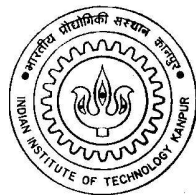


Implementation and Performance Issues of VoIP in Long Distance 802.11b Networks

*A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology*

by

Venkata Rao Chimata



to the

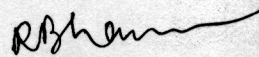
**Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur**

June, 2005

Certificate

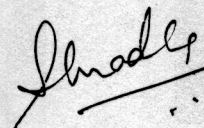
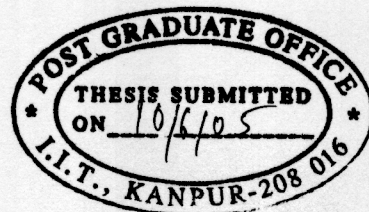
This is to certify that the work contained in the thesis entitled "*Implementation and Performance Issues of VoIP in Long Distance 802.11b Networks*", by *Venkata Rao Chimata*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

June, 2005



(Dr. Bhaskaran Raman)

Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.



Abstract

802.11b seems to be a promising solution to bridge digital barrier between rural and urban areas. The main challenge in long distance 802.11b networks is to avoid the *hidden node problem*. Hidden node problem and contention based DCF (Distributed Coordination Function) protocol degrade QoS (Quality of Service) of VoIP. RTS/CTS may help to avoid hidden node problem, but the overhead associated with RTS/CTS mechanism is too high to use it for services like VoIP where the size of the packet is very small. 802.11b protocol also specifies a contention free protocol, PCF (Point Coordination Function) is not supported in any of the known existing cards.

Our thesis work mainly focuses on two issues, *improving performance* and *implementation (with low cost)* of VoIP in point-to-multipoint networks. In the first part we have analyzed the performance of VoIP on a long distance 802.11b network and shown its bad performance under certain conditions. Later we have designed and implemented a driver based contention free protocol known as Master-Slave protocol for point-to-multipoint links. This protocol cleverly suppresses the behaviour of DCF on off-the-shelf (low cost) 802.11b cards. We have shown that with our protocol the performance from the perspective of QoS is improved significantly, eg jitter (A QoS metric) values with our protocol are around 4ms while with the original setup those values are around 80-100ms in the presence of cross traffic.

In the second part we have engineered and implemented a VoIP system which is best suited to provide telephony/voicemail services to the rural areas with low cost using opensource software like asterisk (PBX software) and openh323 (VoIP client software) to reduce the overall cost.

Acknowledgements

It gives me immense pleasure to thank my thesis supervisor Dr. Bhaskaran Raman for his valuable suggestions, comments, compliments and patience. It is an indelible experience to work with him. I would like to extend my heartfelt thanks and applause to the energetic and dynamic faculty members of Department of Computer Science and Engineering for thier excellent teaching. I sincerely thank Media Lab Asia, IIT Kanpur for providing all the equipment needed for my thesis work. I extend my thanks to Dr. A.R.Harish of Electrical Engineering Department for his help in issues related to RF-Cables and signal levels.

I also wish to thank Sukanto, Anurag, Vishal, Ramchand, AKSingh, Ranjit and Anish for their help in field. I extend my warm regards and thanks to my friends Sateesh, Venu, LovaRaju and DV Janardhan with whom I shared beautiful experiences during my stay at IITK.

I am forever thankful to my parents and my grandfather, for their love and encouragement. I thank my brothers Ram and Lakshman and sister Parvathi for their affection towards me.

Last but not the least , I thank my bestfriends Imran and Susmitha for their constant encouragement at every moment of my life.

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem Statement	3
1.3	Related Work	4
1.4	Approach	4
1.5	Main Results	5
1.6	Organization of the Report	5
2	Background	7
2.1	Quality of Service	7
2.1.1	Jitter	8
2.1.2	Packet Loss	8
2.1.3	Latency/Delay	8
2.2	WiFi Networks	8
2.3	Hidden Node Problem	11
2.4	Software Used	11
2.4.1	Host AP: A Network Driver for Prism2 Cards	11
2.4.2	Asterisk: The Open source Linux PBX	12
2.4.3	OH323: A VoIP Client	12
3	Related Work	13

3.1	802.11b MAC Protocol	13
3.1.1	DCF	13
3.1.2	PCF: Alternative to DCF	15
3.2	802.11e Protocol	15
3.3	Dual Queue Strategy	15
3.4	Our Approach	16
4	Performance of Host AP on a Long Distance 802.11b Link	17
5	Master-Slave Protocol	22
5.1	Performance Analysis of VoIP on DCF	22
5.1.1	QoS on DCF: An Experiment	23
5.1.2	Analysis of the Results	25
5.2	A Driver Based Approach	26
5.2.1	Frame Format At Driver Level	28
5.2.2	Protocol at Master	29
5.2.3	Protocol at Slave	30
5.2.4	Overheads & Optimizations	30
5.3	Implementation of the Protocol	32
5.3.1	Protocol at Master	32
5.3.2	Protocol at Slave	36
5.4	Drawbacks	38
6	Results	40
6.1	Results in Outdoor Setup	40
6.1.1	Cross Traffic is UDP	40
6.1.2	Cross Traffic is TCP	47
6.2	Results in Indoor Setup	48
6.2.1	Original Host AP Driver	48

6.2.2 Master-Slave Driver	48
7 Implementation	53
8 Conclusions & Future Work	55
Bibliography	56

List of Tables

3.1	Comparison of Master-Slave protocol with others	16
4.1	Experimental and Theoretical Throughput Values	20
4.2	Jitter & Packet Loss Values for Different Rates of Traffic	21
5.1	Power Levels at all the sites	24
6.1	Jitter Values with Original driver	42
6.2	Packet Loss with Original Driver	42
6.3	Jitter Values with Master-Slave driver	43
6.4	Packet Loss with Master-Slave Driver	43
6.5	Jitter Values with Master-Slave driver in Indoor Setup with 2-slaves .	50
6.6	Packet Loss with Master-Slave Driver in Indoor Setup with 2-slaves .	50
6.7	Jitter Values with Master-Slave driver in Indoor Setup with 3-slaves .	51

List of Figures

1.1	Complete Testbed of DGP[4]	2
2.1	Typical WiFi Wireless Network	9
2.2	Long Distance Wireless Network	10
2.3	Pigtail	10
2.4	Hidden Node Problem	11
3.1	802.11b Timing Diagram	14
4.1	802.11b Timing Diagram	18
4.2	Setup between Bithoor and MS3	19
5.1	Experimental Out-door Setup	24
5.2	Protocol Stack and Network Driver	27
5.3	Frame Header Format at Driver Level	28
5.4	Frame Header Format in Modified Driver	28
5.5	Master-Slave Protocol	31
5.6	Optimized Master-Slave Protocol	31
6.1	Indoor Setup with 2-way Splitter	49
7.1	FXO Setup	54

Chapter 1

Introduction

1.1 Background

The terms cellular networks and Internet are not new, at least to the people in developed countries, thanks to the communication revolution in the last one and half decades. But 70% of the people in the world live in developing/poor countries. Most of them are yet to reap the benefits of internet revolution. The primary reason for this is the fact that it costs high to provide cellular services, which is unbearable by the poor villagers and also that it is not economically feasible to provide these services in these areas because of the low population density. Thus the digital technology is limited only to developed countries and urban areas in developing countries. One promising solution to bridge the digital barrier seems to be 802.11, since its capability is already tested in DGP(Digital Gangetic Plains) along with others[1]. In this type of network we can employ one station at each village to connect the villages to the external world digitally. Each of the stations communicates to the external world through a land line node which is connected to the Internet. The complete testbed of DGP is shown in Fig 1.1. To know more details about DGP visit <http://www.iitk.ac.in/mladgp>. Though the main focus of DGP is on multi-hop networks, we restrict our discussion to 1-hop networks only. Throughout our discussion we assume that our underlying network has the following characteristics.

Digital Gangetic Plains

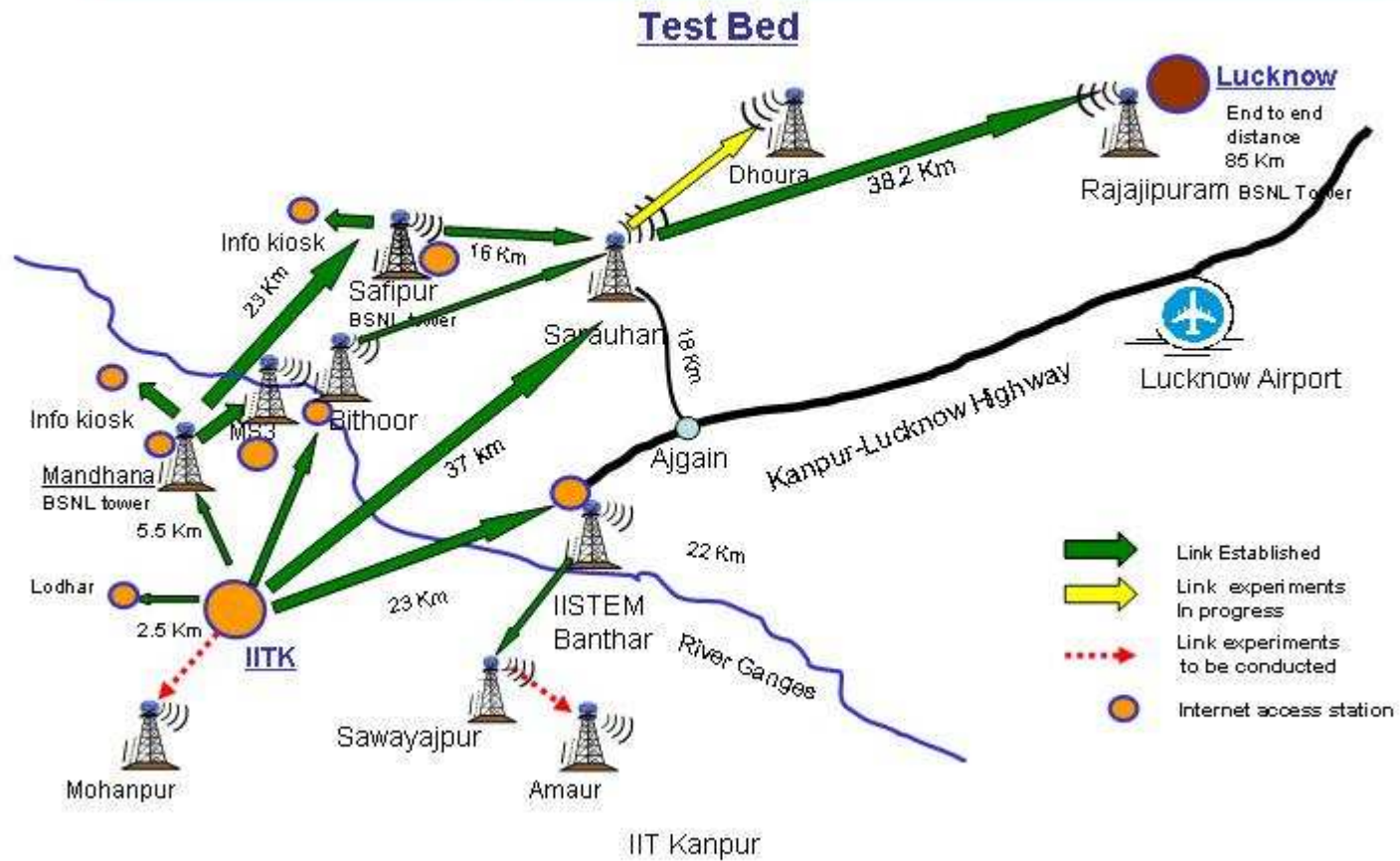


Figure 1.1: Complete Testbed of DGP[4]

1. The network is 1-hop, meaning all the nodes can hear the land line node directly. Land line node is a special node which is connected to the wired Internet.
2. The communication from land line node to any of the remaining nodes is point-to-multi-point and the communication from any of the nodes to the land line node is point-to-point.
3. The distance between any two nodes of each link ranges from 5km to 40km, where as 802.11b is designed for indoor usage only.
4. All the stations are static (do not move).

Since each of the stations cannot see the remaining stations, the probability that hidden node problem occurs is high. Occurrence of hidden node problem directly implies collision of frames at MAC layer which is highly undesirable especially when we are offering Real-Time services like VoIP/video.

1.2 Problem Statement

The main reason for the occurrence of collisions is due to simultaneous transmission of two or more nodes to a common node ie due to the presence of cross traffic. The contention based DCF (Distributed Coordination Function), the underlying 802.11b MAC protocol, cannot handle simultaneous transmissions due to hidden node problem. So DCF is not suitable to support Real-Time services like voice/video. RTS/CTS mechanism is one solution to minimize the number of collisions. But one cannot completely avoid collisions with RTS/CTS. Moreover, the overhead associated with RTS/CTS mechanism is too high to use it for VoIP packets, where the size of each packet is very small. 802.11b also defines contention free MAC protocol, known as PCF (Point Coordination Function) to support Real-Time services. With PCF there would not be any MAC level collisions. But it is not supported by any commercially available products.

In this thesis work we designed and implemented PCF kind of protocol at driver level to support Real Time services in point-to-multipoint networks. It is a TDMA (Time Division Multiple access) style protocol and all transmissions are controlled by a central coordinator. We measured the performance of VoIP in the network when the nodes follow/don't follow our protocol and compared the results. In the second part we implemented a virtual telephone exchange system which supports telephony and voice mail services to rural areas.

1.3 Related Work

In [2] 2P-protocol is proposed which is best suited to multi hop networks with point-point-links. It is also a contention free protocol and aimed at networks with point-to-point links. Our work is an extension to that protocol and is aimed at point-to-multipoint links. In [8] the authors discuss about prioritizing packets above MAC layer there by ensuring QoS to time-critical services. But it is not implemented and can be tried as an extension to this thesis work. IEEE defines 802.11e[6] to provide QoS as an alternative to 802.11b. Even 802.11b standard defines PCF to avoid collisions. But as said earlier PCF is not implemented in any of the existing commercial NICs (Network Interface Cards).

1.4 Approach

In this thesis we divided our work into two parts. In the first part we proposed and implemented a novel *Master-Slave protocol* to avoid collisions at MAC layer, there by improving performance. It is a TDMA style protocol. The Master-Slave protocol works at driver level. Briefly the protocol is as follows. All the transmissions in our protocol are broadcast transmissions only. Any unicast frame that comes from upper layers is converted into a broadcast frame before transmission. Collisions are avoided by controlling the transmissions of all the nodes in the network. All the transmissions are controlled by a central coordinator, called master. Remaining nodes are called as slaves. Any transmission would be instantiated by the master. A slave can transmit

only when it is asked by the master to do so. We modified the network driver Host AP to implement our protocol. The Master-Slave protocol cleverly suppresses the behaviour of DCF on off-the-shelf (low cost) 802.11b Network Interface Cards. This is possible because all the transmissions can be controlled at driver level and the driver can instruct the card for transmission with out much delays. Throughout our discussion we assume that Original Driver means unmodified driver on DCF and Master-Slave Driver means the modified driver on DCF. And also whenever we refer to DCF and don't specify anything about driver, it implies that we use Original Driver there. The meanings of these should be inferred from the context where we refer them.

In the second part we implemented a virtual telephone exchange system where the clients can be either ordinary phones, computers or VoIP phones. We searched on several alternatives for the implementation and finally zeroed on asterisk as the PBX software and openh323 as the VoIP client software as they are open sources and are available for free.

1.5 Main Results

We measured the performance of the network from the perspective of QoS (Quality of Service). We concentrated on two QoS issues, jitter and packet loss. With the Original Driver the jitter values are around 80msec and a packet loss is around 20% even when the rate of the cross traffic is with in the acceptable range for the network. But with the Master-Slave driver the jitter values are around 4msec irrespective of the rate at which the cross traffic is sent. Similarly the packet loss is almost 0% with the Master-Slave Driver, while it is increased with the increase of rate of cross traffic with the Original Driver.

1.6 Organization of the Report

In Chapter 2 we present the QoS issues, wireless networks, hidden node problem and introduce Host AP, Asterisk, Openh323 to the reader. In Chapter 3 we discuss

802.11b and 802.11e protocols and make some observations about how our work differs from them. In Chapter 4 we discuss the issues related to Host AP on a long distance 802.11b link and in Chapter 5 we discuss the proposed protocol. In Chapter 6 we discuss and analyze the results obtained. In Chapter 7 we present the details of implementation of virtual telephone exchange. In Chapter 8 we conclude the important points and discuss the possibilities of future work.

Chapter 2

Background

In this Chapter we start our discussion with QoS issues, why QoS is important in a network. Next we discuss briefly about WiFi networks and long distance networks. Later we will discuss, how long distance wireless networks differ from traditional indoor networks. Finally we end our discussion with the software used in our thesis work. The reader who is familiar with these concepts can skip this Chapter.

2.1 Quality of Service

QoS (Quality of Service) is a major issue to consider in voice/video implementations. It is concerned with ensuring the smooth communication between the two ends of the connection. The main parameters by which QoS is determined are -

- Jitter
- Packet loss
- Latency

Now we briefly discuss about each of these parameters.

2.1.1 Jitter

Jitter can be defined as the variance (standard deviation) in the inter arrival time of packets. It usually arises from the congestion in the network. For delay sensitive traffic, higher values of jitter result in discarding the packets by the jitter buffer (play out buffer) at the receiver. Jitter buffer temporarily stores the arriving packets to minimize the variation. If the buffer size is too high, then additional delay will be increased in the transmission resulting in the overlap of conversation, which makes the conversation difficult. If the buffer size is too low, then more packets would be dropped, degrading the quality of service. A buffer configuration of 30 to 50msec worth of packets is usually acceptable for voice conversation.

2.1.2 Packet Loss

Real time services are not loss tolerant. For example for g-711 codec, even 1% of the loss would degrade the quality of service[11]. The packet loss should be minimized to 0 to ensure a high quality of communication[10].

2.1.3 Latency/Delay

Services like voice/video are delay sensitive. A packet which arrives with an end-to-end delay exceeding a certain limit is useless and would be dropped by the receiver. For eg the end-to-end delay in VoIP connection should not be more than 150ms[11].

2.2 WiFi Networks

Internet services can be provided to any WiFi wireless client as shown in Fig 2.1. There is a special node called Access point, which is connected to Internet. Remaining nodes are called as clients. Any wireless node (client) communicates with the access point using any of the wireless technologies like 802.11a,802.11b etc. All the wireless nodes communicate with the external world through the access point. In this thesis work we consider 802.11b as the underlying technology because it

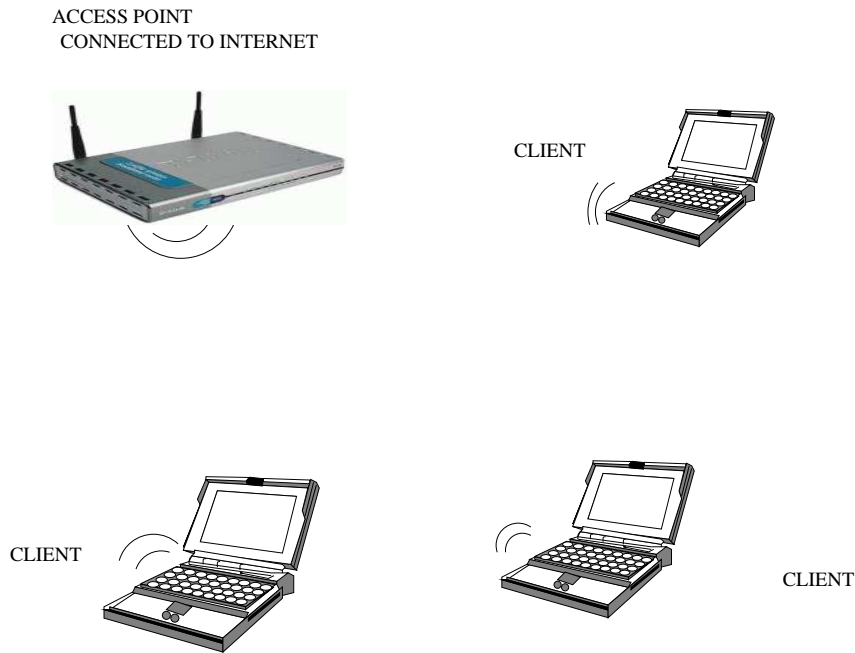


Figure 2.1: Typical WiFi Wireless Network

operates in license free spectrum (2.4GHz)[12].

In long distance networking also, Internet connectivity can be provided to remote nodes in the same manner with a few exceptions. A typical network with long distance links would be as shown in Fig 2.2. Since the nodes are very far from the land line access point, we need to use high gain external antennas and also that we need to mount them on the top of tall towers or buildings to ensure that there would not be any obstacles in the transmission path of the signal. We can use either high gain parabolic directional or sector antennas. Directional antennas have high-gain compared to sector antennas, but their angular radiation pattern is limited. If directional antennas are used to communicate with more than one neighbor, then one antenna is to be used for each neighbor (which is quite common).

Here all the clients are in the vicinity of the land line access point. If we use directional antenna, any two clients may not listen each other. So the probability that *hidden node problem* occurs is more here.

The radio may be connected to high gain external antenna through Pigtail which in turn is connected to RF-cable. A typical pigtail is as shown in Fig 2.3.

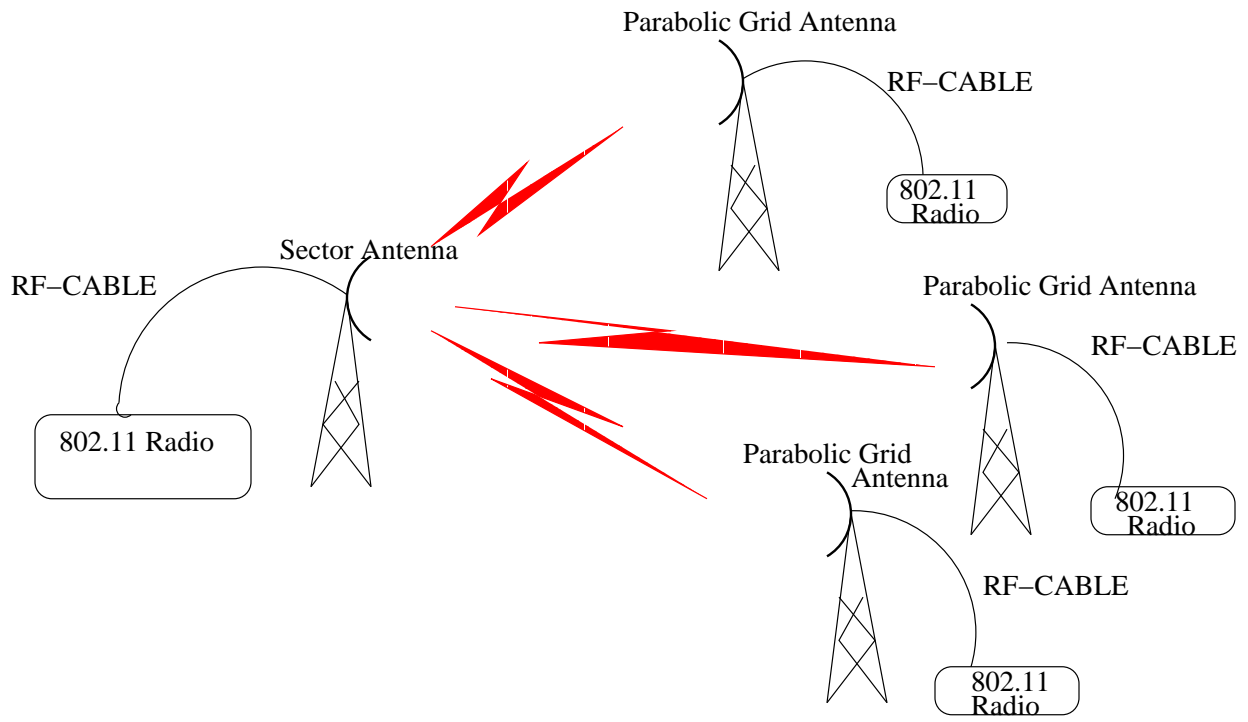


Figure 2.2: Long Distance Wireless Network

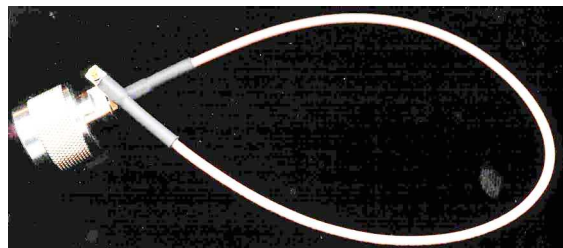


Figure 2.3: Pigtail

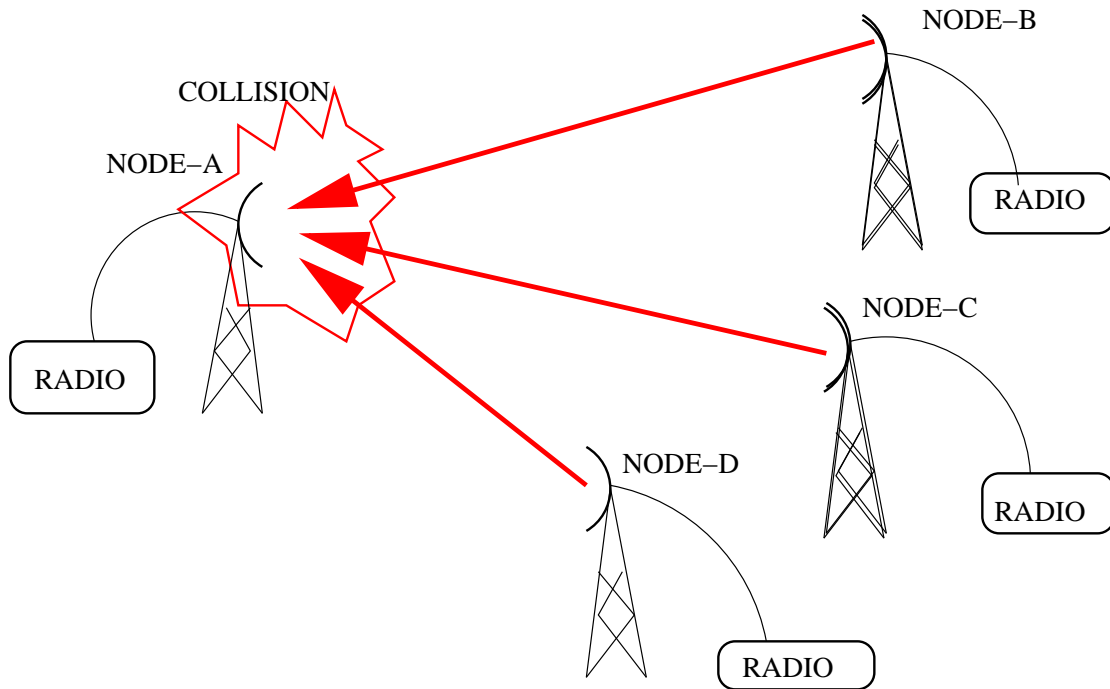


Figure 2.4: Hidden Node Problem

2.3 Hidden Node Problem

Fig 2.4 shows how a hidden node problem occurs. All the nodes try to transmit to node A simultaneously. Each of them is unaware of the others transmissions to node A resulting a collision there by frame loss at node A.

2.4 Software Used

2.4.1 Host AP: A Network Driver for Prism2 Cards

Host AP[5] is a Linux network driver for wireless LAN cards based on prism2/2.5/3 chipset. It is a very popular open source access point software in wireless networks community. In this thesis work I modified the Host AP driver to implement our proposed protocol at driver level.

2.4.2 Asterisk: The Open source Linux PBX

Asterisk[15] is an open source Linux PBX. If installed onto a Linux box, it provides all the services you would expect from a PBX. It supports VoIP and inter-operate with almost all the VoIP technologies like SIP/OH323. It requires relatively inexpensive hardware to operate and offers services like voice-mail, call forwarding.

2.4.3 OH323: A VoIP Client

Openh323 or simply OH323[16] is an open source that implements ITU h.323 teleconferencing protocol. It is used as end client in voice conversation.

Chapter 3

Related Work

In this Chapter we discuss 802.11b protocol, analyze why it is not suitable to offer QoS to time bounded services and discuss some literature related to providing QoS in 802.11 networks.

3.1 802.11b MAC Protocol

The 802.11b standard defines two types medium access modes namely DCF (Distributed Coordination Function) and PCF (Point Coordination Function). While DCF is contention based protocol, PCF is contention free protocol. The timing diagram of 802.11b is as shown in Fig 3.1.

3.1.1 DCF

Before transmitting a frame, station senses the channel. If the channel is idle it defers the transmission for fixed DIFS amount of time, followed by variable back-off*slottime amount of time. Here backoff is a random integer chosen between 0 and CW (Contention Window). For each such slot the station senses the channel. If the channel is sensed as idle then backoff is decreased by one. If the channel is not idle in any of the slots, the station gives up contending for the channel in the current DCF period and compete for the channel in the next DCF period. If

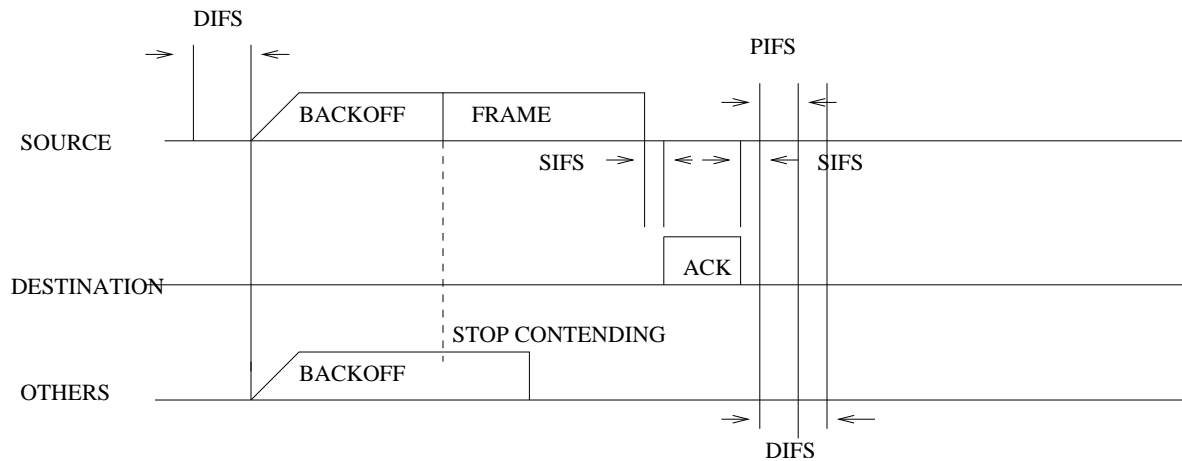


Figure 3.1: 802.11b Timing Diagram

the backoff reaches 0 then the node starts transmitting. The station which chooses lowest backoff value wins the contention and grabs the channel. The problem with DCF is that it is a contention based MAC protocol. Consequently in networks with relatively high load there would be high number of MAC level collisions. Another reason for collisions is the famous *hidden node problem*. So the packet loss, delay and delay variation (Jitter) would be high under high load. Hence DCF is not suitable to support time-critical services like VoIP/video which requires Quality of Service.

RTS (Request To Send)/CTS (Clear To Send) was introduced to avoid hidden node problem. In this mechanism, before any transmission the sender sends a RTS frame to the intended receiver. The receiver responds with a CTS frame if it is ready for reception. RTS and CTS frames contain the information about the duration of the transmission. Consequently all the nodes which are in the vicinity of either receiver or sender would remain silent (do not transmit) during that transmission. RTS/CTS mechanism can reduce the number of collisions but it does not completely avoid collisions. And also the overhead associated with RTS/CTS mechanism is relatively high for VoIP frames whose payload length is low. The time required to required transmit a RTS frame is $216.55\mu\text{sec}$. (SIFS+20 bytes of RTS) and the time to transmit a CTS frame is $212.18\mu\text{sec}$ (SIFS+14 bytes of CTS). So the total overhead associated with RTS/CTS is $428.73\mu\text{sec}$, while the time required to transmit a VoIP packet with g-711 codec whose payload length is 160 bytes is

934.35 μ sec (Refer Chapter4 for more details of the calculation).

3.1.2 PCF: Alternative to DCF

To support time bounded delivery of frames, 802.11b standard defines an optional PCF (Point Coordination Function) as an alternative to DCF . PCF is a contention free TDMA protocol. In PCF the flow of traffic in a network is controlled by a special node called point coordinator. The point coordinator polls each of the stations according to polling list. Only the polled station can transmit. The PCF mode occurs alternately between contention based DCF periods. The point coordinator grabs the channel after every DCF period. It waits for PIFS¹ (PCF Inter Frame Space) amount of time before starting transmission. But PCF is not supported by known existing NICs in the market today as it is optional.

3.2 802.11e Protocol

802.11e standard introduces two protocols as in 802.11b. One is contention based EDCF (Extended DCF). EDCF also doesn't solve the hidden node problem. The other is EPCF (Extended PCF) which is a contention free and is suitable to support Real Time Services. In EPCF at MAC layer, each of the incoming frames from upper layers is given a different priority. Time critical frames like , VoIP/video frames are given top priority. There are no commercial 802.11e products and it is long time to go that an 802.11e product comes into market. Moreover its capability for out-door networks is not tested through simulations also. Even after it comes into market, its interoperability with the existing 802.11b products is not guaranteed.

3.3 Dual Queue Strategy

In [8] the authors discuss about prioritizing packets at driver level. Packets are classified as either Real Time or Non-Real Time. They use port numbers as well as

¹SIFS<PIFS<DIFS

Table 3.1: Comparison of Master-Slave protocol with others

-	Supports QoS ?	Supported Commercially?	Easy to Replace?	Prioritize Frames?
DCF	NO	YES	NO	NO
PCF	YES	NO	NO	NO
EDCF	NO	NO	N/A	NO
EPCF	YES	NO	N/A	YES
Master-Slave	YES	N/A	YES	NO, but can be done

packet type to classify the packets. But their work mainly focuses on indoor WiFi networks. They classify the packets on a single host into Real and Non-Real Time traffic and do not address issues related to cross traffic from other sources in the same network. They did not implement the protocol they have proposed. Implementing any protocol at driver level involves a lot of overhead, which the authors do not take into consideration and we have addressed these issues.

3.4 Our Approach

Master-Slave protocol is different from each of the above schemes. Our solution works at driver level. It is very easy and virtually costs nothing to make changes at driver level. It can be implemented on any of the technologies as the underlying MAC. Our protocol cleverly suppresses the behaviour of underlying MAC protocol, since the driver can instruct the NIC for transmissions without much delay. Though our approach is similar in functionality to PCF of 802.11b, the place where they operate are different. While PCF (if supported) operates in NIC, our protocol works above the firmware of NIC. Modifying a protocol at driver level is relatively easy to modifying the same in NIC. Other techniques like packet aggregation, windowing mechanism can be easily implemented at driver level.

Briefly we compare our work with IEEE802.11e and IEEE802.11b in Table 3.1.

Chapter 4

Performance of Host AP on a Long Distance 802.11b Link

Since we are going to modify the Host AP driver, first we need to check whether it performs well in a long distance wireless link or not. In this Chapter we will verify whether the combination of Linux + Host AP + prism2 card is deployable in a long distance wireless link or not. First we present the details of theoretical estimation of throughput on a long distance 802.11b link, followed by experimental values. After that we compare the theoretical throughput with the experimental value.

802.11b MAC operates in two modes, namely DCF and PCF. While DCF mode is contention based, PCF is contention free. In PCF the access point polls each of the clients for any outstanding frames in an order. But PCF is not supported in most of the commercial products. Hence we consider only DCF here.

The 802.11b frame consists of the following fields.

Preamble(18 bytes)	PLCP Header(6 bytes)	MacData
--------------------	----------------------	---------

MacData consists of the following fields.

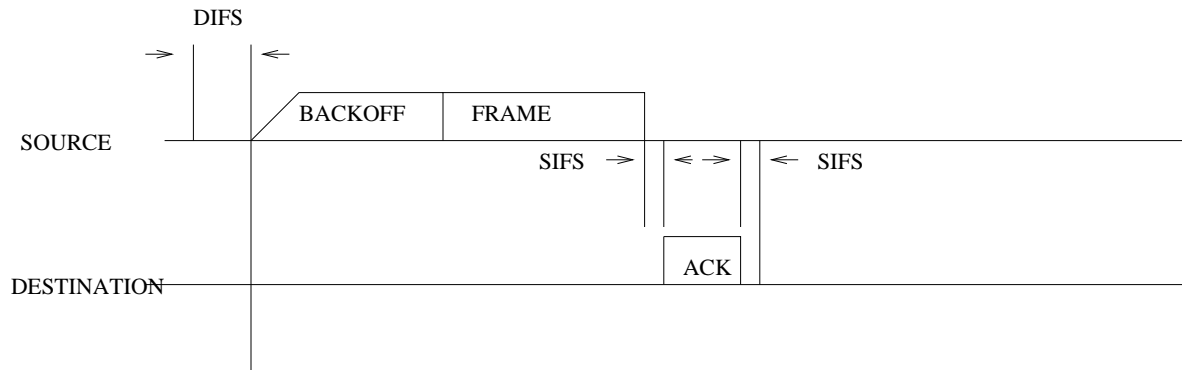


Figure 4.1: 802.11b Timing Diagram

Frame-Control(2 bytes)	DurationID(2 bytes)	Address1(6 bytes) -
Address2(6 bytes)	Address3(6 bytes)	SequenceControl(2 bytes) -
Address4(6 bytes)	FrameBody(0-1470)	CRC(4 bytes)

For better understanding, part of the timing diagram of 802.11b MAC is given in Fig 4.1.

Preamble , and PLCP Header fields are always sent at 1Mbps.

The time taken to send 24 byte preamble and PLCP Header at 1Mbps = 192μ sec.

The time taken to send 34 bytes Frame control data ($34 \times 8 / 11 \mu$ sec) at 11Mbps = 24.72μ sec.

The time taken to send the UDP(8 bytes) and IP(20 bytes) headers at 11 Mbps = 20.36μ sec.

The time taken to send x bytes of data at 11Mbps = $x \times 8 / 11 \mu$ sec.

ACK Frame

The time taken to send 24 byte preamble and PLCP Header at 1Mbps = 192μ sec

The time taken to send 14 byte data at 11Mbps = 10.18μ sec

The values of various constants in the timing diagram of 802.11b are -

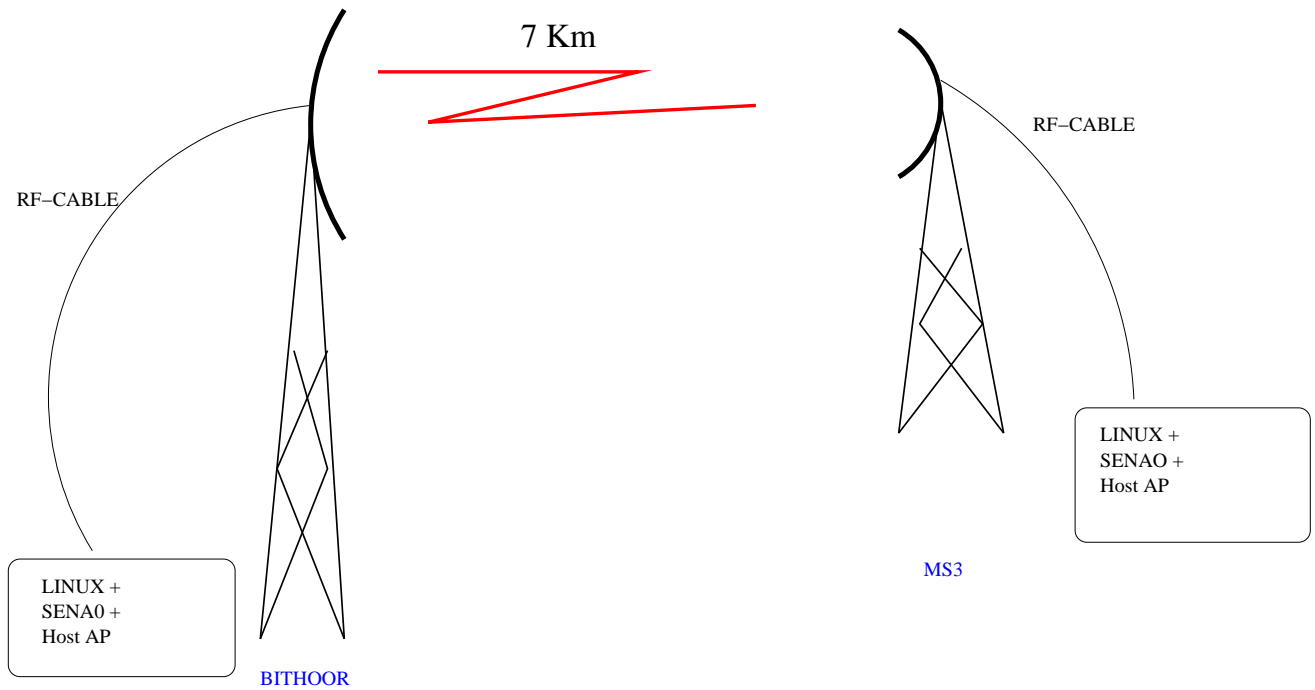


Figure 4.2: Setup between Bithoor and MS3

DIFS = 50 μ sec

Avg CW = 320 μ sec

SIFS = 10 μ sec

So the total time required to send data of size x bytes = $(819.26 + 8*x/11)$ μ sec.

Therefore the throughput = $(8*x)/(819.26 + 8*x/11)$ Mbps

Table 4.1 lists theoretical and experimental values of the throughput for different packet sizes. Our experiment was conducted on MS3-Bithoor link of DGP Testbed[4]. The experimental setup is as shown in Fig 4.2.

It is a 7Km link. At MS3 there was a tower which was 40ft in height. A 24dbi-gain parabolic grid antenna was mounted on that tower facing to wards Bithoor. At Bithoor there was a tower which was around 50ft in height. A 24dbi-gain parabolic grid antenna was mounted on that tower which is facing to wards MS3. At both the sites we used senao cards as the radios. At each site the radio is connected to the antenna through a pigtail and RF-Cable. At each site the transmission power

Table 4.1: Experimental and Theoretical Throughput Values

Packet Size (bytes)	Theoretical Throughput (Mbps)	Experimental Throughput (Mbps)
40	0.38	0.37
180	1.51	1.5
200	1.6	1.55
1000	5.17	4.8
1470	6.22	6.1

of the signal is 20dBm. The signal and noise levels at Bithoor are -63dBm and -95dBm respectively. The signal and noise levels at MS3 are -74dBm and -95 dBm respectively. These power levels are enough to transmit the data at 11Mbps[13]. Each experiment was conducted for 200 seconds.

From the above table we can observe that the theoretical and experimental values are almost same. Hence the combination of Linux+Host AP+prism2 card is deployable as an alternative to commercial access points. An additional advantage here is that the overall cost per access point would be reduced. It costs around 200\$(100\$ for prism2 card and 100\$ for Linux box), whereas a commercial access point costs around 600\$.

Moreover the jitter (One of the metrics of QoS) values and packet loss are also in acceptable range if the traffic rate is with in the supported range. Table 4.2 lists the jitter values for different rates of traffic by keeping the packet size at 180bytes. The first column specifies the rate of the data transmitted in one second. The second and third columns specify the jitter (in msec) and packet loss (%) respectively. Jitter is measured as specified in RFC 1889. Traffic is generated at uniform rate using the tool iperf[14].

Table 4.2: Jitter & Packet Loss Values for Different Rates of Traffic

Rate of Data Transmitted (Mbps)	Jitter Values (msec)	Packet Loss (%)
0.4	0.364	0
0.5	0.536	0
1	0.424	0
1.5	0.546	0.048
1.6	0.336	6.4

Chapter 5

Master-Slave Protocol

We start this Chapter with a discussion on the performance of an 802.11b MAC from the perspective of QoS, show its bad performance in the presence of two or more simultaneous transmissions to a common node. Later we discuss an alternative, driver based approach (Master-Slave protocol) to improve the performance. We confine our entire discussion to 1-hop networks.

5.1 Performance Analysis of VoIP on DCF

In this section we will analyze the performance of VoIP on DCF. We assume that Original Host AP Driver sits on the top of DCF as the network driver. Before going further, we have to recapitulate the fact that DCF (Distributed Coordination Function) is a contention based protocol. In long distance 802.11 networks no station (other than land line node) can hear the transmission of others. Consequently the probability that hidden node problem occurs is high resulting in MAC level collisions. High number of MAC level collisions directly implies excessive retransmissions which results in excessive delay. Also, when two or more stations sharing the same channel, try to transmit at the same time, none is sure of who gets the transmission opportunity. So when all the stations have frames to transmit we are not sure of who will transmit next resulting in variation in the inter transmission time between two consecutive frames.

5.1.1 QoS on DCF: An Experiment

To know how the QoS of any traffic stream would be effected in the presence of cross traffic, we see the results of one of our experiments on the testbed. Our experiment setup is as follows. We have 3 laptops with Senao cards (connected to the high gain antenna through RF-Cable) inserted into the PCMCIA slots of each of them. Host AP is installed as the network driver on all the laptops. All of them are in ad-hoc mode.

We conducted the experiment on the testbed of DGP[1]. We selected FBTOP, MBSNL and Mohanpur as the candidate sites for our experiment. FBTOP is top of faculty building at IITKanpur. MBSNL is located at bsnl office of Mandhana (A village which is around 5Km from Faculty Building of IITK). Mohanpur is a village which is around 2.5Km from the Faculty Building of IITK. Refer [4] to see the geographical locations of these three sites.

2.4.20-8 kernel (compiled) is the underlying kernel on all the laptops. The version of Host AP is 0.2.4. The type of senao card used in our experiments is SL-2511CD PLUS EXT2. We have three laptops with Linux + Host AP + senao card combination on each of them. We call these three laptops as fbtop,mbsnl and mpur. fbtop can hear both mbsnl and mpur. Both mbsnl and mpur can hear fbtop and cannot hear each other. So when either mbsnl or mpur transmit frames to fbtop, there is a chance that hidden node problem occurs.

At MBSNL and Mohanpur the antennas are mounted on towers. The heights of the towers at mbsnl and mpur are are 20ft and 40ft respectively. At FBTOP the antenna is mounted on the faculty building of IITK, which is around 60ft in height. At FBTOP the senao card is connected to external antenna through Pigtail \iff Splitter \iff RF-Cable. At MBSNL and Mohanpur senao card is connected to external antenna through Pigtail \iff RF-Cable.

The setup is as shown in Fig 5.1.

The txpower, signal level and noise level are as shown in Table5.1. These power levels are enough to keep all the transmissions at 11Mbps[13]. Between fbtop and mpur, bidirectional UDP traffic is generated at an uniform rate of 68.8Kbps with

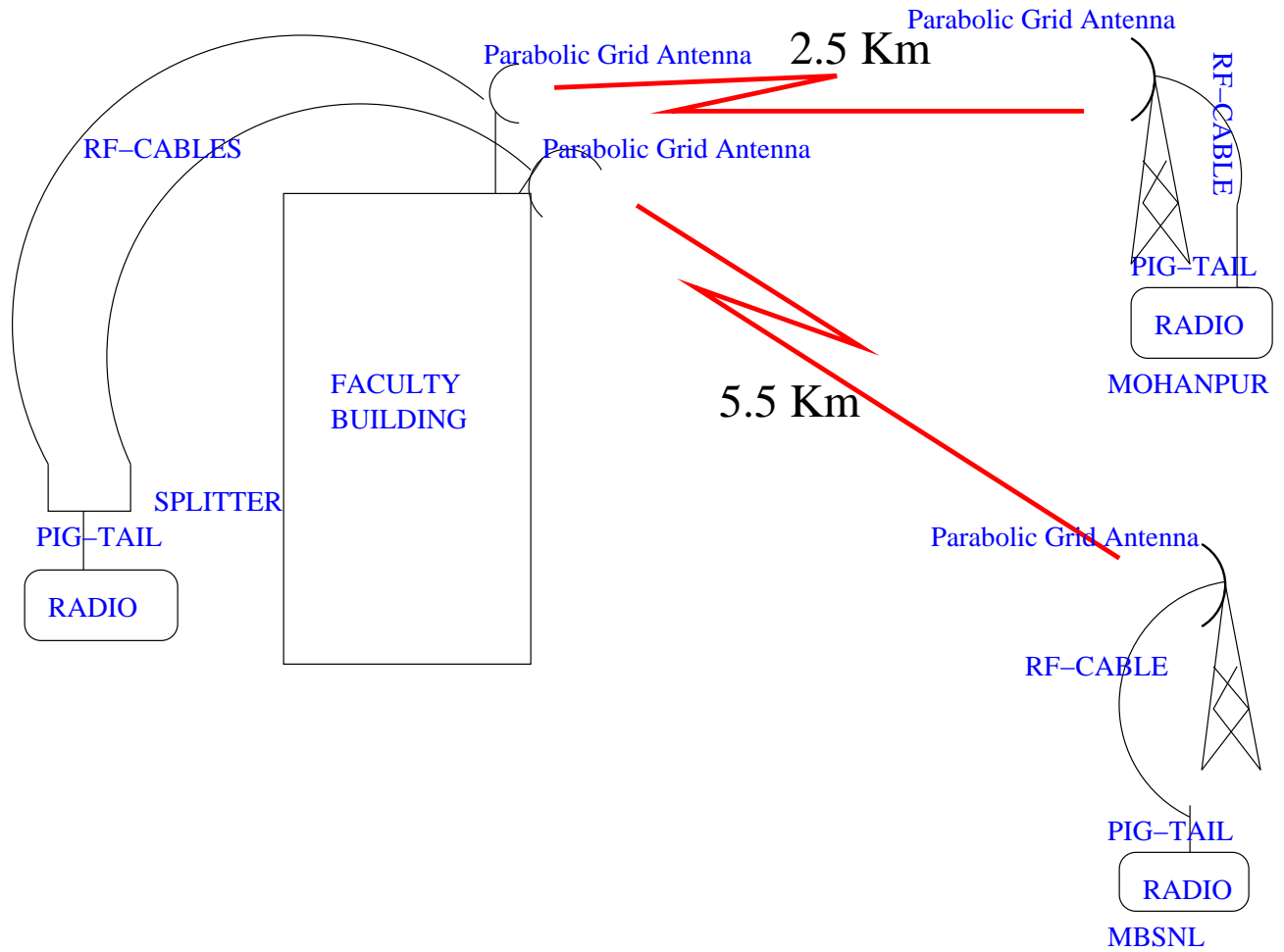


Figure 5.1: Experimental Out-door Setup

Table 5.1: Power Levels at all the sites

-	Txpower(dBm)	Signal Level(dBm)	Noise Level(dBm)
FBTOP	20	-67	-95
MBSNL	20	-71	-95
MPUR	20	-63	-95

the size of each packet being 172bytes. The packets are generated periodically with each packet being generated for every 20msec. Note that we will have the same traffic pattern for a VoIP connection with g711 as the codec. We measured the jitter values of the VoIP stream by varying the rate of UDP cross traffic (between fbtop and mbsnl) by keeping the size of the packet at 1200bytes. When the cross traffic rate is 2Mbps (bidirectional traffic 1Mbps in each direction) the jitter value is around 1msec, which is within the acceptable range. But when the cross traffic rate is increased to 4Mbps (bidirectional traffic 2Mbps in each direction) the jitter value reached to 80msec. The jitter value further gone to 190msec at some point of time, when the rate of the cross traffic rate is increased to 6Mbps. And the packet loss is also increased to more than 50%. Any network on which one would see a jitter value of 190msec and a packet loss of 50% is not suitable to offer time critical and QoS demanding Real-Time services.

5.1.2 Analysis of the Results

First we see whether the network is capable of supporting the above streams. In one second 50 packets each of size 172bytes are sent from fbtop to mpur and 208 packets each of size 1200bytes are sent from fbtop to mbsnl. Since the traffic is bidirectional 100 packets are transmitted between fbtop and mpur and 416 packets are transmitted between fbtop and mbsnl.

The time required to transmit one packet of size 172 bytes = $944.35\mu\text{sec}$.¹

The time required to transmit one packet of size 1200 bytes = $1691.98\mu\text{sec}$.

Therefore the time required to transmit 100 packets of size 172 bytes = 94.43msec

The time required to transmit 416 packets of size 1200 bytes = 703.86msec .

So the total time required to transmit all the packets = 798.29msec .

From the above value we can observe that the time required to transmit all the packets generated in one second is less than one second. Clearly the network is

¹ $819.26+8*172/11=944.35$, Refer Performance Analysis section in Chapter 4 for more details

capable of holding these traffic streams.

Since theoretically the network is capable of supporting the two bidirectional traffic streams, 68.8Kbps and 2Mbps, one may expect a smooth transmission of data. But in reality it doesn't seem so. As we discussed in the beginning of this section, each station has to wait for a different amount of time for each transmission, resulting in variation of inter arrival time at the receiver which results in the high values of jitter and there would be MAC level collisions due to hidden node problem resulting in packet loss.

An alternative to DCF is PCF (point Coordination Function) to support time bound traffic. But as told in Chapter 3, it is not supported by any known existing commercial NICs. Had PCF been supported by the NICs, time bound services would have been provided on wireless networks. Modification at MAC layers requires modification of firmware in the NIC and vendors are not interested in doing it.

5.2 A Driver Based Approach

In this section we propose a driver based approach to improve the performance. The main advantage with this approach is that we do not make any changes to the firmware. We propose a PCF kind of protocol at driver level to support time bounded traffic. The underlying principle here is to avoid contention (there by collision) at MAC layer. We made changes to network driver, is sitting on top of MAC and implements the MAC Management Protocol. The position of the network driver in the protocol stack is shown in Fig 5.2. The driver implements MAC management layer, which is at the same level as the MAC layer.

Any packet that is generated at upper layers reaches to MAC layer via network driver. We take the advantage of this and implement packet control mechanism at the driver level which is similar in functionality to PCF. Like in PCF here also there would be a special node which controls the access to channel. That special node is called as master and the remaining nodes are called as slaves.

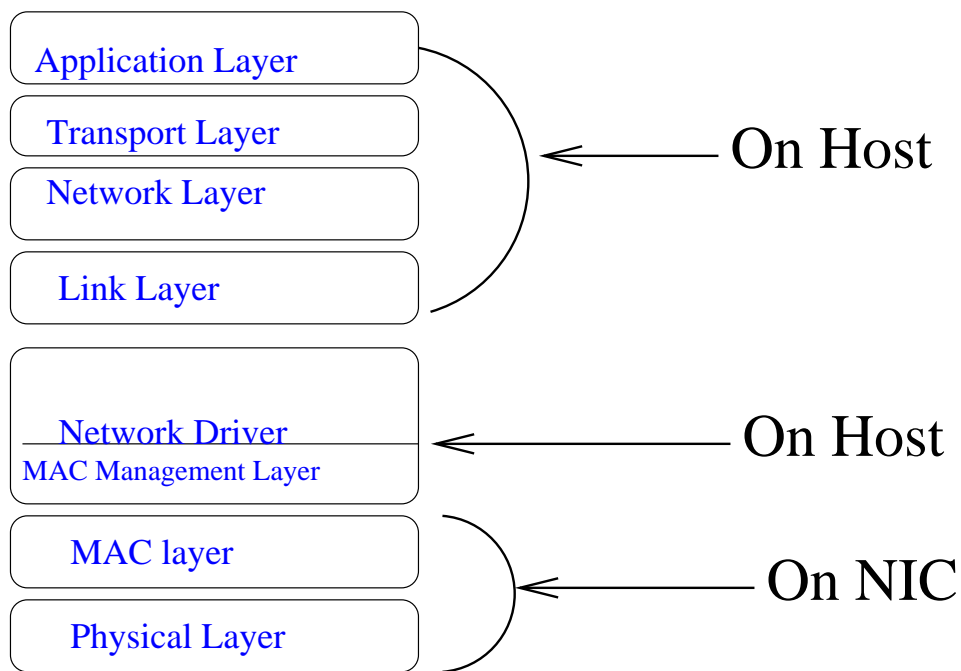


Figure 5.2: Protocol Stack and Network Driver

Each node in the network including the master node is given one driver level unique ID. This ID can be MAC Address also. But to make the implementation simple we use a different set of IDs. This ID is very crucial in the communication and may not be concerned with MAC and IP addresses of the nodes. The master maintains the list of slaves. It probes the slaves in the order of their IDs. A slave node can transmit only when it is probed. The master maintains a queue corresponding to each of the slaves. We call them as Neighbour Queues, shortly Nbr-Queues. There is a one-to-one relationship between the Nbr-Queues and slaves. In each of the Nbr-Queues it buffers the frames those are destined for the slave corresponding to that queue. Each of the slaves also maintains a single queue and stores the frames those are destined for the master.

Since we use driver level addressing, we need to ensure that all the frames reach to the driver level. At MAC level all the frames are broad cast frames. Otherwise at any node, only those frames which are destined for that node reach to the driver level. Frames which are intended to be broadcast type at MAC layer also would be

5.2.2 Protocol at Master

The protocol at master is as follows. Note that any frame that comes from link layer would be *enqueued* into any of the Nbr-Queues maintained at driver. The Nbr-Queue that the frame is to be enqueued into would be selected based on the destination address. The destination address of the frame is modified and make it as a broadcast frame. The main reason behind broadcasting is to make the frame reach the driver level at all the slaves (point-to-multipoint link communication). This is required since we use driver level addressing in our protocol. One additional advantage is that for broadcast frames there would not be MAC level ACK frames (In 2P[2] also there would not be ACK frames). Consequently the ACK overhead would not be there in transmissions. MAC level ACK frames are not necessary since collisions are avoided in our protocol. Problem occurs when a frame is lost due to weak signal. But this is very rare in practice and can be handled by the timer that is maintained at master.

When the master is up, it would probe the slaves in the order of driver level IDs of the slaves. Probing any slave is a 2-step process. In the first step the master dequeues a frame (if the queue is not empty, otherwise a dummy frame) from the Nbr-Queue corresponding to the currently probed slave, constructs a MAC level broadcast frame, fills the *srcID* field with its own ID and *dstID* field of frame with the ID of the slave that is to be probed in next slot, broadcasts that frame and starts a timer. In the second step, the slave which is probed responds if it has any outstanding frames to send. If it doesn't have any outstanding frames it sends an empty marker frame.

When a frame is received from the slave, it probes the next slave in the list. In case any frame is lost in this communication, the timer at the master expires and the master probes the next slave in the sequence. When a frame comes from the upper layers it would be copied into one of the Nbr-Queues. The queue is determined by looking at the destination address.

5.2.3 Protocol at Slave

The protocol at the slave is simple. When it is up, it waits for the driver level probe frame from the master. When a frame comes from the upper layers, it enqueues that frame into the queue it maintains. When it receives a frame it checks whether it is its turn to transmit. If so it would transmit a frame (if it has any outstanding frames, otherwise any empty marker frame).

5.2.4 Overheads & Optimizations

When it is the turn of a node to transmit, the network driver in that node has to copy that frame into one of the buffers of the NIC and instruct the NIC to transmit into air. But copying the frame into the buffer of the NIC consumes significant amount of time, since NIC is an external device. We call this as *to-bap-overhead*. This overhead can be reduced by copying the frame into the buffer before hand and instruct the NIC to transmit it into air when it is its turn to transmit.

When a frame is received by the slave it would check the dstID field of the frame to take the decision of whether it is its turn to transmit (since there are many slaves). So it needs to copy the frame from hardware (NIC) to the driver level buffer to check the dstID. But copying the frame from hardware takes significant amount of time, which results in less throughput, slightly higher values of jitter (around 5-7msec with our experiment) and delay. We call this as *from-bap-overhead*. Here also the overhead can be reduced by making the slave intelligent enough, so that it would decide before hand whether it is its turn to transmit.

These optimizations are analogous to *pipelining* phase in execution of instruction at hardware level in computers. Fig 5.5 show the sequence of steps in the Master-Slave protocol and Fig 5.6 show the sequence of steps done in the optimized Master-Slave protocol in the communication between master and slave. The column Time gives the sequence of actions in the order. The column "At Master" gives the sequence of events happening at master. The column "At Slave" gives the sequence of events happening at slave. In both Fig 5.5 and Fig 5.6 T represents the action, transmission into air. C1 represents copy of a frame from queues at driver into

TIME	AT MASTER	AT SLAVE
0	C1	
1	T	
2		C2
3		D
4		C1
5		T
6		

Figure 5.5: Master-Slave Protocol

TIME	AT MASTER	AT SLAVE
0	T	
1	C1	T
2	T	C2, D, C1

Figure 5.6: Optimized Master-Slave Protocol

the buffer of NIC. C2 represents the action of copying of a frame from NIC buffer into driver level buffer. D represents the action of decoding the received frame and sending it to higher layers.

From Fig 5.5 and Fig 5.6 it is evident that the amount of time that one can save with the optimized version of the Master-Slave protocol is the time consumed by the actions $2 * C1 + C2 + D$. The time taken by the actions C1 and C2 may be high, since those involve copying from/to external device. In optimized version of the protocol, any slave would respond to the probe packet from master immediately after it sees the frame (if it is its opportunity to transmit), before it reads the frame from NIC buffer. After transmitting the frame only it would read the received frame from NIC buffer and copy the new frame to be transmitted into NIC buffer. It is analogous to pipelining concept in Computer Architecture. For any slave the

information about its transmission opportunity would be conveyed by the master in the previous probe itself. In the next section we see how these techniques of reducing the overhead are implemented.

5.3 Implementation of the Protocol

In this section we discuss the implementation details of the protocol. We present the details of how the master and slave responds for various events occur at them.

5.3.1 Protocol at Master

Here we present the details of the protocol at the master side.

■ *Data Structures and Variables at Master*

```
int idx=0;
    int nextSlave=1,prevVal=1;
    buffer *Nbr-Queues[NumOfNbrs];
    u8 *BAP[MaxNumOfBuffersAtCard];
    u8 slaveList[NumOfNbrs];
    txdesc txframe;
```

BAP (Buffer Access Path) is the array of buffers maintained by the NIC. `idx` is a variable used for indexing the BAP. `Nbr-Queues` is the list of queues maintained by master. One queue is maintained corresponding to each of the slaves. `slaveList` is the list of ids of the slaves those are to be probed. `txframe` is a C type structure and contains fields to hold header of the frame followed by payload. `nextSlave` is used to fill the `dstID` field of the frame. `prevVal` is the previous value of the `nextSlave`. The importance of each of these would be clear once we enter into the details of the protocol.

■ When the Master is UP

When the node is up, the following are the steps to be done.

- Initialize each of the Nbr-Queues to NULL.
- Broadcast a frame which is destined to the first slave in the slave list and start the Timer. Suppose the id of the first slave is slave1 and the id of the slave that is to be probed in next slot is slave2. Give sufficient information in the broadcasted frame to make slave2 to be ready for transmission in next slot, so that the from-bap-overhead is reduced at slave2.
- Dequeue a frame (if there is any, otherwise an empty marker frame is to be sent) from the queue corresponding to slave2 and copy it into one of the available buffers at NIC to reduce the to-bap-overhead at the master.

The sequence of steps to be done are listed in more detail.

Initialize each of the Nbr-Queues to NULL.

prevVal=nextSlave;

nextSlave=nextSlave+1;

if(nextSlave>NumOfNbrs)

nextSlave=1;

Construct a new Frame, *txframe* with srcID=ID of master and dstID=slaveList[nextSlave].

The payload of *txframe* is to be filled with data destined (if there is any) to the first slave in slavelist of master.

Get idx, the index of the next available buffer in BAP.

Copy the *txframe* into BAP[idx].

Transmit the frame which is in BAP[idx] into air.

Start Timer

Let idx' be the index of next available buffer in BAP.

idx=idx';

```

prevVal=nextSlave;
nextSlave=nextSlave+1;
if(nextSlave>NumOfNbrs)
    nextSlave=1;
}
Construct a new Frame, txframe with srcID=ID of master
and dstID=slaveList[nextSlave].
//The payload of txframe is to be filled with data destined
//(if there is any)to the slaveList[prevVal] of master.
if(Nbr-Queues[prevVal] is not empty){
    Dequeue a frame from Nbr-Queues[i].
    Insert the payload of the dequeued frame into the payload
    of txframe.
}
Copy txframe into BAP[idx];
break;

```

■ *When a Frame is Received or Timeout*

Precisely the following are the steps to be done when this event occurs.

- Transmit the frame which is in the buffer of the NIC (The index of the buffer is given by *idx*) and start the Timer.
- Let the *dstID* of the slave that is to be probed in next slot is *slave2*.
- Dequeue a frame (if there is any, otherwise an empty marker frame is to be sent) from the queue corresponding to *slave2* and copy it into one of the available buffers at NIC to reduce the to-bap-overhead at the master.

In detail these following are the sequence of steps to be followed.

```

    Transmit the frame which is in BAP[idx] into air.
Start Timer
    Let idx' be the index of next available buffer in BAP.
    idx=idx';
prevVal=nextSlave;
    nextSlave=nextSlave+1;
if(nextSlave>NumOfNbrs)
    nextSlave=1;
}
Construct a new Frame, txframe with srcID=ID of master
and dstID=slaveList[nextSlave].
//The payload of txframe is to be filled with data destined
//(if there is any)to the slaveList[prevVal] of master.
if(Nbr-Queues[prevVal] is not empty){
    Dequeue a frame from Nbr-Queues[i].
    Insert the payload of the dequeued frame into the payload
    of txframe.
}
Copy txframe into BAP[idx];
break;

```

■ *Frame Arrival From Link Layer*

When a frame is arrived from the link layer, the following is the sequence of steps to be done.

- The destination address field is to be changed and make it as a broadcast frame.
- Next it is to be enqueued into one of the Nbr-Queues based on the (original)

destination address.

The pseudo code for this event is as follows.

Determine the index i of the queue in the Nbr-Queues
into which the frame is to be enqueued.

Change the destination address field of the frame to make
it a MAC level broadcast frame.

Enqueue the frame in Nbr-Queues[i].
break;

5.3.2 Protocol at Slave

In this subsection we present the details of protocol at slave.

■ *Data Structures and Variables Used at Slave*

```
int idx;  
buffer* Nbr-Queue-Master;  
int nextTurn=1;  
rxdesc rxframe;  
txdesc txframe;  
int firsttx=true;  
u8 *BAP;
```

idx is used to index the buffers in NIC. Nbr-Queue-Master is the only queue maintained at slave and any frame coming from link layer is enqueued there. rxframe is a C type structure which contains received frame header and payload. txframe is a C type structure which contains frame header and payload fields for the transmitted frame. firsttx is a variable used decide whether a transmission is the first transmission or not.

■ *Frame Arrival From Link Layer*

When a frame arrives from link layer, its destination address field is changed to make it a MAC level broadcast frame and is enqueued into Nbr-Queues-Master.

■ *Frame Reception*

Briefly the following are the steps to be carried out when a frame is received.

- Transmit the frame which is in the buffer of the NIC (The index of the buffer is given by *idx*) if it is its turn to transmit.
- Dequeue a frame (if there is any, otherwise an empty marker frame is to be sent) from Nbr-Queue-Master and copy it into one of the available buffers at NIC to reduce the to-bap-overhead at slave.
- Read the received frame from buffer and gather the information about whose turn it is to transmit in next probe.

Pseudo code follows.

```
if(firsttx==true){
    Let idx' be the index of next available buffer in BAP.
    idx=idx';
    Construct a new Frame, txframe with srcID=ID of slave
    and dstID=ID of the master
    //The payload of txframe is to be filled with data destined
    //(if there is any)to the master.
    if(Nbr-Queue-Master is not empty){
        Dequeue a frame from Nbr-Queue-Master.
        Insert the payload of the dequeued frame into the payload
        of txframe.
    }
```

```

    Copy txframe into BAP[idx];
    firsttx=false;
}
if(nextTurn==myID){
    Transmit the frame which is in BAP[idx] into air.
    Let idx' be the index of next available buffer in BAP.
    idx=idx';
    Construct a new Frame, txframe with srcID=ID of slave
    and dstID=ID of the Master
    //The payload of txframe is to be filled with data destined
    //(if there is any)to the master.
    if(Nbr-Queue-Master is not empty){
        Dequeue a frame from Nbr-Queue-Master.
        Insert the payload of the dequeued frame into the payload
        of txframe.
    }
}
Copy txframe into BAP[idx];
Copy the received frame, rxframe from BAP to driver level buffer,
say buff.
nextTurn=rxframe.dstID;
break;

```

5.4 Drawbacks

There are some drawbacks also to our Master-Slave protocol. They are -

- Since our protocol is a TDMA based protocol, sometimes the channel may be under utilized. This is the main drawback of our approach. But we can defeat

this by introducing a little bit of dynamic nature into our protocol. We can do this by manipulating the functions responsible for transmission dynamically. We will start the system with functions corresponding to Original Driver. Since we can know the number of MAC level errors at driver level, we can switch dynamically to the functions corresponding to Master-Slave Driver when the number of errors exceed a threshold. And also we can switch to the functions corresponding to Original Driver when there are too many dummy marker packets.

- No priority is given to Real-time services over best effort services. But we can implement EPCF kind of protocol, by making small changes to the existing Master-Slave protocol to prioritize the frames.

Chapter 6

Results

In this Chapter we discuss the results we got during our experiments in indoor and outdoor setups. We start our discussion with the results we saw in out-door setup since our protocol is intended for out-door usage. Next we discuss the results we observed in indoor setup.

6.1 Results in Outdoor Setup

Our aim here was to analyze the performance of VoIP in a point-to-multipoint 802.11b network in the presence of cross traffic. We established one traffic stream between fktop and mpur. The traffic pattern was like a VoIP connection with g711 codec ie we transmitted 50 packets per second each of size 172bytes (160bytes packet size + 12bytes RTP header) at uniform rate from fktop to mpur and vice-versa.

6.1.1 Cross Traffic is UDP

We generated the UDP cross traffic between mbsnl and fktop with the size of each packet being 1470bytes (bidirectional traffic with equal number of packets in each direction). By varying the rate of cross traffic (keeping the size of the packet unchanged) we measured the jitter and packet loss at both ends of the VoIP connection. All the packets are transmitted at 11Mbps transmission rate. In all experiments the

jitter values are measured as specified in RFC 1889[20].

■ *Original Host AP Driver*

The txpower, noise level and signal level at each of these sites are shown in Table 5.1. These power levels are enough to keep the transmissions at 11Mbps[13]. The hardware and software used is the same as we discussed in Section 5.1. We observed that though the network was capable of handling the overall traffic generated, there was significant packet loss and high jitter values for VoIP packets. As we discussed in Chapter5 this can be attributed to MAC level collisions and variation in inter transmission times. We don't go for RTS/CTS mechanism since as we have discussed in Chapter3 that the overhead associated with it is too high for a VoIP packet. And also it can reduce the number of collisions but cannot completely eliminate them.

Jitter values are listed in Table 6.1 when the rate of the cross traffic is varied by keeping the size of the packet as 1470bytes. Column1 specifies the number of packets per second transmitted (cross-traffic) between fbttop and mbsnl. Column2 specifies the jitter values of the VoIP flow from mpur to fbttop and column3 specifies the jitter values of the VoIP flow from fbttop to mpur. We follow the same convention throughout this section for all the tables in which jitter values are listed. Each experiment was run for 50 seconds. We conducted seven experiments by varying the number of packets in the cross traffic from 0 to 600 with an increment of 100. Each row corresponds to one experiment. Jitter values are measured for each second. The maximum of the those jitter values in each experiment is given in corresponding row. We will analyze the results once the results with Master-Slave driver are also presented.

Packet loss values are listed in Table 6.2 when the rate of cross traffic is varied between fbttop and mbsnl. Column1 specifies the number of packets per second transmitted (cross-traffic) between fbttop and mbsnl. Column2 specifies the packet loss values of the VoIP connection at fbttop and column3 specifies the packet loss values of the VoIP connection at mpur. We follow the same convention throughout

Table 6.1: Jitter Values with Original driver

Number of Packets/sec (Cross Traffic)	Jitter Values for the VoIP Flow from-mpur-to-fbtop (msec)	Jitter Values for the VoIP Flow from-fbtop-to-mpur (msec)
0	3.816	0.173
100	0.455	0.289
200	8	8
300	45	50
400	51	67
500	63	106
600	58	150

Table 6.2: Packet Loss with Original Driver

Number of Packets/sec (Cross Traffic)	Packet Loss for the VoIP Flow from-mpur-to-fbtop (%)	Packet Loss for the VoIP Flow from-fbtop-to-mpur (%)
0	0	0
100	0	0
200	0	0
300	13	37
400	13	49
500	13	44
600	20	64

this section for all the tables in which packet loss values are listed. The packet loss values of the VoIP stream in the 50 seconds duration are given in each row of Table 6.2.

■ *Master-Slave Driver*

With Master-Slave driver, the power levels are same as those with the Original Driver. The setup is the same as above except that the Original Driver is replaced by Master-Slave Driver. Jitter values and packet loss are listed in Table 6.3 and Table 6.4 respectively. fbtop is the master node and mpur and mbsnl are slave nodes. Each experiment was run for 50 seconds. We conducted seven experiments by varying the number of packets in the cross traffic from 0 to 600 with an increment

Table 6.3: Jitter Values with Master-Salve driver

Number of Packets/sec (Cross Traffic)	Jitter Values for the VoIP Flow from-mpur-to-fbtop (msec)	Jitter Values for the VoIP Flow from-fbtop-to-mpur (msec)
0	3.979	0.173
100	3.979	4.231
200	3.182	3.213
300	2.733	2.392
400	3.330	3.009
500	2.792	3.733
600	2.377	3.259

Table 6.4: Packet Loss with Master-Slave Driver

Number of Packets/sec (Cross Traffic)	Packet Loss for the VoIP Flow from-mpur-to-fbtop (%)	Packet Loss for the VoIP Flow from-fbtop-to-mpur (%)
0	0	0
100	0	0
200	0	0
300	0	0
400	0	0
500	0	0
600	0	0

of 100. Each row corresponds to one experiment. The jitter values of the of the VoIP stream in the 50 seconds duration are given in each row of Table 6.3. The packet loss values of the VoIP stream in the 50 seconds duration are given in each row of Table 6.4.

■ *Analysis of the Results*

We start our discussion with the calculation of the maximum number of packets per second in the cross traffic. We know that 100 packets are generated in a second for a VoIP connection with g711 as the codec.

The time required to transmit one packet of size 172 bytes = $944.35\mu\text{sec}$.

Time required to transmit 100 packets of size 172bytes= 94.43msec .

Therefore the maximum amount of time (in one second) that the channel is idle = $1000 - 94.43 = 905.57$ msec.

The size of each packet in our cross traffic is 1470 bytes.

The time required to transmit one packet of size 1470 bytes = 1888.35 μ sec.

Therefore the maximum number of packets that can be transmitted in the idle time = $905.57 / 1.888 = 480$.

It means that we can ensure smooth communication until the number of packets in the cross traffic does not go beyond 480. But the practical results are quite contradictory. We can observe from Table 6.2 that the loss is 13% and 37% at the two ends of the VoIP communication when number of packets in the cross traffic is 300. And also observe in Table 6.1 that the jitter values are 45msec and 50msec (which are quite high) at the two ends of the VoIP connection. The jitter and packet loss values are increased with the increase of the cross traffic. The most important thing we can infer here is that, with the Original Driver the jitter and packet loss values are not in the acceptable range, even if the total traffic generated is in the supported range of the network.

In all the experiments we have VoIP connection between fbtop and mpur. In each experiment we changed the number of packets generated in one second in the cross traffic between fbtop and mbsnl. As we already said each row in the tables correspond to one experiment. All the experiments in our setup lasted for 50 seconds. We measured the jitter values for each second and also the cumulative jitter in the duration of 50 seconds as specified in RFC 1889[20]. In Table 6.1 the jitter value in any row for a VoIP flow gives the maximum of the jitter values we have observed for that flow during the experiment corresponding to that row. The reason for taking the maximum value is as follows. Since a VoIP service with g711 as the codec cannot tolerate a loss of more than 1%, it is sufficient to show that there would be a loss of atleast 1% with Original Driver to prove that it is not suitable to offer VoIP services. In a VoIP flow a jitter value of more than 40 msec during a a second implies that all the packets are lost during that period. The number of packets generated in the VoIP flow during 1 second amount to 2% of the total number of packets generated in 50 seconds. So it is sufficient to give the maximum of the jitter values measured

for each second to prove that DCF is not suitable to provide Real Time Services.

The main reason for high values of jitter with Original Driver is the occurrence of *hidden node problem*. In our setup due to hidden node problem, at MAC layer the data frames of VoIP stream (from mpur to fbtop) may collide with the data frames of the cross traffic (from mbsnl to fbtop). Similarly the MAC level ACK frames of VoIP stream (from fbtop to mpur) may collide with the data frames of the cross traffic.

In the first case the frames would be lost in both the traffic streams resulting in MAC level re-transmissions. In the second case the fbtop would not receive ACK frame from mpur and try to retransmit the frame though it is already received by mpur, there by increasing the congestion in the network. There would be variation in the inter arrival time at the receiver due to the following reasons. For every collision the contention window is doubled. Each station try to retransmit a frame until the limit on the maximum number of retransmissions is reached. All the frames from a source may not be retransmitted for the same number of times. This is one reason for the variation in the inter-arrival times. The stations give up for transmitting the frame when the limit on the maximum number of retransmissions is reached. It means that frame is dropped at sender side MAC itself. Now next frame generated from upper layers is to be transmitted. Thus the inter transmission time between two successive frames may be different, which leads to the variation in the inter arrival time at the receiver. The frames may not be in sequence, because some of the frames in the middle may be dropped by MAC. This is another reason for the variation in the inter arrival times. So the MAC layer itself is responsible for the variation in inter arrival time at the receiver resulting in high values of jitter and is not suitable to offer Real Time services. When the station tries to transmit each of the frames by attempting retransmissions, obviously the delay to send that frame would be high. And the packets generated at upper layer in the mean time would be queued and packets would be dropped once the limit on the number of packets in the queues is reached. This also results in the variation of inter-arrival time between the frames (the frames may not be in sequence) at the receiver.

Logical Link Control Layer is also responsible to some extent for the variation.

When the sender does n't receive the acknowledgment for a frame (lost due to collision) it would retransmit the frame. The retransmitted frame would compete with the freshly generated frame to grab the channel. This would lead to the increase in the amount of overall traffic which in fact leads to more uncertainty in who would grab the channel next, thereby increasing the variation.

Now we discuss the packet loss values of VoIP stream. We can observe that packet loss values are increased with the increase in the number of packets/sec in the cross traffic. There is packet loss in the VoIP stream even when the over all traffic in the network is within the acceptable range. Once again this can be attributed to hidden node problem. Due to hidden node problem, MAC level collisions would occur frequently and frames would be dropped after the limit on the maximum number of retransmissions is reached. This is one reason for packet loss. When the station tries to transmit each of the frames by attempting retransmissions, obviously the delay to send that frame would be high. And the packets generated at upper layer in the mean time would be queued and packets would be dropped once the limit on the number of packets in the queues is reached. This is another reason for packet loss.

Observe that the values of jitter and packet loss are high for the VoIP flow from-fbtop-to-mpur when compared to those of the flow from-mpur-to-fbtop. This is due to dropping of frames by fbtop. Note that fbtop has to contend for the channel for both VoIP frames and cross traffic frames, since both the cross traffic and VoIP streams are bidirectional. If a frame comes from upper layers when a frame is pending for transmission, the former may be buffered at upper layers. But there would be a limit on the number of packets that can be buffered. So more number of packets would be dropped at fbtop compared to any other place. So the number of the frames of the VoIP stream from-fbtop-to-mpur, dropped by fbtop is higher than the number of frames dropped by mbsnl in the VoIP stream from-mpur-to-fbtop. Similarly the variation in inter transmission time is also high which results in high variation in the inter arrival times at the receiver. So the jitter and packet loss values are high for the VoIP stream from-fbtop-to-mpur compared to those of the VoIP stream from-mpur-to-fbtop.

With Master-Slave driver, the packet loss values are always 0 since we avoid collisions (see Table 6.4). The loss values for the VoIP connection are zero even when the number of packets in the cross traffic are more than the maximum number of packets that can be supported by the network. The reason for this is as follows. In our protocol the master gives equal number of transmission opportunities to each of the slaves and each opportunity occurs alternately. It means that total time (there by bandwidth) is shared equally between the two links, mpur \iff fbtop and mbsnl \iff fbtop. So in the duration of one second each of the links roughly get 500 msec. Even if the number of packets in the cross traffic is increased, there will not be any change in equal sharing. We know that 100 packets are generated in the VoIP connection in one second. Time required to transmit 100 packets of size 172bytes= 94.43 msec which is very less than 500 msec. The packet loss if at all occurs would occur only in the cross traffic if it tries over-utilize the bandwidth allocated to it. Actually that also do not happen, since in Master-Slave protocol buffering of the frames would be done by all the nodes at driver level. We can put a limit on the maximum number of packets that can be buffered at driver level depending on the capability of the system and type of traffic.

And the jitter values are always with in the acceptable ranges with the maximum value being 6msec (at some point of time during communication). The jitter value in any case is not zero due to small variation in inter-transmission time due to queuing delays at driver level. And also the transfer of packets from/to upper layers involves some delays. The variation in the delay would be smaller and hence the jitter values are low.

6.1.2 Cross Traffic is TCP

We established a VoIP connection between fbtop and mpur. TCP traffic is generated from mbsnl to fbtop after 5 seconds. This experiment lasted for 40 seconds. With Original Driver we observed a significant amount of packet loss(10-20%) and jitter values going upto 20msec. With Master-Slave driver there was no packet loss and jitter values are normal (3-5msec) even in the presence of TCP cross traffic. The TCP throughput in both the cases is around 3Mbps. The reason for this may be as

follows. When a packet loss occurs TCP engine adjusts its window size according to slow start algorithm and reduces the rate of transmission. Consequently there would be consistency on the number of packets transmitted in one second and TCP tries to transmit the maximum number of packets it can send. That is why the TCP throughput is always 3Mbps throughout the experiment.

6.2 Results in Indoor Setup

6.2.1 Original Host AP Driver

In indoor setup there would not be hidden node problem. So there would not be any collisions. We tried to create a situation where hidden node problem occurs. We established the setup shown in Fig 6.1. Our aim was to make radios RADIO2 and RADIO3 not to hear each other. But we couldn't succeed. The reason for that is as follows. The loss of signal strength when it is transmitted from one end of the RF-Cable to the second end is 3db. When signal generated at RADIO2 is transmitted to the other end of the RF-Cable (connected to splitter) it can be listened by RADIO3 also because of the leakages in the splitter. The leakage is sufficient enough for RADIO3 to listen to the transmissions of RADIO-2 and vice-versa. So there would not be hidden node problem in indoor setup.

6.2.2 Master-Slave Driver

We modified the Master-Slave driver slightly to suit indoor setup. The most important modification we did to the Master-Slave driver is that in this version of the driver, the from-bap-overhead and to-bap-overhead are not removed. The reason for this is the difference in the architecture of the network, in indoor and out-door setups. In out-door setup each slave can hear the master only and cannot hear other slaves. We took the advantage of this fact and effectively removed the to-bap-overhead and from-bap-overhead to improve the throughput. But in indoor setup all the slaves also can hear each other. That's why we cannot remove to-bap-overhead

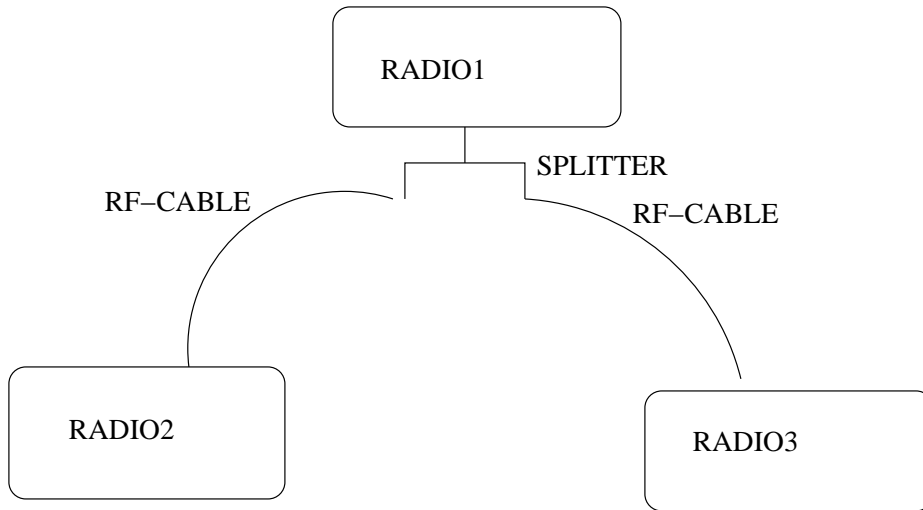


Figure 6.1: Indoor Setup with 2-way Splitter

and from-bap-overhead. So the throughput would be less in indoor setup. And also the delay in transmission of a frame would be higher when compared to that of the out-door setup.

We tested the driver with 2-slaves and 3-slaves.

■ *Master and Two Slaves*

We call the three nodes here as master, slave1 and slave2. VoIP traffic is generated between master and slave1. UDP traffic is generated between master and slave2. This is cross traffic. As in the outdoor setup we varied the number of packets in a second (in cross traffic) by keeping the size of the packet fixed at 1470. All the nodes are in the vicinity of each other. The jitter and packet loss are listed in Table 6.5 and Table 6.6. We can see that both jitter and loss are in control. In Table 6.5 column1 specifies the number of packets each of size 1470 bytes generated in one second between master and slave2, columns 2 and 3 specify the jitter values at the two ends of the VoIP connection. In Table 6.6 column1 specifies the number of packets each of size 1470 bytes generated in one second between master and slave2, columns 2 and 3 specify the packet loss values at the two ends of the VoIP connection. Observe that the jitter values are slightly high in indoor setup when

Table 6.5: Jitter Values with Master-Slave driver in Indoor Setup with 2-slaves

Number of Packets/sec (Cross Traffic)	Jitter Values for the VoIP Flow from-master-to-slave1 (msec)	Jitter Values for the VoIP Flow from-slave1-to-master (msec)
0	3.514	3.286
100	4.12	3.627
200	3.991	4.289
300	4.419	3.966
400	4	3.737
500	4.285	3.803
600	3.932	4.153

Table 6.6: Packet Loss with Master-Slave Driver in Indoor Setup with 2-slaves

Number of Packets/sec (Cross Traffic)	Packet Loss for the VoIP Flow from-slave1-to-master (%)	Packet Loss for the VoIP Flow from-master-to-slave1 (%)
0	0	0
100	0	0
200	0	0
300	0	0
400	0	0
500	0	0
600	0	0

compared to those in our-door setup. As we already discussed in the beginning of this subsection the delay in transmission of a frame is relatively high in indoor setup when compared to that in out-door setup. This results in slightly high values of variation a VoIP frame has to wait at driver level. (Refer to Analysis subsection in section Results in Out-door Setup).

As expected, with TCP cross traffic (from slave2 to master) also the jitter and packet loss are in acceptable range with the jitter value being 3.35msec and packet loss being 0%. And the throughput of the TCP cross traffic is around 2.2Mbps. The throughput is less than that we got in out-door setup (In out-door setup it is 3Mbps). The reasons behind this are

- to-bap-overhead and

Table 6.7: Jitter Values with Master-Slave driver in Indoor Setup with 3-slaves

Number of Packets/sec (Cross Traffic)	Jitter at master (msec)		Jitter at slaves (msec)	
	for the VoIP Flow from-slave1 -to-master	for the VoIP Flow from-slave2 -to-master	for the VoIP Flow from-master -to-slave1	for the VoIP Flow from-master -to-slave2
0	2.416	3.162	3.521	3.158
100	4.814	4.392	2.258	6.884
200	2.079	3.659	2.612	4.733
300	4.550	3.895	4.480	4.371
400	4.439	5.857	3.794	4.741
500	4.792	4.745	3.935	4.016
600	3.862	3.582	2.984	4.293

- from-bap-overhead

Both of them takes significant amount of time and the channel is unused during that period.

■ *Master and Three Slaves*

Here we have one master and three slaves. Assume that the three slaves are named as slave1, slave2 and slave3. We established two VoIP connections, one between master and slave1 and the other between master and slave2. We tested the performance of these two connections by varying the traffic rate between master and slave3 and keeping the size of the packet fixed. Here also all the nodes are in the vicinity of each other. The jitter values are listed in Table 6.7. column1 specifies the number of packets each of size 1470 bytes generated in one second, columns 2 and 3 specify the jitter values at master for the two VoIP connections. Column 4 lists the jitter values at slave1 and column 5 lists the jitter values at slave2. With TCP cross traffic the jitter values are 4.348 msec and 4.512 msec at master and 3.955 msec and 4.07 msec at slaves 1 and 2 respectively. The TCP throughput in this case is around 1.5Mbps.

In all the cases, with Master-Slave protocol, whatever may be the rate or whatever may be the type of the cross traffic the QoS parameters are always in acceptable

range. So smooth communication is ensured with Master-Slave driver, while it is not ensured with a Original Driver.

Chapter 7

Implementation

In this Chapter we discuss the implementation details of VoIP. We mainly focus our attention on engineering the VoIP system with low cost. We investigate software available to suit our system and later implement the system. In all the cases we go for open sources since they are available for free.

The goals of our system are-

1. The cost should be low.
2. The overall system should be integrated with the existing telephony network.

There would be multiple number of clients. There should be a way to forward the calls to appropriate client. All the clients would be having IP addresses. To able to communicate with the external telephony network they should be addressed using plain telephone numbers. So there should be a mapping between IP addresses and plain digit numbers. Put in one sentence our system should have a PBX (Private Branch eXchange). There comes *asterisk* into picture.

Asterisk is open source software which when installed onto a Linux box acts as a PBX. It supports most of the VoIP technologies like h.323 and SIP (Session Initiation Protocol). It provides facilities like call forwarding, voice mail, call queuing etc. To communicate with any telephony network it needs an FXO (Foreign eXchange Office) card. The combination of Linux box (soekris board[19]) + FXO card[18] +

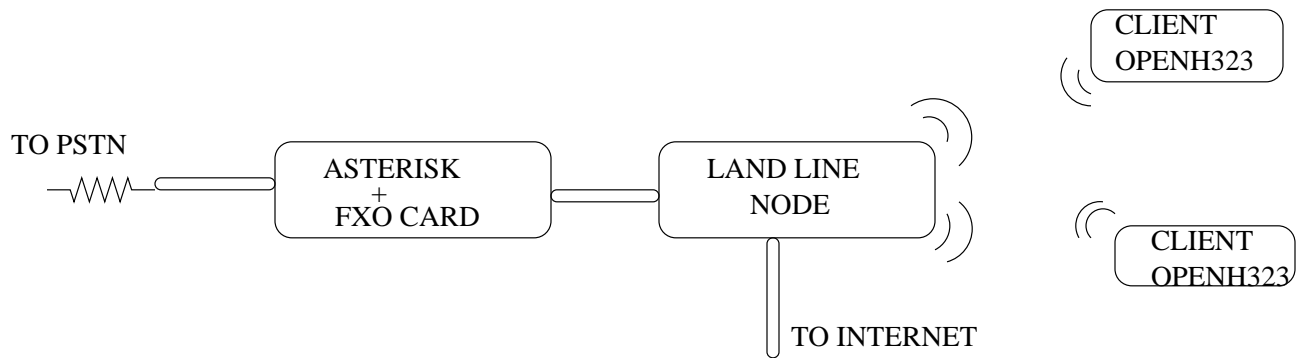


Figure 7.1: FXO Setup

asterisk acts as an FXO, which acts as a gateway between computer network and telephony network. The main advantage of using this combination is its low cost. While a commercial FXO box costs around 500\$[17], this combination costs around 200\$ (100\$ for FXO card and 100\$ for a soekris board).

Openh323 is used as client software in our system. Openh323 is a VoIP client open source software which implements h.323 protocol. It provides facilities for teleconferencing also.

The setup which can be used for providing telephony services to the wireless clients is shown in Fig7.1.

Chapter 8

Conclusions & Future Work

The existing 802.11 MAC(DCF) is not suitable to provide Real Time services in long distance networks. We designed, implemented and tested a contention free protocol which works at network driver level. We analyzed the performance of VOIP from the perspective of QoS with and without our protocol. We proved that with our protocol the QoS is far superior. Later we engineered a virtual telephone exchange system which is best suited to rural areas in developing countries.

Our main concentration was on 1-hop networks only. This type of contention free protocol can be designed for multi-hop networks also. In our protocol we give the same priority to all the packets. One possible extension for this work would be designing a mechanism to prioritize the different types of traffic. With this mechanism one can ensure that time-critical services are given higher priority than best-effort traffic.

Bibliography

- [1] Pravin Bhagwat, Bhaskaran Raman, and Dheeraj Sanghi. Turning 802.11 Inside-Out *Second Workshop on Hot Topics in Networks(HotNets-II)*,20-21 Nov 2003,Cambridge, MA, USA.
- [2] Bhaskaran Raman and Kameswari Chebrolu. Revisiting MAC Design for an 802.11-based Mesh Network *Third Workshop on Hot Topics in Networks (HotNets-III)*, 15-16 Nov 2004, San Diego, CA, USA.
- [3] The DGP Media Labs Asia Team Digital Gangetic Plains (DGP): 802.11 based Low-Cost Networking for Rural Areas 2001-2004: A Report
- [4] <http://www.iitk.ac.in/mladgp/>
- [5] <http://hostap.epitest.fi>
- [6] <http://standards.ieee.org/getieee802/802.11.html>
- [7] Can I add a VoIP call? *Communications,2003*, USA.
- [8] Enhancement of VoIP over IEEE 802.11 WLAN via Dual Queue Strategy *WCNC-05,IEEE Wireless Communications and Networking Conference*,Newyork, NY,USA.
- [9] http://aqua.comptek.ru/test/HiddenNode/hidden_node_en.html
- [10] http://www.cisco.com/en/US/tech/tk652/tk698/technologies_white_paper09186a00800d6b73.shtml

- [11] <http://www.voip-info.org>
- [12] Registered no. D.L.-33004/99, Extraordinary Gazette, Part - II - Section 3 - Sub Section (i) No. 36 dated 28 January 2005
- [13] <http://www.seattlewireless.net/index.cgi/SenaoCard>
- [14] <http://dast.nlanr.net/projects/Iperf/>
- [15] <http://www.asterisk.org>
- [16] <http://www.openh323.org>
- [17] <http://www.payphoneoutlet.com/mts-mvp130.html>
- [18] <http://www.digium.com>
- [19] <http://www.soekris.com>
- [20] <http://www.ietf.org/rfc/rfc1889.txt>