used on the Internet, User Datagram Protocol (UDP), which provides a lightweight and unreliable transport service, and the Transmission Control Protocol (TCP), which provides a reliable and controlled transport service. The majority of Internet applications uses TCP, because of its reliability and flow control services, which ensures that data do not get lost or corrupted. However, many applications that do not require the overhead found in TCP, or that cannot use TCP because the application has to use broadcast or multicast, will use UDP instead. UDP is more appropriate for any application that has to issue frequent update messages, those messages have time delivery deadlines, and the application does not require every message to get delivered [CHIII-1]

The TCP variant used in the Internet today is called Reno, but its exact behavior varies depending on the operating system and/or TCP-stack used, since various suggested improvements to TCP may be implemented and used by default [CHII-11]. Since the chosen operating system of the work reported in this thesis is Linux, a brief background on the Linux implementation of TCP, as well as, the usage of TCP in a 802.11 WLAN are explained in this chapter.

3.1 User Datagram Protocol

The user datagram protocol (UDP) [CHIII-2] is the standard transmission layer protocol defined by the TCP/IP architecture for the provision of connectionless service. It is called connectionless since it provides a way for the application layer to send data without having to establish a connection with the receiving layer. With lack of any connection state in end systems, UDP does not provide any protection against packet losses or congestion, thus it is unreliable. The connectionless service makes UDP rather fast and because of that, it is the preferred protocol for many real-time applications.

UDP transmits datagrams, the encapsulated data, consisting of an 8-byte header followed by the payload. The header is shown in figure 3.1. The header contains information about the source port, destination port, length of the datagram in bytes, and a checksum. Without the port fields, the transport layer would not know what to do with the packet, so they are needed to deliver the segments correctly.

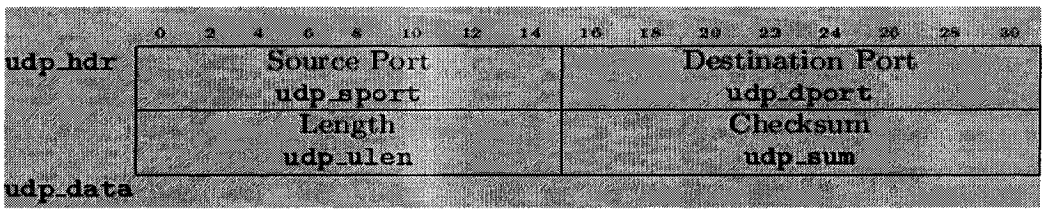


Figure 3.1: The UDP header

3.2 Transmission Control Protocol

The Transmission Control Protocol (TCP) [CHIII-3] is the standard transmission layer protocol of the TCP/IP protocol architecture, responsible for the provision of a point-to-point full duplex connection-oriented service.

It is important to note that a TCP connection is full duplex, with each end point maintaining independently its own sequence number.

In contrast to UDP, connections are established in TCP by using three-way handshake. This is done to inform each other about allocated buffers and initial values of sequence numbers for both, sender and receiver. The 32-bit sequence number field identifies the position of the first data byte of this segment in the byte stream during data transfer. The sequence number is increased with each new segment sent out and is used to keep track of the segment in the transmission. Note that TCP identifies the sequence number for each byte (rather than for each segment), i.e. it does not reflect the number of segments sent but instead it specifies the number of the first byte in every segment. The receiver in turn, ACKs the data received by specifying the next byte expected from the sender [CHII-11]. Thus, if a packet is lost, the sender will notice it and will retransmit it; therefore, TCP is providing a reliable transfer of data. As an example of these rules, the following example was captured in our experiments during the connection establishment process for a file transfer

34891 > 10845 [SYN] **Seq=455324449** Ack=0 Win=5840 Len=0

7 0.007923 192.168.1.5 192.168.1.7 TCP 10845 > 34891 [SYN, ACK] Seq=3497314015 **Ack=455324450** Win=5792 Len=0 8 0.008014 192.168.1.7 192.168.1.5 TCP

The above shows that the Acknowledgement number in [SYN, ACK] is equal to the Sequence number in [SYN] +1, which is 455324449 + 1. And during the actual data transfer:

DATA FTP Data: 1448 bytes

13 0.123401 192.168.1.7 192.168.1.5 TCP 34891 > 10845 [ACK] Seq=455324450

Ack=3497315464 Win=8688 Len=0

14 0.125768 192.168.1.5 192.168.1.7 FTP-DATA FTP Data: **1448 bytes**

15 0.125838 192.168.1.7 192.168.1.5 TCP 34891 > 10845 [ACK] Seq=455324450

Ack=3497316912 Win=11584 Len=0

(Ack) = (previous Ack) + 1448 (bytes)

3497316912 (bytes) = 3497315464(bytes) + 1448 (bytes)

To regulate the sending rate of segments, TCP implements two control mechanisms; the flow control and the congestion contrTCP Flow Control

Different hosts in a network might have different characteristics, including processing capabilities, memory, and network bandwidth, other resources. For this reason, not all hosts might be capable of sending and receiving data at the same or even similar rate. TCP must be able to deal with these dissimilarities. Furthermore, TCP must be able to accomplish this, without expecting any "co-operative" action taken by the applications in use. Therefore, this is accomplished through flow control[2], applied over the connection, implemented by dynamically advertising the window size. A change in rate over time may be required due to a variety of reasons, including the available buffer space on the destination system and the packet handling characteristics of the network. For this reason, TCP is controlling the limited amount of unacknowledged segment allowed at any given time during the lifetime of a connection, by a receiver window (R_{wnd}) variable, maintained at the sender. The size of R_{wnd} is determined at the receiver, by subtracting the

[2] Flow control is the process of regulating the traffic flow between two points and is used to prevent the sender from overwhelming the receiver with too much data, which will happen if the received data is not processed quickly enough. With the flow control process the sending system will adjust the rate at which it tries to send data to the destination system.

buffer space used from the total buffer space allocated. The sender gets an updated value of R_{wnd} with every ACK it receives.

TCP Congestion Control

The congestion control mechanism of TCP consists of four different algorithms: slow start, congestion avoidance, fast retransmit, and fast recovery [CHIII-16]. They act in a coordinated fashion, in order to prevent the overflowing of intermediate router buffers in the network located between sender and receiver, and the overflowing of the receiver itself. TCP attempts to achieve this goal by dynamically manipulating the window size of the network, by using a variable congestion window (C_{wnd}), which regulates the transmission rate of the sender. (C_{wnd}) is defined in [CHIII-16] as follows: "a TCP state variable that limits the amount of data a TCP can send. At any given time, a TCP must not send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of C_{wnd} and R_{wnd} ". Another state variable, the slow start threshold (ssthresh), is used to determine whether the slow start or congestion avoidance algorithm is used to control data transmission, as discussed below.

When a connection is established, the sender initializes C_{wnd} to the size of one maximum segment size (MSS); it then sends one maximum segment. It is doubling the window size (exponential growth) until it reaches a certain threshold. The initial value of ssthresh may be arbitrarily high (for example, some implementations use the size of the advertised window), but it may be reduced in response to congestion. The threshold for changing

from exponential to linear growth is reduced (by half) every time there is congestion, and eventually, the connection might end up increasing linearly its rate.

The slow start algorithm is used when C_{wnd} < ssthresh, while the congestion avoidance algorithm is used when C_{wnd} > ssthresh. When C_{wnd} and ssthresh are equal the sender may use either slow start or congestion avoidance.

During slow start, a TCP increment C_{wnd} by at most MSS bytes for each ACK received that acknowledges new data. Slow start ends when C_{wnd} exceeds ssthresh (or, optionally, when it reaches it) or when congestion is observed [CHIII-16].

During congestion avoidance, C_{wnd} is incremented by 1 full-sized segment per round-trip time (RTT). Congestion avoidance continues until congestion is detected. One formula commonly used to update C_{wnd} during congestion avoidance is given in Equation 3.1 [CHIII16]

$$C_{wnd} \mathrel{+}= \text{MSS}*\text{MSS}/C_{wnd} \qquad (3.1)$$

Equation (3.1) provides an acceptable approximation to the underlying principle of increasing C_{wnd} by 1 full-sized segment per RTT.MSS is defined as the maximum segment size not including any header. The size of the header for a TCP segment is 40 bytes. The congestion control keeps growing exponentially until either a timeout occurs or the receiver's window is reached. If it is a new connection, C_{wnd} will double every

RTT until the first packet loss is detected or the threshold is reached. After that C_{wnd} will increase by 1 for each RTT. As [CHIII-4] states, packet loss is defined by either 3 or more duplicate ACKs, or by a timeout of the retransmission timer associated with the oldest outstanding ACK.

In the case of duplicate ACKs, a variable slow start threshold (ssthresh) is set to one half of the size of C_{wnd} plus three segments, accounting for segments having left the network due to duplicate ACKs. Moreover, C_{wnd} is set to the size of (ssthresh). As we explained earlier, ssthresh is used for determining when the sender should enter congestion avoidance, i.e. increase of C_{wnd} linearly, by one MSS every RTT. Recovery from duplicate ACKs is initialized by a fast retransmit of the missing packet, without waiting for a timer timeout. After a fast retransmit, the sender uses fast recovery to control the sending of new packets until a non-duplicate ACK is received. During fast recovery, C_{wnd} is increased by one MSS for every additional duplicate ACK received and a new segment is transmitted if allowed according to the value of C_{wnd} and R_{wnd}. When an ACK for new data arrives, the value of C_{wnd} is set to the value it had upon receiving the first three duplicate ACKs, and fast recovery is exited.

In the case of a retransmission timeout (RTO), ssthresh is set to one half of the current C_{wnd}, but C_{wnd} is now set to one MSS. The reason for this is that a timeout is considered to be caused by a sudden change is the network load (ACK is lost), whereas a duplicate

ACK indicates that packets are still arriving at the receiver. Instead of entering congestion avoidance, the sender will enter into slow start.

The RTO is calculated from the following three equations (3.1, 3.2, 3.3) [CHIII-5]. To compute the current RTO, a TCP sender maintains two state variables, SRTT (smoothed round-trip time) and RTTVAR (round-trip time variation)

$$RTTVAR_{new} <- 3/4 * RTTVAR_{old} + 1/4 *|SRTT_{old}-R| \quad (3.2)^3$$

$$SRTT_{new} <-7/8*SRTT_{old} + 1/8*R \quad (3.3)$$

$$RTO <-SRTT_{new} + MAX (+4*RTTVAR_{new}, 1s.) \quad (3.4)$$

Where R is the recent round-trip time measurement.

3.2.1 Improvements to TCP Reno (New Reno, SACK, D-SACK)

Several improvements to the TCP have been proposed since the introduction of its original form (Reno) in 1990. Some of the proposals are: Selective Acknowledgment [CHIII-6], Duplicate Selective Acknowledgment [CHIII-7], and New Reno [CHIII-8].

3.2.2 Linux TCP

The Linux TCP implementation supports many of the suggested improvements, including the ones mentioned in section 3.2.1. Linux also takes some different approaches for congestion control [CHIII-9]. The author of [CHIII-9] describes the Linux-specific

[3] In computer science equation like a = b + c sometimes is written as: a <- b + c, which means that "<-" (the left pointing arrow) is an assignation operator, not a "less than equal sign".

protocol enhancements; moreover he points out the details where Linux TCP behavior differs from the conventional TCP implementation or the RFC specifications.

Congestion window is traditionally evaluated against the difference of the highest data segment transmitted (SND.NXT) and the first unacknowledged segment (SND.UNA). Although Linux conforms to the TCP congestion control principles, it takes a different approach in carrying out the congestion control. Instead of comparing the congestion window to the difference of SND.NXT and SND.UNA, the Linux TCP sender determines the number of packets currently outstanding in the network. The Linux TCP sender then compares the number of outstanding segments to the congestion window when making decisions on how much to transmit. Linux tracks the number of outstanding segments in units of full-sized packets, whereas the TCP specifications and some implementations compare C_{wnd} to the number of transmitted octets. This generates a different behavior if small segments are used: if the implementation uses a byte-based congestion window, it allows several small segments to be injected in the network for each full-sized segment in the congestion window. Linux, on the other hand, allows only one packet to be transmitted for each segment in the congestion window, regardless of its size. Therefore, the TCP implementation in Linux is more conservative compared to the byte-based approach, when the TCP payload consists of small segments.

When it comes to the RTO timer, Linux uses a value of 200 ms rather than 1second [CHIII-5]. Also, Linux TCP has a retransmission timer granularity of 10 ms and the sender takes a round-trip time sample for each segment. Therefore it is capable of achieving more accurate estimations for the retransmission timer. Linux TCP deviates from the IETF specification by allowing a minimum limit of 200 ms for the RTO. The

traditional algorithm for retransmission timeout computation has been found to be problematic in some networking environments [CHIII-10]. According to [CHIII-10], the first problem is that when the round-trip time decreases suddenly, the RTT variance increases momentarily and causes the RTO value to be overestimated. The second problem is that the RTT variance can decay to a small value when RTT samples are taken for every segment while the window is large. This increases the risk for spurious RTOs that result in unnecessary retransmissions. This could cause problems with the RTO during slow start, since then there are few ACKs being received and the RTO estimate will not increase (i.e. is underestimated) fast enough. The Linux RTO estimator attacks the first problem by giving less weight for the measured mean deviance (MDEV), when the measured RTT decreases significantly below the smoothed average. The reduced weight given for the MDEV sample is based on the multipliers used in the standard RTO algorithm. First, the MDEV sample is weighed by 1/8, corresponding to the multiplier used for the recent RTT measurement in the SRTT equation given in Section 3.1. Second, MDEV is further multiplied by 1/4, corresponding to the weight of 4 given for the RTTVAR in the standard RTO algorithm. Therefore, choosing the weight of 1/32 for the current MDEV neutralizes the effect of the sudden change of the measured RTT on the RTO estimator, and assures that RTO holds a steady value when the measured RTT drops suddenly. This avoids the unwanted peak in the RTO estimator value, while maintaining a conservative behavior. If the round-trip times stay at the reduced level for the next measurements, the RTO estimator starts to decrease slowly to a lower value. In summary, the equation for calculating the MDEV is as follows: [CHIII-9]

if (R < SRTT and |SRTT - R| > MDEV){

$MDEV \leftarrow 31/32 * MDEV + 1/32 * |SRTT - R|$

} *else* {

$MDEV \leftarrow 3/4 * MDEV + 1/4 * |SRTT - R|$ (3.5)

Where R is the recent round-trip time measurement, and SRTT is the smoothed round-trip time. Linux does not directly modify the RTTVAR variable, but makes the adjustments first on the MDEV variable, which is used in adjusting the RTTVAR, which determines the RTO. The RTO estimator and SRTT variables are set according to the standard specification ((3.2) and (3.3))

A separate MDEV variable is needed, because the Linux TCP sender allows decreasing the RTTVAR variable only once in a round-trip time. However, RTTVAR is increased immediately when MDEV gives a higher estimate, thus RTTVAR is the maximum of the MDEV estimates during the last round-trip time.

Linux also selected Linux TCP features that differ from a typical TCP implementation. Linux implements a number of TCP enhancements proposed recently by IETF, such as Explicit Congestion Notification [CHIII-11] and D-SACK [CHIII-7]. These features are not yet widely developed in TCP implementations, but are likely to be in the future because they are promoted by IETF [CHIII-9].

Presently, BicTCP [CHIII-12] has been implemented into the standard release of the Linux 2.6.kernels [CHIII-13]. In this work all the tests were done under the Linux kernel 2.4.20, therefore we will not cover the details of BicTCP implementation in Linux. Notice that Linux 2.4 by default has TCP SACK option enabled.

3.3 TCP performance in 802.11 Wireless LANs

Researchers have realized that TCP performance can degrade over wireless networks. One reason is that, in a wired network, TCP interprets a packet loss as an indication of congestion in the network. This assumption is based on the small loss of wired links. However, most packet losses in wireless networks are from packet corruptions due to fading and interference in the wireless channel [CHIII-17]. Since the wireless medium is broadcast and the TCP traffic is bi-directional, packets might collide over the wireless segment with ACKs sent from destination to the source, passing through the same "broadcast" wireless network, which indeed is not caused by congestion. Moreover, as described in section 3.2, TCP will react the same way as it does on a wired network, i.e. reacting to congestion. Signal fading due to multipath, or interference from other transceivers (e.g. other STAs/cells in cellular networks, microwave ovens in the home environment etc) are typical problems in wireless media. Although some recovery is performed at the MAC layer, the performance of TCP will still suffer, and it may also trigger TCP retransmissions. Link latency is another factor that might affect TCP performance. It is not uncommon that the link latency contributes to the majority of the RTT, leading to higher RTO values, potentially degrading performance. A high RTT variance is not uncommon in a wireless network, which ultimately affects the RTT estimation of TCP [CHII-11].

Many solutions, including link-layer and transport-layer modifications, are proposed to overcome the TCP problems such as the one presented in [CHIII-14]. In this article the authors examine the wireless link characteristics, outline the performance problems they cause to TCP, and then present a wide range of solutions to these problems. In [CHIII-15]

the authors, identify the fundamental reasons for TCP performance degradation over asymmetric networks and present several techniques to address this performance degradation problem. [CHIII-18] focuses on illustrating the typical wireless TCP techniques with examples. The authors discussed about several wireless TCP solutions[4] in wireless networks; they concluded that a universal solution for all types of wireless networks is unlikely to be developed; since the characteristics of wireless networks vary as access technologies differ.

[4] For example, ATCP in Ad Hoc Networks, Freeze-TCP in Cellular Network, and etc.

Chapter 4

System description

In this chapter we present the equipment and software that was used in our test-bed, as well as the encountered problems. Before proceeding to the description of the test-bed, we discuss briefly about the Linux operating system and signal strength measurements.

4.1 Linux Networking Overview

In this section we present a brief overview of the Linux operating system, with our focus placed on the network subsystem. We also survey available Linux device drivers for wireless 802.11 based networking.

4.1.1 The Linux Operating System

A good description of Linux, according to [CHIV-1] is: " Linux is a freely available multiuser, multitasking, multiprocessor, and multiplatform UNIX operating system". It is an open source operating system, which means that the source code is available so that everyone can modify it, even distribute his/her modifications, but under its original copyright.

The primary author of Linux, Linus Torvalds, released the first versions of Linux on the Internet in 1991. One of the important facts about Linux is that its development occurs simultaneously around the world and is continuously improved by countless number of people.

Originally, the term Linux described only the operating-system kernel, however over time the term evolved to mean the kernel together with the entire system environment. It now refers to the operating-system kernel (currently versions 2.0, 2.2, 2.4, or 2.6), the system programs (compiler, libraries, tools, etc.), the graphical user interface (e.g., Xfree), a window manager or an application environment (KDE, Gnome, etc.) and a large number of applications from all areas (editors, browsers, office applications, games, etc.). When it comes to networking, Linux offers a rich functionality and robustness under heavy traffic loads. Moreover, many applications are freely available in Linux, which allow us to implement and evaluate new theories and protocols in real-world networks. For example Linux has been used to study various modifications of the TCP transport protocols, to develop frameworks for QoS support (e.g., the Karlsruhe Implementation of Differentiated Services (KIDS), project developed at the University of Karlsruhe, Germany) and to develop a high-resolution timer (also developed at the University of Karlsruhe) [CHIV-1].

Linux also supports many different networking protocols such as: TCP/IP, IPv6, IPX/SPX, Appletalk, WAN Networking (X.25, Frame-relay, etc.), ISDN, PPP (Point-to-Point-Protocol), SLIP (Serial Line IP), PLIP (Parallel Line IP), amateur radio protocols, and ATM. The development of the Internet and its services is inseparably linked to UNIX systems, which is why the properties of the TCP/IP protocol family and its behavior can be properly studied and controlled in a UNIX system rather than in other operating systems.

A usual criticism has been that the driver support for Linux is not up to the level provided by competing commercial systems. This lack of driver support has become one of its

major problems. For example during the set up of the systems used in this work the required drivers for the wireless network cards were available from the manufacturer only for the Windows operating system, whereas it was necessary to install open-source project drivers [CHIV-2] as it will be detailed later. Nevertheless, this situation is continuously changing and it is very likely that the Linux community will provide a matching driver for each new device appearing in the market.

In order to have a better understanding of Linux network performance, it is necessary to review some critical parts in the network subsystem of the Linux kernel. The following sections describe this subsystem in detail.

4.1.2 Overview of the Directory Tree for the Linux Networking Code

The high-level networking functions can be found in the /usr/src/linux/net directory. The low-level networking functions are carried out by the drivers. They pass received packets up to higher level protocols or get packets from them and pass them to the lower level. They may also discard the data, or use it in-kernel functions, depending on the packet. The net/core directory contains useful code for most of different network protocols, as do some of the files in the net directory itself. Specific network protocols are implemented in subdirectories of net/. For example, IP (version 4) code is found in the directory net/ipv4 [CHIV-3]

Figure 4.1 is a list of directories in /usr/src/linux/, which shows where the networking code is located in the Linux kernel source code, in other words it is a list of directories which are in the directory linux. Most of the code is in net/ipv4. The rest of the relevant code is in net/core and net/sched. The header files can be found in include/linux and include/net [CHIV-5]

Figure 4.1 is a list of directories in /usr/src/linux/, which shows where the networking code is located in the Linux kernel source code, in other words it is a list of directories which are in the directory linux. Most of the code is in net/ipv4. The rest of the relevant code is in net/core and net/sched. The header files can be found in include/linux and include/net [CHIV-5]

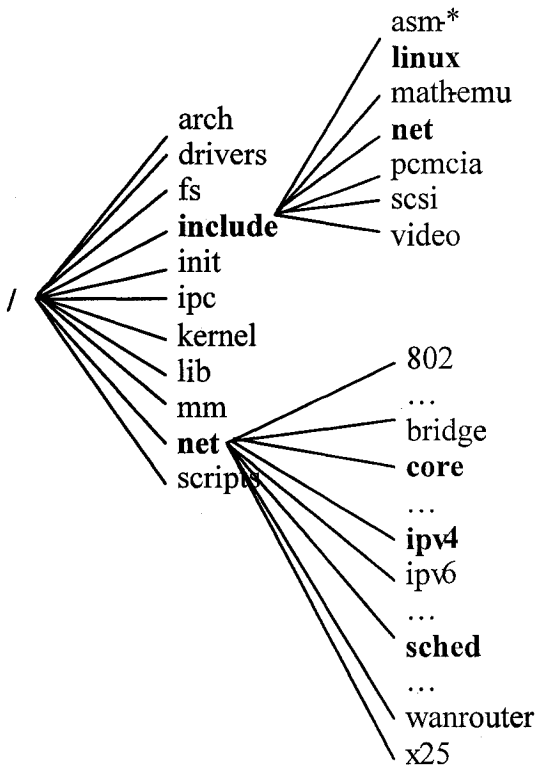


Figure 4.1: List of directories in Linux

4.1.3 The Linux architecture and the network subsystem

The Linux kernel is part of a larger system. This way, it makes sense to discuss the kernel in the context of the entire system. Figure 4.2 shows a decomposition of the Linux operating system.

The Linux kernel abstracts and mediates access to the hardware resources, including the CPU. The Linux kernel is composed of five main subsystems: the process scheduler, the memory manager, the virtual file system, the network interface, and the inter-process communication interface. These subsystems interact with each other using function calls and shared data structures.

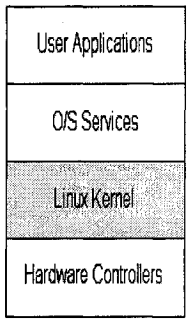


Figure 4.2: Decomposition of the Linux system into major subsystems, according to [CHIV-9]

The network subsystem allows the Linux systems to connect to other systems over a network. There are several hardware devices that can be supported, and several network protocols that can be used. All network operations, such as handling an incoming packet, have to be managed by the operating system and be allocated the appropriate level of priority. Incoming packets are asynchronous events. They have to be collected, identified, and forwarded with low delay. This is the reason why the kernel is responsible for the handling of packets across program and network interfaces [CHIV-1].

4.1.4 Network Devices

As we know each communication task carried out over a network requires a physical medium, which is accessed through a network adapter (network interface card). For

example, in the ISO/OSI reference model, the task of a network adapter spans over the layers 1 and 2.

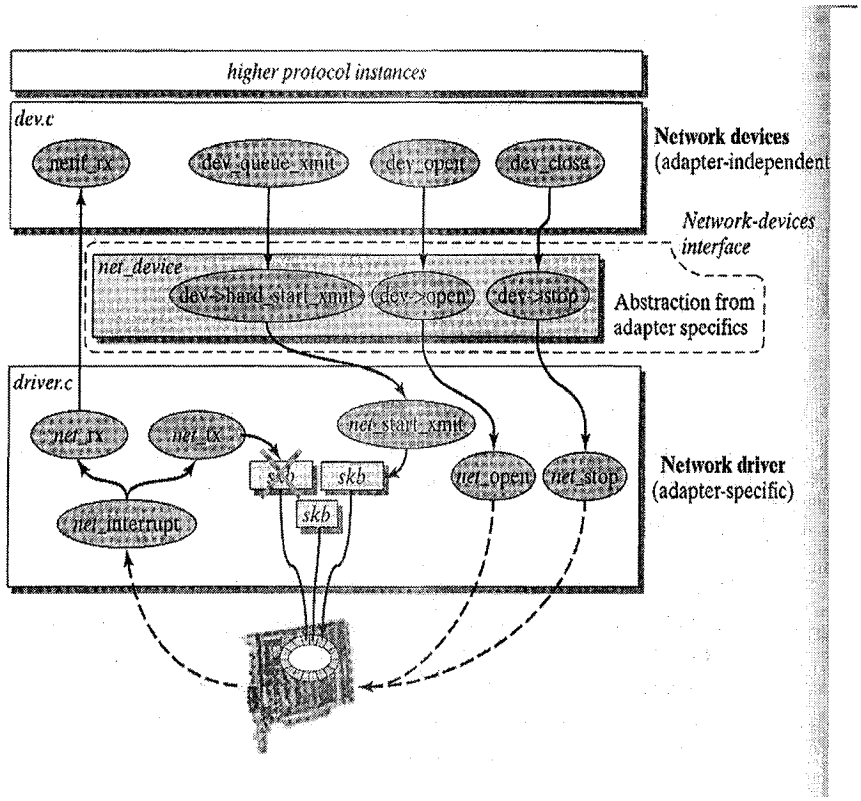


Figure 4.3: A structure of a network device interface [CHIV-1]

In the Linux network architecture this interface between software-based protocols and network adapters is implemented by the concept of a network-device. A network-device interface should meet two properties. The first one is to hide the implementation-specific details due to the fact that network adapters are manufactured by different vendors and they might differently implement layer-1 and layer-2 protocols. This means that their configurations are individual and specific to each network adapter. For this reason a driver, a piece of software is needed for each adapter to communicate with the operating system. The second property that a network device must have is to provide a uniform

access interface to protocol instances. In a system like Linux there are several protocol instances using the services of network adapters. These instances should be implemented independently of a specific type of the adapter, which means that network adapters should have a uniform interface to the higher layers (Figure 4.3)

Each data transmission in the Linux network architecture occurs over a network card. A network card is an interface adapter that automatically transmits and receives network packets according to a defined MAC protocol (Ethernet, Token Ring, etc.). This means that a network adapter has an independent logic that works in parallel to the regular central processor(s). The network adapter uses interrupts and its driver-specific interrupt-handling routine to communicate with the operating system. (Figure 4.4) When the adapter wants to pass data to the processor (e.g. a packet it received), it triggers an interrupt, and the processor uses the interrupt-handling routine of the network adapter to serve the network adapter [CHIV-1]

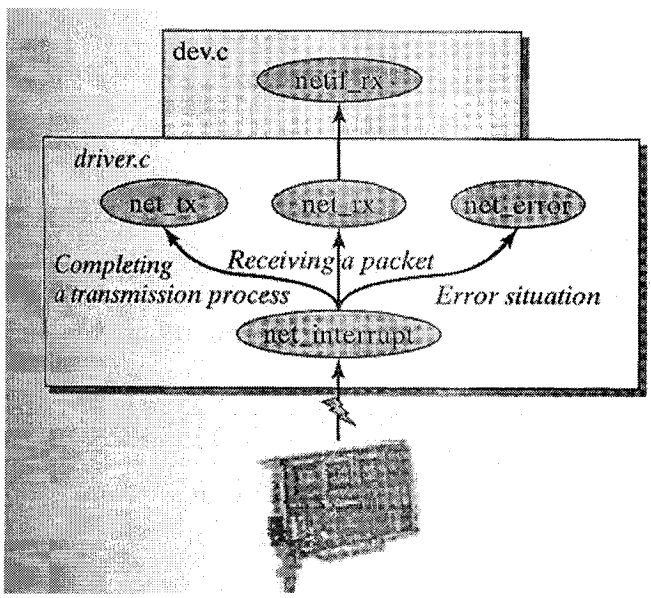


Figure 4.4: A network adaptor uses an interrupt to send message [CHIV-1]

4.1.5 Data packet transmission

The function hard_start_xmit (skb, dev) of the C the file isa_skeleton.c, located in directory /usr/src/linux/drivers, is responsible for forwarding a data packet to the network adapter in order to be transmitted. The data packet of the socket buffer is copied to an internal buffer located in the network adapter and a time stamp is attached, marking the beginning of that transmission. If this copying action was successful, it is also assumed that the transmission will be successful. In this case hard_start_xmit () will return a value of 0. Otherwise it will return 1, alerting the kernel that the packet could not be sent.
After forwarding the network packet between the operating system and the network adapter, either one of the following two different technologies will be applied, depending on the kind of the network adapter.
Older network adapters (e.g., 3Com 3c509) have a limited internal buffer size memory on the adapter for packets to be sent. This means that the kernel can always forward only a single packet to the adapter at a time. If the buffer in the adapter is free, a packet will be copied into the adapter right away and the kernel can delete the corresponding packet in the socket buffer. More recent network adapters work differently. The driver manages a ring buffer consisting of 16 to 64 pointers, pointing to socket buffers, where the packets are also ordered using a ring buffer. When a packet is ready to be sent, the corresponding packet in the socket buffer is arranged within this ring, and a pointer to this data packet is passed to the network adapter. Subsequently, the packet in the socket buffer remains in the ring buffer until the network adapter, using an interrupt, notifies that the packet was transmitted. Finally, the packet in the socket buffer is removed from the ring buffer and the memory is freed. If an error occurred during the transmission, then the packet in the

socket buffer should not be removed because the kernel will most likely attempt to retransmit the packet. [CHIV-1]

4.1.6 Data packet reception

When the network adapter receives a packet it has to be forwarded to the kernel. Packet reception is an asynchronous operation therefore it might occur whereas the processor may be busy doing other tasks. In general, there are two methods to inform the kernel that a packet has arrived. The first method is called the polling method. With this method, the system periodically asks the network adapter whether any data has been received or not. One major problem of this method is the selection of the time interval, used to ask the network adapter. If it is too short, computing time is wasted. If it is too long, it might cause unnecessary delay to the data exchange, as well as the network adapter might not be able to buffer all incoming packets. The second, and better method, which modern network adapters follow, uses an interrupt and an appropriate interrupt handling routine to inform the operating system about an incoming packet. The system processor is briefly interrupted in order to accept the received packet and to store it in a queue. The packet will further be handled at a later time. This interrupt method has a better performance than the polling method and it adapts itself better to current system loads.

4.2 Representation of Signal Strength in wireless network

4.2.1 "signal strength", "signal quality", and "signal to noise ratio"

There is an inconsistency in the way terms such as, "signal strength", "signal quality", and "signal to noise ratio" are used by researchers, developers and users, when it comes

to the 802.11 standard. At the beginning of our test, it was not clear what the signal strength values reported by each of our computers represented or if they were correct. From our personal observations, and after researching this subject, we came to the conclusion that, this confusion arises when common terms are used with inconsistent definition, or misunderstanding of the common terms leads to false conclusions. In [CHIV-4] we find the precise definitions of the above terms (those definitions coming from reliable sources such as 802.11 standards), as follows:

"• *Signal strength* is defined in 802.11 as the Received Signal Strength Indicator (RSSI). RSSI, is intended to be used as a 'relative value' within the chipset. This is a 1-byte value so that it could have values ranging from 0 to 255, but vendors prefer to use arbitrary scales from 0 to RSSI_Max where the latter is vendor-specific (for instance, Cisco uses 101, Symbol 31, Atheros 60). It is not associated with any particular power scale (e.g. mW) and is not required to be of any particular accuracy or precision. The RSSI value is used internally by the microcode in the adapter and this is why vendors are not forced to use a compatible standard. As an example of its use, if the RSSI value is below some threshold, the NIC knows that the channel is idle. Therefore, the signal strength numbers reported by an 802.11 card will probably not be consistent between two vendors, and should not be assumed to be particularly accurate or precise (Details are provided below).

• *Signal quality* is defined very briefly in the 802.11 standard. Common definitions have arisen, but they are usually incorrect. The correct definition hinges on the

term, "PN code correlation strength," which is a measure of the match (correlation) between the incoming DSSS signal and an ideal DSSS signal.

• *Signal to noise ratio* is a general term that is used in a novel way by 802.11 administrators. Most usages of the term refer to the strength of the signal relative to thermal noise within a circuit , but many professionals, use the term to refer to the strength of the signal at the receive antenna relative to the ambient, non-802.11 RF power that is present at the bandwidth occupy by the signal. According to the standard communications systems terminology, SNR is defined as the ratio of received signal power to the power of the additive Gaussian noise that appears at the output of the receiver. While thesedefinitions are not wrong, they may lead to confusion when 802.11 professionals communicate among themselves.

• *Receive sensitivity* refers to the weakest power level the card's internal thermal noise will allow it to receive. It is unrelated to the ambient, non-802.11 RF energy in the environment. [CHIV-4].

4.2.2 Measurement Units for RF Signal Strength

There are four Units of measurements that are used to represent RF Signal Strength: mW(milliwatts), dBm ("dB"-milliwatts), RSSI (Received Signal Strength Indicator), and a percentage measurement.

Equation (4.1) shows that "dBm" is a logarithmic measurement of signal strength and dBm values can be directly converted to and from mW values.

$$dBm = \log_{10}(mW) * 10 \qquad (4.1)$$

RSSI, which has been explained earlier, is an arbitrary integer, with an allowable range of 0-255 (a 1-byte value) defined in the 802.11 standard. This value is used internally by the micro-code of the adaptor and by the device driver. For example, when an adaptor wants to transmit a packet, it is checking whether the channel is clear (i.e.: no one else is transmitting) or not. If the RSSI value is below a certain low value, the chipset knows that the channel is clear [CHIV-6]. No vendors have chosen to measure 256 different signal level values. Each of the 802.11 NIC's vendors adopts and uses a specific maximum RSSI value (RSSI_Max). For example, Cisco chooses to measure 101 separate values for RF energy and their RSSI_Max is 100. Symbol uses an RSSI_Max value of 31. The Atheros chipset uses the RSSI_Max value of 60. In Appendix B, more details information is provided.

When using RSSI as basis for reporting dBm signal strength, it is common to see the signal strength been represented as a percentage. The percentage represents the RSSI for a specific packet, which constitutes practically the division between the measured to the maximum (RSSI_Max) value, multiplied by 100 (in order to derive the percentage). For example, when the signal level is 50%, this is reported with different values of RSSI, depending on the vendor; a Symbol card would convert to an RSSI of 16, because its RSSI_Max =31, Atheros, with RSSI_Max = 60, would convert it to an RSSI of 30, and for Cisco, which is the easiest one because its RSSI_Max = 100, RSSI is 50. Unfortunately, we still do not know how all vendors map RSSI to signal strength percentage. This lack of consistency between vendors, does not allow for direct comparison of performance evaluation results, performed with equipment of different vendors.

"Signal quality" is also reported by vendors' client utilities. Those, who are working with IEEE 802.11 products, are aware that these two parameters, "signal strength" and "signal quality", are the two metrics for assessing the "goodness" of the 802.11 signal. Unfortunately, other that the definition of "signal quality" we provided earlier in section 4.2.1, the 802.11 standard does not offer further information. Based on the definition, the authors of [CHIV-6] conclude that "signal quality" "reflects the amount of signal within the channel formed between the two communicating stations (e.g. an AP and a client). The above description is consistent among the manufacturers. For example a manufacturer might say that an IEEE 802.11b chipset needs a minimum of 20 dBm signal quality in order to achieve 11 Mbps data rate. But the 802.11 standard, does not define a specific method of calculating and reporting the "signal quality", and, as in the case of "signal strength", the vendors measure it using inconsistent methods. The IEEE 802.11 standard defines "Signal to Noise Ratio" (SNR), as can be seen in section 4.2.1. However, since 802.11 cards do not typically report SNR, practically the concept of SNR is not used.

4.2.3 The Linux Wireless Extension

The Linux Wireless Extension (WE) and the Wireless Tools is an Open Source project. It has been sponsored by Hewlett Packard since 1996, and progresses with the help of many Linux users all over the world.

In directory /proc/net, in the file named wireless, we can find some wireless statistics on each wireless interface in the Linux system, which gives us the standard driver statistics.

According to [CHIV-7], the Wireless Extension allows a driver to expose to the user space, configuration, and statistics specific to common Wireless LANs A single set of tools can support all the variations of Wireless LANs as long as the driver supports them. As well as, these parameters may be changed on the fly without restarting the driver (or Linux). As an example of the information that can be obtained with the Linux wireless extensions, by using the following command

$cat /proc/net/wireless, we got the following information:

		Link	Level	Noise
Atheros:	ath1:	38	199	161
Orinonco:	eth1:	57	219	161

The numbers on the right hand side are: link quality, signal level and noise level. This is device-dependent information and they are defined in [CHIV-7] as below:

Quality - link : general quality of the reception.

Quality - level : signal strength at the receiver.

Quality - noise : silence level (no packet) at the receiver.

From the above reported values, "level" gives the signal strength in arbitrary units whereas "noise" gives the noise level in the same units. The value reported by "link quality" is not SNR. It measures the degree of correlation between what was received and what should have been received. This is not SNR and although it may be related to the SNR, it is not clear how it is calculated. One way to estimate the SNR is to convert the reported values of "level" and "noise" to dBm, as was explained in the previous section as well as in the Appendix B, and then from there compute the SNR. All this is just

approximate since the cards are not intended to provide high-accuracy readings. The important point according to [CHIV-7] is that "the RSSI is measured between the beginning of the start frame delimiter (SFD) and the end of the PLCP header error check (HEC)" corresponding to the current packet being received. It means that RSSI is updated once each packet is being received. This measurement remains there until the time another packet is received. In other words, if no new packets are received, the values for "old" signal strength value remain, since there are no new measurements taken. The noise level is determined by the strength of the RF power in the channel, and can be measured during periods where there is not any packet transmission occurring.

4.3 Test-bed description and equipment

Our experimental performance evaluation tests were conducted on a network consisting of an access point (AP) and wireless stations. The access point was implemented using a desktop computer running Linux and appropriate device drivers. We used a laptop computer as a mobile station and another desktop computer as a stationary end point. The equipment is described below in more detail.

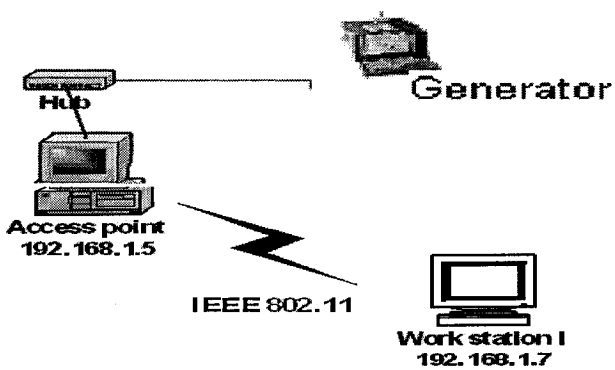


Figure 4.5: Network Topology

On the other hand, the tests of the second set were carried out the corridor outside the same laboratory previously mentioned (See Figure 4.6b). In this second set, the PC was kept stationary at one end of the corridor and the laptop computer was placed at different spots along the corridor. . This corridor is mostly made of concrete walls with a large glass window at one end (the one opposite to the position of the stationary station). This setting is representative of school buildings, underground tunnels connecting building, or malls, like our tunnel system in the University of Ottawa and Carleton University.

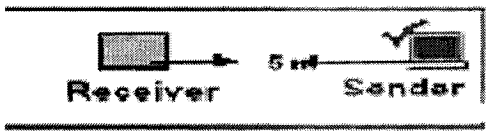


Figure 4.6a: Performance test in the Lab

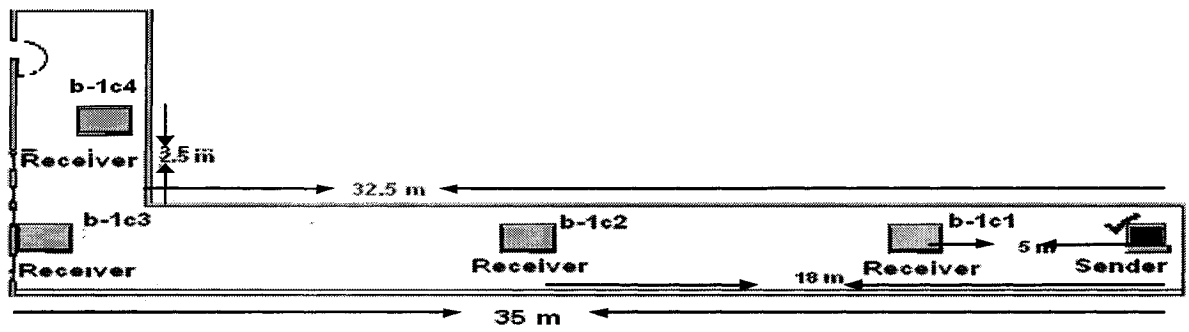


Figure 4.6b: Performance test in the corridor

The AP is a standard IBM P4 PC with a Linksys 802.11b wireless PCMCIA wireless card. This wireless standard works in the band of 2.4 GHz and can provide a data rate up to 11Mbps. The computer runs the Linux RedHat distribution with kernel 2.4.20. The AP

functionality is achieved using the HostAP Driver, which is configured as "master" mode.

The mobile receiver is an IBM R50p laptop. We ran tests with this computer and several Linux distributions. Linux RedHat, with kernel 2.4.20 was used for the tests with the Orinoco Gold 802.11b wireless card [CHIV-12]. We also used more recent distributions, which simplify the task of installing and configuring the wireless card. We used Linux Mandriva kernel 2.6.X [CHIV-11] to carry out the tests with the D-link DWL-AG660 PCMCIA wireless card in standards 802.11a and 802.11g. The Orinoco Gold wireless cards use the Orinoco driver included in the package PCMCIA-CS-3.1.29. It has to be configured as "managed" mode to work as a slave station. The Orinoco Gold card works at 2.4 GHz and can support 1 Mbps, 2 Mbps, 5.5 Mbps and 11 Mbps. The D-link DWL-AG660 card is not supported by a standard driver. We used a driver supplied by the Open Source project "MadWifi", which stands for "Multi-band Atheros Driver for Wireless Fidelity" [CHIV-2]. This project provides a Linux device driver for Atheros-based Wireless LAN devices. The driver makes the WLAN card to appear as normal network interface in the system. Additionally there is support for the Wireless Extensions

4.3.1 Network load generators

In order to generate network traffic for testing purposes, an InterWatch 95000 (IW95000) network analyzer/traffic generator [CHIV-13] was used in the test-bed to produce external UDP traffic (see Figure 4.7). As shown in Figure 4.5, this equipment is connected to the computer acting as the access point through a Fast Ethernet network. It is interesting to mention that although in our tests we set this generator to produce CBR traffic, the generated traffic pattern shows slight variations in packet inter-arrival times to

the access point. However, on average the equipment produces the data rate that we can specify through the parameters values. An example of this, in Figures 4.8a and 4.8b, we show the probability density function and cumulative distribution function of the packet inter-arrival time for a case with a data rate of 1000 packets/s. We took 10,000 samples and obtained a mean inter-arrival time of 0.001005 (s) with a standard deviation of 0.000888 (s), the minimum inter-arrival time was 0, and the maximum inter-arrival time was 0.00201 (s). It is shown that with probability 0.4 packets are arriving with small inter arrival time and with probability of 0.25 packets are coming with larger inter arrival times.

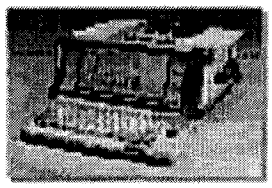


Figure 4.7: InterWatch 95000 Network Analyzer [CHIV-13]

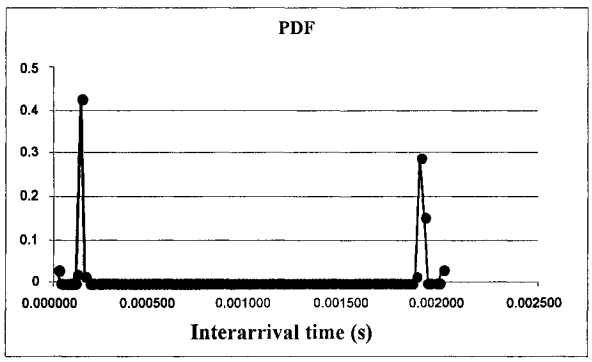


Figure 4.8a: Probabilty Density Funcion Funcion of Packet Interarrival time

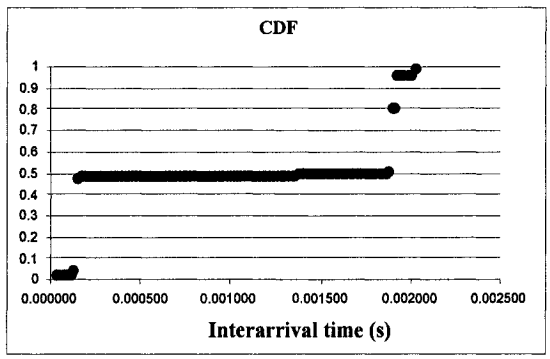


Figure 4.8b: Probabilty Density Function of Packet Interarrival time

The packet interarrival statistics were determined as follows. At the access point we used the Ethereal software tool in order to capture arriving packets coming from the traffic generator. Upon reception, Ethereal time stamped each packet with the local time. At this

point it is important to mention that Ethereal uses libpcap, a capture library (described below), which supports a resolution of 1 microsecond. It means that if two packets arrive within the same microsecond the software tool will report the same time stamp for both of them. We determined the interarrival times by taking the difference in arrival times for two consecutive packets and we performed this procedure for all packets in the test. We are reporting that some inter-arrival times are recorded equal to zero, we know that there cannot be zero interarrival times (we must have at least one frame transmission time between two consecutive frames). This is likely due to the fact that if the processor performs a higher priority task when a frame arrives this frame will have to wait in the NIC buffer until the processor is free to retrieve it, but it may happen that another frame may have arrived by the time the processor is ready to retrieve data. Therefore, both packets will be stamped with the same time stamp since this procedure is faster than the time resolution of the monitoring tool, which is 1 microsecond. Now why this is possible? There are at least two reasons:

1) If a higher priority task is running in the processor, the interrupt caused by a frame arrival will have to wait. This is due to the fact that in the PC architecture the interrupts are prioritized [CHIV-14]. According to the wiring diagram in [CHIV-14], the interrupt priorities are (from highest to lowest): 0-2, 8-15 and 3-7. NIC cards are usually given IRQ (interrupt request) 11, which means that interrupts 0-2, 8, 9 and 10 have higher priority!

2) Modern NICs use what is called "interrupt coalescing". The problem that this feature addresses is as follows. If an interrupt happens too often there will be significant CPU overhead because for every interrupt the CPU has to save a task state in order to resume

the task at a later time. Some high performance NICs delay the generation of the hardware interrupts for some time in order to try to capture another frame (or a group of frames) before the interrupt is generated. In this way with only one interrupt the NIC will deliver a group of frames.

[CHIV-14] addresses the impact of interrupt coalescing on performance:

In summary, inter arrival times of zero are not happening in reality. They are due to the limitations of the monitoring tool (the PC).

In order to generate TCP traffic, we used file transfers using the file transfer protocol (FTP). We transferred 1Gbyte files from one system to another under user control. When a user engages in a file transfer, FTP establishes a TCP connection to the target system for the exchange of control messages. When the transfer is complete the control connection is used to signal the completion and to accept a new file transfer.

4.3.2 Software tools

In order to generate trace files, the UNIX command TCPDUMP was used. It prints out the headers of packets on a network interface that matches a Boolean expression [CHIV-8]. It uses the libpcap (packet capture library)[1], which is freely available. The libpcap library is versatile and works with BSD packet filter [CHIV-10], the SVR4 Data-link Provider Interface (DLPI) and the Linux SOCK_PACKET interface. TCPDUMP works

[1] The Packet Capture Library provides a high-level interface to packet capture systems. In the operating system, the Berkeley Packet Filter (BPF) is the packet capture system. This library provides user-level subroutines that interface with the BPF to allow users access for reading unprocessed network traffic. By using the Packet Capture Library, users can write their own network-monitoring tools. Applications using the Packet Capture Library subroutines must be run as root user. A reference for BPF is in UNIX® Network Programming, Volume 1: Networking APIs: Sockets and XTI, Second Edition by W. Richard Stevens, 1998.

by capturing data link layer frames, for instance Ethernet frames, since Ethernet is a standard for the data link layer. (It does not capture IP packets, nor TCP segments directly.) It then extracts network layer information and transport layer information from the captured data link layer frame. It is thus able to report the information that is contained in the data link layer header, such as MAC address. Finally, in order to analyze the trace files, we wrote a set of programs in C language as is described below.

- **Wireless Channel Monitor**

One of the main factors affecting the performance of a wireless communications system is the time variability of the wireless channel. This applies to the case of WLANs as well. In order to register these changes, a wireless channel monitor was designed to record the condition of the wireless channel during data transmission. This program uses the library provided by the wireless tools package and records the values of link quality, signal strength, and noise level, which have been explained in section 4.2.3, after each packet reception.

- **Network Throughput Monitor**

This program is based on the library of PCAP package. It captures all packets arriving or leaving from a particular wireless network interface. Then it records the packet's ID, size and time stamp for later analysis. It also filters, based on the type of the traffic, and calculates and records throughput for that specific traffic.

- **Transmission Delay monitor**

Transmission delay, which is the NIC delay, is the time the wireless card takes in order to successfully transmit a frame. Commercially available 802.11 wireless NICs do not provide this information to the upper protocol layers. Instead, we used an alternative delay measure, collected by a custom-made software tool developed by a colleague (Mr. Dong Liu) as part of his MASc research in the BWRL. The tool measures the difference between the time the Internet layer protocol delivers a frame to the driver of the wireless card, and the time the driver generates an interrupt, informing the socket that is ready to receive a new packet.

- **Strings of consecutive received / lost packets**

This C program is based on the library of PCAP package. It captures all the packets arriving or leaving the specified network interface. We designed a filter based on packet IP header value and then we recorded the strings of consecutive packets received (without a loss of a packet(s) in-between), and strings of consecutive packets, which are not received (in other words, size of a burst of lost packets).

4.4 Knowledge acquired through hands-on experience, major difficulties- encountered, and solutions

4.4.1 Linux support for novel technologies

During the development of this work we observed that long-established technologies are likely to have a smooth operation under Linux. In our tests we did not have problems

with devices using the IEEE802.11b standard such as the Orinoco Gold 802.11b wireless card, and Linksys 802.11b wireless PCMCIA wireless card. In these cases the drivers were available for Linux from their specific vendors.

However, there is a constant lag between the appearance of new devices and the availability of proper drivers for the Linux operating system. The current trend is to ship new devices with Windows-only drivers, leaving to the Linux community the responsibility of developing the appropriate driver. Eventually the driver comes out, although first versions are prone to errors. This seems likely to continue since software development in Linux is based on "good will".

Newer devices of hot technologies such as WLANs are a different story. At the time of this writing, there is a generalized lack of support for new devices since the corresponding drivers are not produced fast enough to cope with the rate at which new products with different chipsets are introduced into the market. As an example of this, the support for devices using the currently popular WiFi "11g" can be considered as deficient. To make things worse, new variations, termed 11g "turbo" or "super" recently came to the market. Now it is possible to find PCI, or PCMCIA wireless cards that work in three modes (11a/b/g) with some of them also support turbo g, but they may probably use totally unsupported chipsets.

4.4.2 Linux drivers for WiFi devices

Some working WiFi drivers on Linux are listed below:

- MADWIFI. This is an open source Linux driver for 802.11a/b/g universal NIC cards: Cardbus, PCI, or miniPCI with Atheros chip sets [CHIV-2].

- Intel. The company has provided open source drivers for their Centrino WIFI chipsets [2200BG/2915ABG].

- Ralink. This Taiwanese company has GPLed the drivers for their RT2400 (11b only) and RT2500 (11g) chipsets [rt2x00.serialmonkey.com/]

- NDIS Wrapper. This is not a real driver, but a piece of software that allows us to use the Windows driver, making the card to work under Linux. Nevertheless, the functionality that can be achieved is very limited and restricted to what can be done with the Windows driver.

In addition to tests with the previously mentioned cards (which were supported by standard Linux drivers) we also carried out tests with the more modern cards DWL-AG660 PCMCIA, and DWL-AG530 PCI. Both of them have the Atheros chipset, thus we tried to use the MADWIFI project to make them work in Linux.

The MADWIFI driver depends on a binary-only HAL (hardware abstraction layer) for regulatory reasons. A user does not have access to the source code in the HAL module (closed source) since this piece of software stops the driver from setting illegal transmission signal power among other things. Atheros provided the HAL for use with Linux.

The MADWIFI project produced a working driver but with the time it was evident that the initiative had lack of project management, lack of documentation, lack of transparency, lack of developers and lack of development (http://article.gmane.org/gmane.linux.drivers.madwifi.devel/1421). In the fall of 2005, in

an attempt to change this situation driver development had a major change including a name change from MADWIFI to MADWIFI next generation or MADWIFI-NG. MadWifi-ng's work is in progress but currently there is no a product release available yet. This does not mean that we cannot use the driver under development, but it will probably need some hacking and debugging. Therefore, many problems may occur here and there. Although there is available support via the Support Section in their web site and various problems are documented, many current problems remain unsolved. According to [CHIV-2], New Madwifi Code From Atheros, which claimed that "This page is a brief description of the features of the latest lot of code from Atheros. It will probably be incomplete for some time, so please bear with us". Although we were aware of the possible problems, it was our only alternative and started to compile Madwifi and follow the instructions. We should point out that compiling the driver for different hardware /software, had different errors. Based on our experience, Madwifi was more compatible with PCMCIA in the Laptop and Linux kernel 2.6, and we were able to find a solution for the problems that arose.

4.4.3 Linux kernel support for wireless communications

Issues with the MADWIFI driver

One of the first problems that we encountered when trying to install the MADWIFI driver was that available installation scripts were not flexible and general enough as to work under different Linux distributions and failed to build the driver with no indication as to what the problem was. This issue has already been reported to Madwifi developers [CHIV-2]. The failed procedure produced the following lines:

> >>Make [3]: *** [ieee80211_linux.o] Error 1

> >>Make [3]: Leaving directory `/lib/modules/2.4.20-8/madwifi-ng/net80211'

> >>Make [2]: *** [_mod_/lib/modules/2.4.20-8/madwifi-ng/net80211] Error 2

> >>Make [2]: Leaving directory `/usr/src/linux-2.4.20-8'

> >>Make [1]: *** [all] Error 2

> >>Make [1]: Leaving directory `/lib/modules/2.4.20-8/madwifi-ng/net80211'

> >>Make: *** [all] Error 1

These errors occurred when working with kernel version 2.4.20-8. After doing some research and tinkering, we found out that working with kernel version 2.6 produces less problems, so we decided to upgrade our systems to kernel version 2.6.11.

Once we upgraded the system we faced some other errors of different nature during module load. The solution was to use the compiler "gcc" of same version that was used to compile the kernel; otherwise it would cause "Invalid module format" errors during module load.

When compiling the driver it is important to compile the Madwifi driver specifying the location of the kernel build tree, e.g.: make KERNELPATH=/usr/src/linux-2.6.11.

If the kernel was built outside the source directory, the variable KERNELPATH should point to the output directory, not to the sources.

If the driver is successfully built, the procedure will produce numerous loadable modules:

---

ath/ath_pci.ko (Atheros driver for PCI/Card bus devices),

ath_hal/ath_hal.ko (Atheros HAL), and

net80211/wlan.ko (802.11 support layer)

net80211/wlan_wep.ko (WEP cipher support)

net80211/wlan_tkip.ko (TKIP cipher support)

net80211/wlan_ccmp.ko (AES-CCMP cipher support)

net80211/wlan_xau th.ko (external authenticator)

net80211/wlan_auth.ko (802.1x authenticator)

net80211/wlan_radius.ko (RADIUS client for 802.1x authenticator)

net80211/wlan_acl.ko (MAC ACL support for AP operation)

ath_rate/sample/ath_rate_sample.ko (SAMPLE rate control)

The first three files must be loaded manually. The remaining ones are loaded by the WLAN modules when needed. The necessary modules are loaded when an Atheros device is recognized, although the exact procedure varies from system to system.
We also faced problems during module loading. One of our common problems was related to the loading of modules "ath_pci.ko" and "wlan.ko", which produced different errors in different systems, such as:

---

modprobe wlan and # modprobe ath_pci we had:

WARNING: Error inserting wlan

(/lib/modules/net/wlan.ko): Unknown symbol in module, or unknown parameter

FATAL: Error inserting ath_pci

(/lib/modules/net/ath_pci.ko): Unknown symbol in module, or unknown parameter

---

The solution was to rebuild the kernel since this error generally means that our kernel was built with CONFIG_MODVERSIONS set, but the Madwifi modules got compiled

with different options. But it might also happen after building and installing Madwifi if modprobe does not find the modules. We had the same problem in one of our machines; the reason is that on some distributions the running kernel has another name as the installed kernel source. Madwifi takes information from the installed kernel source. If this is not correct, it can happen that the modules are installed in the wrong location. To prevent this sort of strange behavior from occurring, we should set the KERNELRELEASE environment variable before running make: export KERNELRELEASE=`uname -r`

We also faced some other problems, which fortunately were easier to fix. These problems and their solutions are as follows:

It was necessary to make KERNELPATH=/path/to/your/kernel/source, this makes the kernel source for our running kernel installed. Some kernels seem to lack crypto support, which is an option CONFIG_CRYPTO in kernel .config file (AES support is used if present, otherwise the AES-CCMP cipher module falls back to a private implementation) [CHIV-2].

After all, we could make the cards work in the laptops and with some limitations in the PCs. The PCMCIA card was working perfectly in the laptop in the three modes, 11a, 11b, and 11g, as well as "managed" and "master" modes.

The problems we were not able to solve were:

In the PCI card the mode 11a does not work in the 5 GHz range. The problem is related to the "regdomain" value on the dwl-ag530 card (18). Apparently D-link created their own value instead of using the FCC value (16). Originally, Madwifi used to fail with this value but something changed along the way and it now will give the public safety

channels (4,9GHz) only but none of the valid 5GHz channels (b/g channels seem unaffected). As result, users of D-Link cards, which have a regdomain of 0x18 in their EEPROM, seem to be unable to use any of the 11a channels with the current HAL release. Madwifi filed that issue at http://madwifi.org/ticket/298 to forward it to Atheros. There is no solution available at this time. We most probably need to wait until Atheros provides a new HAL with either a work-around or a fix for this particular problem.

The other problem that made us to give up continue working with this card was that the drivers make the computer to crash when the network load increases. This problem also happens "randomly" on normal traffic.

After all this steps we had:

a) Two cards IEEE 802.11b (the Orinoco Gold 802.11b wireless card, and Linksys 802.11b both PCMCIA) fully functional under Linux. b) Two more cards (DWL-AG660 PCMCIA and DWL-AG530 PCI) working with reduced functionality.

Chapter 5

Experimental Setup, Results, and Discussions

This chapter describes the test methodologies and results. The test methodology is first explained followed by the test cases and the results.

5.1 Test Methodology

All tests were carried out in a typical indoor environment. As we explained before (Section 4.3, Figures 4.6(a) and 4.6(b)), part of the tests took place inside the Broadband and Wireless Research Laboratory of the School of Information Technology and Engineering (University of Ottawa). This is a typical schoolroom with several workstations, office furniture and people working in it. The second set of tests took place in the corridor outside the previously mentioned laboratory. The corridor is made of concrete walls with a large glass window on one end. This hallway is fairly long, measuring 35 meters, and ends at a 90 degrees turn that connects to a second corridor, whose length is 2.5 meters. This is depicted in Figure 5.2. One of the computers of our test system was placed at one end of the corridor (the farthest from the window) and the other computer was placed at various locations along the corridor.

We divided the tests in several sets, which are detailed below (Section 5.2). Each test was repeated 10 times and in each time 100,000 packets were transmitted. This large amount of samples let us ensure statistical correctness in our results. We wrote several

custom-made C programs and shell scripts in order to simplify tasks such as network card configuration, traffic generation, data collection and analysis.

In this study we assess the performance of wireless systems in terms of throughput, delay, and packet losses since the system's performance ultimately depends on these metrics. To have a broad view of the functionality and the performance of the systems implemented in our testbed, we considered the two commonly used transport layer protocols UDP, and TCP.

We had to take into consideration that the propriatry algorithm, for dynamic rate adaptation and automatic fallback could lead to bandwidth thrashing as pointed out in [CHV-1] and [CHV-2]. These algorithms for rate adaptation might be implemented in the NIC itself (if they have an on-board processor) or be left as a task to be executed by the operating system. This situation occurs when a wireless sender has to resend a frame because it is corrupted, and does the resending with such a high data rate that basically causes bandwidth trashing. This way, the retransmitted frame is also likely to be corrupted; therefore wasting valuable bandwidth by causing unnecessary retransmissions. In order to observe the system's performance without bandwidth trashing, the system was tested at limited data rates. But in order to also observe the system's performance with all the implications that use of the 802.11 standard implies; some tests with unlimited data rate were carried out.

Limited data rate means that the maximum data rate was set to be less than what is allowed by the standard whereas unlimited means that the system can transmit at the maximum possible that the conditions of signal strength and noise would allow. In Linux we could easily control this condition as follows. Unlimited data rate can be achieved

through the use of the auto mode, which automatically selects the bit-rate mode (fallback to lower rate on noisy channels), which is the default for most cards. Linux drivers also allow specifying a fixed data transmission rate and we can also specify a bit-rate value and "append auto", so the driver would use all bit rates lower than or equal than this value. For instance, to set the data rate of our AP for 11b operation we used the command: "iwconfig eth0 rate 11auto".

5.2 Description of the experiments and the followed procedure

The first set of experiments had as objective to identify the proper operating range of the system. In order to assess the system's performance with different data rates, we considered different combinations of packet sizes and packet transmission rates. We considered the following packet sizes (which are generated at the network layer) small (100, 500) medium (1000) and large (1500), with all packet sizes mentioned, being in bytes. The value of the packet transmission rate is set by considering the packet size and the desired data rate (Tables: 5.1, 5.2, 5.3 and 5.4).

pk size (bytes)	speed (packet/s)	rate (Mbps)
100	625	0.5
100	1875	1.5
100	3125	2.5
100	4375	3.5
100	5625	4.5
100	6875	5.5
100	8125	6.5
100	9375	7.5
100	10625	8.5
100	11875	9.5
100	13125	10.5

Table 5.1

pk size (bytes)	speed (packet/s)	rate (Mbps)
500	375	1.5
500	625	2.5
500	875	3.5
500	1125	4.5
500	1375	5.5
500	1625	6.5
500	1875	7.5
500	2125	8.5
500	2375	9.5
500	2625	10.5
500	2875	11.5

Table 5.2

pk size (bytes)	speed (packet/s)	rate (Mbps)
1000	**188**	1.5
1000	**313**	2.5
1000	**438**	3.5
1000	**563**	4.5
1000	**688**	5.5
1000	**813**	6.5
1000	**938**	7.5
1000	**1063**	8.5
1000	**1188**	9.5
1000	**1313**	10.5
1000	**1438**	11.5

Table 5.3

pk size (bytes)	speed (packet/s)	rate (Mbps)
1500	**125**	1.5
1500	**208**	2.5
1500	**292**	3.5
1500	**375**	4.5
1500	**458**	5.5
1500	**542**	6.5
1500	**625**	7.5
1500	**708**	8.5
1500	**792**	9.5
1500	**875**	10.5
1500	**958**	11.5

Table 5.4

The packet transmission rate is calculated as follows: In order to maintain a constant rate with different packet sizes we have to take into consideration that the product of packet size times the number of packets per second must remain constant (i.e., we may increase packet size, but we have to reduce the packet transmission rate accordingly, as to keep the number of bits per second constant). Therefore, our settings in the traffic generator were based on the tables shown above, where the second column of the table was calculated according to the specific values of packet size and desired data rate as described below:

$$packet\_generation\_rate = \frac{desiered\_data\_rate}{packet\_size * 8} \tag{5.1}$$

Where the data rate is measured in bits per second, the packet generation rate is the number of packets generated per second and the packet size is measured in bytes. As an example, for a packet size of 100 bytes and a target data rate of 1Mbps the required packet generation rate is X = 1000000/(8*100) = 1260 [packets/s] as shown in the corresponding table. Please note that what we call "packet size" is the PDU size at the network layer including the header, not only the payload.

Although the amount of possible variations in position are endless, we chose to place our equipment in the most sensible area of interest, which is in the middle and run a large number of tests and got the average, in order to take into account possible variations in our measurements.

In these tests we used a standard setting using an AP and a wireless station, as shown in Figure 5.1. We used UDP and TCP traffic streams and the experiment was performed with a 5-meter distance between the two stations. Objective of this test was to acquire more information regarding the performance of our system and to find out which is the maximum throughput of the system under "perfect" conditions. The points where data were collected are specified in Figure 5.1

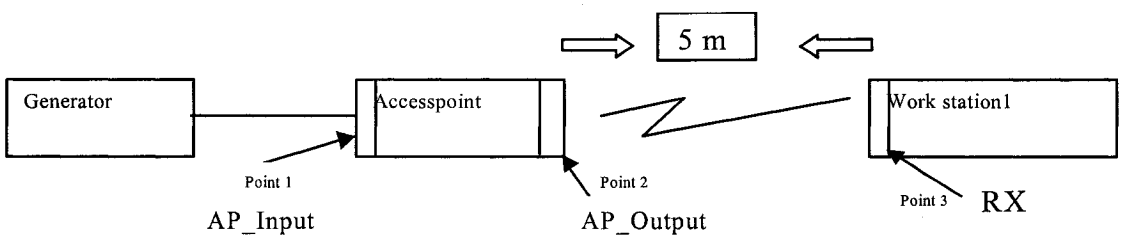


Figure 5.1: Experimental set-up used for experiments b-1c

As we mentioned earlier, the objective of this experiment is to identify the proper operating range of the network system (i.e., appropriate throughput) that can be obtained through the network between two stations, when there is no interference from other stations. We did not have any spectrum analyzer to ensure there weren't any other wireless devices operating, so we tried to ensure that our results were the least affected by interfering stations. Thus we worked at late hours, when the chance of having others use wireless access was less, we coordinated with other researchers, working with wireless LANs related experiments, and also we "checked" before starting and during the experiments if there was any other network "broadcasting" itself (even though, for higher security, APs don't have to advertise themselves, when security is used).

The second set of experiments is an experimental performance evaluation of UDP over 802.11 where we tested how various transmission impairments affect the performance including distance, walls, windows and corners (in the corridor of a building). The evaluation scenario is shown in Figure 5.2.

The set of tests b-1c were performed at 5, 18, and 35 meters of distance with line of sight between the two antennas, except the one at the spot labeled as b-1c4 in the Figure 5.2.Besides the line of sight between the antennas, we also took into consideration some ground clearance for the antennas. The laptop antenna was kept constant at 1 meter above ground level, and the AP antenna was also kept constant at 10 cm above the ground level. Moreover, both antennas were kept horizontal during the test. Besides the distance, these last three factors mentioned are important enough to affect on the signal strength. The AP used standard settings. The obtained results are shown and discussed in the following Section.

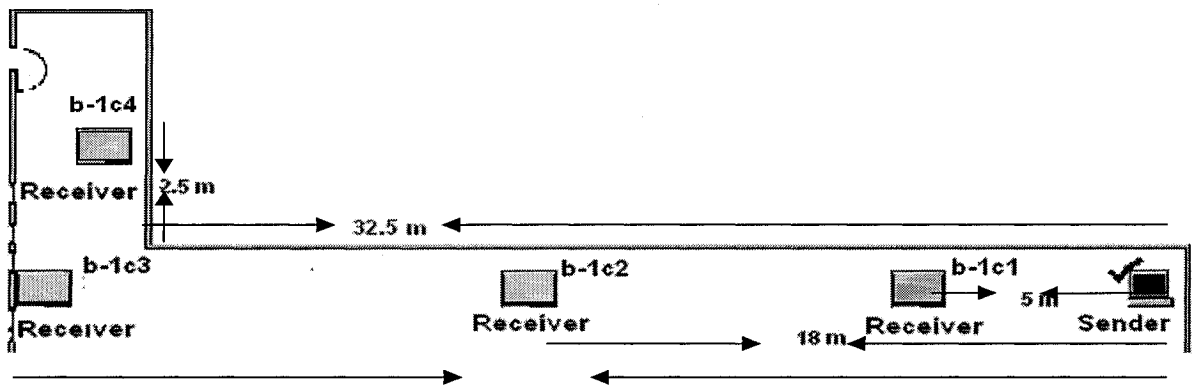


Figure 5.2: Experiments b-1c

5.3 Results and Discussion

In this Section we present the results that were obtained through the data collected during previously described tests.

It is important to mention that although the presented results can provide a very useful insight on the performance of similar wireless devices, some differences are expected to occur among different vendors or even among different models of the same maker. Nevertheless, judging from our experience, these differences are not significant. For example, we tried to compare the performance of the system when: 1) Orinoco and 2) D-

Link cards were used at the receiving node. We observed that although it seems that Orinoco gives a better performance than D-Link, the differences were rather marginal and the results were relatively close to each other. On the other hand, processor speeds of the host computers turned out to be a more significant factor in the system's performance. For instance, we realized that a laptop with Pentium 4 processor had a sensibly better performance than Pentium Mobile. This is likely due to the fact that the Pentium Mobile is a low-power version and therefore it is usually slower.

Through this section we show the achieved TCP/UDP performance in different scenarios that were specified in Section 5.2.

5.3.1 Experimental performance evaluation of UDP over 802.11b in an optimal environment

- Maximum performance of the system

The purpose of this set of experiments is to find out the maximum achievable throughput of our system in order to identify the proper parameter settings in which our system can operate. We decided to do this in order to avoid including in our results the effect of other factors not related with wireless transmissions, such as processor speeds of the host computers. Once the proper settings had been determined, all the tests were performed within this operating range, in order to observe the effect of various transmission impairments, such as distance and corners on the selected performance metrics (i.e. delays and packet losses).

In order to determine the capacity of the wireless link, first we had to determine at what point the transmission capability of the sender starts overflowing. For this purpose, we sent packets at different rates and checked the amount of losses. In order to determine when losses are occurring, we compared the rate at which UDP packets are entering the Access Point (Point 1 in Figure 5.1) and the rate at which UDP packets are passing the output interface of the AP (Point 2 in Figure 5.1). We started

by sending packets at a low rate and we increased this rate in steps of 1 Mbps, until losses started to occur. This way, as we can see in Figures 5.4a, and 5.4b, we could determine the maximum throughput that can be sustained without losses due to the wireless link capacity.

Figures 5.3 (a) and 5.4 (a), (b) present the results acquired through the measurements in terms of throughput and losses. These measurements were taken at the points shown in the Figure 5.1. The term "Forwarding loss" indicates the loss between Point 1 and Point 2, while "Wireless loss" indicates the loss between Point 2 and Point 3. In Figures 5.3(a1), (a2), (a3) and (a4), APInput refers to Point 1 in Figure 5.1, and the same for APOutput and RX is the receiver input, which refer to Points 2 and 3 respectively. To ensure the results were not dependent on the particular computer that was used, we switched the role of computers in these tests ((the transmitter became the receiver and vice versa). The results remained practically unchanged.
Figures 5.3 (a1), (a2), (a3), and (a4) present the results in terms of throughput for different packet sizes. The curves clearly show where the AP starts overflowing because the transmission capacity of the AP overflows. Figures 5.4(a1), (a2), (a3), (a4), and (b) show the effect on packet losses for different packet losses.

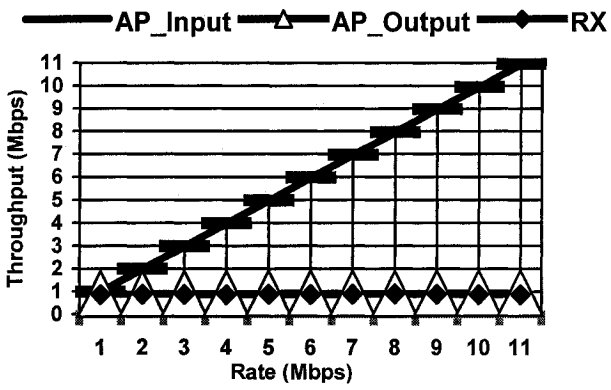


Figure 5.3(a1): Throughput Pk_Size100(bytes)

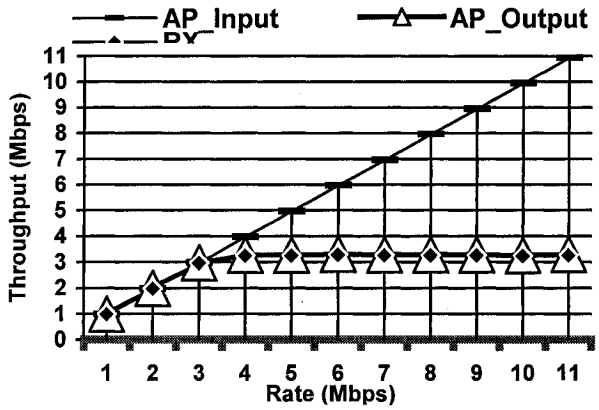


Figure 5.3(a2):Throughput Pk_Size500(bytes)

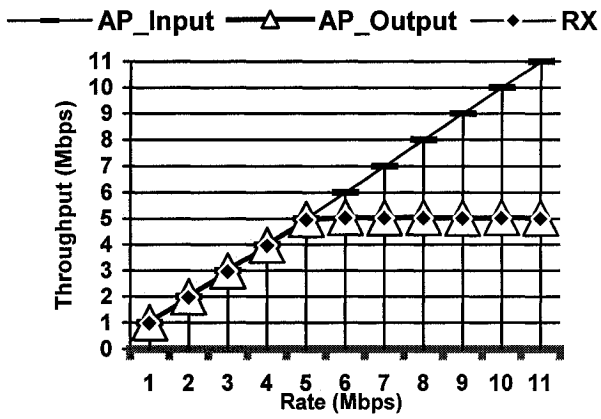


Figure 5.3(a3): Throughput Pk_Size1000(bytes)

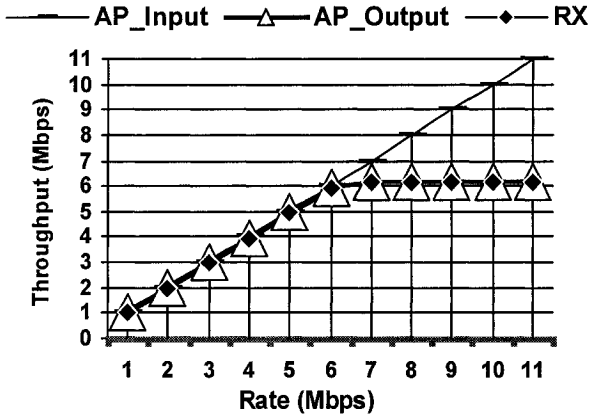


Figure 5.3(a4): Throughput Pk_Size1500(bytes)

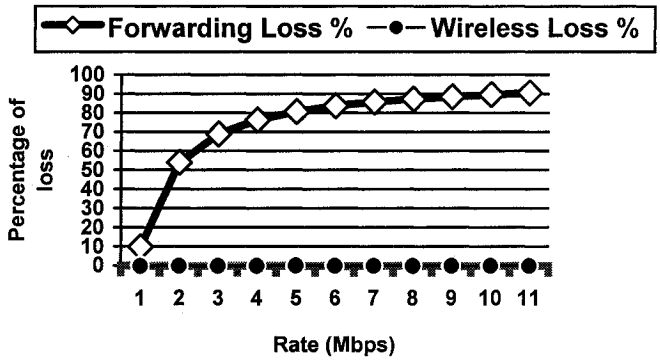


Figure 5.4(a1): Percentage of loss PkSize100(bytes)

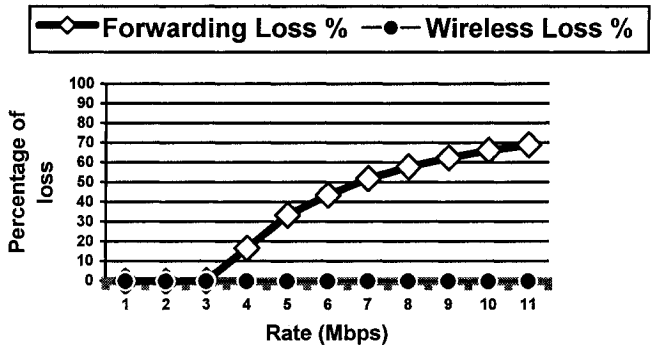


Figure 5.4(a2): Percentage of loss PkSize500(bytes)

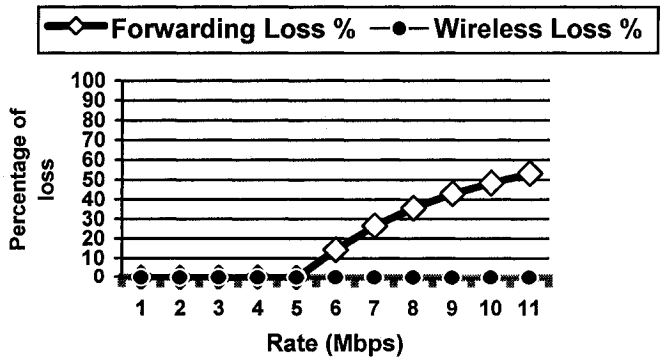


Figure 5.4(a3): Percentage of loss PkSize1000(bytes)

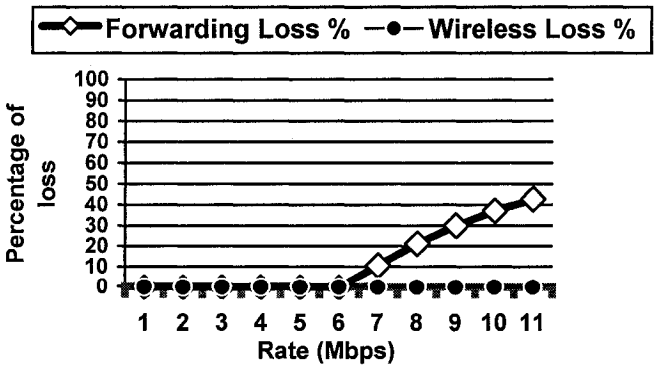


Figure 5.4(a4): Percentage of loss PkSize1500(bytes)

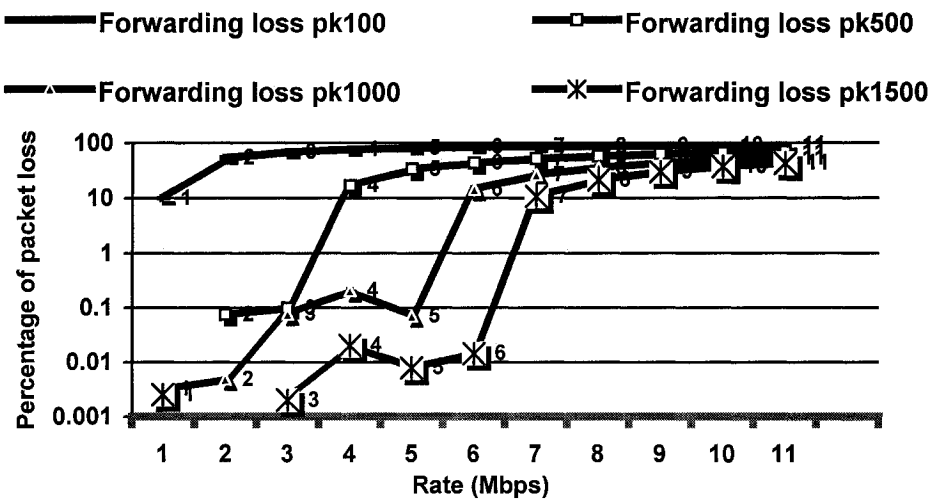


Figure 5.4(b): percentage of packet loss(log scale)

From the tests previously described and the corresponding results we can determine when performance starts to diminish because the output interface of the sending station starts overflowing. From the figures previously shown we can clearly appreciate which are the optimum packet transmission rates for this system. These values are summarized in Table 5.5 shown below

pk Size (bytes)	speed (packets/s)	rate (Mbps)	Expected packet interdeparture time (msec)
100	625	0.5	1.14
500	625	2.5	1.6
1000	563	4.5	1.7762
1500	458	5.5	2.1834

Table 5.5: Maximum rate that could be supported without buffer overflowing occuring at the Internet layer

From the presented results, we can identify the maximum achievable throughput at the network layer of this system. Figure 5.3(a), clearly indicates that no matter how much traffic is offered to the system, the maximum achievable throughput at the network layer

is limited by the wireless link to about 6 Mbps. This value repersents the information throughput, and of course it is less than 11 Mbps.

In fact, 11 Mbps is the rate that can be achieved at the physical layer by a specific combination of modulation and coding with high enough SNR. This means that under this scheme, when a station transmits, it does it at 11Mbps, but is does not do that continuously. The effective data rate is decreased because of the efficiency of protocols at the data link and network layers. For instance, the MAC layer establishes some silences that are used to coordinate channel access among diverse stations, a wireless station has to stop and wait some time before accessing the channel again reducing the effective throughput. We should also recall that in the wireless 802.11 each MAC frame becomes the payload part of a physical layer frame (PLCP). The header of the physical layer frame is always transmitted at 1 Mbps, and one of the fields in the header indicates which is the signaling used to transmit the payload part (the MAC frame). Therefore, the 11 Mbps is achieved at the physical layer but only one part (the payload containing the MAC frame) of the physical frame is transmitted at such rate.

- Effect of traffic load on perceived link quality and traffic pattern

In this Section we show the effect of the traffic loading has on the perceived link quality (at the wireless receiver), interdeparture time (from the AP) and interarrival time (measured at the wireless station). These results are shown in Figures 5.5 (a) to (f).

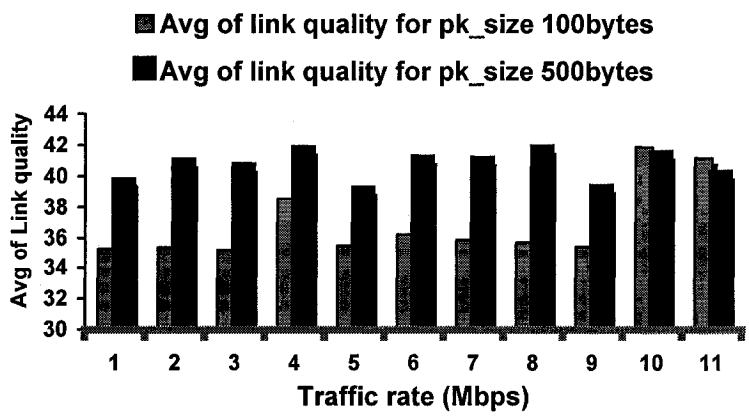


Figure 5.5(a): Average of Link quality versus Traffic rate

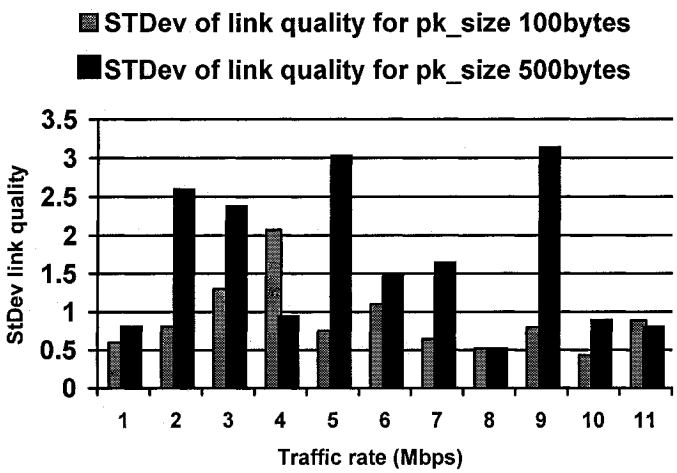


Figure 5.5(b): Standard Deviation of Link quality versus Traffic rate

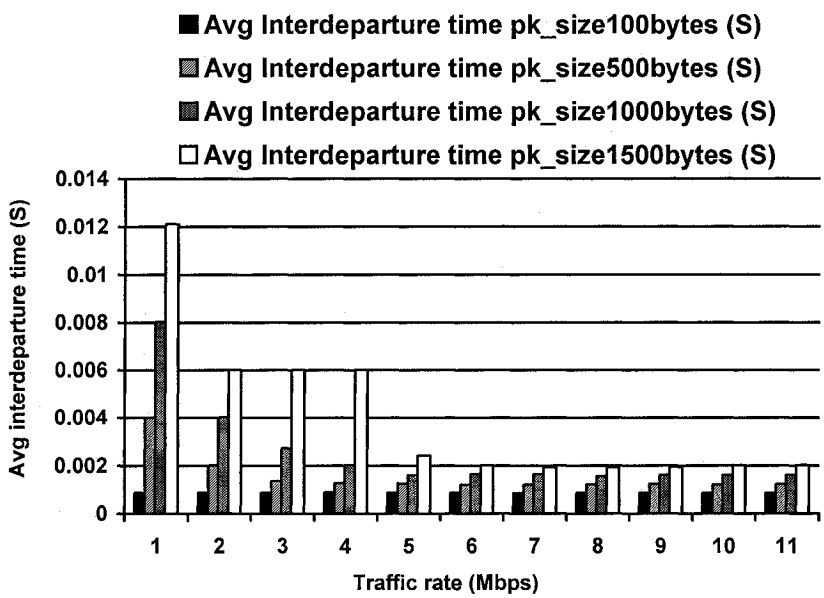


Figure 5.5(c): Average of interdeparture time of packets in the sender station versus traffic loading

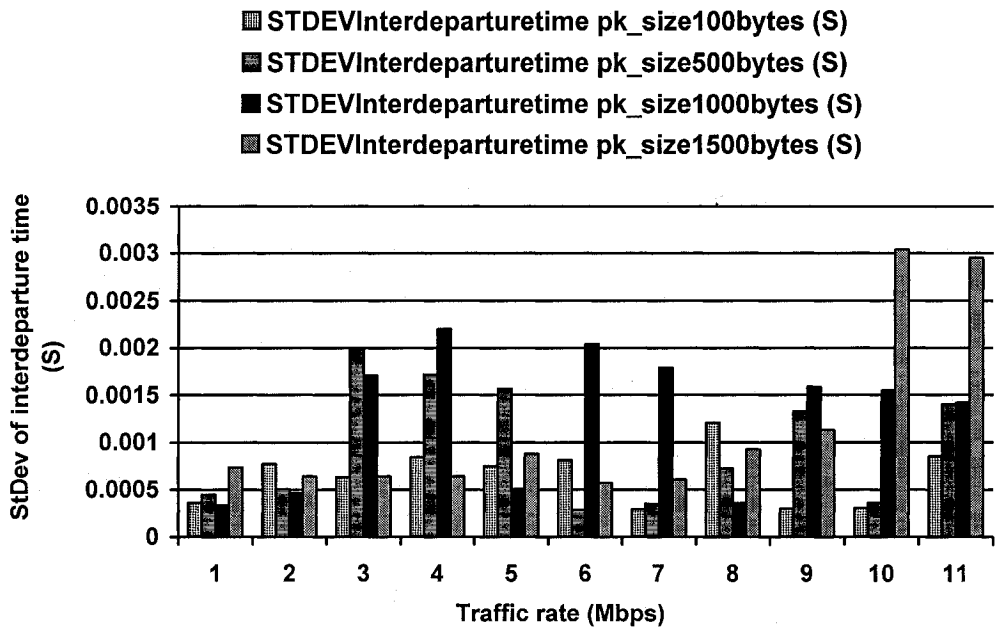


Figure 5.5(d): Standard Deviation of interdeparture time of packets in the sender station versus traffic loading

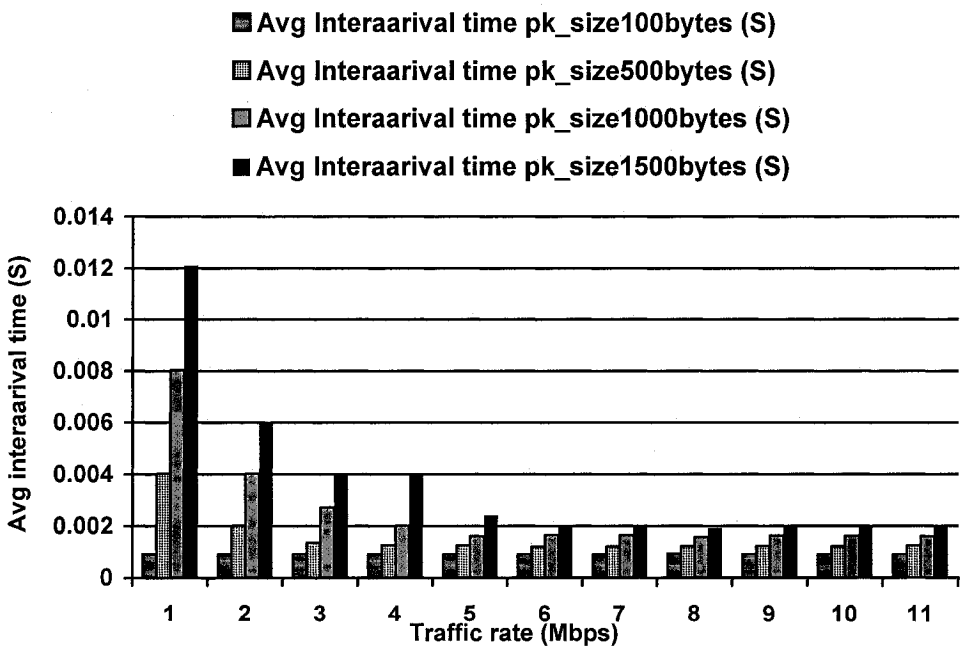


Figure 5.5(e): Average of interarrival time of packets in the receiving station versus traffic loading

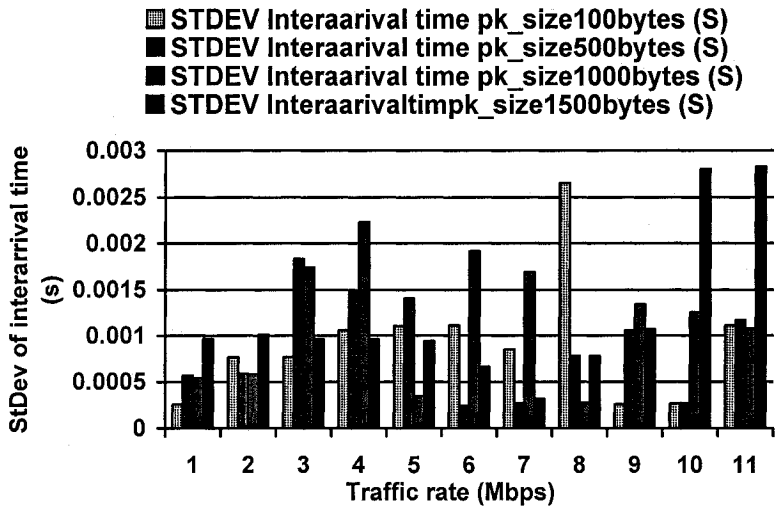


Figure 5.5(f): Standard Deviation of interarrival time of packets in the receiving station versus traffic loading

From Figures 5.5(a) and (b) we can confirm that, in agreement to what we were expecting to occur, the link quality sensed by the wireless network card is independent of the traffic loading. Figures 5.5(c), (d), (confirm that the generator produces CBR traffic since the packets are sent at (almost) regular intervals (there is a small standard deviation). The pattern is not significantly affected by the wireless link as can be deduced from the graphs of interarrival times. However, this is so because the channel conditions are excellent (proximity of stations, absence of sources of interference) as well as there isn't contention with other stations.

Although there are a number of studies on TCP/UDP performance over wireless links (either based on simulations or on measurements), the related papers we found do not allow for a direct comparison with our results due to differences in diverse settings, scenarios or technology. However, some results can be compared with the data provided by Atheros Communications (a chip manufacurer) in their white paper [CHV-4]. In this paper they provide the maximum theoretical rate that can be achieved with UDP under the following settings: packet size 1500 bytes, zero packet errors and maximum channel bandwidth (i.e., close operating range). For UDP over IEEE 802.11b they computed a maximum theoretical value of 7.1 Mbps. Under similar settings we experimentally obtained a maximum of 6.2 Mbps as seen in a Figure 5.3a4. This decrease in performance

can be explained by a number of factors such as packet losses and variations on the channel capacity, which were not considered in the theoretical computation.

- Other performance measurements

All tests described in this section were carried out using the values shown in the Max-Throughput table (Table 5.5). This ensured that the system was operating within a meaningful[5] traffic load. For example, tests with a packet size of 100 bytes indicates that the test was carried out at a rate of 0.5 (Mbps) with a packet generation rate of 625 (packets/s) as shown in the table.

In the following pages we provided the graphs of Probability Density Functions (PDF) and Cumulative Distribution Function (CDF) for tests carried out at the rates indicated by the table of Maximum Throughput (Table 5.5). Recall that each test was repeated 10 times and in each time 100,000 packets were transmitted. We are providing these statistics for the quality channel indicator and for the packet interarrival/interdeparture times. These measurements are the outgoing traffic at the internet layer of the access point and incoming traffic at the internet layer of the receiving station).
We also provided Cumulative Distribution Function (CDF) describing the packet delays at the wireless network card. As we explained earlier in 4.3.2, the NIC delay indicates how fast the low-level NIC hardware can transmit the packet.
Figures 5.6 (a) to (d) show that the perceived link quality does not vary too much when using different packet sizes. However, there is a slight tendency to perceive a better channel quality with smaller packet sizes. This seems to be due to the fact that the wireless cards keep the same rate during a packet transmission. Therefore, rate adaptation algorithms can track more closely the channel conditions if packets are small. Figures 5.9a to 5.9d show the statistics of the delay at the NIC is independent of the packet size.

[5] We use the term "meaningful" in the context that whatever losses or other impairments might be occurring during the experiments, they can't be contributed to sending traffic above the level, the "channel" (and the system as a whole) can support under ideal conditions (no losses due to channel impairments, no losses because of competing nodes).

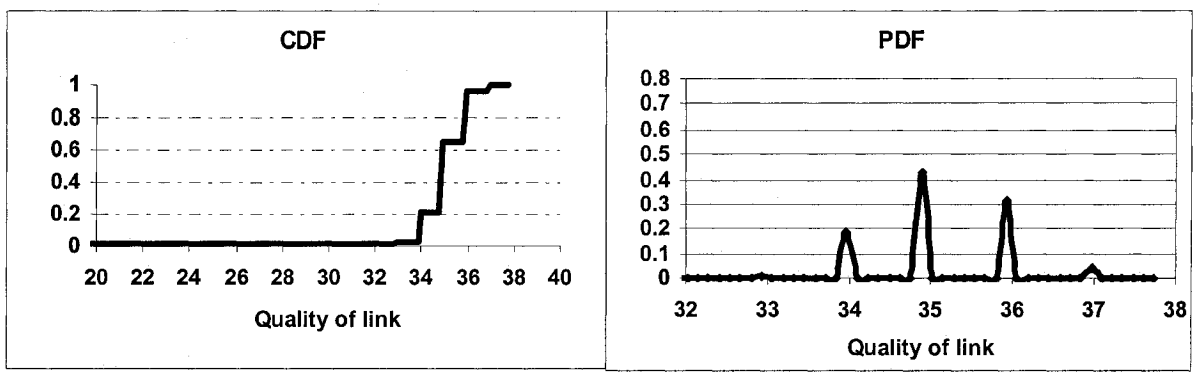


Figure 5.6(a): CDF and PDF of Quality of link, UDP traffic Packet Size 100 bytes

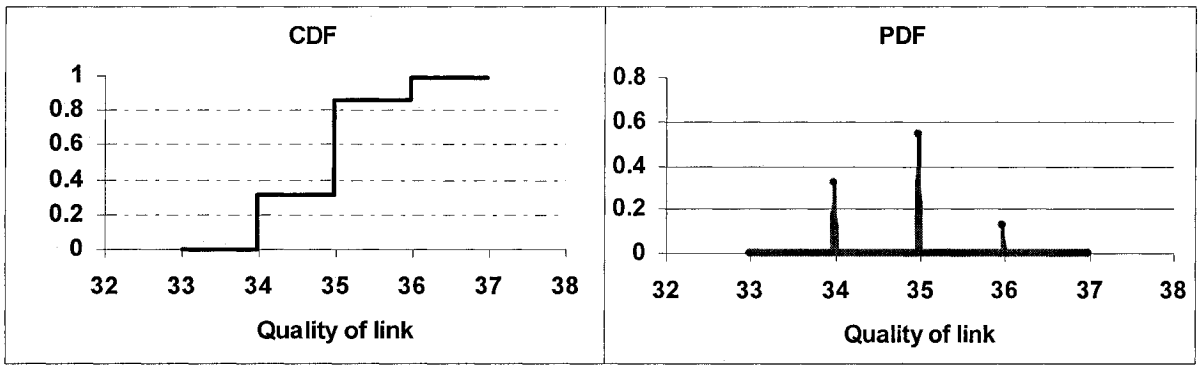


Figure 5.6(b): CDF and PDF of Quality of link, UDP traffic Packet Size 500 bytes

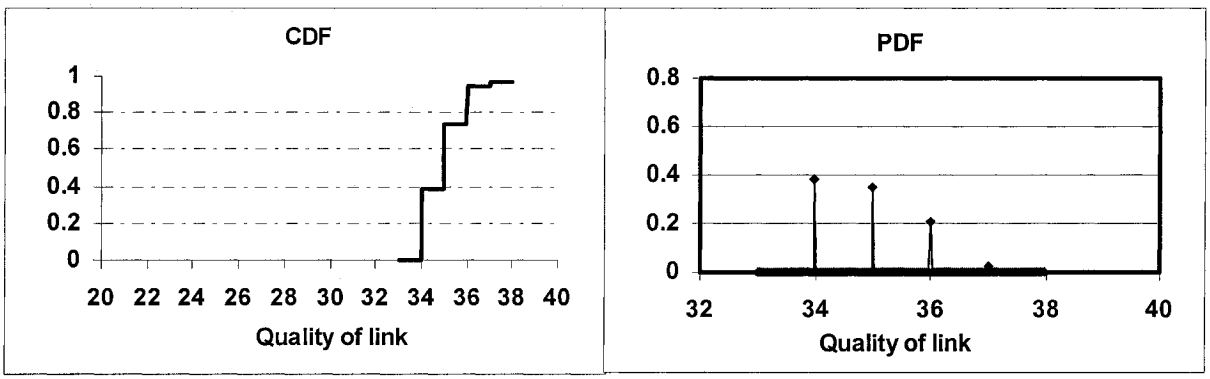


Figure 5.6(c): CDF and PDF of Quality of link, UDP traffic Packet Size 1000 bytes

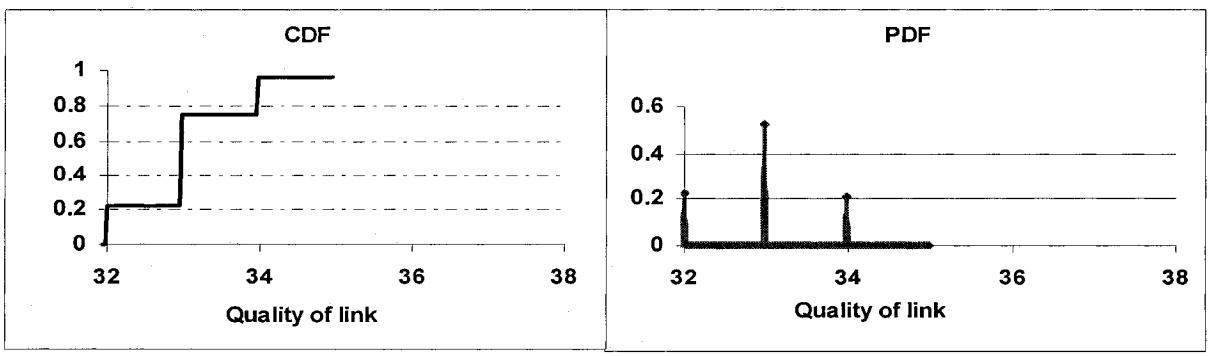


Figure 5.6(d): CDF and PDF of Quality of link, UDP traffic Packet Size 1500 bytes

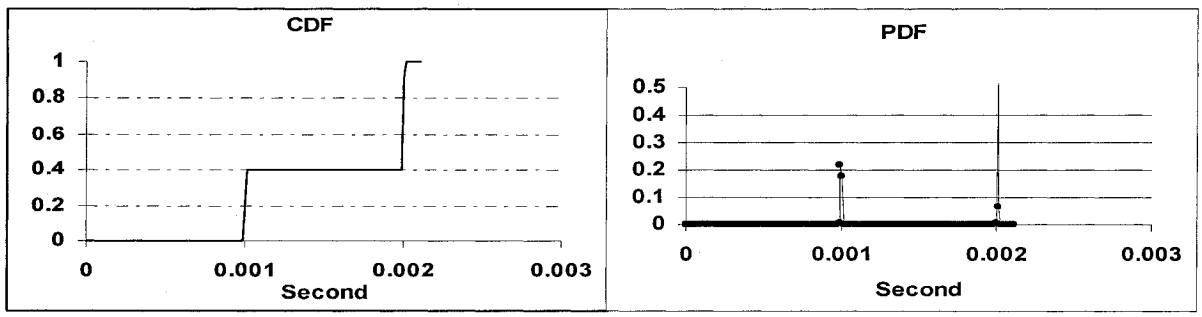


Figure 5.7(a): CDF and PDF of Interdeparture time, UDP traffic Packet Size 100 bytes

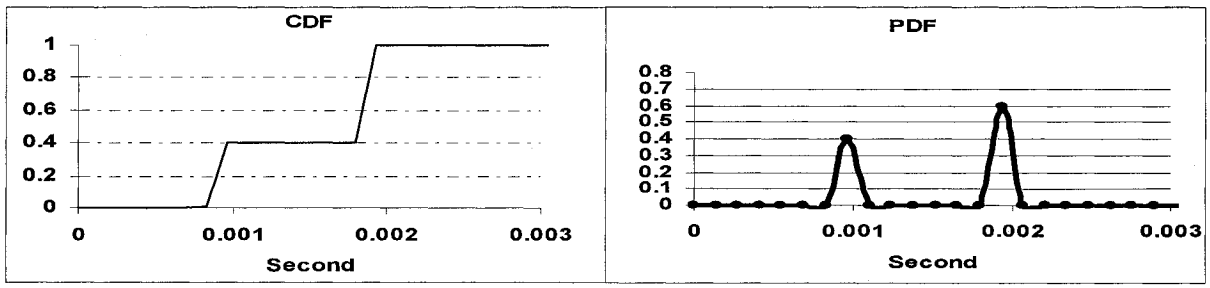


Figure 5.7(b): CDF and PDF of Interdeparture time, UDP traffic Packet Size 500 bytes

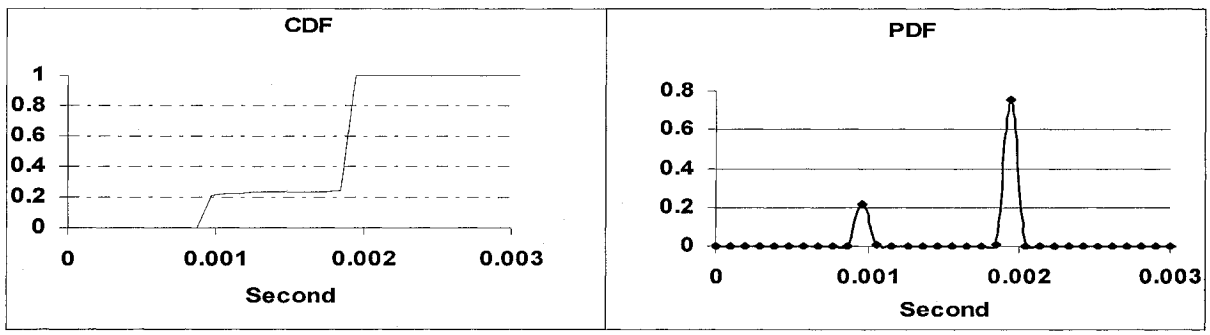


Figure 5.7(c): CDF and PDF of Interdeparture time, UDP traffic Packet Size 1000 bytes

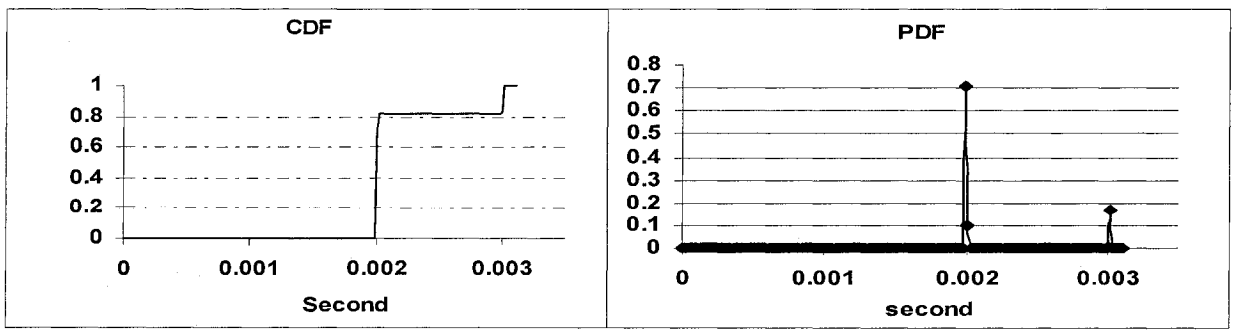


Figure 5.7(d): CDF and PDF of Interdeparture time, UDP traffic Packet Size 1500 bytes

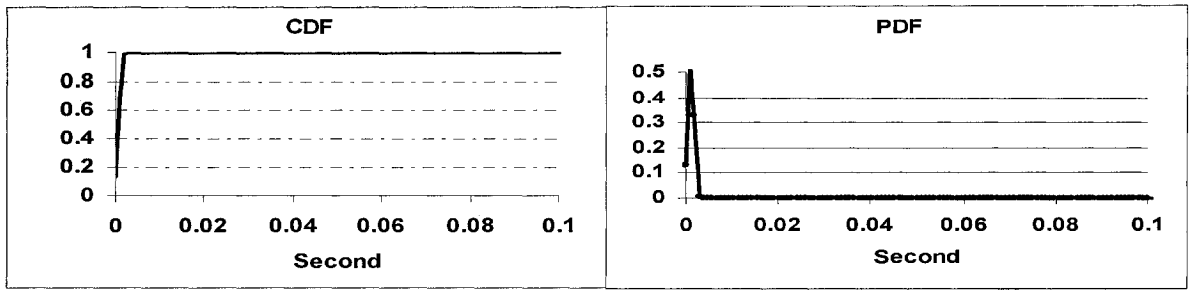


Figure 5.8(a): CDF and PDF of Interarrival time, UDP traffic Packet Size 100 bytes

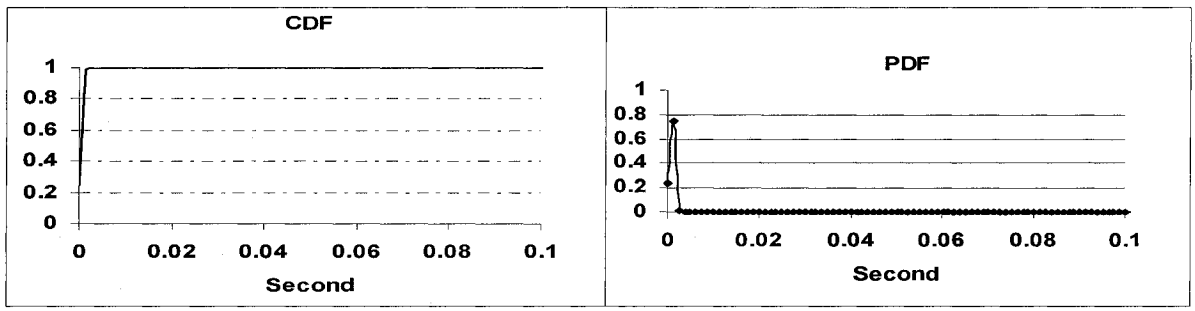


Figure 5.8(b): CDF and PDF of Interarrival time, UDP traffic Packet Size 500 bytes

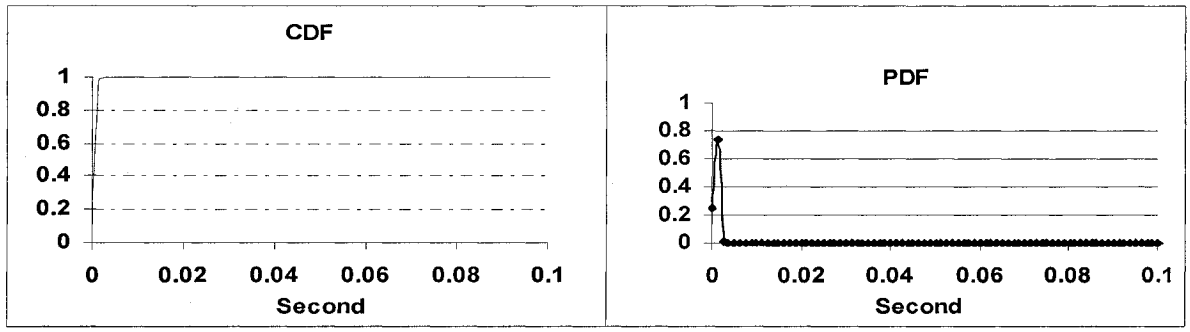


Figure 5.8(c): CDF and PDF of Interarrival time, UDP traffic Packet Size 1000 bytes

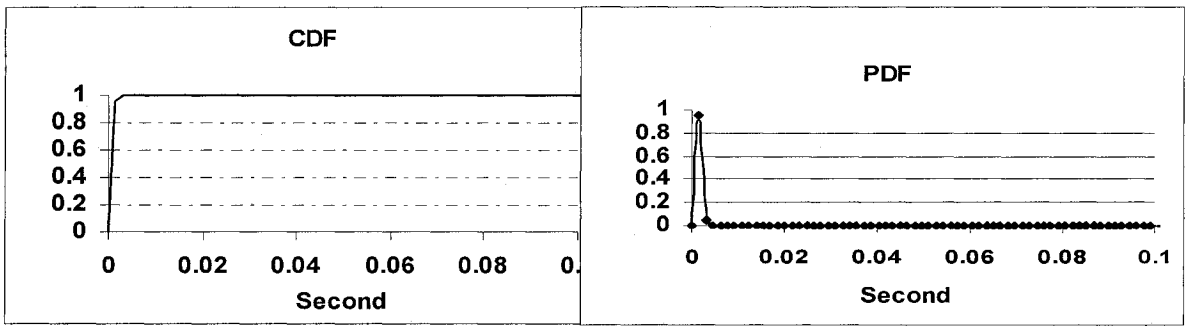


Figure 5.8(d): CDF and PDF of Interarrival time, UDP traffic Packet Size 1500 bytes

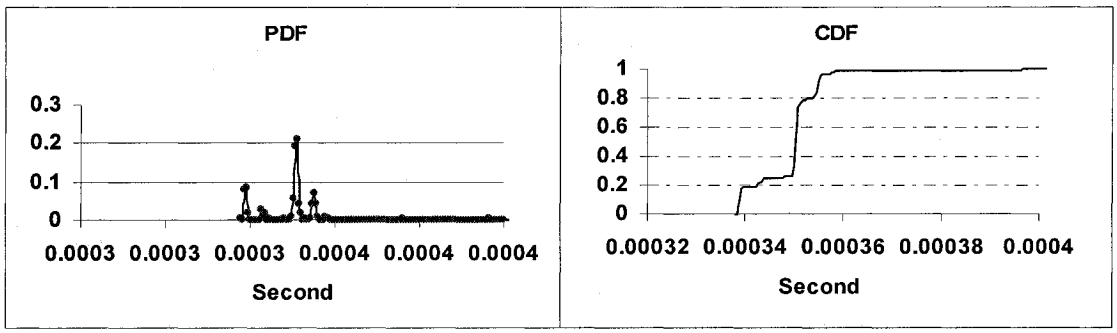


Figure 5.9(a): CDF and PDF of packet delay, UDP traffic, Packet Size 100 bytes

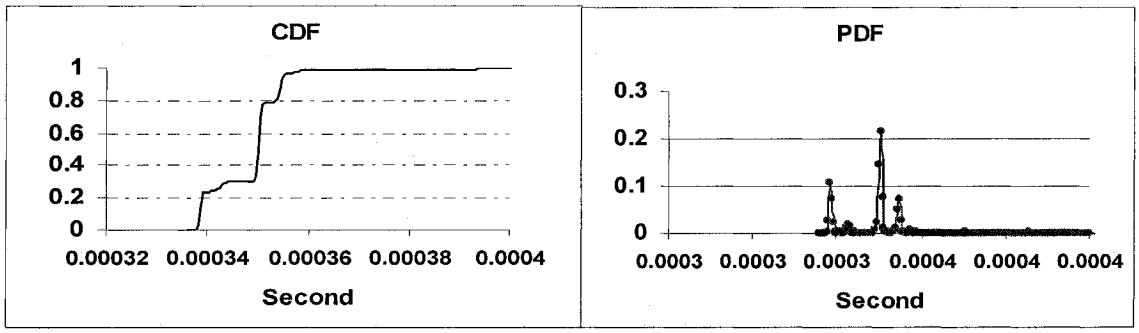


Figure 5.9(b): CDF and PDF of packet delay, UDP traffic, Packet Size 500 bytes

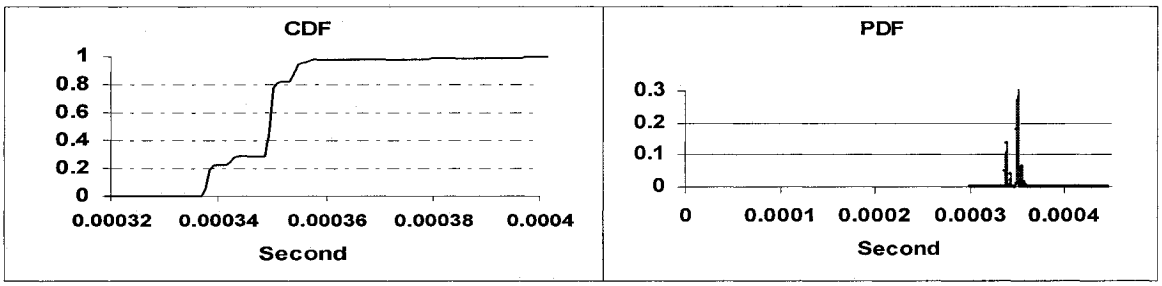


Figure 5.9(c): CDF and PDF of packet delay, UDP traffic, Packet Size 1000 bytes

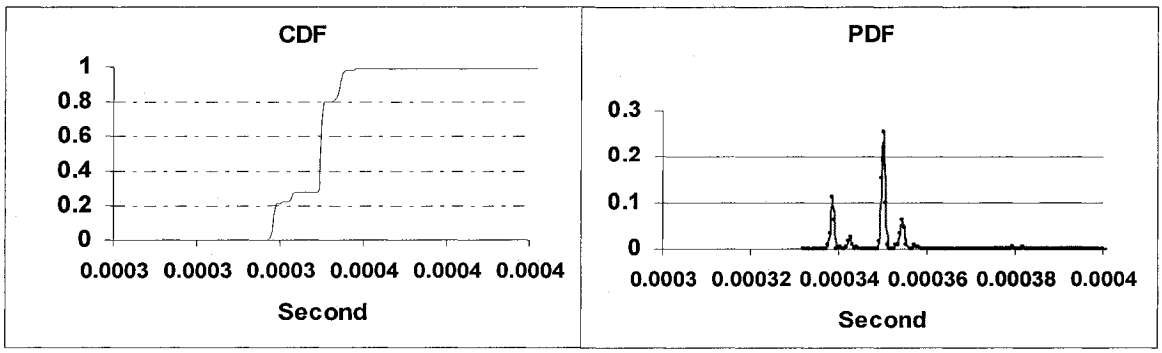


Figure 5.9(d): CDF and PDF of packet delay, UDP traffic, Packet Size 1500 bytes

5.3.2 Experimental performance evaluation of TCP over 802.11b under perfect channel and network conditions (inside the lab)

In this section, we study the performance of TCP with a wireless link. An important feature of TCP is its congestion control algorithm, which is essential for preserving network stability.

We study the performance of TCP in terms of throughput and goodput. In order to generate TCP sessions we ran file transfers using FTP since it uses TCP as its transport protocol. In order to simulate run long ftp transfer we transferred large files of 78 Mbytes, and have it run in a promiscuous mode.

In addition to the FTP session, we also used the traffic generator to create UDP cross traffic in order to take part of the available bandwidth away from the TCP connection. For UDP we chose a packet size of 1000 bytes and performed experiments with different volumes of UDP traffic loading.

We produced the graph describing the congestion window for each test. As we made clear in section 3.2.2, the Linux TCP implementation differs from a typical TCP implementation, and we described these differences when it comes to the congestion control. Our tests were performed under the Linux kernel 2.4.20. Linux 2.4 by default has the TCP SACK option enabled. Figure 5.10 shows the slow start phase of the Linux TCP that we were testing.

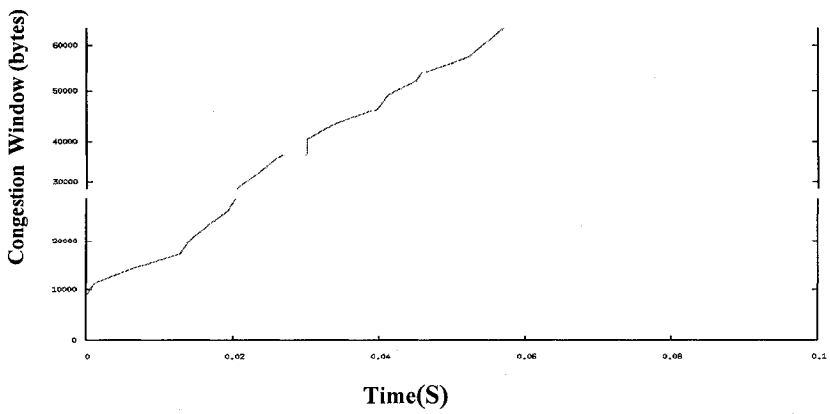


Figure 5.10: Congestion Window Versus Time (s)

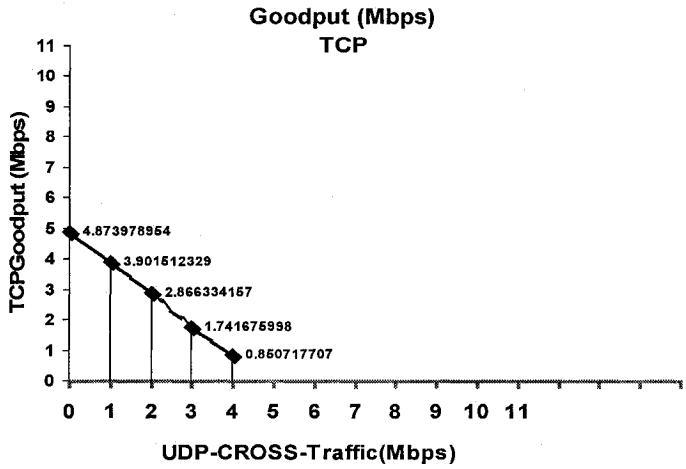


Figure 5.11a: TCP Goodput

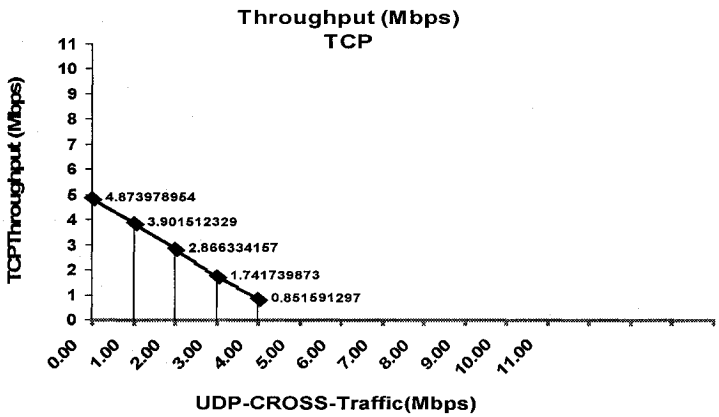


Figure 5.11b: TCP Throughput

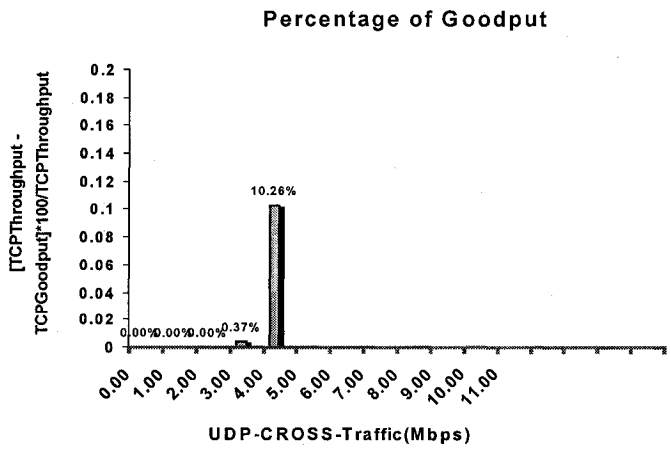


Figure 5.11c: Percentage of Goodput

The following figures are the plots of congestion window at the receiver, when there is just TCP traffic (UDP-Cross traffic is 0 Mbps), and UDP-Cross traffic is 1,2,3,4 Mbps

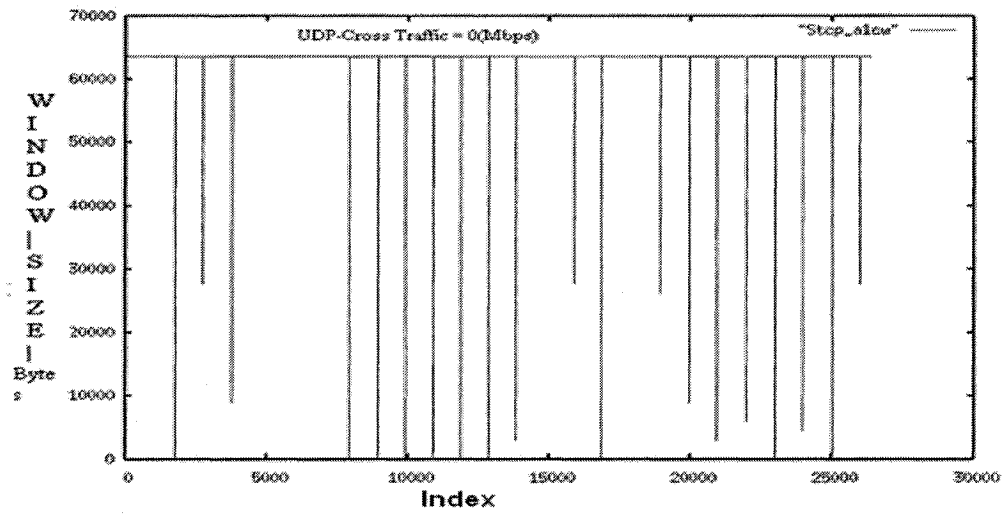


Figure 5.12a: Congestion window, UDP cross traffic=0(Mbps)

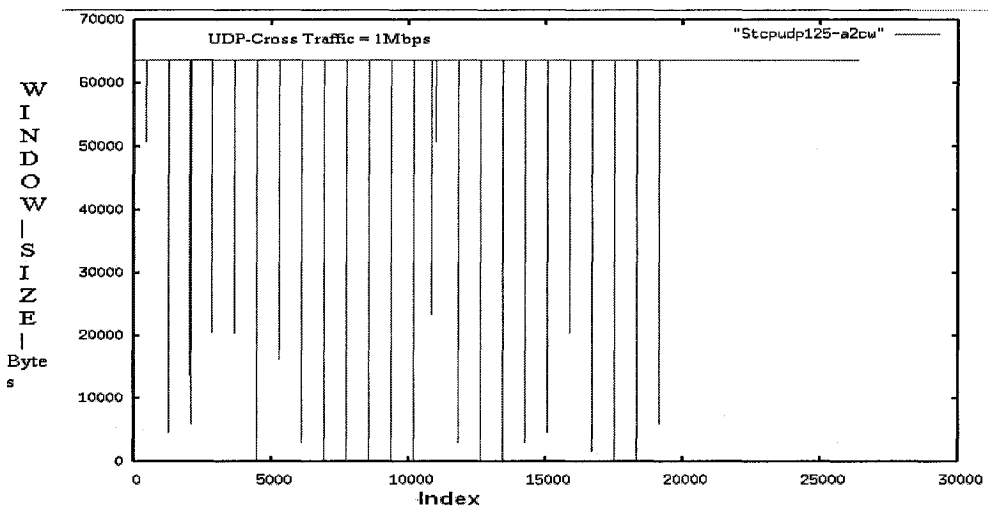


Figure 5.12b: Congestion window, UDP cross traffic=1(Mbps)

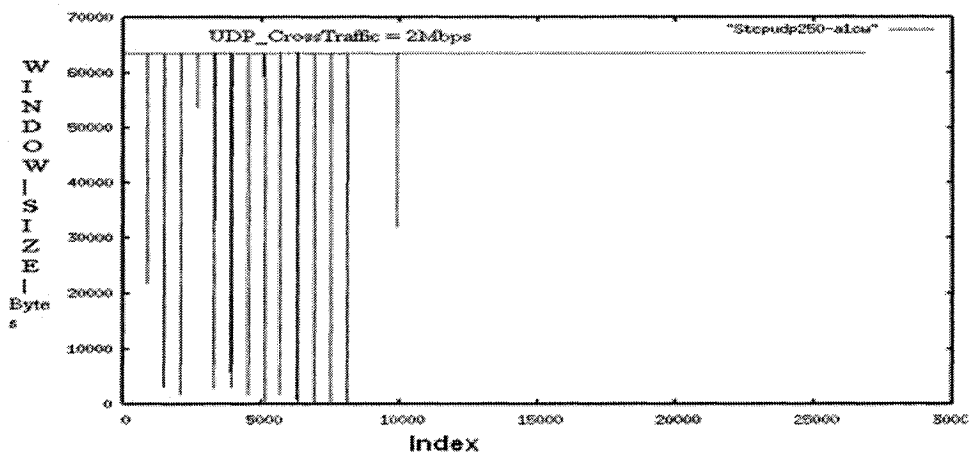


Figure 5.12c: Congestion window, UDP cross traffic=2(Mbps)

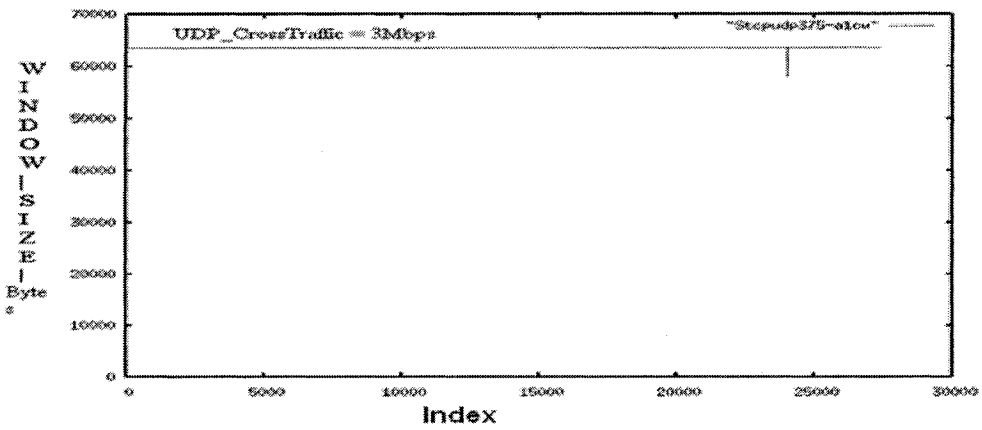


Figure 5.12d: Congestion window, UDP cross traffic=3(Mbps)

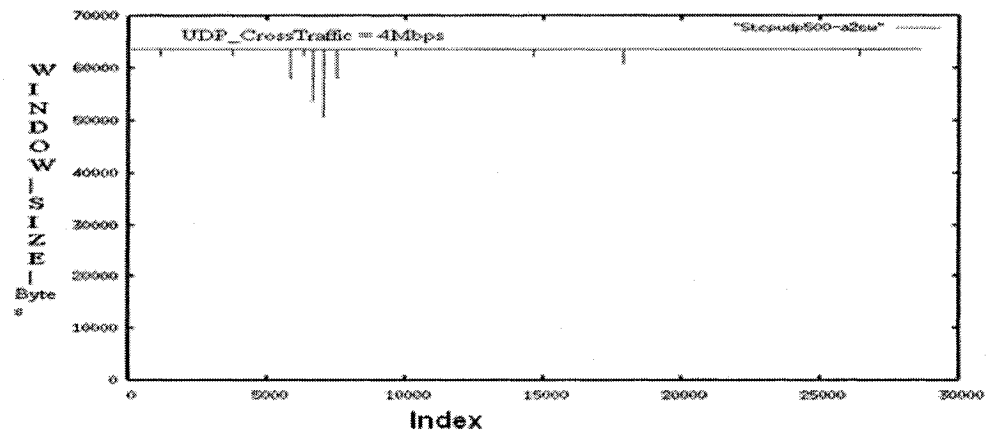


Figure 5.12e: Congestion window, UDP cross traffic=4(Mbps)

According to the figures 5.11a, 5.11b, and 5.11c, when the UDP-Cross traffic rate is 0, 1, 2 (Mbps), there aren't packet losses or retransmissions. At these rates throughput and goodput are equal. Note that the aggregate throughput remains constant for up to 4.9 Mbps approximately.

On the other hand, the congestion window results in Figures 5.12a, 5.12b, and 5.12c show that there are several drops of the window size to its minimum value, with no apparent reason. According to [CHIII-1], "The amount of buffer space that a system advertises depends on how much buffer space it has available at that given moment, which is dependent upon how quickly applications can pull data out of the receive buffer. This in turn is driven by many factors, such as the complexity of the application in use, the amount of CPU time available, the design of the TCP stack in use, and other elements". This means that sometimes the congestion window will decrease, not because a packet loss was detected, but because the receiver cannot pull data out of the input buffer as fast as needed. We observe that the speed of the host computer can play a major role in the performance of wireless stations.

There is one issue worth mentioning, which relates to both, Linux kernel version 2.4 and version 2.6. For paths with a very large bandwidth delay product (BDP) where the TCP window is greater than 20 Mbytes, it is likely to hit a Linux SACK implementation problem. If

Linux has too many packets in flight when it gets a SACK event, it takes too long to locate the SACK packet, and we get a TCP timeout and CWND goes back to 1 packet. Restricting the TCP buffer size to about 12 Mbytes seems to avoid this problem, but clearly limits our total throughput. [CHV-3]. The other solution is to disable SACK[CHV-3]. The above results are with enabled SACK option, because in the Linux kernel version 2.4 it is enabled by default. However, it is extremely unlikely that this problem occurred in our experiments because the propagation delay is small (the stations were placed just 5 m apart), which produces a negligible propagation delay and therefore a low BDP. In addition, the maximum TCP congestion window in our tests was 64 Kbytes as can be seen in the graphs.

5.3.3 Experimental performance evaluation of UDP over 802.11b in a corridor of a school building

These categories of experiments took place in the corridor of CBY (Figure 5.2) in four spots; b-1c1, b1-c2, b-1c3, and b-1c4, and focused on the performance of the wireless system against distance, walls, windows and corners.

- Quality of Link

As we explained earlier in section 4.2.3, "link quality" just measures the degree of correlation between an expected sequence and the received bit stream. Of course there is some degree of relation between SNR and "link quality", since a high SNR means that we will likely receive the right sequence. There is another metric termed "signal level" or just "level" which is meant to give an indication of the signal level in an arbitrary scale which also may change from vendor to vendor. Although this metric is also related to the SNR, there is no standard way to convert these measurements to SNR and the conversion is not usually provided in the open literature.

The following figures show the way "link quality" changes by distance and around a corner.

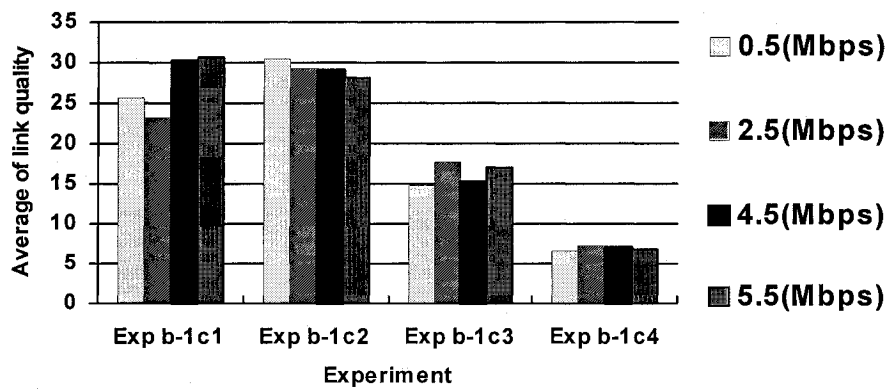


Figure 5.13a: Average of link quality versus experiment for different rates

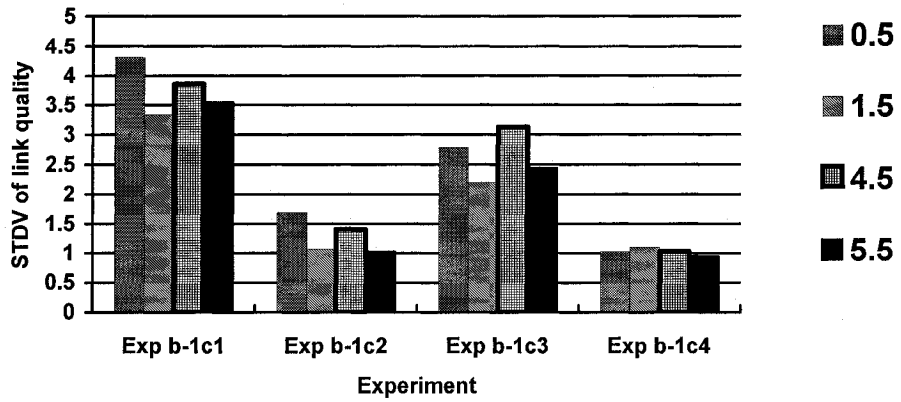


Figure 5.13b: STDV of link quality versus experiment for different rates

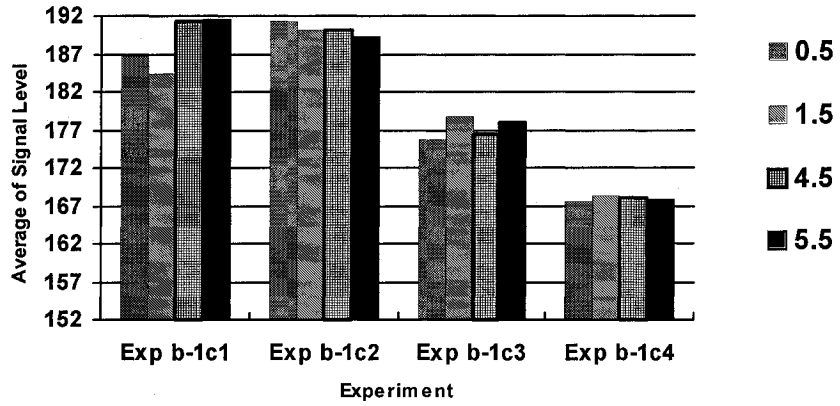


Figure 5.13c: Average of signal level versus experiment for different rate

The above figures show that; lower values of "link quality" were achieved as we got farther from the transmitter. As soon as we went around the corner, the "link quality" shows a big drop. In particular, for experiment b1-c4 it is very clear from the "link quality" metric that the corner is severely affecting the transmission. Because "link quality" is measured in percentage (according to [CHIV-4]) it is very easy to get an idea of what the link status is. In contrast, this is not so evident from the values shown in the graph "level" for the same experiment.

- Throughput, losses and other metrics of performance

In the following figures, we report the results for throughput and packet loss.

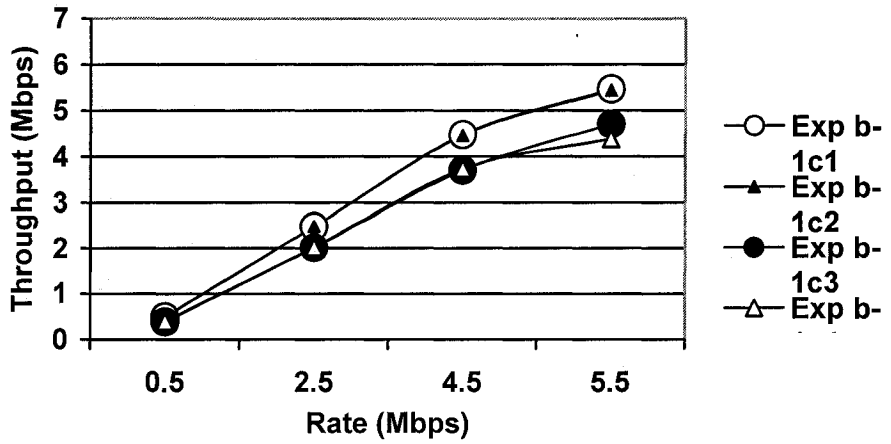


Figure 5.14a: Throughput versus traffic rate for each experiment in the corridor

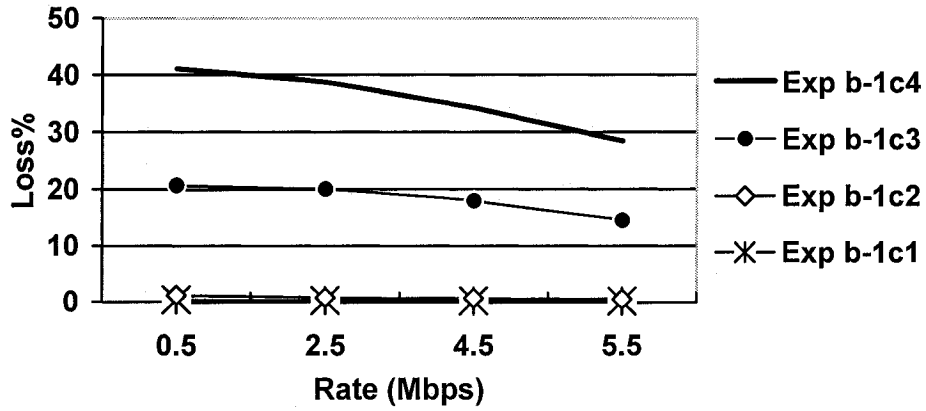


Figure 5.14b: Percentage of wireless loss versus traffic rate for each experiment in the corridor

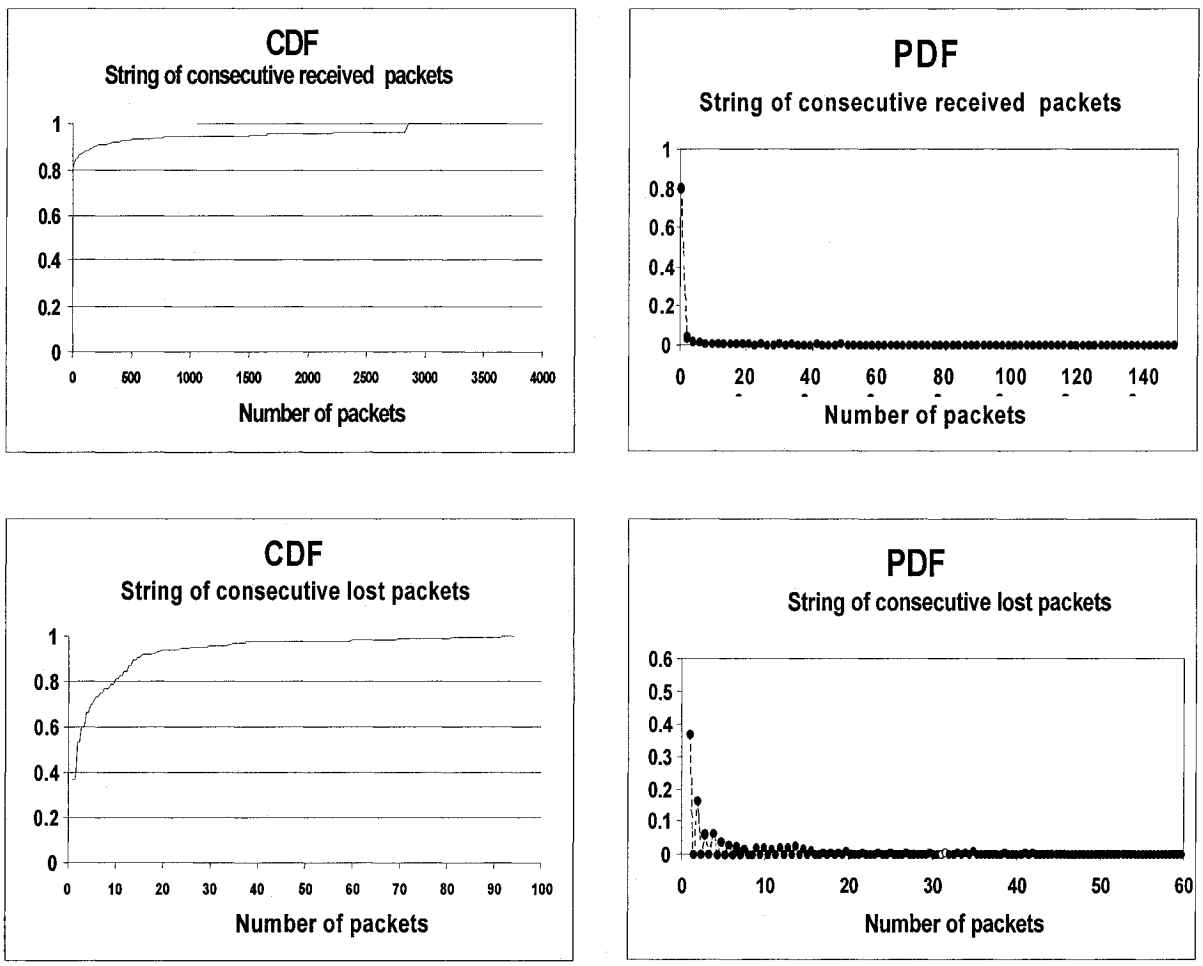
Figure 5.14b shows that by increasing the distance, the loss is increased and the throughput is decreased. Points b-1c3 and point b-1c4 have almost equal distance from

the sender (34 m). In figure 5.14b, from the experiment b-1c4 it is clear that the corner produces an increase in the packet loss.

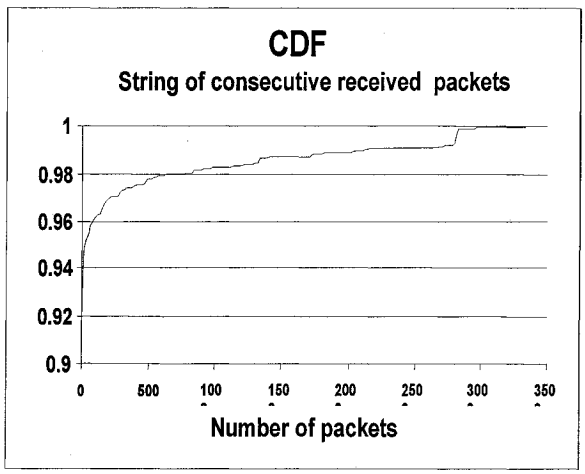
The following figures show the statistics for the number of properly received packets and the number of consecutively lost packets. These sets are grouped by the packet size used for each case.

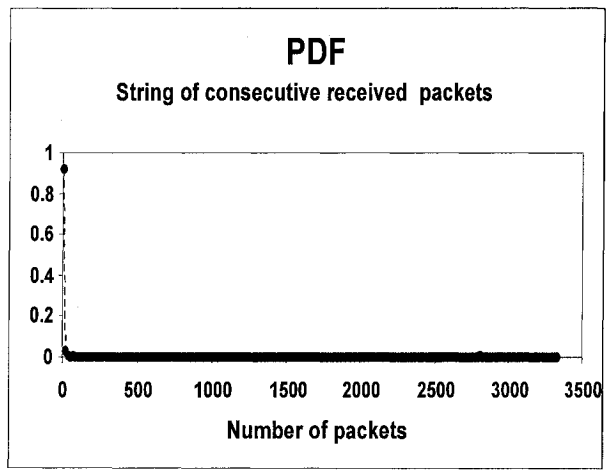
Figure 5.15: PDF and CDF of string of consecutive lost and received packets (Experiment b-1c4)

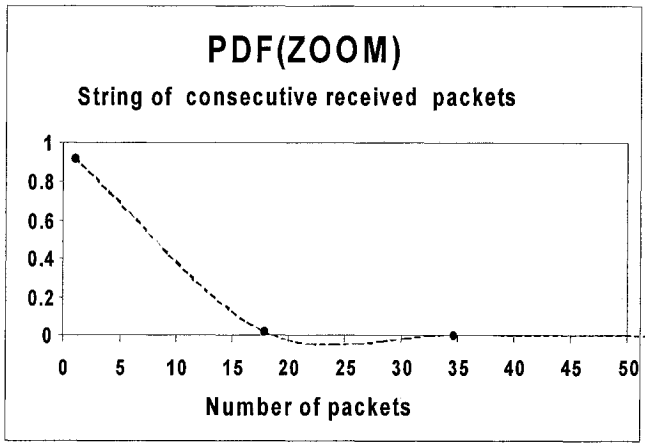
5.15a: Pk Size 100 bytes

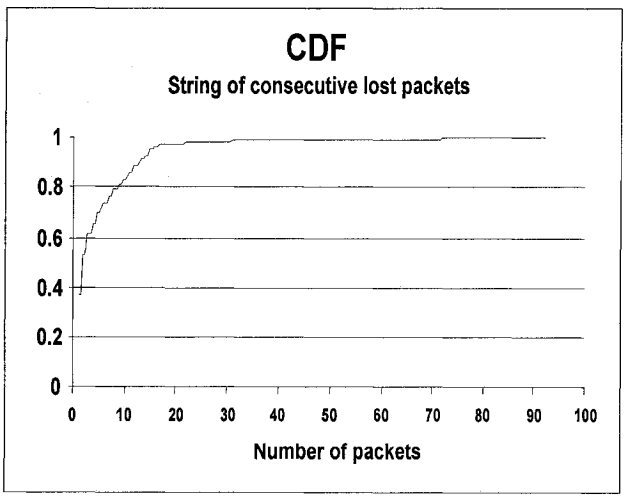


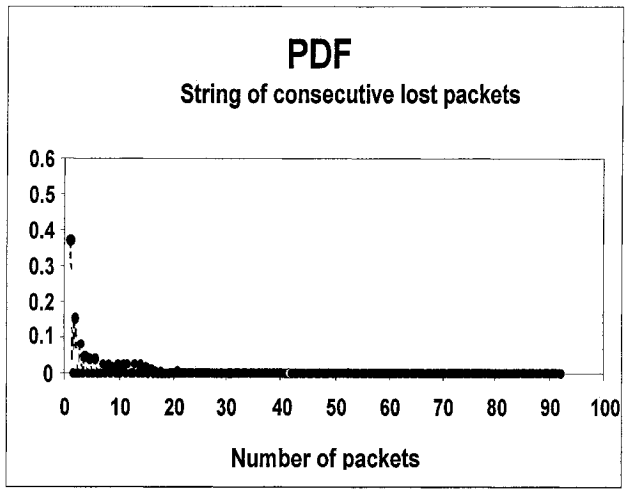
5.15b: Pk Size 500 bytes



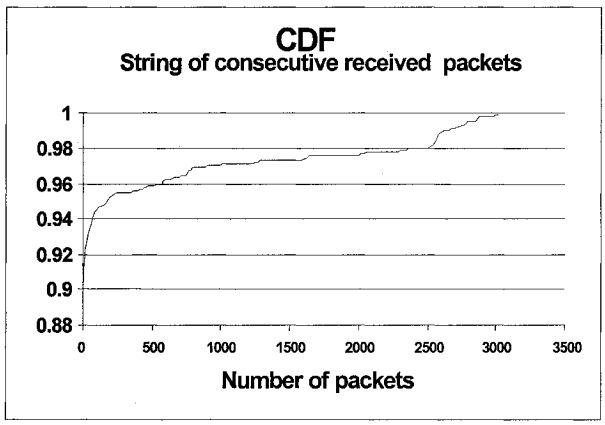


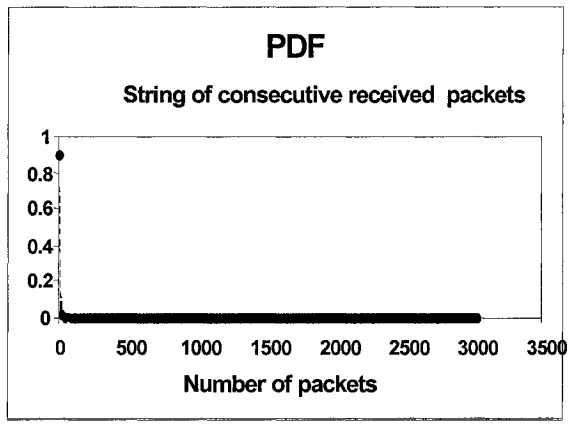


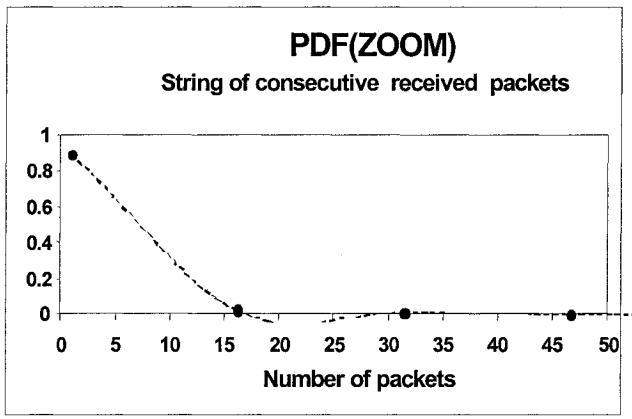


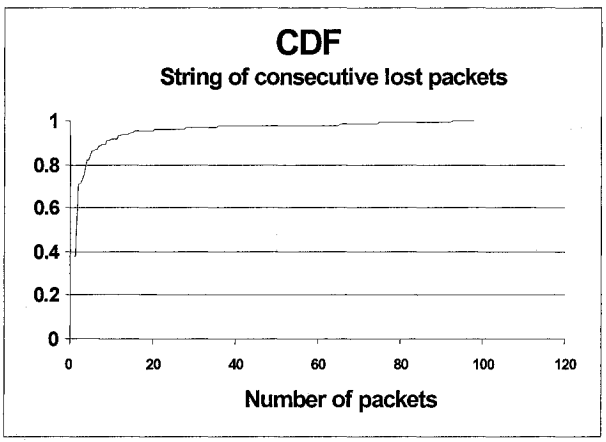


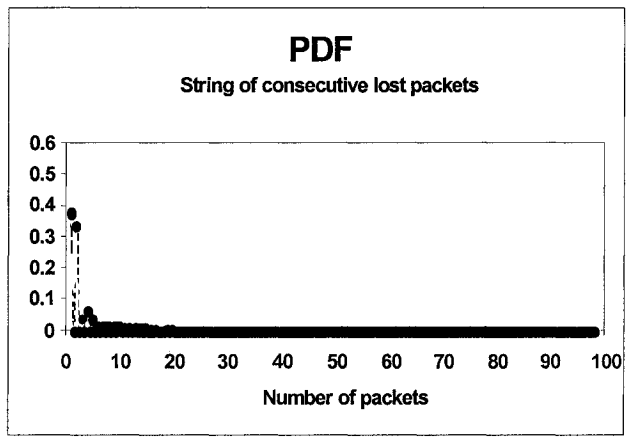
5.15c: Pk size 1000 bytes



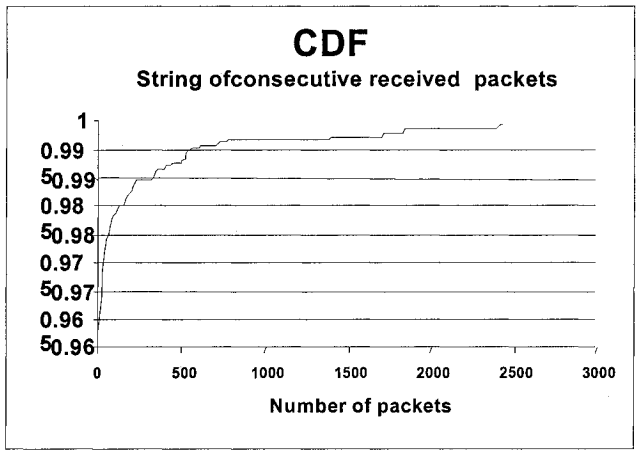


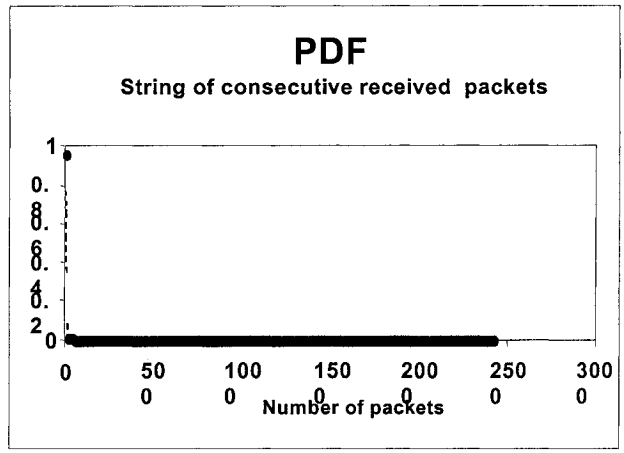


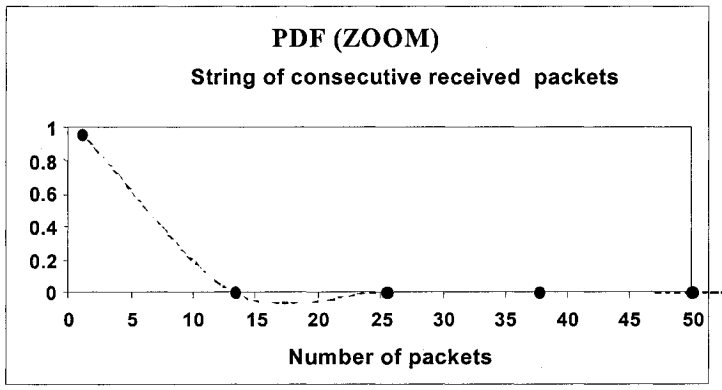


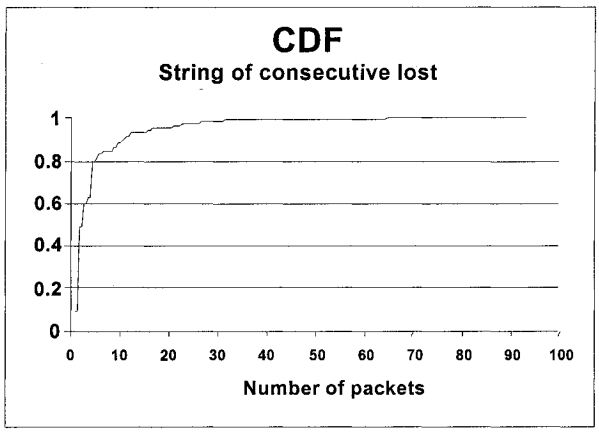


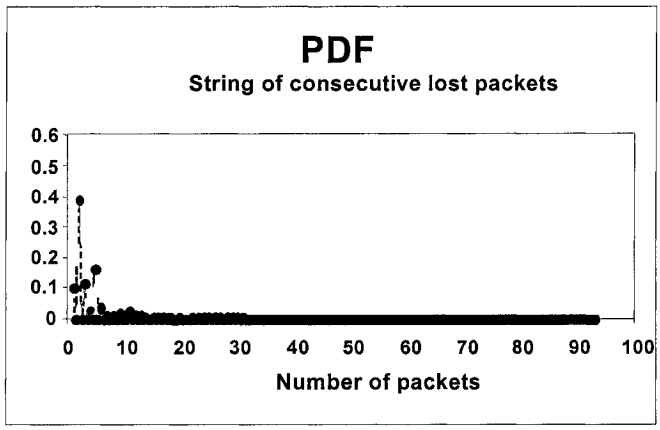
5.15 d: Pk Size 1500











The above experimental results show that the length of the sequence of properly received packets and lost packets is not significantly dependent on the packet size.

Chapter 6

Conclusions

In this thesis we studied how diverse factors affect network performance in real wireless LANs. In our experiments we took into consideration a large number of parameters in order to obtain a better understanding of the way diverse factors and their combination impact the performance. Our most important conclusions follow.

We observed that proprietary algorithms for dynamic rate adaptation and automatic fallback can lead to bandwidth thrashing thus reducing throughput. Packet sizes also may play an important role in performance, especially when the computer has to carry out forwarding functions since smaller packets impose a heavier load to the system. Although new computers are more powerful and this issue may become less relevant, there is also the trend to use low power devices in mobile stations. These devices have usually lower computational processing capabilities and may experience significant decrease in performance. On the other hand, the speed of the host computer may also affect TCP performance if the computer is not fast enough to pull out data from the input buffer, thus forcing the TCP receiver to advertise a smaller congestion window size even if there have not been packet losses. Thus, we observed that processor speeds of the host computer turn out to have an effect on perceived performance.

Another issue that is worth mentioning is the effect on performance of the so-called corner effect. We observed that diverse metrics of performance such as throughput, losses, link quality and signal level indicate that the reception of packets is significantly impacted when there is a corner in the path signal between the sender and the receiver. This issue is rarely mentioned as a source of problems but we consider that it is important since WLANs are usually deployed in indoor environments. This factor should be taken into consideration when planning the coverage of a WLAN.

With all these observations we conclude that network performance of a WLAN is the result of a complex interaction of many factors. Development of accurate tools in software and hardware to help designing and planning network coverage should take into consideration such factors.

Here we give some suggestions for future research:

- Development of more reliable software tools for standard wireless equipment
- Development of more complete simulators that take into account more transmission impairments such as the building layout and construction materials
- Performance evaluation of newer wireless standards such as turbo g

BIBLIOGRAPHY

[CHI-1] M.M. Carvalho. J.J Garcia-Luna-Aceves, "Delay Analysis of IEEE 802.11 in Single-Hop Networks", *in Proc. of 11th IEEE International Conference on Network Protocols (ICNP),* pp.146-155, Atlanta, November 2003.

[CHI-2] C.K. Toh, M. Delwar, and D. Allen, "Evaluating the communication performance of an ad hoc wireless network", *IEEE Transactions on Wireless Communications*, Vol. 1, No. 3, pp. 402-414, July 2002.

[CHI-3] M. Yue, J.J. Han, .S. Trivedi, "Composite performance and availability analysis of wireless communication networks". *IEEE Transactions on Vehicular Technology,* Vol. 50, Issue 5, pp. 1216-1223, Sept. 2001.

[CHI-4] A. Messier, J. Robinson, K. Pahlavan, "Design and Performance Monitoring of a Wireless Campus Area Network". http://www.cwins.wpi.edu/projects/scripts/nsf_wlan.html, 1997.

[CHI-5] G. Xylomenos, G.C. Polyzos, "TCP and UDP Performance over a Wireless LAN". *Proceedings of the IEEE INFOCOM Conference*, pp.439-446, 1999.

[CHI-6] J.C. Amaro, R.P. Lopes, "Performance analysis of a wireless MAN" *Proceedings of IEEE International Symposium on Network Computing and Applications,* pp. 358 –361, 2001.

[CHI-7] G. Xylomenos, G.C. Polyzos, P. Mahonen, "TCP Performance Issues over Wireless Links". *IEEE Communications Magazine,* Vol. 39, No. 4, pp. 52-58.

[CHI-8] C. Komar, C. Ersoy, "Measured Performance of an IEEE802.11 Wireless LAN" in *Proceedings of the Fifteenth International Symposium on Computer and Information Sciences,* pp.246-254, Istanbul/Turkey, October 2000.

[CHI-9] A. Botta, D. Emma, S. Guadagno, A. Pescape, "Performance Evaluation of Heterogeneous Network Scenarios" , *Proceedings of 11th International Conference on Parallel and Distributed Systems - Workshops (ICPADS),* pp. 120-124, 2005.

[CHII-1] IEEE std 802.11; part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, 1999.

[CHII-2] M.S. Gast, *802.11 Wireless Networks: The Definitive Guide*, O'Reilly Networking, April 2002.

[CHII-3] M.S. Gast, "When Is 54 Not Equal to 54? A Look at 802.11a, b, and g Throughput", O'Reilly, Aug. 2003. http://www.oreillynet.com/pub/a/wireless/2003/08/08/wireless_throughput.html

[CHII-4] S. Kapp, "802.11: Leaving the Wire Behind", *IEEE Internet Computing*, Vol. 6, No.1, pp.82-85, January 2002.

[CHII-5] S. Kapp, "802.11a: More Bandwidth without the Wire", *IEEE Internet Computing*, Vol.6, No.4, pp.75-79, July-August 2002.

[CHII-6] S.M. Cherry, "Wi-fi takes new turn with wireless-g". *IEEE Spectrum Magazine*, pp 12-13, August 2003.

[CHII-7] G. Bianchi, A. Capone, and C. Petrioli, "Throughput Analysis of End-to-End Measurement-Based Admission Control in IP," in *Proceedings of IEEE INFOCOM*, pp. 1461-1470, 2000.

[CHII-8] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE J. Selected Areas in Comm.*, Vol. 18, pp. 535-537, March 2000.

[CHII-9] F. Calì, M. Conti, E. Gregori, IEEE 802.11 Wireless LAN: Capacity Analysis and protocol Enhancement, *Proceedings of INFORCOM*, pp.142-149, 1998.

[CHII-10] H. Wu, Y. Peng, K. Long, S. Cheng, J. Ma, "Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement", *Proceedings of the IEEE INFOCOM*, pp. 599- 607, June 2002.

[CHII-11] M. Eriksson, "A Performance Evaluation on the use of IEEE 802.11 for Long Range Communication", Master Thesis, Lulia University of technology, February 2005.

[CHII-12] D. Liu, "A QoS Capable Architecture for IEEE 802.11 Wireless LAN and the Prototype Implementation on Linux". Master Thesis, School of Information Technology and Engineering, Faculty of Engineering, University of Ottawa, Jan. 2004.

[CHIII-1] E.A. Hall, *Internet Core Protocols, The Definitive Guide*, O'Reilly, 2000.

[CHIII-2] J. Postel, RFC 768: User Datagram Protocol, August 1980.

[CHIII-3] J. Postel, RFC 793: Transmission Control Protocol, September 1981.

[CHIII-4] S. Deering, RFC 1112: Host extensions for IP multicasting, August 1989.

[CHIII-5] V. Paxson and M. Allman, RFC 2988: Computing TCP's retransmission timer, November 2000.

[CHIII-6] M. Mathis, J. Mahdavi, S. Floyd and A. Romanov, RFC 2018: TCP selective acknowledgement options, October 1997.

[CHIII-7] M. Mathis, J. Mahdavi, S. Floyd and M. Podolsky, RFC 2883: An extension to the selective acknowledgement (SACK) option for TCP, July 2000.

[CHIII-8] S. Floyd, T. Henderson and A. Gurtov, RFC 3782: The New Reno medication to TCP's fast recovery algorithm, April 2004.

[CHIII-9] A. Kuznetsov, and P. Sarolahti, "Congestion control in Linux TCP", 2002. *In Proc. of USENIX'*, June 2002.

[CHIII-10] R. Ludwig, and K. Sklower, "The Eifel Re-transmission Timer". *ACM Computer Communication Review,* Vol. 30, No. 3, pp.17-27, July 2000.

[CHIII-11] K. Ramakrishnan, S. Floyd, and D. Black,"The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[CHIII-12] Y.T. Li, and D. Leith, "BicTCP Implemenation in Linux Kernels" *Hamilton Institute , NUI Maynooth*, February 2004.

[CHIII-13] The linux kernel archives. http://www.kernel.org

[CHIII-14] G. Xylomenos, G. Polyzos, P. Mahonen, and M. Saaranen, "TCP performance issues over wireless links", *IEEE Communications Magazine*, Vol. 39, No. 4, pp. 52–58, 2001.

[CHIII-15] H. Balakrishnan, and V.N. Padmanabhan, "How Network Asymmetry Affects TCP". *IEEE Communications Magazine*, April 2001.

[CHIII-16] M. Allman, and V. Paxson, RFC 2581: TCP Congestion Control, April 1999.

[CHIII-17] D. Eckhardt, and P. Steenkiste, "Measurement analysis of the error characteristics of an in-building wireless network", *Proceedings of ACM SIGCOMM'96*, pp.243-254, August 1996.

[CHIV-1] K. Wehrle, F. Pahlke, and H. Ritter, *The Linux Networking Architecture*, 2005.

[CHIV-2] Madwifi Project, http://madwifi.org/, http://madwifi.org/wiki/ngFeatures, http://madwifi.org/ticket/179, http://madwifi.org/browser/trunk/INSTALL#L19, 2006.

[CHIV-3] V. Henson, and L. Martensson, "Hacking the Linux 2.6 kernel, Part 2: Making your first hack", IBM, 02 Aug 2005.

[CHIV-4] J. Bardwell, "You Believe You Understand What You Think I Said..." Connect802 Corporation, 2004.

[CHIV-5] Datatag/papers, http://datatag.web.cern.ch/datatag/papers/tr-datatag-2004-1, 2004.

[CHIV-6] J. Bardwell, "The Representation of Signal Strength by Wireless Analysis Tools", Connect802 Corporation, 2004.

[CHIV-7] J. Tourrilhes, "Wireless Extensions for Linux", http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/LinuxWireless.Extensions.html , May 2003.

[CHIV-8] Linux man pages, http://72.14.203.104/search?q=cache:qdREokYJxpkJ:man.he.net/+linux+man&hl=en&ct=clnk&cd=2&client=firefox-a

[CHIV-9] I.T. Bowman, R.C. Holt, and N.V. Brewster, "Linux as a Case Study: Its Extracted Software Architecture", *Proceedings of ICSE*, pp.555-563, 1999.

[CHIV-10] J. Ousterhout, H. Costa, D. Harrison, J. Kunze, M. Kupfer, and J. Thompson, "A Trace-Driven Analysis of the UNIX 4.2 BSD File System".*In Proceedings of the Tenth ACM Symposium on Operating System Principles*, pp. 15-24, Orcas Island, WA, December 1985, ACM.

[CHIV-11] Linux Mandriva. http://www.mandriva.com/, 2005.

[CHIV-12] ORiNOCO Classic Gold PC Card, http://www.proxim.com/products/wifi/client/goldpccard/index.html, 2005.

[CHIV-13] InterWatch 95000 network analyzer, http://www.dtikorea.co.kr/Navtel/Iw95000.html.

[CHIV-14] Frequently Asked Questions (IRQs - Interrupt Request), http://www.duxcw.com/faq/irq/irq.htm, 2003.

[CHIV-15] M. Zec, M. Mikuc, M. Žagar, "Estimating the Impact of Interrupt Coalescing Delays on Steady State TCP Throughput", *Proceedings of the 10th SoftCOM Conference,* 2002.

[CHV-1] T. Kuang, F. Xiao, and C. Williamson, "Diagnosing Wireless TCP Performance Problems: A Case Study". University of Calgary, 2003

[CHV-2] M. Lacage, M.H. Manshaei, and T. Turletti, "IEEE 802.11 Rate Adaptation: A Practical Approach". Institut National de Recherche enAutomatique (INRIA) Planete Project, 2004

[CHV-3] TCP Tuning Guide, http://www-didc.lbl.gov/TCP-tuning/linux.html, Version 1.0 - Last published Feb 13, 2006.

[CHV-4] Atheros Communications, "802.11 Wireless LAN Performance", http://www.atheros.com/pt/papers.html , published April 2003.

Appendix A

IEEE 802.11 Taskgroups

The following list is not complete, but it contains the most important extensions.

802.11	Legacy 802.11, 2.4GHz, data rate of up to 2Mbps
802.11a	5GHz, data rate of up to 54Mbps
802.11b	2.4GHz, data rate of up to 11Mbps. Backwards compatible with 802.11
802.11e	Quality of Service extension
802.11g	2.4GHz, 54Mbps. Backwards compatible with 802.11 and 802.11b
802.11i	Enhanced security and authentication mechanisms
802.11n	Data rate of at least 100Mbps
802.11p	WAVE, wireless access for the ability in vehicular environments
802.11r	Fast hand-o (roaming) between APs in the same WLAN
802.11s	Wireless performance prediction

APPENDIX B: Converting Signal Strength Percentage to dBmValues [CHIV-6]

Conversion for Atheros
Unlike the other vendors described, Atheros uses a formula to derive dBm.
RSSI_Max = 60
Convert % to RSSI
Subtract 95 from RSSI to derive dBm
Notice that this gives a dBm range of –35dBm at 100% and –95dBm at 0%.

Conversion for Symbol

RSSI_Max = 31
Convert % to RSSI and lookup the result in the following table:
RSSI <= 4 is considered to be –100dBm
RSSI <=8 is considered to be –90 dBm
RSSI <=14 is considered to be –80 dBm
RSSI <=20 is considered to be –70 dBm
RSSI <=26 is considered to be –60 dBm
RSSI greater than 26 is considered to be –50dBm
Notice that this gives a dBm range of –50dBm to –100dBm but only in 10dBm steps.

Conversion for Cisco
Cisco has the most granular dBm lookup table.
RSSI_Max = 100
Convert % to RSSI and lookup the result in the following table. The RSSI is on the left, and the corresponding dBm value (a negative number) is on the right.
Notice that this gives a range of –10dBm to –113dBm. Bearing in mind that a Cisco card will have a Receive Sensitivity of –96dBm at its lowest, it is impossible to obtain an RSSI value of less than 16. Note, also, that all RSSI values greater than 93 are assigned –10dBm, and that there are multiple places in the table where two adjacent RSSI values are assigned the same dBm value.

0= -113	12= -101	24= -88	36= -75	48= -60	60= -47	72= -37	84= -22	96= -10
1= -112	13= -99	25= -87	37= -74	49= -59	61= -46	73= -35	85= -20	97= -10
2= -111	14= -98	26= -86	38= -73	50= -58	62= -45	74= -34	86= -19	98= -10
3= -110	15= -97	27= -85	39= -72	51= -56	63= -44	75= -33	87= -18	99= -10
4= -109	16= -96	28= -84	40= -70	52= -55	64= -44	76= -32	88= -17	100= -10
5= -108	17= -95	29= -83	41= -69	53= -53	65= -43	77= -30	89= -16	
6= -107	18= -94	30= -82	42= -68	54= -52	66= -42	78= -29	90= -15	
7= -106	19= -93	31= -81	43= -67	55= -50	67= -42	79= -28	91= -14	
8= -105	20= -92	32= -80	44= -65	56= -50	68= -41	80= -27	92= -13	
9= -104	21= -91	33= -79	45= -64	57= -49	69= -40	81= -25	93= -12	
10= -103	22= -90	34= -78	46= -63	58= -48	70= -39	82= -24	94= -10	
11= -102	23= -89	35= -77	47= -62	59= -48	71= -38	83= -23	95= -10	