Implementation of WiFiRe MAC

Framing, memory and wireless modules

M.Tech. Project Dissertation

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

by

Janak Chandarana

Roll No: 05329R04

under the guidance of

Prof. Sridhar Iyer

and

Prof. Anirudha Sahoo

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

Mumbai

Dissertation Approval Sheet

This is to certify that the dissertation entitled

Implementation of WiFiRe MAC

by

Janak Chandarana

(Roll no. 05329R04)

is approved for the degree of **Master of Technology**.

Prof. Sridhar Iyer

(Supervisor)

Prof. Anirudha Sahoo

(Co-supervisor)

Prof. Varsha Apte

(Internal Examiner)

Dr. Vijay Raisinghani (Patni Computers Ltd.)

(External Examiner)

Prof. T.K.Biswal

(Chairperson)

Date: ______________

Place: ______________

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

CERTIFICATE OF COURSE WORK

This is to certify that **Mr. Janak Chandarana** was admitted to the candidacy of the M.Tech. Degree and has successfully completed all the courses required for the M.Tech. Programme. The details of the course work done are given below.

Sr.No.	Course No.	Course Name	Credits
		Semester 1 (Jul – Nov 2005)	
1.	HS699	Communication and Presentation Skills (P/NP)	4
2.	IT601	Mobile Computing	6
3.	IT605	Computer Networks	6
4.	IT619	IT Foundation Lab	8
5.	IT623	Foundation course of IT Part II	6
		Semester 2 (Jan – Apr 2006)	
6.	IT680	Systems Lab	6
7.	IT614	Internet Technologies	6
8.	IT620	New Trends in Information Technology (Optical Networks)	6
9.	IT694	M.Tech. Seminar	4
		Semester 3 (Jul – Nov 2006)	
10.	IT625	ICT For Socio-Economic Development	6
11.	IT634	Communication Networks	6
12.	CS681	Performance Evaluation of Computer Systems and Networks (AU)	6
		Semester 4 (Jan – Apr 2007)	
13.	HS701	Development, technology and Global Order (Institute elective)	6
		Semester 5 (Jul – Nov 2007)	
14.	CS620	New Trends in IT: wireless networks (AU)	6
		M.Tech. Project	
15.	IT696	M.Tech. Project Stage - I (Jul 2007)	18
16.	IT697	M.Tech. Project Stage - II (Jan 2008)	30
17.	IT698	M.Tech. Project Stage - III (Jul 2008)	42

I.I.T. Bombay Dy. Registrar(Academic)

Dated:

Acknowledgements

I would like to express my sincere gratitude toward my guide Prof. Sridhar Iyer for his constant support and guidance. There were many great ideas, insightful suggestions and constructive criticism given to me during my stay in IIT. I hope that I will get to work with him in near future. I also thank my co-guide Prof. Anirudha Sahoo for various technical and non-technical discussions we had and experience that he poured to the project.

I also thank my team-mate Ranjith for sharing project responsibility equally. We learned quite a few things while writing code. All my friends in MTech-05 and Mtech-06 batch of KReSIT: Kushal, Karthik, Moniphal, Sandeep, Saurabh, Sameer, Manoj, Jeevan, Kedar, Ajay, Avadhoot, Nithin and many others who made my life at IIT cheerful. We had incredible fun and these were probably the best days of my life so far.

I have been fortunate to get scholarship from Development Gateway Foundation (DGF) and R V Nilekani Endowment Fund. I thank them for financial support they provided me. I also got opportunities to work with people in CEWIT (IIT Madras), SynerG group (IIT Bombay) and Intel (Bangalore), it was good learning experience.

Finally, I thank my parents and Charul for great affection and constant encouragement.

Janak Chandarana

IIT Bombay

July 2008

Abstract

Long range wireless for data and voice connectivity is being considered as viable and affordable solution for rural India for few years now. Numerous solutions were presented to bridge the digital divide; WiFiRe is one of them. Basic idea is to change CSMA/CA MAC of 802.11 with more efficient TDMA MAC. WiFiRe promises higher data-rate, longer range and cost-effective solution as compared to WiFi based solution.

Preliminary implementation was started last year with design phase and we are extending that work. This year, WiFiRe MAC team at IIT Bombay has implemented most of the WiFiRe MAC modules mentioned in the WiFiRe draft and they can be integrated with 802.11b PHY easily. Our MAC implementation is able to support web and voice traffic for multiple subscriber terminals and their clients. It also supports end-to-end connectivity for all the clients, while they are unaware of underlying layer-2 protocol.

We have come across numerous implementation issues and design decisions with respect to WiFiRe MAC testbed. While developing WiFiRe MAC, we got insights into various system and networking issues with hands-on experience. We have also suggested various extensions possible to this work in PHY integration and QoS area. Initial results achieved by this project are encouraging and will lead to full deployment of system soon.

Contents

List of Figures

Chapter 1

Introduction

1.1 Preamble

Long range wireless for data and voice connectivity is being considered as a viable and affordable solution for rural India. As described in [1], the average user cannot spend more than Rs. 300 per month for Internet. There is a need for affordable and easily available system, using which broadband access can be spread to every village in India. WiFiRe [2] presents one such solution, using cheap 802.11b chip-set but modifying the MAC for longer range, support for QoS and better efficiency. Initiatives reported in [3] and [4] address very similar issues.

1.2 Background

As shown in Figure 1.1, Indian villages are located 2-3 km apart, spread around town or city. City or town are present every 40km or so. All such towns and cities are connected with high quality fiber (with huge bandwidth unused) to backbone network across the country. Each such central point provides WiFiRe cell with the radius of 20km [1].

This high quality fiber does not reach till villages because of low revenue and high cost involved. There are basic fixed wired telephone available at some places but many places are still not connected. Problem is more severe for broadband connectivity.

There is a wide gap between urban tele-density (47.5%) and rural tele density (1.8%). The rural telephony has not kept pace with the impressive growth in urban connectivity. In the context of global growth pattern and indicators, we need to achieve more in terms of tele-density [5].

The challenge is to connect all such remote villages, unconnected due to remoteness from city and lack of basic telephone line nearby. One of the options available to provide broadband

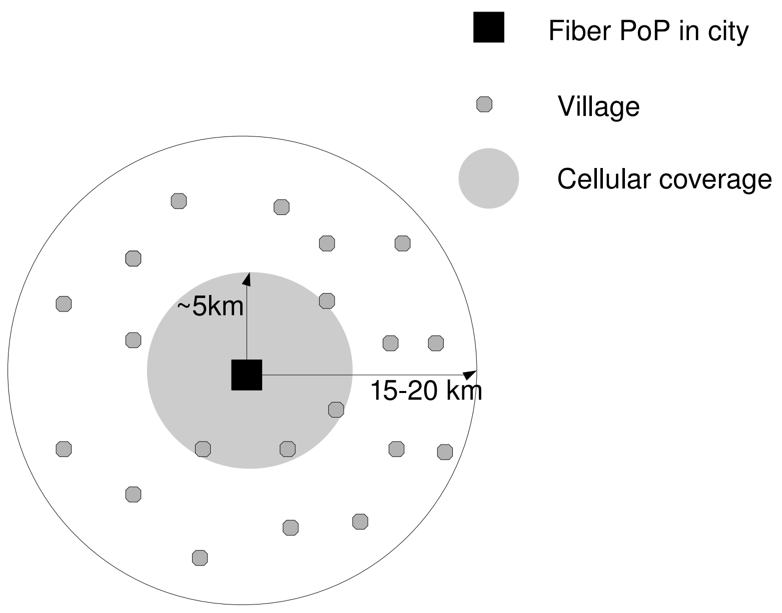


Figure 1.1: Communication in rural India

access today is DSL. ISP provides this connectivity using copper (or similar cable) to most of the urban part of India. It is possible to get enough revenue because of dense population in urban area. In rural areas (specially remote), it is not feasible to have copper cable because of higher cost, long distance and very low revenue. In most of the remote villages, broadband on copper wire is not a viable option. For all these cases, wireless connectivity seems to be a viable solution because of ease of configuration, rapid deployment and low cost. Wireless connectivity can be point to point or point to multi-point basis which will connect all remote areas to the nearest city in quick time.

Currently available cellular technologies (e. g. GSM, CDMA) are meeting the cost targets but not feasible for broadband services because of very low bandwidth availability. IEEE 802.16d [6], is a new technology for fixed wireless broadband access which provides required data rate but industry has not adopted it because of certain issues. It does not meet rural sustainable cost due to ignorance from industry, low volume and low revenue. 802.16e is a mobility aware WiMAX standard which is widely adopted by industry. It promises to have more volume and lower cost to end users. However, if we see current trends in India and outside, WiMAX adoption is very slow and it will take few years for the costs to reduce and become viable for rural deployment. WiFi is an inexpensive LAN technology and provides sufficient data rate to be called as broadband for rural areas. WiFi can provide this data rate for shorter distance (approximately 50 meter) only. One advantage with WiFi is the free spectrum and low-cost chip-set. 802.11 based devices are widely available and can be used off-the-shelf.

Over the years, 802.11 technology has matured and being used extensively everywhere now a days. As 802.11 (commonly known as WiFi) is designed to serve normal office and home usage, it covers area of 50 meter or so. Although, one can extend its range using better chip-sets, antenna and other such devices, it does not perform well for the longer range. Problem lies in 802.11 MAC which is designed to operate in smaller area. People have identified this problem and several solutions have been proposed in last 3-4 years. Channel sensing mechanism of 802.11 does not work well with longer range because of propagation delay involved. If all wireless nodes work in a slotted system where they are already aware of Tx and Rx timing, such a system will have higher throughput compared to CSMA/CA based MAC. Several projects have identified this solution and various kind of wireless networks has been deployed using that. Most common are mesh and point-to-point networks where number of nodes is very less and chances of interference is negligible.

1.3 WiFiRe

WiFi Rural Extension (WiFiRe) [2] uses the licence-free 2.4 GHz spectrum and cheap 802.11b RF chip-set to provide long-range wireless communication. It replaces 802.11b MAC mechanism (DCF/PCF) with a MAC similar to 802.16d, while retaining 802.11b PHY. WiFiRe architecture is based on star topology - a Base Station (BS) at the fiber Point of Presence (PoP) and Subscriber Terminal (ST) in the villages nearby. BS has six sectorised antennas covering area of 15-20 km. ST is connected to end users with different LAN technologies [7] and communicates with BS through directional antennas. Within a sector, WiFiRe follows time division multiplex (TDM) frame structure similar to 802.16d. WiFiRe is explained in detail in chapter 3.

WiFiRe is promoted by CEWiT India[1] and the project is spread across IIT Bombay, IIT Madras and IISc Bangalore.

[1]The Center of Excellence in Wireless Technology (CEWiT) is an independent research organization set up by the Ministry of Information Technology, Government of India (MIT) in partnership with industry.

1.4 Thesis scope and outline

As mentioned above, WiFiRe project is combined project, work presented here is mainly on MAC layer. WiFiRe team at IIT Bombay focussed on MAC protocol details for WiFiRe. Basic WiFiRe LAN emulator was designed last year and this work is extension to it. There are various MAC modules presented here which deals with framing, queueing, upper layer processing, memory management, transmission and reception of frame etc. Once MAC is prepared, it can be integrated with PHY layer of 802.11b. Proposed integration work is also presented in the end.

WiFiRe MAC team work is divided in two parts as shown in Figure 1.2. Modules which are developed by Janak are in grey color while modules developed by Ranjith [8] are in white color. In this report, we discuss all the modules shown in grey color. All modules integrate with each other and make complete WiFiRe MAC prototype.

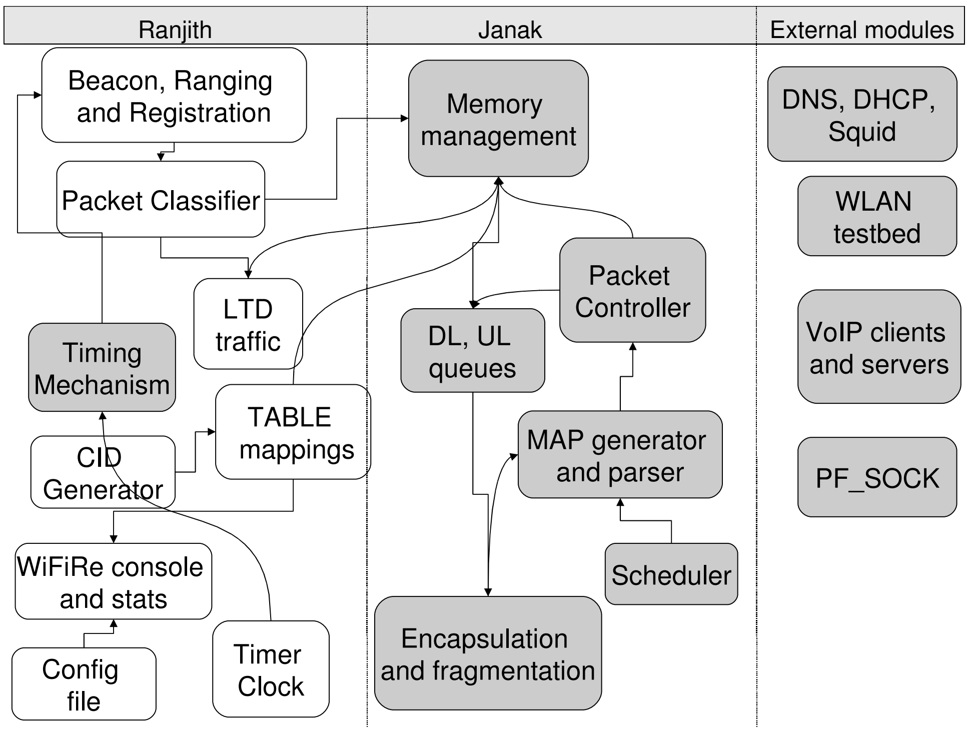


Figure 1.2: Division of work, WiFiRe MAC modules

The content of thesis is arranged in following order: Chapter 2 describes literature survey and research projects similar to WiFiRe. Chapter 3 introduces WiFiRe, protocol details and work done previously. Chapter 4 explains WiFiRe LAN testbed and few design decisions and assumptions. Chapter 5 explains various MAC modules of BS and ST and their implementation details. Chapter 6 explains learning during the development of project and results achieved. Chapter 7 verdicts possible extensions to this work, new directions for research in this area and summary of work. Chapter A in appendix contains details of various server configurations on proxy machine.

Chapter 2

Related work

There has been significant work in the field of long range wireless since last 5 years or so. There are three main focus areas which are important to us. Researcher are interested in providing QoS to important traffic like voice, streaming video etc. First section describes efforts in that area. Second section includes projects which used WiFi hardware to establish communication over long range. It describes deployment difficulties, hardware issues and changes done in MAC to support long range. Last section includes work where authors have tried changing MAC protocol to optimize channel utilization. Mostly, they have proposed TDMA similar protocols and proved their betterment using simulation or prototype.

2.1 WiMAX scheduling, CAC and QoS

This section includes MTech projects done in WiMAX related area at various IITs. They provide closer look at some QoS and scheduling designs for WiMAX. QoS guarantee is claimed and separate service queues have been proposed. This section also reminds that there is enough ground work in this area and it is good time to evaluate some of these concepts in reality.

Efficient Call Admission Control for IEEE 802.16 Networks

This report [9] presents CAC and QoS for WiMAX services. It considers different kind of services like UGS, rtPS, nrtPS and BE. Paper presents CAC architecture on BS and SS considering different QoS parameter. Bandwidth based CAC does not consider delay requirement and priority for real time traffic. This paper adds delay into CAC's consideration. Simulation is done using C on Linux platform. Simulation considers lots of input parameters like maxrate, minrate, total number of slots, nominal grant interval, jitter, polling interval etc. CAC considers all these input parameters and takes decision based on existing data it has. The main question here is, how do client actually give this input to WiMAX CAC. 802.16 specify convergence sub-layer which deals with different upper layer protocol. This CS layer will parse TCP packets and retrieve those service parameters. Parsing and storing them in database will take considerable amount of time. 802.16 devices will be having small processor and limited amount of memory to use. For VoIP kind of traffic, it may happen that total time spent in processing this parameters may be more than total QoS benefit received by packet (in terms of ms). To prove this fact, we can have small performance based module which runs on actual hardware and takes few parameter into consideration. Then we can formulate equation like: for X number of parameters time taken is Y. For Z parameters, what will be the processing time. Paper presents some innovative ideas about different service queues, periodic grant generator and pseudo code to prove algorithm proposed.

QoS Scheduling Architecture for IEEE 802.16 Wireless MANs

Report [10] presents Efficient QoS scheduling architecture in 802.16 networks. It tries to provide delay and bandwidth guarantee, fairness maintenance and higher bandwidth utilization. Simulation is done in Qualnet and contains modules like GPSS, TDD frame, scheduling services, aggregate bandwidth requests etc. All nodes are using 802.16 MAC and 802.11b PHY layer. Main traffic is VoIP, FTP, telnet etc. This work was done just after introduction of 802.16 standards; it was one of the very initial work in WiMAX MAC scheduling implementation. Author used min-max fair allocation for uplink scheduling and WFQ for downlink scheduling. It introduces MAP generator, grant generator, data classifier, scheduler and traffic shaper.

MAC Scheduling Architecture for IEEE 802.16 Wireless MANs

In this report [11], author used WFQ for both uplink and downlink. He simulates his proposed algorithm using NS-2. Paper claims to provide delay bound scheduling for real-time traffic. Author considers GPC mode because he believes that connections are more important than number of SS. NS-2 simulation with modules for TDD frame structure, GPC mode and bandwidth allocation, Ranging request and type of services are described. NS-2 architecture is modified slightly to support WiMAX modules. Result describes the effect of one type of flow to other flow and choice of correct bandwidth contention period.

2.2 Deploying long range WiFi

802.11 based wireless is used for indoor use since last decade. It has been mainly used for home settings but never considered as an option for long range or broadband access. Over the years, 802.11 based devices became really cheap, researcher started trying to use 802.11 based RF chip-set for long range communication. WiFi based networks are important for us because WiFiRe also uses 802.11b chip-set for PHY layer. This section includes some academic deployment projects for the same.

Rethinking wireless for the developing world

Paper [4] describes technical and non-technical challenges associated with wireless deployment in developing regions. Paper looks at larger scenario where authors think about long wireless links, affordable pricing model, Intranet usages, traffic support etc. Authors also describe some working deployments in India, Ghana and San Francisco. WiFi-based Long Distance (WiLD) links provides many interesting challenges like routing, interference, multi-path etc. It also describes ACK timeouts, collisions, scheduling in wireless links and various QoS mechanism. Overall, paper had compiled all major issues concerned with wireless deployment in developing regions.

Turning 802.11 inside-Outs

This is probably the first paper [3] to describe long range wireless in India. Paper describes digital divide, Indian Telecom sector and cellular wireless technology in India. DGP [3] is having a network of eight nodes and eight point-to-point links. The longest point-to-point link spans over 38km. It claims to get voice (using VoIP) and cheap Internet to villages. DGP describes various technical challenges like PHY performance in outdoor channels, power efficiency, 802.11 MAC issues, timeouts, contentions, routing etc. Author presents real life deployment and problems involved in making all pieces working. Paper is highly cited in other such deployments.

DakNet: commercial deployment

DakNet is commercial project [12] started in MIT Media Lab. Initial deployments are already there in some part of Orissa and Rajasthan by United Villages Ltd. They have specialized hardware (mainly 802.11 based) to have delay tolerant communication. There is a vehicle which comes to village once a day, kiosk operator will communicate with antenna kept on top of vehicle. They have added plenty of services like voice-mail, SMS, email, retail sales, railway tickets etc. This is probably the first viable commercial deployment of such concept. DakNet also manufactures low cost end-user device for daily usage in kiosk.

Arvind Eye wireless

Intel and TIER group from UC Berkeley had established long distance WiFi links in southern part of India. They have connected remote centers of Arvind eye hospitals using 802.11. Some wireless links are 50km long as well. Using this approach, they eliminate shortage of doctor in any particular center. Any patient coming to remote center can talk with expert doctor sitting at main hospital and doctor can describe remedies and medicine. Deployment is in working condition and being used on day-to-day basis.

2.3 Changing 802.11 MAC

This section describes projects that replace 802.11 MAC with more efficient MAC protocol. Most of them use open-source MAC implementation and modify it to make TDM similar system. New MAC gives better throughput and more control on frame. Challenge here is to port this MAC on appropriate hardware and communicate efficiently. WiLDNet and DGP also change MAC but not included here.

MadMAC: Building a Reconfigurable Radio

In this paper [13], author tries to change 802.11 CSMA/CA MAC with more efficient TDMA MAC. They used madwifi drivers and wrote their MAC on top of it. Its kernel level module which takes care for packet scheduler, slot processing and channel switching. They observed 20% improvement over conventional WiFi MAC in 2-node testbed. Important thing to note is, they retain madwifi based MAC as it is. Project used batch processing for multiple packets which is very similar to what is mentioned in WiFiRe draft [2].

SoftMAC Flexible Wireless Research Platform

This project [14] tries to change CSMA/CA behavior by changing few 802.11 functionalities. They have changed/removed 802.11 behaviors like RTS/CTS, acknowledgement, back-off timers etc. They have made changes at driver level and modules are written as kernel modules. They have achieved precision in micro second level. They introduce TDMA based MAC where slots are pre-defined. They built software defined radio at very cheap rate of Rs. 3000 approximately.

MultiMAC - An Adaptive MAC Framework

This project [15] has proposed hybrid MAC for wireless communication. In case of low contention, they use normal CSMA/CA MAC while in case of higher contention, they use TDMA similar MAC. Their module acts as mediator and senses environment and changes MAC accordingly. It used softMAC mentioned above for their underlying hardware use. This MAC has precision of 100 μSec.

Comments

There is significant work done by researchers in the area of QoS, CAC and scheduling. It would be interesting to see how these algorithms work in actual deployment. There are numerous wireless deployments (based on 802.11) for longer range but they are either point-to-point or mesh configuration. In case of WiFiRe, we have star topology with point-to-multipoint support. To change 802.11 MAC, there are two main approaches. First, change at driver level where one can get frame level control. Second, overlay MAC which runs on top of 802.11 MAC with less control on framing and slots but very easy to implement. We have opted second approach for WiFiRe wireless link as mentioned in Section 5.4

Chapter 3

WiFiRe

WiFiRe stands for WiFi Rural extension. It uses licence free 2.4 GHz spectrum and cheap 802.11b RF chip-set as PHY layer. It replaces 802.11b MAC mechanisms (DCF/PCF), with long range MAC (like 802.16 [6]), keeping 802.11b PHY same. WiFiRe is a star topology - a Base Station (BS) at the fiber Point of Presence (PoP) and Subscriber Stations (ST) in the villages nearby with 6 directional and sectorised antennas at the System. It follows TDM frame structure which is similar to WiMAX and GSM. WiFiRe is promoted by CEWiT, India and project is spread across IIT Bombay, IIT Madras and IISc Bangalore.

3.1 WiFiRe architecture

The basic design of WiFiRe comprises of a central System (S) with dedicated Internet lines (e.g. fiber OC9 or OC12). This system provides the connectivity to the Internet and normal PSTN line. This system is usually placed in city or town (fiber PoP) and covers villages nearby. This area is covered in sectorised manner and each sector has its dedicated base station (BS). BS is a sectorised antenna with wireless chip-set and mounted on top of tall tower with height of 40m approximately. All BSs are synchronized such that they can send and receive data without interfering each other. There are Subscriber Terminals (ST) situated in the villages with wireless chip-set and directional antenna pointed towards BS. ST is mounted on comparatively smaller tower of 10-15 meter height. Both BS and ST are static and does not support mobility as of now. Clients can be connected to ST using Ethernet LAN or wireless LAN. End user with client devices can be connected to outside world via ST. The system will be configured as a star topology. The network topology will be as shown in the Figure 3.1. Each BS usually covers range of 15 - 20km, covering around 200 villages for whole system(S). ST is usually connected with kiosk which provides voice and data services. In case ST gets signals from more than one

BS, signal with higher RSS will be considered.

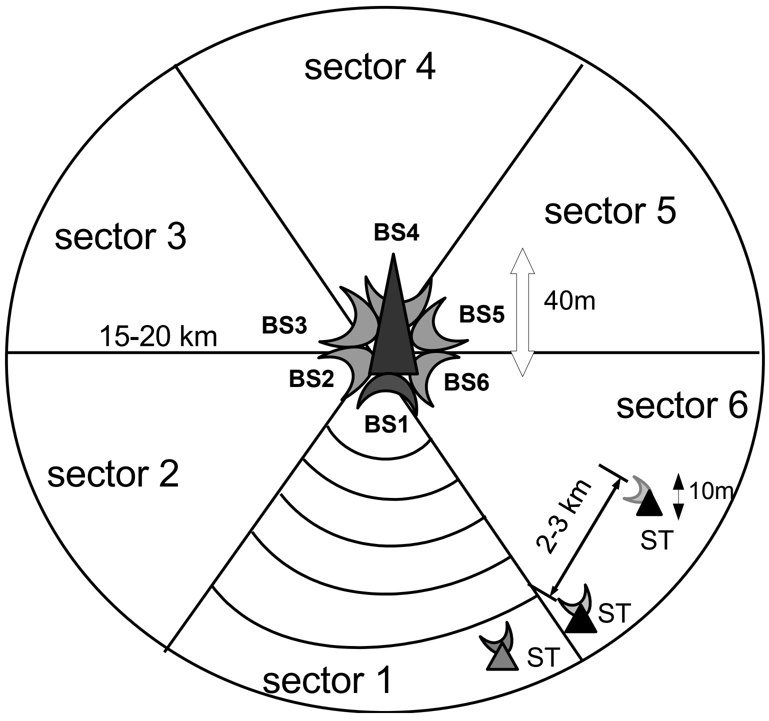


Figure 3.1: WiFiRe sectorised system

Opposite BS can operate parallel because their signal will not interfere with each other. WiFiRe supports time division duplex (TDD) over single channel with multi-sector TDM (MSTDM) mechanism, which supports about 25 Mbps (for both uplink and downlink) for a cell. In TDD, the uplink (ST to BS) and downlink (BS to ST) share the same frequency but are activated at different time. BS and ST operate in synchronization with each other. Each frame divided into downlink(DL) and uplink(UL) subframe.

WiFiRe link layer is designed to provide long distance reliable communication, and supports service guarantee for real time and non-real time applications. WiFiRe uses time division duplex multi-sector TDM (TDD-MSTDM) MAC. Scheduling of slots is done so as to maximize simultaneous transmission in multiple sector while keeping co-channel interference within prescribed limit. We assume that scheduling is done in a Round Robin (RR) fashion, where each sector is scheduled one after another. For example, in Figure 3.1 in case of RR scheduling, STs in sector 1 will get scheduled first, followed by STs in sector 2 and then sector 3. Transmissions in the opposite sectors can be scheduled in parallel, hence sector 1 & sector 4; sector 2 & sector 5; and sector 3 & sector 6 will get scheduled in parallel.

3.2 WiFiRe frame, messages and system

WiFiRe being a TDMA based system, time is divided into frames and frames are further divided into slots. WiFiRe frame structure is shown in Figure 3.2. Frame duration depends on the VoIP packet generation period and slot duration depends on the VoIP packet size. In [2], frame duration is chosen as 10 mSec and slot time in 32μsec. Figure 3.2 shows WiFiRe frame structure. Frame is partitioned into downlink (DL) and uplink (UL) segments. In DL segment we have transmission from BS to ST and in UL segment transmission takes place from STs to BS. These segments are separated by a guard band of 4.5 slots to account for propagation delays and transmitter-receiver turnaround. At the start of the frame we have beacon transmissions, which contains system information, control information and DL-MAP, UL-MAP. DL-MAP, UL-MAP specifies DL and UL slot allocations respectively.

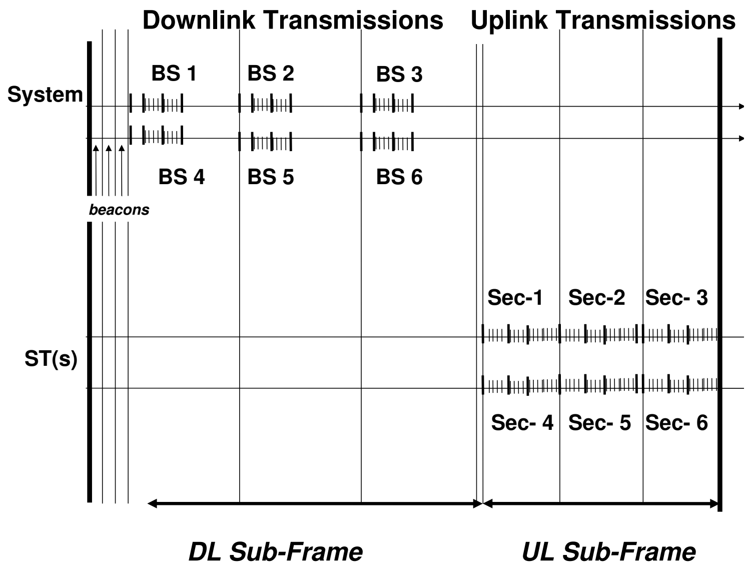


Figure 3.2: WiFiRe frame structure

3.3 First phase of Implementation

WiFiRe's MAC software stack was partially implemented at IIT Bombay [16]. This work was with the assumption of programmable 802.11b chip-set running WiFiRe MAC on top of it. Implementation proposed establishing connection, basic packet flow, MAC header construction etc. Initial implementation was done using C sockets (on layer 7). Later on, MAC was running

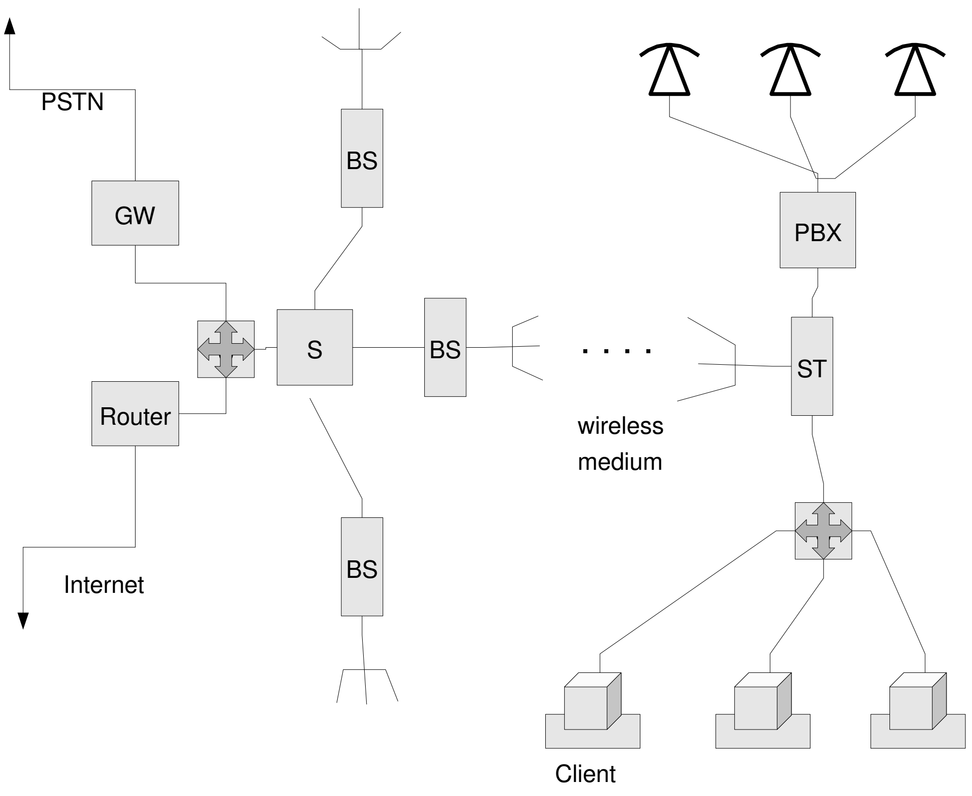


Figure 3.3: WiFiRe system architecture

as program on layer-2 and talking with NIC directly. This was done using PCAP libraries in Linux. This (layer-2 implementation) gave opportunity to integrate other programs with PCAP modules. Layer-2 code was not flexible as it was sending MAC packets from BS to SS where program does not have much control on NIC's output. The latest C sockets worked like peer to peer application and could easily integrate with WiFiRe's other MAC code. Simulation's behavior and packet structure remains same in layer-2 and layer-7's code. It can be verified using *tcpdump* or *ethereal* filters on respective NIC.

Implementation of MAC was done using gcc in Linux where program runs in user space. Linux terminal with 2-3 NICs works as BS and STs. This whole scenario gives us emulation of actual protocol. Many network parameters (static IP, ARP cache, DNS, proxy etc.) were kept fixed to keep design as simple as possible. In most of the cases, BS and STs were made layer-2 device where any packet going to layer-3 or above will be ignored.

BS connection with ST is having DIX based (similar to 802.3) Ethernet. This configuration depends on NIC drivers being used, connecting media and firmware's support for flexible frame structure. This link is treated as wireless link of WiFiRe. As MAC program runs on Ethernet

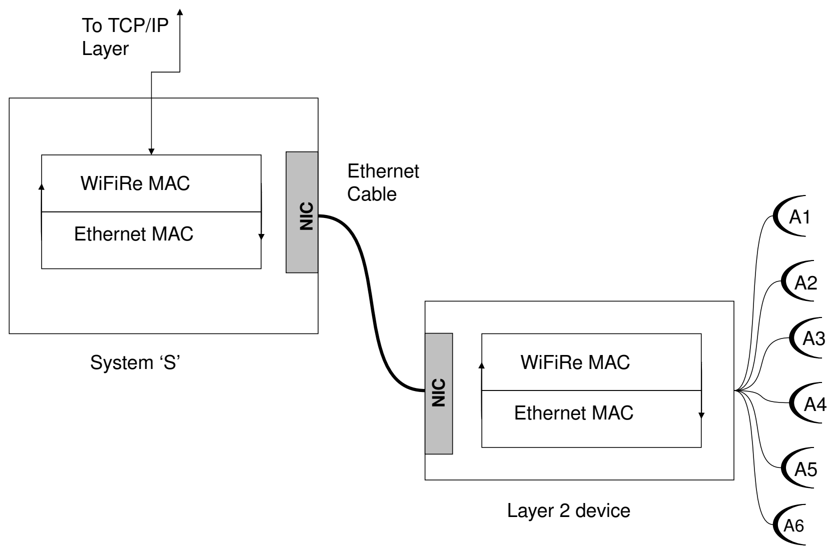


Figure 3.4: WiFiRe MAC and 6 BSs

and with PC connected directly, it can't get total flexibility on frame structure, size, CRC etc. This link restricts size of individual packets and mpdus as well. Both BS and ST were kept next to each other which eliminates the problem of propagation delays, ranging, synchronization and other such wireless specific issues. It also assumed that some of those wireless specific issues will be taken care by underlying hardware in actual implementation.

BS and STs handle packets based on two threads: OUT-THREAD and IN-THREAD (See figure 4.4). Initially, these mechanism lead to recursive Tx and Rx calls. When program captures packets on BS's NIC-1 which is connected to outside world, packet get processed and given to NIC-2. When NIC-2 tries to send packet to ST, it will be captured again and this recursive cycle goes on. By applying 'smart' filters on PCAP, we could remove this recursive property. To keep design simple, there were no shared variables or semaphores used for co-ordination among different functions. No hardware based interrupts and triggers were supported because there is no predefined hardware property.

Many clients connected to ST using layer-2 switch. ST is able to recognize and treat clients differently. CID was given on per client per TCP connection basis. For example, if client have one VoIP and one FTP connection, there will be 2 CIDs for that client. If client have 2 VoIP and 2 FTP connections running, it will get 4 CID in connection database. CS layer in WiMAX specification takes care for different connections. Similarly, WiFiRe also keeps a pool of CIDs in CS layer. CS layer can be interfaced with 802.3 based media only.

3.4 Approaches and assumptions taken

WiFiRe presents classical networking project where many of theoretical concepts came into real application and gave wide range of design options. It also represents typical software engineering problem where design, execution and review cycle comes in picture. WiFiRe draft being initial design of protocol, implementation part considered various strategies for same problem. WiFiRe gives opportunity to see how protocols are built with their development life cycle. Lot of assumptions and dependencies came as implementation is spread across various groups (Figure 3.5).

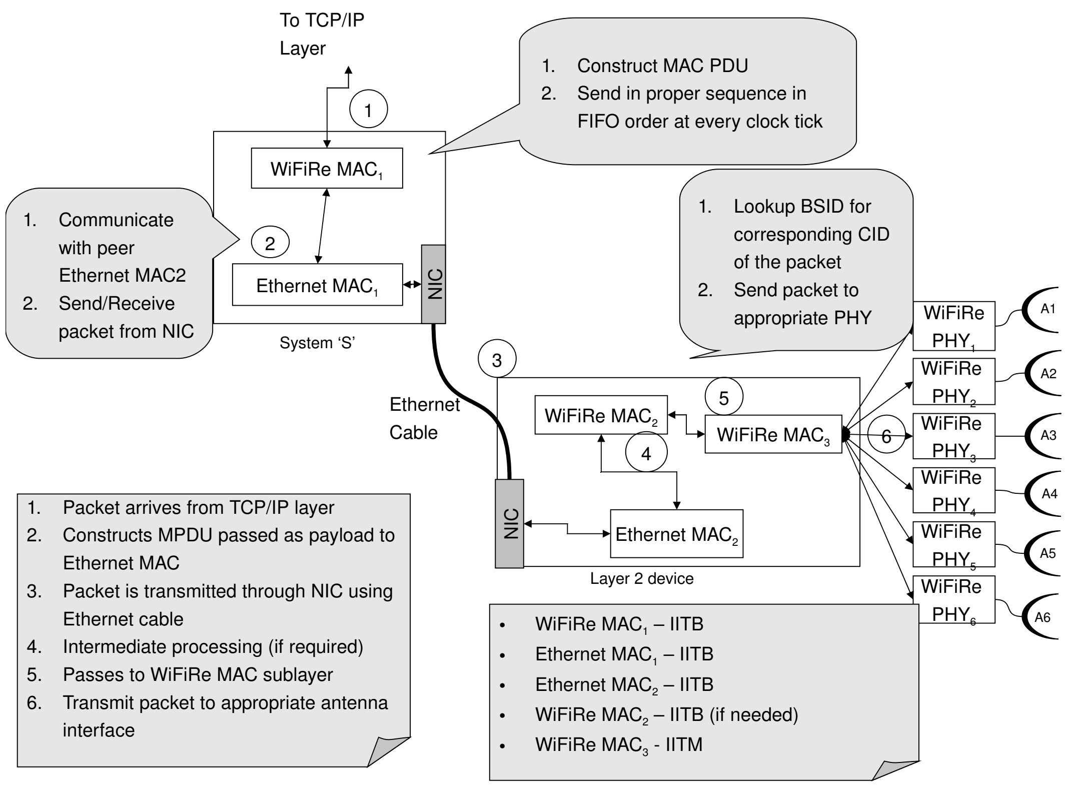

To TCP/IP Layer
1. Construct MAC PDU
2. Send in proper sequence in FIFO order at every clock tick
1
WiFiRe MAC$_1$
1. Communicate with peer Ethernet MAC2
2. Send/Receive packet from NIC
2
Ethernet MAC$_1$
NIC
System 'S'
1. Lookup BSID for corresponding CID of the packet
2. Send packet to appropriate PHY
3
Ethernet Cable
WiFiRe MAC$_2$
5
WiFiRe MAC$_3$
4
6
NIC
Ethernet MAC$_2$
Layer 2 device
WiFiRe PHY$_1$
WiFiRe PHY$_2$
WiFiRe PHY$_3$
WiFiRe PHY$_4$
WiFiRe PHY$_5$
WiFiRe PHY$_6$
A1
A2
A3
A4
A5
A6
1. Packet arrives from TCP/IP layer
2. Constructs MPDU passed as payload to Ethernet MAC
3. Packet is transmitted through NIC using Ethernet cable
4. Intermediate processing (if required)
5. Passes to WiFiRe MAC sublayer
6. Transmit packet to appropriate antenna interface
• WiFiRe MAC$_1$ – IITB
• Ethernet MAC$_1$ – IITB
• Ethernet MAC$_2$ – IITB
• WiFiRe MAC$_2$ – IITB (if needed)
• WiFiRe MAC$_3$ - IITM

Figure 3.5: MAC and dependency on PHY

WiFiRe's implementation was based on PCAP libraries. PCAP relies on NIC and kernel's configuration heavily. Most of these properties can not be modified to make it WiFiRe compatible. Linux kernel is assuming 802.3 based networks and sets network parameters (like frame length, firmware property etc.) accordingly. BS-ST link was also Ethernet and WiFiRe MAC has to be written keeping that (802.3) in mind. The reasons are: separation of hardware from

MAC (modularity), good design assumptions (dependency) about chip-set in WiFiRe draft and unavailability of 802.11b chip-set. Because of that approach, we concentrated heavily on ”802.3 compatible WiFiRe MAC”. In reality, it should not matter which physical layer will be used because once 802.11b chip-set is available, all 802.3 based assumption are not required.

WiFiRe's MAC used cross layering at several places. Few Examples: To decide new connection, it tries to check new TCP SYN message. For VoIP, SIP and RTP packets are treated as separate connections. For ARP, there are dummy replies. For QoS, it checks TCP/IP PDU type and decides service flow. WiFiRe also assumes web proxy and DNS server running. Some of these issues were discussed in design and few were implemented as well. This all leads to not just violation of classical layering approach but addition of great complexity to system. Argument against classical layering approach is that, now a day, almost all vendors are violating it!

WiFiRe classifier and CAC (decides QoS type) recognize real time and best effort flows. Now, when ever a new packet comes to BS or ST, classifier must be aware of TCP/UDP structures, addressing parameters, delay requirement etc. Packet's header will go through different filters and several parameters will be captured from it. These parameters will serve as inputs to classifier and CAC. This complex CAC and classifier [9] will take significant amount of time to give output and then packet will be processed further. This processing delay is not desirable for VoIP packet which has very stringent latency requirements. All cross layer assumptions are invalid if packet is encrypted. Further details are given in Section 5.3.

WiFiRe design is very much similar to WiMAX which has several advantages. Each design and implementation decision can be compared with WiMAX's current status. Mistakes done by WiMAX will not be repeated in case of WiFiRe (like complex and expensive SS). Both have similar layering approach, modularity and abstraction in those layers is also same. WiMAX is closely 'talking' with hardware while WiFiRe will keep modular approach and 'minimum talk' with hardware. This design decision was very helpful to keep software MAC in developing state and not worrying about hardware.

There are few more debatable things like fixed slot size, PHY overhead, header compressions, VoIP codec, total capacity of system etc. But, first iteration of implementation was supposed to be simple and basic working prototype. It had gone in right direction.

Chapter 4

WiFiRe MAC implementation

4.1 Related work

SRAWAN MAC [17] is TDMA MAC similar to 802.16 developed at IIT Kanpur in collaboration with Zazu Networks, Bangalore. In the implementation of MAC, author changes driver of 802.11 off-the-shelf device to support their protocol. SRAWAN MAC has frame structure which is similar to WiMAX [6]. It supports UL-MAP, DL-MAP, uplink and downlink sub-frames etc. Implementation is done using Atheros madWiFi drivers.

WiFi-based Long Distance (WiLD) networks [4] has proved longer range wireless using 802.11 hardware. They have changed CSMA/CA with MAC which is more proper for longer range. Implementation uses adaptive loss-recovery mechanism using FEC and bulk acknowledgements for better performance.

Intel's WiMAX chip-set for 802.16d [18] uses MAC described in WiMAX recommendation for their development. This board is developed with intention to use it for rural development.

4.2 WiFiRe LAN emulation

Simulations have been done to check WiFiRe protocol's correctness, performance, scheduler design and few PHY related issues like antenna behavior [16]. We would like to test protocol behavior in actual implementation and emulation on LAN is the first step towards it. We are using layer-2 functionalities of C sockets to implement WiFiRe MAC and test it on DIX2 Ethernet based LAN. Emulator is carefully designed so that some modules are PHY dependent and some are PHY independent. PHY dependent modules will be replaced appropriately when WiFiRe hardware becomes available.

There are few assumptions taken while developing the testbed as mentioned below.

- Single sector, few STs and clients
- Clients are connected to ST using 802.3 Ethernet only
- Single proxy machine to handle all the requests coming from clients
- MAC code in user space

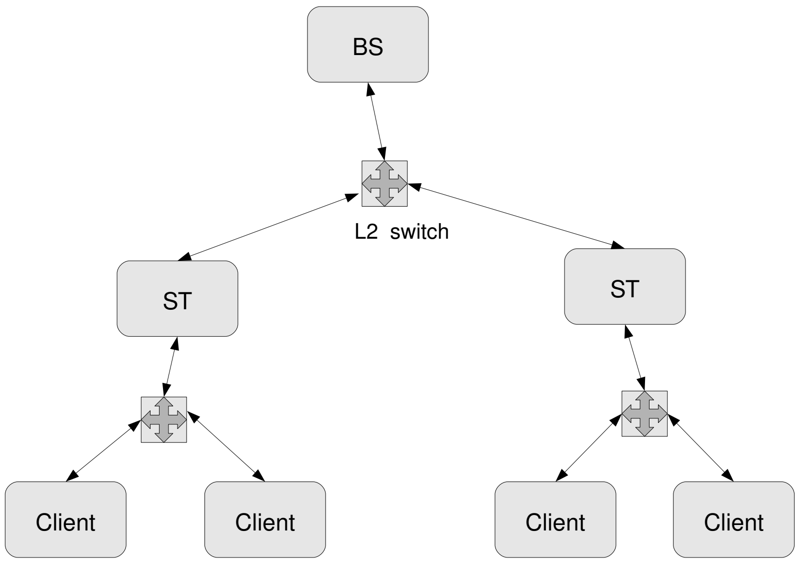


Figure 4.1: WiFiRe LAN emulation prototype

Implementation with PF_SOCK in user space

While observing need to interact with wireless hardware, there is urge to develop kernel module for WiFiRe. Implementation is done in C sockets instead of kernel for following reasons:

- A user level communication component, which provides direct access to low level communication (like putting bytes directly on PHY) mechanisms, bypasses operating systems complexity (like interrupts, sock-buff) in critical paths of communication.
- Overheads of kernel traps and memory copies along with various dependencies between user space and kernel space are avoided.
- To develop a prototype when WiFiRe protocol still evolving and changing, it is difficult to change code and debug in kernel module frequently.

- Debugging is much easier, in case of frequent changes.

We have used PF_SOCK for preparing this layer-2 sockets. PF_SOCK reads and writes directly to NIC. Basic calls to PF_SOCK are presented in chapter C of appendix.

4.3 Ethernet Testbed

As mention earlier, WiFiRe PHY board is under development. In the absence of PHY, development of MAC is done using LAN Emulation. Ethernet is used to emulate the behavior of wireless PHY. Few assumption taken while development are described below.

- Both BS and ST are connected using a cross cable, which eliminates problems of propagation delays, ranging, synchronization and other such wireless specific issues. We assume that such issues will be taken care by underlying hardware in actual implementation.
- Implementation does not perform real ranging procedure as test-bed uses Ethernet cable where propagation delay is not variable. In this scenario, Beacon transmission is enough for synchronization. Purpose of ranging here is to assign and transfer basic and primary CIDs only.
- As our emulation runs on Ethernet, and PCs are connected directly using RJ-45 cables, it cannot get total flexibility on frame structure, size, CRC etc. This Ethernet link restricts size of individual packets and WiFiRe MPDUs as well.
- Due to absence of real hardware clock, software timers are used in implementation which guarantees precision only up to milliseconds.
- CS layer (Convergence Sub layer) of WiFiRe keeps a pool of CIDs. CS layer have details of Ethernet to WiFiRe mapping and their header details. Current CS layer can be interfaced with Ethernet based media only and we wish to add other LAN technologies like 802.11 in future.

Implementation of WiFiRe MAC is done using C sockets on Linux platform. Program runs in user space. BS and ST are Linux terminals with two NICs each. Network parameters like IP address, ARP cache, DNS, proxy settings are kept fixed to keep the implementation simple.

WiFiRe testbed (Figure 4.2) includes a BS with two NICs, an ST with two NICs, a server which acts as gateway, FTP, proxy and web server. Several clients are connected to ST through Ethernet switch. BS is connected to ST using DIX based Ethernet cross cable. Link configuration depends on NIC drivers being used, connecting media and NIC firmware's support for flexible frame structure.

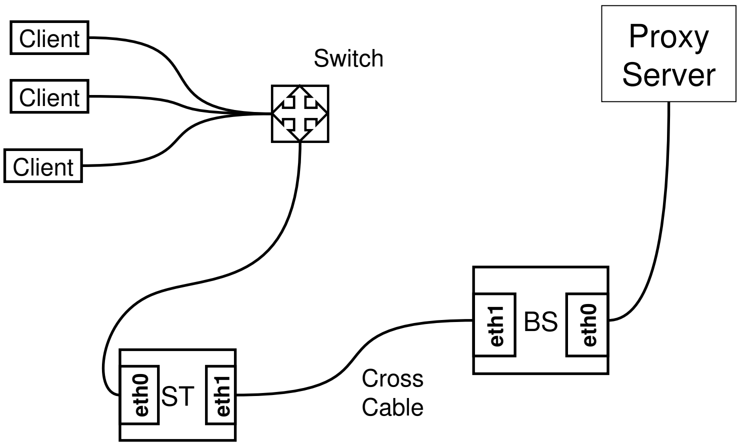


Figure 4.2: Ethernet Testbed

This link is treated as wireless link of WiFiRe. Subsequently, it will be replaced with the hardware for WiFiRe PHY (Figure 4.3. ST's *eth0* is connected with end users via switch (Figure 4.2). BS is connected to server using *eth0* and sends data to ST by other interface (*eth1*). *eth0* of ST and BS have standard 802.3 MAC based connectivity and it makes the network transparent for end users and Server (S) respectively. End-user clients send packets to ST, ST follows WiFiRe MAC slot structure defined in the UL-MAP and sends packets to BS in designated slots. BS receives these packets, processes them using WiFiRe modules and forwards them to S as normal Ethernet packets. Server sends reply to BS, which forwards it to appropriate ST and finally, client receives the packets.

4.4 Modules of BS and ST

As shown in Figure 4.4, there are various modules in BS dedicated to different functions of WiFiRe MAC. As expected, BS and ST have significantly different functionality in WiFiRe MAC. BS MAC contains packet classifier, CID generator, packet controller, scheduler and timer clock. Packet classifier detects packet type, QoS requirements and connection details. It makes

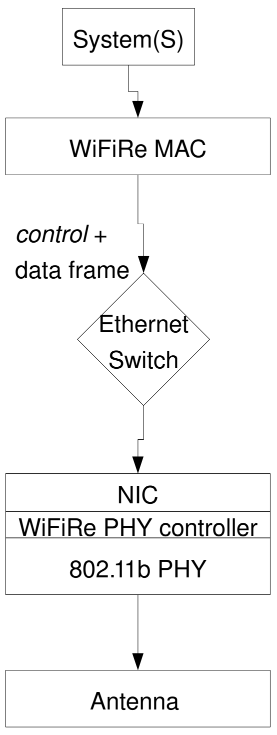


Figure 4.3: Proposed IITM PHY integration with MAC

appropriate entries in the mapping table (like BS-TABLE or MC-TABLE). CID generator module generates 16-bit CID for each new connection. These 16 bits are divided in subgroups, such that each group of bits represents ST-ID, Type of Service (ToS) and client. Packet controller sends packets to outgoing queues, buffer queues or simply drops them based on MAC requirement. Scheduler keeps track of all connection's requirements and generates DL-MAP and UL-MAP. Both MAPs are sent in first slot of downlink segment to all STs. Scheduler keeps dedicated slots for real-time traffic like VoIP. Timer clock reads UL-MAP and generates a sequence for packet reception.

ST has simpler MAC compared to BS. ST has MAP parser, packet controller, connection classifier and CS layer. MAP parser reads both MAPs and prepares ST for uplink and downlink slots. Packet controller maintains incoming and outgoing queue, filled with raw IP packets. Packet controller takes inputs from parser for packet sequence. Connection classifier detects packets coming from different end-users and makes new dynamic service addition (DSA) request for each new connection. CS layer understands Ethernet and WiFiRe header, converts packets appropriately from WiFiRe to Ethernet and vice-versa.

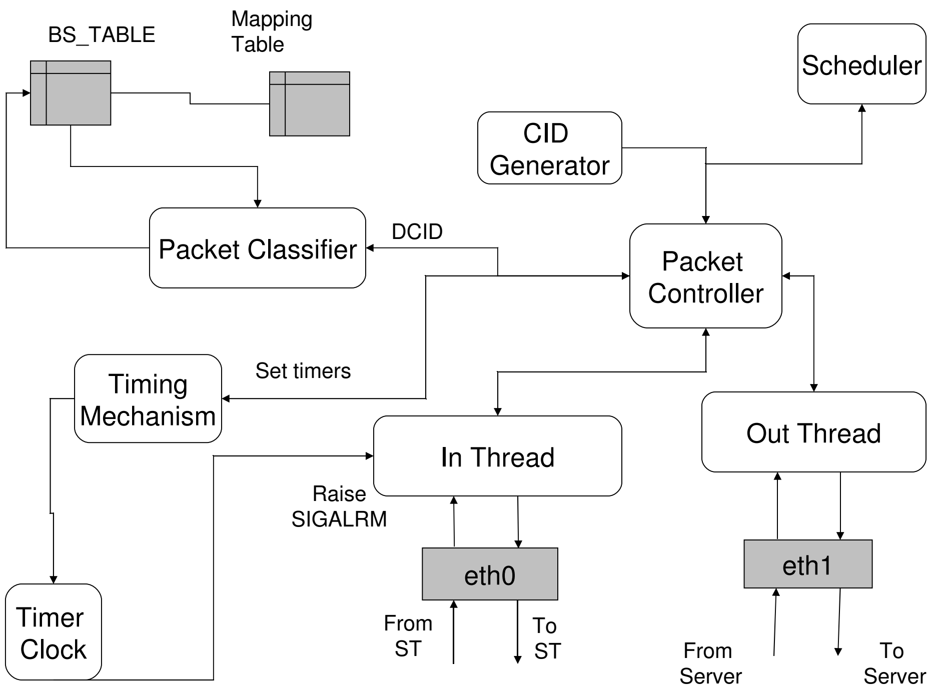


Figure 4.4: MAC modules at BS

4.5 Control and data flow for various events

Initially BS comes up and starts execution of its routine procedure of beacon broadcast, which is handled by packet controller. Whenever an ST comes up, it starts listening for a beacon. ST will go through ranging procedure, and ST finally get registered. ST is now ready to serve its client's requests, which were not served earlier. Timing mechanism module generates local sequence based on timer from DL-MAP/UL-MAP. This timing sequence is fed to the Clock to raise interrupts in the beginning of designated slots to wake up ST at required slot.

Each ST maintains ST-TABLE which includes MAC address of client, CID and ToS. Each row of the table indicates unique connection from ST to BS. BS also maintains two tables: BS-TABLE and MC-TABLE. BS-TABLE entry includes BSID (indicates the sector), STID (indicates the ST) and CID while MC-TABLE entry includes CID, client IP and ToS. Here, we give an example (Figure 4.5) of typical HTTP transaction to illustrate actions taken in WiFiRe MAC at ST and BS.

Whenever ST receives Ethernet packets from client(s), following actions are performed in uplink:

1. Packet classifier checks if received packet belongs to any existing connection by scanning

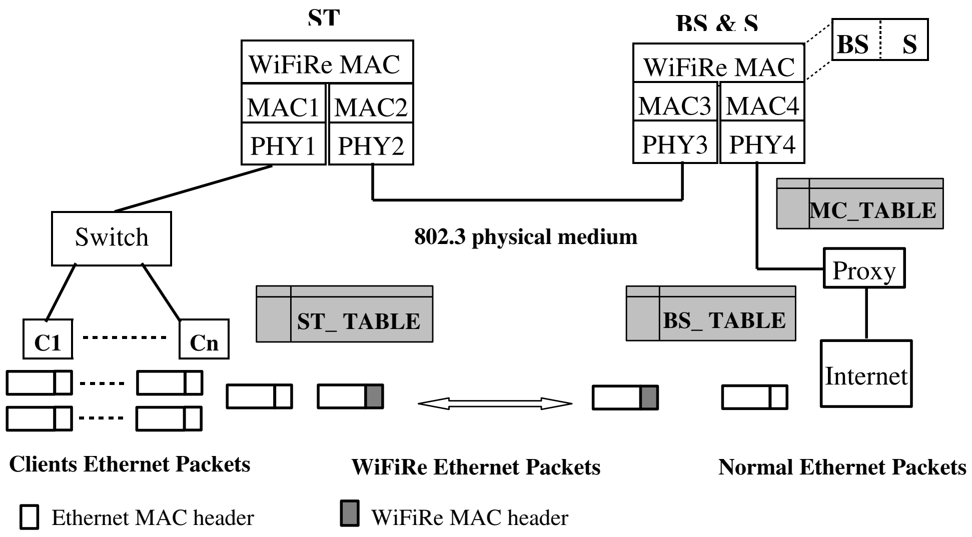


Figure 4.5: Layering approach in WiFiRe emulation

table entries. If matched, goto to step 4. If no match found, then make new entry in the table with temporary CID.

2. ST creates DSA request and waits for DSA response from BS. Ethernet packet will be stored in the buffered queue till the response arrive.

3. Once ST receives DSA response, it updates the table entry in ST-TABLE with newly allocated CID.

4. ST removes existing Ethernet header from the packet. ST's CS layer creates a WiFiRe header (based on CID of the table entry) for raw IP packet PDUs.

5. ST transmits WiFiRe packet to BS through outgoing socket according to UL-MAP.

Action at BS:

1. BS captures the packet from *eth1* (Figure 4.2) using socket.

2. BS reads the WiFiRe header and updates the table with IP address to CID mapping entry.

3. Packet controller removes the WiFiRe header after making table entry and constructs an Ethernet MAC packet for proxy.

4. Transmits Ethernet packet to proxy, which ultimately forwards these packets to Internet and responds back to BS.

5. BS buffers the responses and send them in downlink frame according to DL-MAP.

Similar steps are followed for downlink as well.

Chapter 5

MAC modules in testbed

WiFiRe draft [2] has mentioned every single detail about control messages, structure of all the fields in frame, frame size in seconds and bytes, registration, ranging and other such details. Earlier development [16] has followed these details and wrote code accordingly. There are several modules in WiFiRe MAC which are highly dependent on each other. It was assumed that, if draft is followed correctly, all modules will work correctly.

Current WiFiRe MAC development has followed slightly different method while writing code. Instead of writing whole code for all the modules, it was started with basic steps for communication. For example, transmit single packet from client to Internet which goes through ST and BS (tunneling). While writing code for this module, WiFiRe draft is not followed at all. Once given module is working correctly, its behavior will be changed/tweaked to follow WiFiRe draft specification. This method has speed-up development process and showed actual working of protocol from initial stage itself. It is much easier to observe packet flow and write code accordingly. Using this approach, it is easy to detect the problem faced in actual client usage (for example, web access) in initial phase. There are several restriction on client applications and their configuration (details in Appendix A). All the completed modules of MAC are described in this chapter.

5.1 Uplink and Downlink frame

WiFiRe frame (Figure 3.2) is divided in two major parts: uplink and downlink. Uplink sub-frame contains ranging and UL-MAP/DL-MAP slot as well. Packets from BS to STs will be transmitted in downlink frame while packet from STs to BS will be transmitted in uplink frame.

Figure 5.1 describes the procedure used to construct uplink frame at ST. All the clients send their packets to ST with whom they are attached. Now, ST will receive this packet and keep

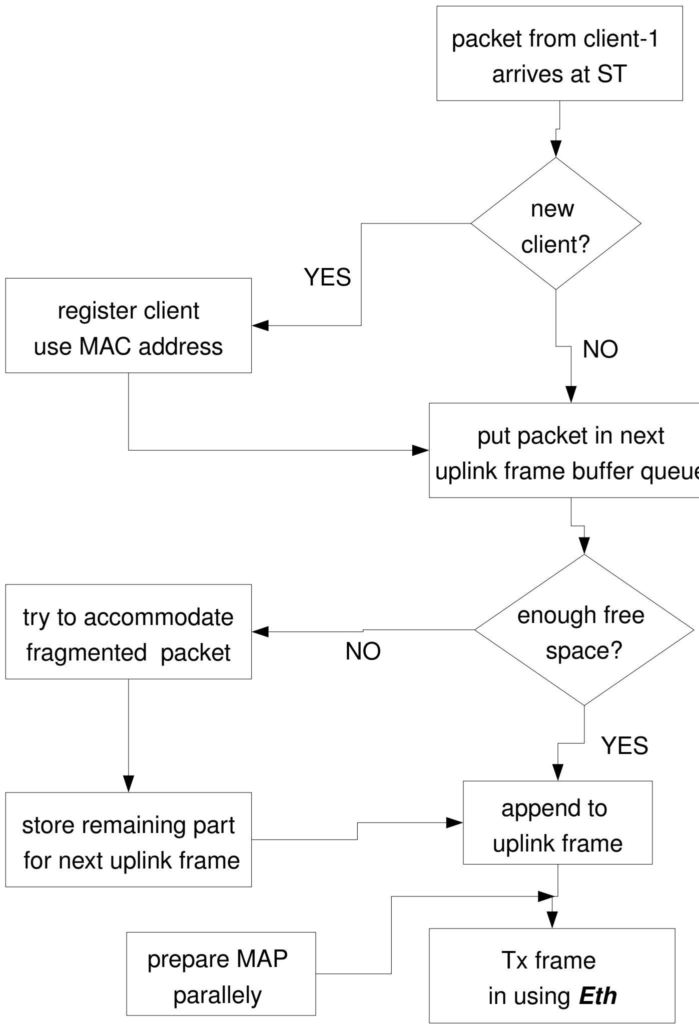

packet from client-1 arrives at ST
new client?
YES
register client use MAC address
NO
put packet in next uplink frame buffer queue
enough free space?
NO
try to accommodate fragmented packet
YES
store remaining part for next uplink frame
append to uplink frame
prepare MAP parallely
Tx frame in using *Eth*

Figure 5.1: Construction of uplink frame

it in buffer without removing 802.3 header. All such packets will be encapsulated in single meta-frame called uplink frame. ST will transmit this frame with UL-MAP (Figure 5.3). Once uplink frame is received at BS, modules at BS will *decapsulate* frame; individual packet will be forwarded to their destination (to proxy machine). Here, proxy will receive this packets as normal 802.3 packet (Figure 5.2). These packets will have source MAC address as client's MAC address and destination as proxy's MAC address. Because of this method, client and proxy are unaware of underlying WiFiRe protocol. Other implementations also follows similar approach [18].

In many cases, packet size is large and can not be accommodate in single frame. In this case, packet will be fragmented on ST. First part of this packet will be sent with uplink frame while second part of packet will be buffered and sent in next frame. On BS side, BS will

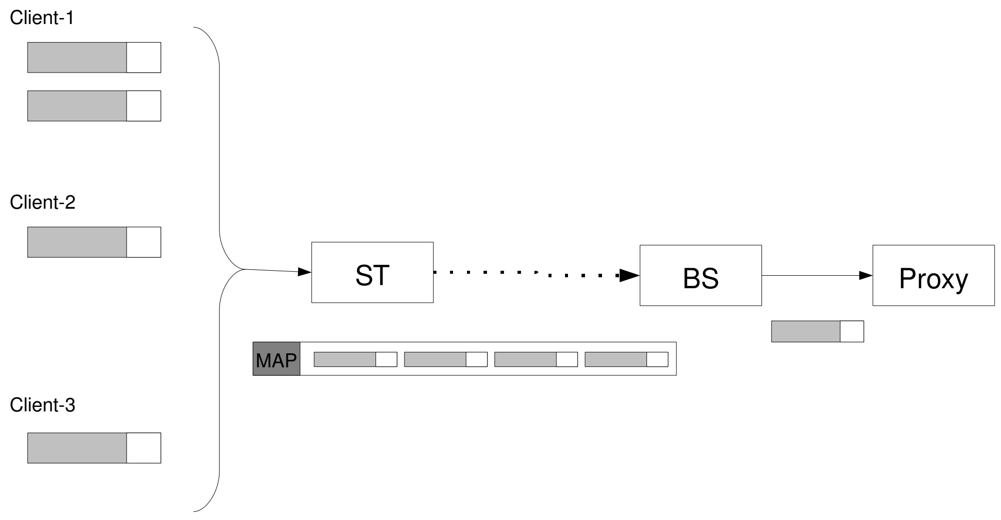


Figure 5.2: Encapsulation of Ethernet packets

check IS-FRAG flag from MAP it receives. If flag is set, packet will be buffered and it will be re-assembled at the time of next frame's arrival and remaining part of packet (Figure B.3). Downlink frame also follows similar approach.

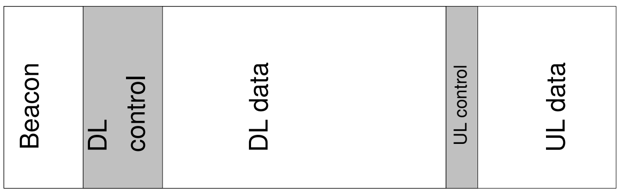


Figure 5.3: DL and UL frame of WiFiRe

5.2 Registration and Ranging

WiFiRe draft [2] has mentioned ranging slot as first slot in downlink frame. Currently, we are using Ethernet testbed and propagation delay is negligible. Ranging here is just done as part of protocol and main aim of that is registration only. Beacons will be transmitted at every 10 mSec with destination as broadcast address for all STs. Whenever ST comes up first time, it will receive this packet and send registration request (Figure B.1). Once BS get this request packet, it will allocate ST_ID to that particular ST (Figure B.2). BS will also make entry in ST-TABLE which describes which ST has which MAC address. Registration process can be described in following steps:

- BS sends periodic beacons
- ST sends registration request
- BS adds ST in list, allow access
- ST starts transmission
- Client can not start communication before registration
- BS and ST shut down / restart condition handled

5.3 Scheduler design

As current testbed does not take care of timing issues, scheduler works on soft slots. Timing calculation is highly dependent on hardware on which MAC is running. Actual WiFiRe frame will have slot length in microsecond which is not achievable using PC.

Current scheduler is similar to FIFO and assumes that there is always slot available for ST. To summarize, scheduler is in very basic, naive mode and will be enhanced in next stage of project. We propose scheduler design based on following assumptions.

- FIFO packet schedule
- No service distinction for current version (VoIP and HTTP are at same level)
- There is always slot available
- Packets are never dropped, they will be kept in buffer till their turn comes, maximum buffer size is 100 packets for each ST
- All STs get equal share of resources
- Unused slots can be (re)allocated to other STs on request
- UL-MAP and DL-MAP to support PDU encapsulation

5.4 WLAN testbed

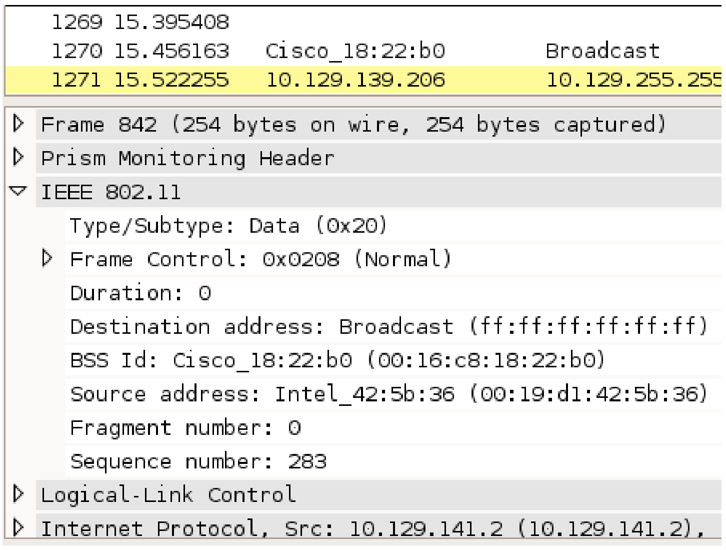
WiFiRe testbed is emulation of MAC on Ethernet media instead of 802.11 PHY. We opted for Ethernet because of its simplicity. Ethernet has very simple 14 byte 802.3 MAC header, we used switch/hub which connected BS to STs as broadcast medium. While doing tunneling from BS to ST, Ethernet was obvious choice because it allows transmission of packet without specifying its source or destination. Ethernet has several other advantages like extremely reliable connections, easy fault detection in case of failure, NIC configuration and drivers etc.

Actual BS to ST link will be wireless 802.11b with WiFiRe MAC running on top of it. Replacing Ethernet with WiFi will give more realistic view of WiFiRe. Following are the difficulties involved in wireless testbed.

- Working with wireless devices is much harder than working with Ethernet.
- Connectivity in wireless is more unreliable with varying packet error rate.
- 802.11 MAC header with CSMA/CA and channel sensing is more complex than Ethernet.
- Ethernet allows transmission of packet with wrong source/destination address while WiFi does not.
- Interference from other wireless sources should be dealt with proper module which drops them without any memory leak.
- Adding two wireless card on same machine is not advisable. If we treat WiFiRe link and client connection on same card, detecting client packets from source need some filtration criteria.

If we maintain MAC header as shown below, we can create wireless sockets using PF_SOCK.

```
ieee80211header {
u16 ver:2,
type:2,
subtype:4,
flags:8;
u16 duration;
```



```
1269 15.395408
1270 15.456163   Cisco_18:22:b0        Broadcast
1271 15.522255   10.129.139.206        10.129.255.255
▷ Frame 842 (254 bytes on wire, 254 bytes captured)
▷ Prism Monitoring Header
▽ IEEE 802.11
    Type/Subtype: Data (0x20)
  ▷ Frame Control: 0x0208 (Normal)
    Duration: 0
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    BSS Id: Cisco_18:22:b0 (00:16:c8:18:22:b0)
    Source address: Intel_42:5b:36 (00:19:d1:42:5b:36)
    Fragment number: 0
    Sequence number: 283
▷ Logical-Link Control
▷ Internet Protocol, Src: 10.129.141.2 (10.129.141.2),
```

Figure 5.4: 802.11 MAC header captured in wireshark (monitor mode)

```
u8 mac1[6];
u8 mac2[6];
u8 mac3[6];
u16 SeqCtl;
} Ieee80211Header;
```

As 802.11 MAC header has many fields, maintaining state of all these fields is not trivial.

Testbed is equipped with D-link DWL-650 wireless 802.11 card. Card uses madWiFi drivers on ubuntu Linux. MadWiFi driver creates pseudo interface for each wireless card (Figure B.4). To create this interface, following command is used. It will put wireless card in monitor mode.

```
$wlanconfig ath1 create wlandev wifi0 wlanmode monitor
```

Monitor mode will represent 802.11 MAC header completely. In disadvantage, card in monitor mode will not able to transmit packets at all. It is similar to passive listening or promiscuous mode. If we keep wireless card in normal (AP mode) mode, it will hide internal details of wireless parameters and will work as normal Ethernet interface. This pseudo interface can be easily used by normal sockets and makes testbed transparent of underlying PHY. In our case, Ethernet and wireless card has same header and same byte field. We can address them using *eth0* and *wlan0* respectively.

Once socket of Ethernet is replaced with wireless, testbed has following setting: BS and ST is connected using wireless access point. Currently, they operate in AP mode using channel 6 in 2.4GHz band. BS is connected to outside world using proxy via Ethernet interface as earlier (Figure 5.5 and 5.6). Similarly, ST is connected to clients using switch. Wireless link between ST and BS is very similar setting as mentioned in WiFiRe draft [2]. In actual deployment, distance between BS and ST will be more while other connections will remain same. This wireless link has MTU of 2312 bytes and follows CSMA/CA mechanism.

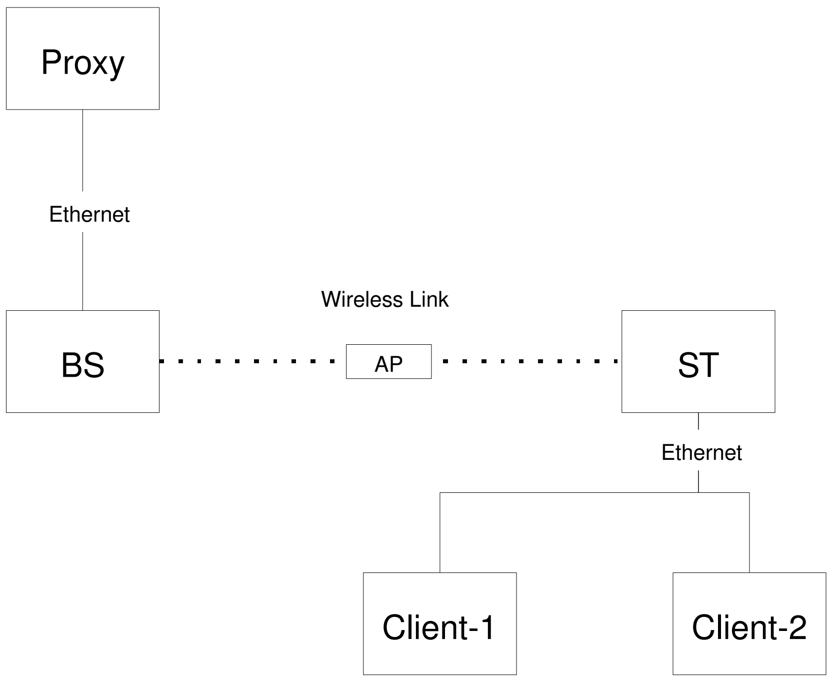


Figure 5.5: Wireless testbed setup

Although performance evaluation and actual throughput calculation is out of scope for this report, we predict 10-15% improvement using this wireless link compare to CSMA/CA link. Actual reason for CSMA/CA's poor performance is channel sensing and back-off interval. If traffic pattern is not predictable (given high load), there is very high probability of channel sensing and back-off. This event reduces throughput of system and frame might suffer delay in transmission. In case of WiFiRe WLAN testbed, downlink and uplink frame transmission interval is pre-define at fixed periodicity (10 mSec). Due to this fixed interval, this link will never go for back-off period because there will not be any simultaneous transmission. We assume absence of interference in this case. Testbed with wireless link gives better idea about actual deployment (interference, packet drop, etc) as well.

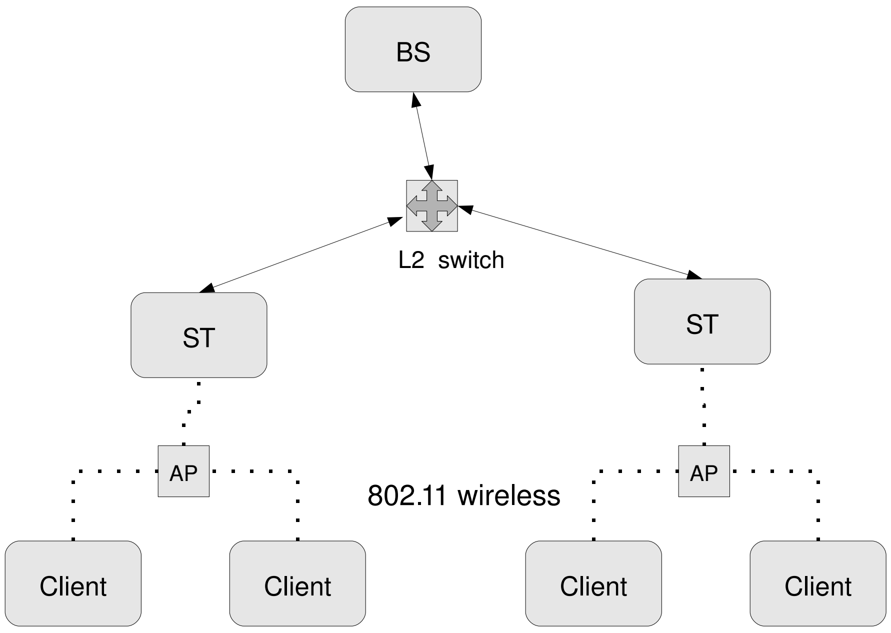


Figure 5.6: Wireless testbed : alternate setup

5.5 GPSS mode

WiFiRe draft recommends GPC (grant per connection) mode of operation. WiMAX scheduler usually prefers GPC mode for better QoS. In the case of GPC, BS maintains CID, slots, buffer, timing details per connection. For example, client-1 is using VoIP and HTTP, it will have two CIDs. Both of them will get different QoS level and scheduler's priority. GPC mode leads to very complex scheduler and MAP design at BS.

We have adapted GPSS (grant per SS) mode of connection in WiFiRe MAC. GPSS is prescribed in WiMAX standard but rarely used. In case of GPSS, connection identification is done at ST level. We treat all the packets of single ST at same level. Scheduler will grant connection resources like slots, CID, bandwidth, QoS service guarantees per ST basis. It leads to very simple scheduler design at BS with no QoS guarantee for individual packets/clients. GPSS mode needs look-up table for client to ST mapping. Whenever packet comes from Proxy, BS will look into BS-TABLE (As mentioned in Table 3.1), will find appropriate ST-ID, append packet in particular Queue. To add client level QoS support, ST will need some more intelligence by adding client scheduler. In current implementation, ST distributes total slots among all the clients equally.

ST-ID	client MAC
1	AA-AA-AA-AA-AA-AA
2	BB-BB-BB-BB-BB-BB
1	CC-CC-CC-CC-CC-CC
1	DD-DD-DD-DD-DD-DD
2	EE-EE-EE-EE-EE-EE

Table 5.1: BS-TABLE: mapping from client MAC to ST in GPSS mode

Encapsulation of Ethernet packets

In earlier basic implementation of WiFiRe, ST removes Ethernet header of packets coming from client and send PDU to BS. This was done to remove confusion of multiple header and optimize bandwidth. In that case, ST will remember mapping each packet's ID with corresponding Ethernet header. This method is not scalable and there are doubts on its correctness as well. In the current implementation, packet is preserved as it is. ST does not remove Ethernet header before putting it in queue. This same packet reaches BS and BS will send it to proxy. This procedure is transparent to external nodes. Proxy and clients are not aware of underlying protocol and feels that they are communicating directly (Figure 5.2).

5.6 Packet chains and memory management

BS and ST stores packet before forwarding it to its destination. These packets are processed based on their source/destination address and data it carries. All these packets are stored in link-list format. As of now, they follow FIFO approach but we can selectively insert/remove packets if required. If we consider current design on base station (BS), packets are stored on destined ST basis. Scheduler keeps track of all these queues and decides DL-MAP based on frame length. On ST side, there is single FIFO queue. One can introduce different queues for different traffic like VoIP and web if required.

This module is also coupled with memory management unit which actually deals with system's memory. Earlier, WiFiRe used system's default memory management which is not so reliable. For any wrong memory call, program will come to halt with *segmentation fault*. Using our own module, such problems can be trapped. This system also takes care of 64-bit processor architecture. Memory management module also checks each queue's memory occupancy. It

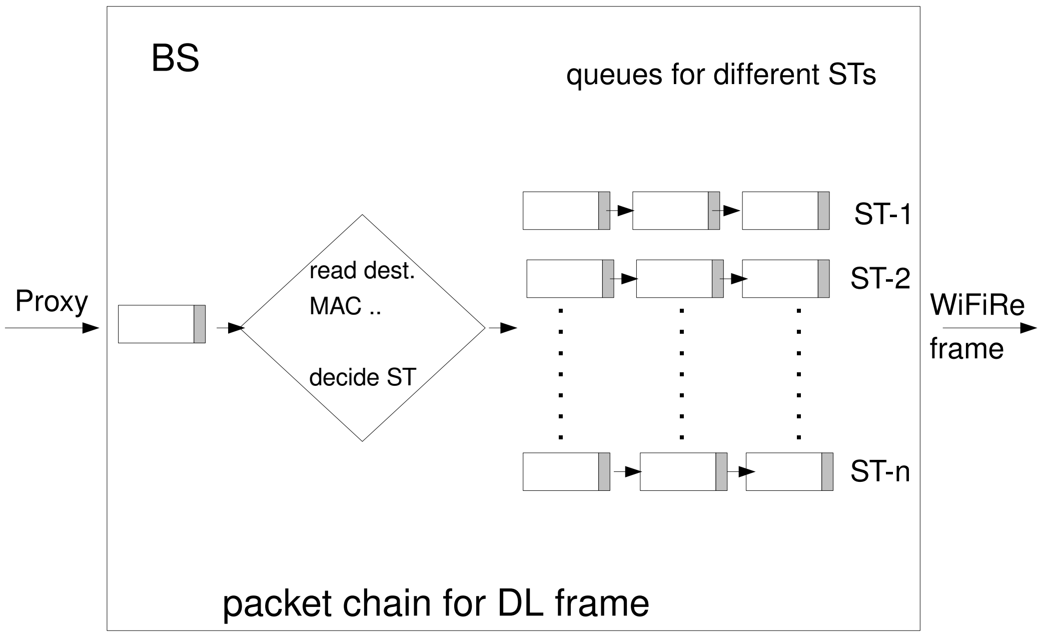


Figure 5.7: Packet queues at BS

does not allow any queue to grow indefinitely because program can start memory stack trashing and eventually comes to halt. It can be determined that which ST is getting longer queue (number of packets) in buffer.

In case of QoS, if someone wants to provide different queues for various traffic, it can be easily done with very little change in this module.

5.7 Filters and local traffic

There are modules written to filter packets on BS and ST. This module uses BS_TABLE and ST_TABLE to detect whether packet should be allowed to enter system or not. On ST, once DL frame is received, ST checks DL-MAP and scans if any packet/slot belongs to it. It reads given bytes from frame and drop remaining packets. For broadcast packet, ST makes sure that it does not forward packets which has actually came from itself. This scenario is explained in Section 6.2.7.

Whenever BS receives packet, it reads its destination MAC address and start matching it

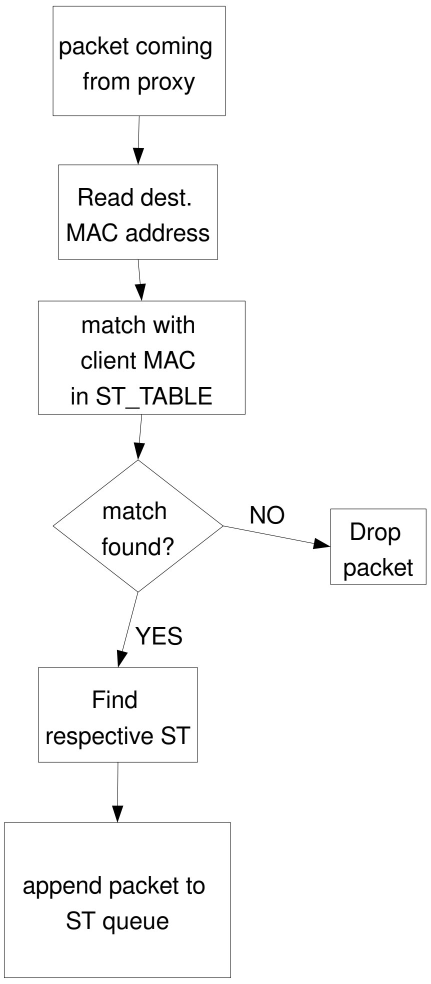


Figure 5.8: Packet filter at BS

with client MAC in ST_TABLE. If destination is one of the client, it adds the packet to respective ST queue (based on ST_TABLE). If it does not belong to any of its clients, it will be dropped. If destination MAC is broadcast address (FF), packet will be appended to broadcast queue and will be handled accordingly. Usually, BS should not receive broadcast packet (e.g. ARP query) intended for clients because clients are hidden from external world and their address (IP, MAC etc.) are not known beyond gateway machine.

5.8 Log and config files

As WiFiRe system handles numerous clients and STs with variety of traffic, there is chance that something does not work correctly. For example, client is not able to connect or voice communication is affected severely. In all such cases, its extremely important to know the cause of problem and solve it. We have module for BS and ST which write all such details in log file. Its very easy to detect problem/bug once we have log file. There is an option to specify the verbosity level (for various messages) in log file.

WiFiRe modules are layer-2 program and can be run on any machine with at least 2 network interface. BS and ST module can be installed if it gets correct details of its parameter. There are configuration files which contains all such details. Few important parameters (with their default value) are: interface name (eth0), frame periodicity (10 mSec), MTU size(1450 bytes), proxy's MAC address, verbosity level for log files etc.

Chapter 6

Design and implementation issues

As mentioned in previous chapters, this project involves development of layer-2 protocol from the beginning. It requires knowledge of system and networking related concepts with their practical implementation. There are various system level details like memory, clock timer, processes and file read/write encountered while writing code for these modules. It also involves sockets, framing, TCP/IP stack details and application layer protocol from networking point of view.

6.1 Design decisions

From beginning, there were several design decision made regarding implementation of WiFiRe MAC. While developing this MAC, we realized that some changes are required to optimize certain aspects of MAC. Following are design decisions which we took while writing code for WiFiRe. Some of these are implementation specific and some are protocol level changes.

- In the beginning of the project, our goal was to produce layer-2 protocol only. We assumed that upper layer handling is not the scope of the project. Over the time we realized that, if WiFiRe MAC module wants to show results to end-user client, we have to deal with TCP/IP stack and above. This forced us to add proxy machine to testbed and various application layer protocol server are installed. Few results are discussed in section 6.3.

- We started with implementing all modules mentioned in WiFiRe draft [2] in the beginning. As code grew, we realized that not all modules are required for basic working prototype. We sidelined few modules and concentrated on important ones. Focus of work was shifted to end user's perspective. Once we had all basic modules working, we thought about all functionality mentioned in draft.

- WiFiRe draft presents GPC based connection identification for various traffic. It will detect each connection (same or different clients) and grants number of slots based on its bandwidth requirement. We believe that it has very high overhead of traffic detection and processing. Basic prototype does not need such finer details for scheduler. We adapted GPSS mode which is mentioned WiMAX standard as well.

- We started implementing BS and ST with assumption that BS is heavy system where all MAC modules has to be written. ST should be naive software which simply works as gateway. As project progressed, we added some intelligence in ST and found it more balanced system.

- As draft mentions BS and S to be same machine, we started our design in that manner. We felt that keeping all modules on one machine was bit complex because we want BS to be layer-2 device. It also adds great deal of confusion in deployment and execution. We added one more machine to the system as proxy machine which handles all application layer protocol details (e.g. DHCP and DNS). BS and S are separate machine now.

6.2 Implementation issues

6.2.1 RTP point-to-point connection

- Problem: VoIP communication between two clients of different ST can not be established although, call between two clients of same ST is working.

- Troubleshooting:

  - Both clients (from different ST) are getting registered to VoIP PBX using SIP.
  - Whenever call is made, call is logged on VoIP server's log file.
  - RTP works correctly within ST because it has direct (P2P) communication. RTP makes bridge between two peers and no server is involved in this call.
  - Whenever client is sending packets to client in same ST, it goes from same switch because both clients (and ST) are connected to same L2 switch.
  - Calls to client in other ST fails because BS does not forward such packets to ST. BS assumes all traffic is for proxy and forwards accordingly.

- Solution: Created module for *local traffic diversion*. This module detects traffic intended for other clients, does not forward it to proxy. All such packets are queued back for DL traffic.

6.2.2 TCP checksum offloading

- Problem: Client is able to access http (web) for smaller pages (e.g. Hello World!) but normal web access does not work.
- Troubleshooting:
  - Client is able to ping proxy.
  - If we observe packets in *tcpdump*, client is receiving ICMP packets (ping) and TCP packets (for web) correctly.
  - Browser does not display bigger web-page (e.g. wikipedia). If we observer packets in *tcpdump* on client, it displays TCP checksum off-loading error but packet size is shown correctly on proxy and client (usually 1514 bytes).
  - TCP Offloading is used in high-speed network where whole TCP/IP stack is implemented as part of NIC.
  - In high-speed network, calculating checksum for each packet takes significant time and processor is always loaded with checksum calculations. This functionality is delivered by NIC in this case. NIC will get packet from processor without checksum (sometimes, dummy checksum is appended) and NIC calculates the correct one.
  - We found out that our card does not have TCP off-loading functionality. We concluded that TCP off-loading was *false alarm*.
- Solution: We found bug in our code which was not handling packets with size of more than 1500. Mistake was in copying 1 Byte incorrectly. It affects checksum calculation and TCP/IP stack drops the packet. Bug was resolved and web-access started working.

6.2.3 Firewall on proxy

- Problem: VoIP clients are not able to register with VoIP PBX.

- Troubleshooting:
  - Clients are able to ping proxy and can have web-access.
  - PBX software is running on proxy itself.
  - Whenever client sends registration request, it gets ICMP destination host unreachable message.
  - There is firewall running on proxy machine which blocks all the requests to TCP ports (except 80). It blocks SIP requests and replies with ICMP destination unreachable message.
- Solution: Disable *iptable* service on proxy machine. It will stop firewall and access to VoIP server will be open.

6.2.4 64-bit architecture

- Problem: WiFiRe MAC module runs fine on IITB's machine but crashes in IITM's machines.
- Troubleshooting:
  - Symptoms: IITB machine are old (128MB RAM, Pentium-2 architecture) while IITM's machine are with higher configuration (512MB RAM etc.)
  - IITM machines are 64-bit architecture and WiFiRe's memory management module has not taken care for 64-bit address space.
  - *malloc* and *free* memory sytem call does not work properly and program crashes once gcc memory stack is full.
- Solution: Memory management module is updated to work with 32-bit and 64-bit architecture.

6.2.5 Background process of BS

- Problem: Client is receiving duplicate packet for each packet.
- Troubleshooting:

  - All ping acknowledgement have DUP ack (duplicate packet).
  - ST is receiving 2 DL frame in every cycle (10 ms). Everything is same (DL-MAP, number of packets) in two frame except frame number.
  - There might be two instance of base station program running at same time.
  - Previous BS program was not halted properly by program's *exit* system call. It was still running as a background process.

- Solution: Simple script which checks weather any instance of old BS program is running before starting BS program. If any process is still running, kill it: *pkill bs*.

6.2.6 Multicast packet from switch

- Problem: ST program crashes with memory error *segmentation fault* in 30-40 minutes of execution.
- Troubleshooting:
  - This behavior is very rare. After reading log files, we found that it received frame from BS which was not following WiFiRe frame structure. It didn't carried WiFiRe header (with frame number) as well.
  - BS and ST are connected using some L3 devices which generates this frame for multicast group message. ST assumed it to be WiFiRe frame and memory calculation failed.
- Solution: We developed small filter module which checks whether it is actually WiFiRe frame or not. If non-WiFiRe frame comes to NIC, it will be simply dropped.

6.2.7 ARP cache flush

- Problem: Clients of different ST are not able to communicate but clients within ST are able to communicate.
- Troubleshooting:
  - Both clients are connected and registered with ST (and BS).
  - Client generates ARP request before actual communication.

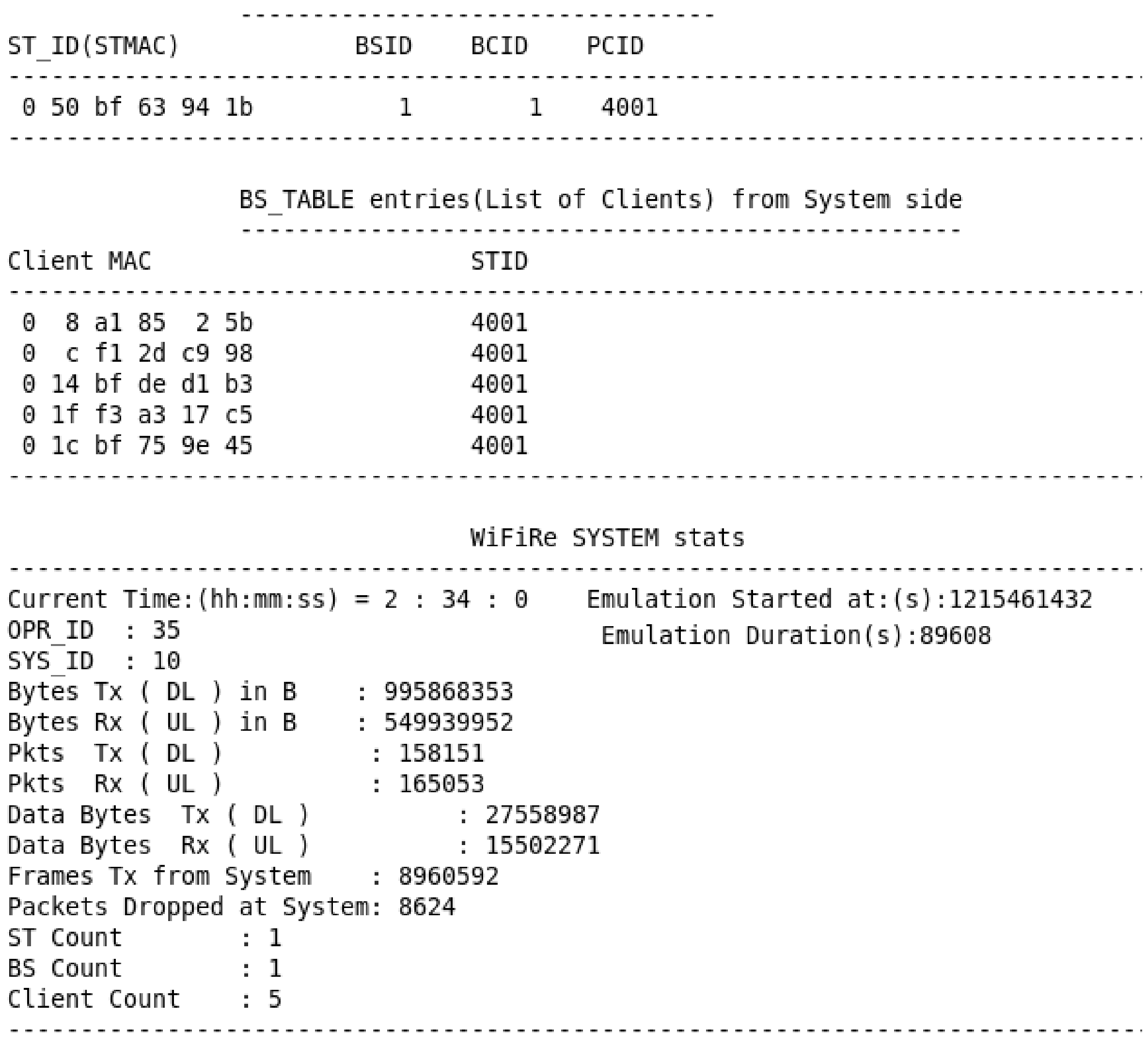
  - It gets reply from other client after 1 cycle but it is keep repeating ARP query.
  - ARP cache table is empty all the time.
  - Whenever client sends ARP query, its destination is broadcast (MAc address as FF) and source as its own MAC address. This packet reaches to BS, BS forwards this to proxy and append it to broadcast queue (As mentioned in 5.6). When ever this ARP query comes back to client itself, it flushes its ARP cache.
  - When ever machine gets ARP query with source address as its own address, it flushes ARP cache.
- Solution: On ST, one filter module to check if source address of the packet is one of the client's MAC address, don't send packet to any client.

6.2.8 DHCP's negative ACK

- Problem: One client is not able to acquire IP address using DHCP but client from other ST is able to acquire it.
- Troubleshooting:
  - Client from other ST is able to acquire IP address from DHCP server.
  - DHCP Server is running on proxy machine.
  - Client is connected to ST using old Access Point (working as Ethernet switch). It runs DHCP server inside it, but does not have client's MAC address in authorized list. It replies with NACK (negative acknowledgement), client assigns IP address as 0.0.0.0 and process repeats itself.
  - AP always reply before actual DHCP server because it is connected directly while DHCP server (running on proxy) has to wait for one cycle.
- Solution: Access point is replaced with L2 switch.

6.3 Results and traffic analysis

Figure 6.1 shows snapshot of WiFiRe console. There was bursty web traffic and some ICMP packets at the time of capturing this statistics. ST_TABLE gives all the details about current STs connected with BS. It has ST's MAC address, BSID, BCID and PCID. Other table is client's table where MAC address and association with ST is shown. Console also shows statistics on BS and ST about data transmitted and received. It shows 'Total Bytes' (control+data) transmitted in DL frame and received in UL frame. Similar for packets and actual data. It also counts total STs and clients currently present in system.



```
                --------------------------------
ST_ID(STMAC)            BSID    BCID    PCID
-----------------------------------------------------------------------------
 0 50 bf 63 94 1b          1        1    4001
-----------------------------------------------------------------------------

                BS_TABLE entries(List of Clients) from System side
                --------------------------------------------------
Client MAC                      STID
-----------------------------------------------------------------------------
 0  8 a1 85  2 5b               4001
 0  c f1 2d c9 98               4001
 0 14 bf de d1 b3               4001
 0 1f f3 a3 17 c5               4001
 0 1c bf 75 9e 45               4001
-----------------------------------------------------------------------------

                                WiFiRe SYSTEM stats
-----------------------------------------------------------------------------
Current Time:(hh:mm:ss) = 2 : 34 : 0    Emulation Started at:(s):1215461432
OPR_ID  : 35                             Emulation Duration(s):89608
SYS_ID  : 10
Bytes Tx ( DL ) in B    : 995868353
Bytes Rx ( UL ) in B    : 549939952
Pkts  Tx ( DL )          : 158151
Pkts  Rx ( UL )          : 165053
Data Bytes  Tx ( DL )          : 27558987
Data Bytes  Rx ( UL )          : 15502271
Frames Tx from System    : 8960592
Packets Dropped at System: 8624
ST Count        : 1
BS Count        : 1
Client Count    : 5
-----------------------------------------------------------------------------
```

Figure 6.1: Snapshot of WiFiRe console

Current statistics shows that, code is running for 89608 seconds (approximately 24 hours). Total bytes in DL is 995868353 which consists of beacon, data and control frames. Total bytes in UL is 549939952 which consists of data, UL-MAP and control frame. If we see actual data (e.g. TCP packets), DL is 27558987 mainly due to TCP and ICMP traffic. On uplink, data is 15502271 which is mainly ICMP and TCP acknowledgements.

We also tested the accuracy of SIG_ALRM timer using above statistics. We found that there were 8960592 frames sent in 89608 seconds. We have set periodicity of frame to 10 mSec, there should be 8960800 frames. We calculate accuracy as frames missed in running time of MAC. We get 208 frames missing in 89608 seconds, resulting as accuracy of 99.997%. We repeated same experiment with different parameters and STs and found similar results. Although, missing frame might look like bad design, it does not affect working of MAC much because, there is beacon every 10 mSec which takes care of synchronising all STs to BS.

Following is the ping statistics gathered on one of the clients. Client is pinging proxy machine (172.168.1.1) which involves ST and BS as intermediate hop. Once transmitted from client, this packet will be queued at ST, Once ST gets UL frame, it will be sent to BS. BS will forward this packet to proxy machine, proxy replies back to BS immediately. This packet will be again queued at BS for downlink, will be sent to respective ST in DL frame. Packet reaches to ST and it forwards it to client back.

```
--- 172.168.1.1 ping statistics ---
8978 packets transmitted, 8967 received, 0% packet loss
rtt min/avg/max/mdev = 9.866/15.200/39.305/3.112 ms
```

Result shows that average delay is 15.2 mSec. This statistics were taken when there was very little traffic from other clients. As WiFiRe frame periodicity is 10 mSec, reply can vary from 10 mSec to 20 mSec in uniform distribution which justifies average delay of 15 mSec. Our aim here is to quantify delay occurring in WiFiRe system in best case. If we communicate with some other machine (e.g. some machine in Internet), total delay will be more than 15 mSec.

We added traffic from other sources and tried seizing bandwidth to 100%, we saw delay of ICMP traffic in that case.

```
64 bytes from 172.168.1.1:  icmp_seq=2126 ttl=64 time=16.1 ms
64 bytes from 172.168.1.1:  icmp_seq=2127 ttl=64 time=12.1 ms
64 bytes from 172.168.1.1:  icmp_seq=2128 ttl=64 time=36.8 ms
64 bytes from 172.168.1.1:  icmp_seq=2129 ttl=64 time=110 ms
64 bytes from 172.168.1.1:  icmp_seq=2130 ttl=64 time=69.9 ms
64 bytes from 172.168.1.1:  icmp_seq=2131 ttl=64 time=64.8 ms
```

Datarate	more than 120KBps
Avg. delay	15 mSec
SIG_ALRM Timer accuracy	99.997%
Max. run time tested	more than 38 hours
Time to load web-page (56KB) (home page of wikipedia)	5 sec (avg.) (including delay in Internet)

Table 6.1: Statistics on BS

```
64 bytes from 172.168.1.1:  icmp_seq=2132 ttl=64 time=140 ms
64 bytes from 172.168.1.1:  icmp_seq=2133 ttl=64 time=150 ms
64 bytes from 172.168.1.1:  icmp_seq=2134 ttl=64 time=146 ms
```

Observe packets from 2128 to 2133, delay got knee-jerk increase with effect of other traffic. As traffic increases, there are more packets in queue for transmission. We used 100% of bandwidth to get above statistics.

WiFiRe's desired applications are web-access and VoIP calls and we did basic testing for the same. We downloaded file with size approximately 1GB from WiFiRe proxy to client. Datarate achieved during this was 120KBps with no significant other traffic. As WiFiRe DL frame is sent every 10 mSec periodically with MTU size of 1400 bytes, we get datarate mentioned above (1400 x 100 Bytes per second). Assuming some traffic from other client and control header, 120KBps seems to be correct data-rate. If we want to increase datarate, we can add multiple Ethernet frame in single cycle.

We also tested normal web access with following command:

```
wget --no-cache http://en.wikipedia.org/wiki/Main_Page
```

We found average delay to be 5 second. We performed this operation several times and got similar results. If we access this page in normal IITB's LAN, it takes approximately 3 second. Additional delay is due to queueing at BS and delay in TCP ACK.

We tried VoIP calls with two different setup. First, we setup call between two clients using VoIP soft-phone with the help of Asterisk PBX. This call works perfectly fine and voice clarity and delay fulfills the requirement prescribed by WiFiRe [2]. Secondly, we setup a call from client to normal PSTN phone using VoIP-PSTN gateway. This call also gave similar results. For both calls, we used G.711 codec which generates packet every 10 mSec.

As WiFiRe MAC deals with many packets and very high periodicity (10 mSec), we did basic profiling for BS program. We used *gprof* and *valgrind* profiler for this purpose. These profilers are useful to analyse our program, find bugs and unobvious behavior by any function. We used *gprof* on MAC modules and found following results (Table 6.2). These results are taken when BS was running for more than 700 second and more than 6000 ICMP packets were transmitted by BS.

Function	Calls	Comments
getSTindex, dequeueLTQ	159171	Per ST, per frame
getSTID, my_malloc my_free, SetGetFrameTime	86632	Each cycle for each packet
GetSTList, beacongenerator Process_packet_bs_time, ParseUlHeaderBS	73770	Used for each DL frame and beacon
getClientST addClient, enqueueLTDQ	12859 6361	Called per packet basis
getBScount, getClientCount displayTime, showSystemStats	45	Called for Summary on Console

Table 6.2: Profiling with *gprof*

We used *valgrind* to analyse memory consumption by MAC modules. We found some bugs in our code related to memory access and threading, they were rectified and following is the output from the latest code.

```
==17330== LEAK SUMMARY:
==17330== definitely lost:  272 bytes in 26 blocks.
==17330== possibly lost:  272 bytes in 2 blocks.
==17330== still reachable:  53,902 bytes in 11 blocks.
```

This output was taken after 90 minutes of program's run. It shows memory leak of 272 bytes in code. This is very good improvement as earlier code had memory leak of more than 1MB. As of now, 272 byte leak is mainly from the system library (*libc* and similar program).

Chapter 7

Future work and conclusion

WiFiRe design is very much similar to other WiMAX deployment which we came across. Modularity given to MAC was great benefit as it helped development of WiFiRe MAC in absence of 802.11 PHY devices. As basic WiFiRe LAN emulation prototype is working, it proves correctness of protocol, design of MAC modules and integration. Next task ahead is to add PHY integration support for current MAC code and better scheduling algorithm.

Integration with chip-set will provide interesting wireless specific problems which we may not have expected.

7.1 Short-term goals

WiFiRe MAC modules are written with a flexibility of adding new features easily. Following are the features that can be written with very little additional work. These functions were not implemented because of lack of time and they are not so crucial for basic working prototype.

- Current WiFiRe frame has 1514 byte Ethernet frame for DL and UL. This frame is transmitted every 10 mSec. Actual data-rate is around 120KB/sec with given frame size. We can not increase frame size because of restriction from Ethernet MTU limits. If we want to increase date-rate, we can have multiple Ethernet frames in each cycle. This module can be written to support fragmented frames for DL and UL.

- Current packet queues on BS are designed to support ST level support. For each ST, there is a queue which has packets for all its clients. We can give each ST priority based on preference. If we want to support QoS for different traffic (VoIP, web etc.), we can have queues for different kind of traffic. This can be done easily without any change in data structure implemented for queues.

- WiFiRe LAN testbed has wireless link for WiFiRe frames but clients are always connected using L2 Ethernet switch. If required, we can connect all such clients using 802.11 (ad-hoc or managed mode). For wireless clients, all subnets should be operating in different 802.11 channels (e.g. 1,6 and 11).
- Actual WiFiRe draft has recommended 6-sector system with parallel Tx in opposite direction. We can have 3-sector system on BS with 3 different network interface (NIC). We need additional table which does the mapping of ST-ID to interface. We will have 3 DL frames per cycle of 10 mSec.

7.2 Long-term goals

Following are the long-term goals and can be implemented as separate projects.

- Explore the possibility to implement MAC as part of kernel module
- Driver code of 802.11 and integration with WiFiRe
- Adding bulk ACK support for WiFiRe frame
- Performance evaluation of WiFiRe testbed
- Time synchronization among 3 BSs
- Long range deployment and study of propagation delay
- Exploiting Ethernet MTU size of 1500 (with specialized hardware)

7.3 Integration with 802.11b PHY

In a typical implementation, such as WiMAX (802.16d), all the modules (MAC, RF, antenna, modem, and memory) are in same *box* [18]. Every component is closely coupled with each other and they exchange configuration parameters with each other. For example, memory has driver and API through which, it can store/retrieve data to/from buffer. This buffer is shared by MAC and PHY.

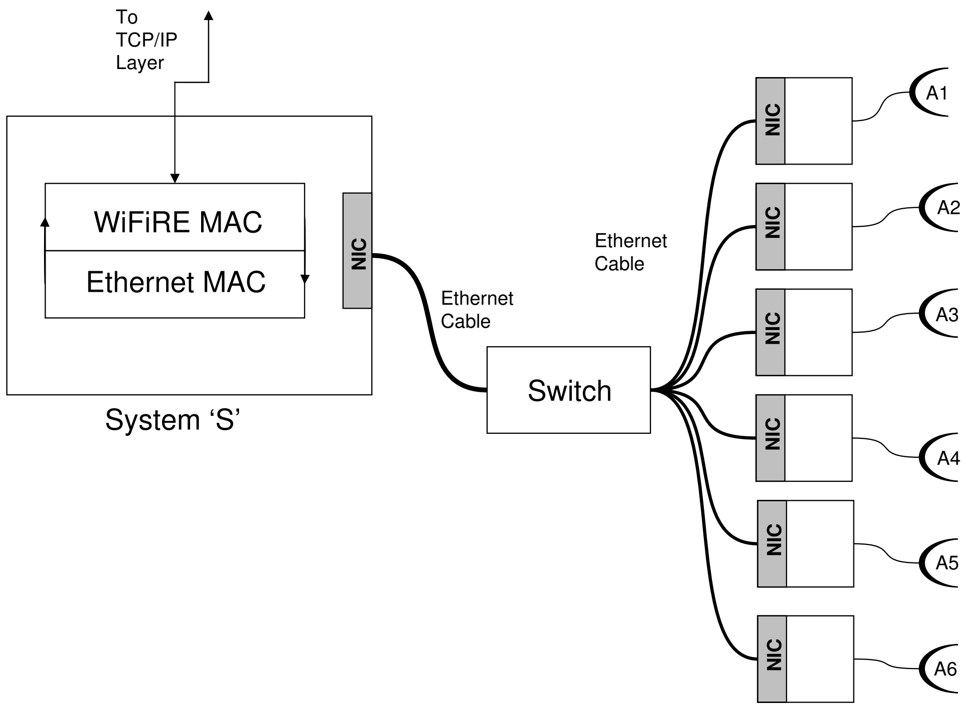


Figure 7.1: BS hardware: proposed WiFiRe real system component

In current WiFiRe implementation plan, WiFiRe MAC and BS-PHY are separate from each other (Figure 7.1). PHY boards are being developed independently and require MAC frame to be delivered on Ethernet cable. Eventually, MAC and PHY will be integrated as single entity. Current hardware keeps six different NICs for six BSs. These six NICs are connected by single switch.

This leads to Ethernet cable as single communication medium. All the non-WiFiRe control messages such as 2 byte control data (explained later in this section) have to be transferred using same link. Extra control modules have to be written to handle such messages which adds to the complexity in implementation. WiFiRe frames are to be sent to this switch with destination address as a given BS-PHY MAC address. Frame structure and slot size is fixed. There is a centralized module which instructs all six BS-PHYs to synchronize with each other and send data as and when required.

As mentioned in figure 7.1, all 6 BS have their NIC card respectively. This NIC is usually normal Ethernet NIC with 6 Byte MAC address. When ever system(S) wants to send any data

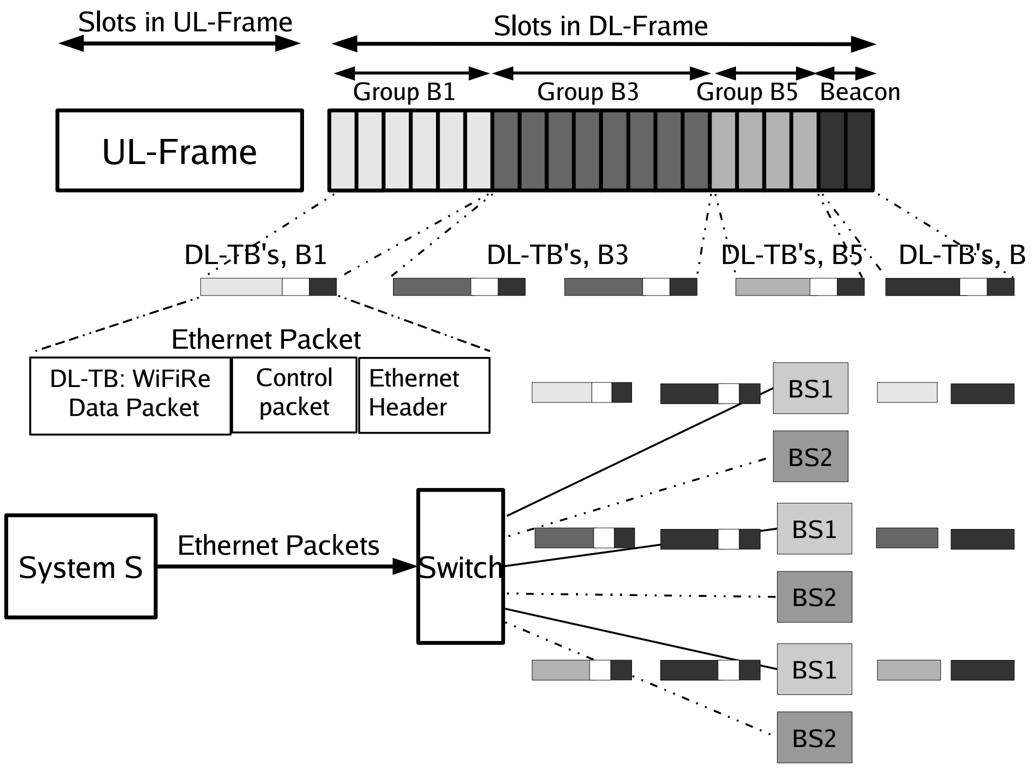


Figure 7.2: Consecutive allocation of DL slots for each BS

for transmission to BS, it constructs packet destined to that particular BS (using MAC address) and data is appended as PDU. Once this data packet reaches to switch, switch will lookup MAC address and forwards it to appropriate interface. Synchronization among all BSs is also important and explained below.

We also need to optimize the PHY overhead, which is of 4 slots, for every new transmission at BS-PHY. Among the available methods, concatenation has low PHY overhead. Using concatenation we allocate all slots which belong to same BS consecutively, thereby reducing PHY overhead. We use a 2 byte control packet per DL-TB (down link transport block), having information <starting slot, number of consecutive slots> and then transmitted to BS-PHY. At BS-PHY, it reads the information present in control packet and acts accordingly.

Figure 7.2 shows how this approach works. Total DL frame is divided into groups (B1, B3, B5) and each group belongs to the respective BS-PHY. Beacons are transmitted to every BS-PHY. Down link Transport Blocks (DL-TB is a group of slots which belongs to the same BS. Each BS will have zero or more DL-TBs and this number depends on Ethernet packet size.) are created before inserting the WiFiRe frame into Ethernet packets.

For example, when BS-PHY receives an Ethernet packet having a control packet <05,15>, it transmits from slot 5 to 19. This method has advantage of fixed size memory requirement (PHY buffer need to store single frame at any given time) at BS-PHY, minimal control and PHY overhead. However it increases the complexity of scheduler. Scheduler has to take care of constructing DL-TBs.

7.4 Future directions

While working for WiFiRe, we came across some problems which are not described in 802.16 recommendation. There is active work going in this direction. Some of these are open research problems and can be explored if time permits.

Generic Packet Convergence Sublayer (GPCS)

It is difficult for the industry to accept a set of 802.16 convergence sub-layers that all devices must implement to be called 'WiMAX compliant'. For example, if Ethernet CS, why should a phone have to implement Ethernet frame formats? Why device need to participate in IP address assignment, IP mobility, and tunneling, etc if its 802.16d ST? And if a vendor implements a proprietary upper layer protocol, how can its 802.16 layer are tested to be compliant? GPCS [19] suggests that a generic packet convergence sub layer can help 802.16 CS by simplifying a *compatibility* that is independent of the upper layer protocol.

The 802.16 convergence sub-layers do not define the capability for multiple upper layer protocols to transport the SDUs on a single 802.16 connection. A connection ID (CID) is a valuable resource in both base stations and subscriber stations. An 802.16 system should not be forced to open a different CID for each upper layer protocol if packets for upper layer protocols have the same QoS requirements (802.16 scheduling service types). The generic convergence sub-layer provides a simple way to transport multiple protocols over a single 802.16 connection.

In the current 802.16 specification, address-based classification rules define how data packets of different users are mapped to different CIDs, so that the differentiated QoS and/or security provisions can be provided. The address-based classification rules require the current 802.16 convergence sub-layer to maintain the mapping information between upper-protocol-defined addresses (e.g., IP or Ethernet) and CIDs. This ultimately forces 802.16 CS to implement some upper layer functions. For example, the IPv4 CS at BS maintains a mapping state for IP addresses to CIDs. Whenever there is a change in IP addresses, the mapping state needs to be

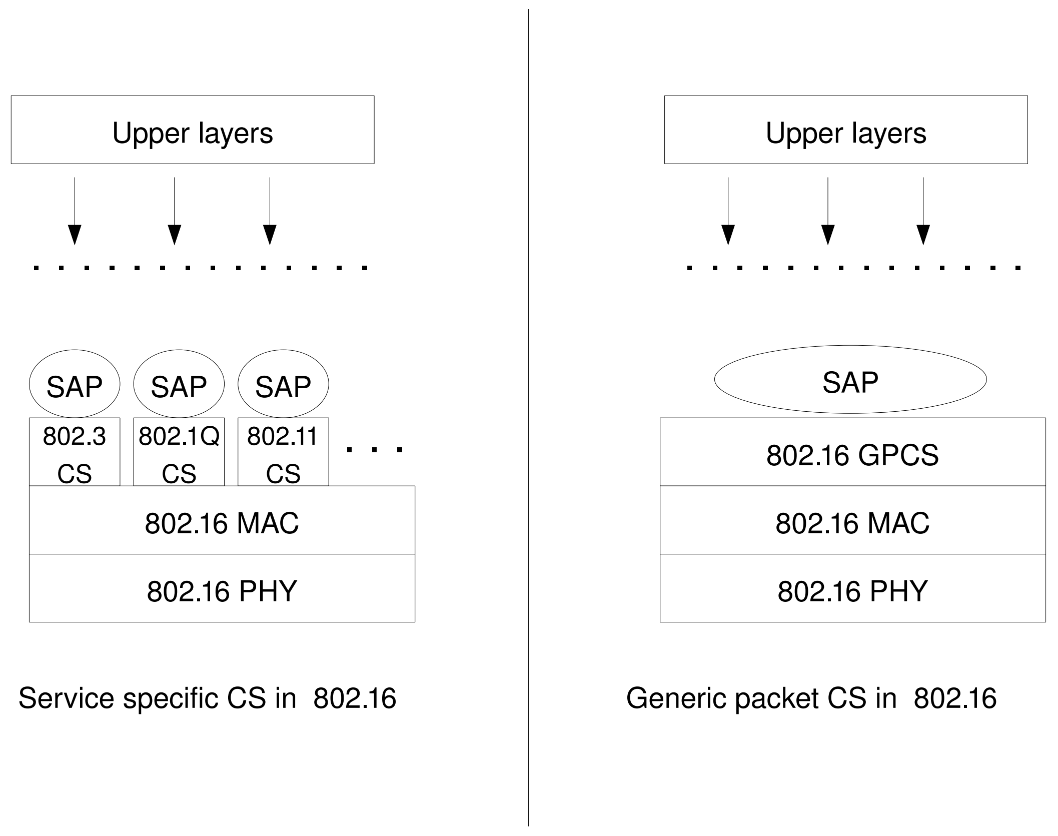


Figure 7.3: different CS and GPCS

updated. In non-802.16 systems and protocol stacks, upper layer address assignment and mapping to link layer entities is typically part of a routing function at the network layer and not at MAC.

GPCS is equally important in case of WiFiRe as well. WiFiRe considers 802.3 based network for external connectivity. [7] describes intra-village connectivity and how it can be incorporated with WiFiRe. WiFiRe SS can be connected with 802.11 based access point. In that case, we may need to have CS which 'knows' 802.11. For voice connectivity, if we have RJ11 based network connecting to SS with TDM then we need this CS as well. 802.16e network has much wider range of clients. Beside normal SS, we can have laptop, mobile handset and other small devices which may use WiMAX. This all devices are having different usage and requirements (like mobility, security, QoS etc.). These devices may not want to implement CSs which are not required.

Performance Measurement in WiFiRe/WiMAX

There is large possibility to have good performance evaluation based simulations in WiMAX. Here, we have listed some of them with their possible outputs.

WiMAX provide four different kinds of services: UGS, rtPS, nrtPS and BE. As we have described above in literature survey section, most of them recommend having different queue for different priority traffic. At the time of Tx, those queues will be appended one after another and then transmitted as a single PDU. This means that when ever we get a packet from Ethernet, it will be processed by CS and identified according to its service requirement. Then CPS will send them in different queue instead of FIFO. We argue that this process requires much higher processing time and may not provide significant QoS improvement. As we have seen so far, WiMAX frame is usually 10 ms. Considering 2:1 ratio for DL to UL, we get approximately 6 ms for DL. Now, putting UGS queue ahead of rtPS will give it advantage of 1-2 ms. Also note that, UGS is periodic request and does not get affected by re-arrangement of packets in priority sequence. To prove these results, one can setup small testbed and have fixed traffic of VoIP, FTP and HTTP.

WiMAX and WiFiRe currently support GPC based connections. WiMAX also prescribes GPSS mode for simpler design. There can be hybrid model which gives connection based on service flow. It means that each SS will have 3-4 different connection and there would be no more differences within that flow. It would be interesting to see that if we provide GPC based connection, what would be the table lookup time in CID database, classifier, header generation etc. It will be interesting to see that what the advantages of GPC mode against GPSF mode are.

There are few more interesting questions on traditional WiMAX settings like software-hardware coupling, complex scheduler, pending queues, VoIP header compression etc. If explored with realistic setting and environments, we may get surprising results.

Open Base Station architecture

An open base-station architecture[20] allows manufacturers to focus their research and development efforts on their core competencies. They also can buy selected base-station modules from each other and other specialist module vendors. These aspects will result in faster development of innovative, cost-effective base stations. They also will result in earlier and lower-cost introductions of new technologies and services to network operators. It supports different access technologies such as GSM/EDGE, CDMA2000, WCDMA and IEEE802.16/WiMAX. It has common operational and management interface. OBSAI is common forum joined by all major Telecom companies. OBSAI allows multiple concurrent operation of air interface as well. It has different plane for control and user data. This problem is interesting because WiFiRe's one of the major cost in deployment is tower itself. India specific case where we can use WiFi using GSM/CDMA tower is described in [1]. If we can use existing tower (at least in urban area) of GSM or CDMA, spreading WiFiRe/WiMAX would be much easier. Here, we are not just sharing physical space but most of the common blocks are shared if we have more than one air interfaces. The OBSAI organization consists of more than 130 component, module and base station vendors. OBSAI specifications allow module vendors to manufacture modules that are capable of operating in any OBSAI-compatible BTS, thereby reducing substantially the development effort and costs involved in the introduction of a new range of BTS products. One more similarity with WiFiRe is that, OBSAI recommends RF components are to be kept remote from other modules. This means that RF module will take care for all Tx and Rx for GSM, WiMAX etc. and remaining software system will take care for operational and management functions.

7.5 Summary

All the primary modules of WiFiRe are implemented in user space with socket libraries. We have achieved frame timer of 10 mSec for UL and DL frame. WiFiRe Ethernet testbed is able to show end-to-end connectivity. Clients and external servers are transparent to WiFiRe protocol and operate same as LAN connection. WiFiRe MAC is robust, modularised, easy to extend and maintain. We also achieved desired results in terms of delay and bandwidth requirement. Basic intended applications like web and VoIP are working as expected and WiFiRe MAC prototype is completed with all primary goal gained. Next task ahead is to integrate this MAC with 802.11b PHY and full fledged deployment.

Bibliography

[1] Bhaskar Ramamurthi, Anand Kannan, Ashok Jhunjhunwala, “Interim 3.5G broadband wireless system for india: Framework, requirement, performance needs,” *CEWiT*, 2005.

[2] Sridhar Iyer, Krishna Paul, Anurag Kumar, Bhaskar Ramamurthy, “WiFiRe: Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” *CEWiT, India*, 2006.

[3] P Bhagwat, B Raman, D Sanghi, “Turning 802.11 inside-outs,” *ACM SIGCOMM Computer Communication Review*, 2004.

[4] Lakshminarayan Subramanian, Sonesh Surana and Eric Brewer, “Rethinking wireless for the developing world,” *Hot Topics in Networks*, 2006.

[5] Department of telecommunications, Govt. of India, “Report of working group on the telecom sector for the eleventh five year plan 2007-2012,” Oct-2006.

[6] IEEE Std 802.16-2004, “IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems,” 2004.

[7] J Chandarana, Sravana K, S Perur, R Rangarajan, S Sahasrabuddhe and S Iyer , “VoIP-based Intra-village Teleconnectivity: An Architecture and Case Study,” *WiSARD, Comsware*, 2007.

[8] R. Madalapu, “Implmentation of WiFiRe MAC protocol,” *MTech Technical report, IIT Bombay*, 2008.

[9] S. Chandra, “An Efficient Call Admission Control for IEEE 802.16 Networks,” *MTech Technical report, IIT Bombay*, 2006.

[10] S. Maheshwari, “An Efficient QoS Scheduling Architecture for IEEE 802.16 Wireless MANs,” *MTech Technical report, IIT Bombay*, 2005.

[11] A. Maheshwari, “Implementation and Evaluation of a MAC Scheduling Architecture for IEEE 802.16 WirelessMANs,” *MTech Technical report, IIT Kanpur*, 2006.

[12] A. Pentland, R. Fletcher, and A. Hasson, “Daknet: Rethinking connectivity in developing nations.” *IEEE Computer*, vol. 37, no. 1, pp. 78–83, 2004.

[13] Ashish Sharma, Mohit Tiwari, Haitao Zheng, “Madmac: Building a reconfigurable radio testbed using commodity 802.11 hardware,” *Networking Technologies for Software Defined Radio Networks*, 2006.

[14] Michael Neufeld, Jeff Fifield, Christian Doerr, Anmol Sheth and Dirk Grunwald, “Softmac: Flexible wireless research platform,” *Fourth Workshop on Hot Topics in Networks*, 2005.

[15] Doerr, C. Neufeld etc., “Multimac - an adaptive mac framework for dynamic radio networking,” *New Frontiers in Dynamic Spectrum Access Networks*, 2005.

[16] Sameer Kurkure, “Design and implementation of wifire mac layer protocol,” *MTech Technical report, IIT Bombay*, 2007.

[17] N. P. Reddy, “The SRAWAN MAC protocol to support real-time services in long distance 802.11 networks,” *MTech Technical report, IIT Kanpur*, 2006.

[18] Intel Corporation, “Intel PRO/Wireless 5116 Broadband Interface,” *http://www.intel.com/network/connectivity/products/wireless/307327.pdf*, 2007.

[19] IEEE 802.16 Broadband Wireless Access Working Group, “GPCS for Supporting Multiple Protocols over 802.16 Air Interface,” 2006.

[20] OBSAI technical working group, “Open base-station architecture initiatives: BTS system refrence, Version 2.0 ,” 2006.

[21] Squid proxy server website, “http://www.squid-cache.org/.”

[22] Open-source PBX for VoIP, “http://www.asterisk.org/.”

[23] DHCP server package by ISC, “http://www.isc.org/sw/dhcp/.”

[24] DNS package from maraDNS, “http://www.maradns.org/.”

Appendix A

WiFiRe proxy server

Configuration

WiFiRe proxy has two interface: one connects to WiFiRe network other connects to external world. First interface connects to BS using cross-cable. It can be connected using L2 Ethernet switch as well. Other interface connects to external Router/Gateway using 802.3 based Ethernet LAN. Throughout this document, we will assume First interface as *eth0* with IP address of 172.168.1.1. Second interface is *eth1* with IP address of 10.129.41.2.

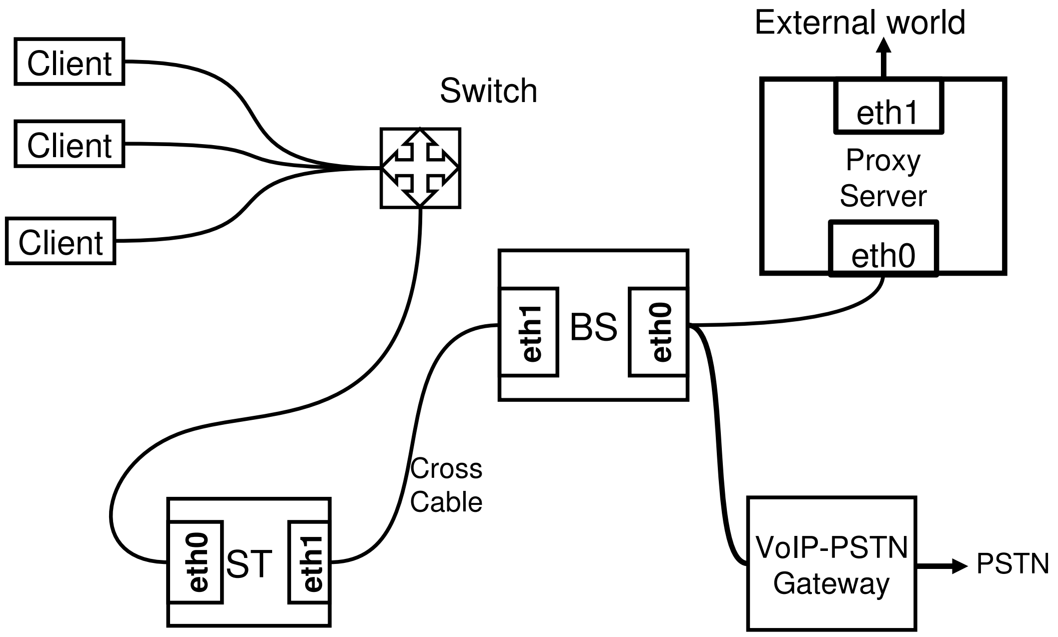


Figure A.1: Proxy Set-up

Squid: HTTP proxy

Squid is proxy server and web cache daemon. It has a variety of applications including speeding up a web server by caching repeated requests, to computer network lookups for a group of people sharing network resources, to filtering traffic [21]. It supports HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages.

Configuration details of /etc/squid/squid.conf:

```
http_port: 8080
cache_peer proxy.it.iitb.ac.in parent 80 0 no-query no-digest
```

Configuration details on proxy.it.iitb.ac.in:

```
acl wifire_MTPexp src 10.129.41.2
```

Command on shell for Squid:

```
#/etc/init.d/squid start
#/etc/init.d/squid stop
#/etc/init.d/squid restart
```

WiFiRe proxy listens on TCP port number 8080 and its parent is proxy.it.iitb.ac.in.

Here proxy.it.iitb.ac.in is main proxy which handles web access of KReSIT building. WiFiRe proxy forwards all its request to this proxy. At proxy.it, we should mention that WiFiRe proxy is authorized proxy and it should entertain the requests coming from it. We setup acl for the same.

Asterisk: VoIP proxy

Asterisk[22] is an open source implementation of a PBX mainly used for VoIP. It supports SIP, H232 and many other protocols. It supports PSTN integration with IP telephony. It has AGI scripts which allows asterisk to integrate with other programs and daemon like LDAP, perl scripts, SMTP servers etc.

Configuration of /etc/asterisk/sip.conf:

```
   [janak]
type=friend
```

```
callerid=janak <100>
host=dynamic
username=janak
context=sip
```

Configuration of /etc/asterisk/extensions.conf: `[sip]`

```
exten => 100,1,dial(SIP/janak)
exten => janak,1,dial(SIP/janak)
```

Here, janak is the user name for SIP client which has extension number of 100.

Command on linux shell to start asterisk:

```
#asterisk -vvvvc
```

Once started, command on Asterisk shell:

```
sip show users
stop gracefully
```

DHCP server

We used DHCP server from version 3 of the Internet Software Consortium DHCP package[23]. Dynamic Host Configuration Protocol (DHCP) is a protocol like BOOTP. It assigns IP addresses to clients based on lease times. It is probably essential in any multi-platform environment. Most of the WiFiRe clients does not know their network configuration like Default gateway or DNS server. DHCP server automatically assigns all this details to client and overhead of configuring clients is reduced by this.

Configuration of /etc/dhcp3/dhcpd.conf:

```
   option domain-name-servers 172.168.1.1;
default-lease-time 86400;
max-lease-time 604800;
authoritative;

    subnet 172.168.0.0 netmask 255.255.0.0
```

```
range 172.168.0.200 172.168.0.229;
option subnet-mask 255.255.0.0;
option broadcast-address 172.168.0.255;
option routers 172.168.1.1;
```

Here, 172.168.1.1 is Server's IP and range is client IP's range.

Command on shell for DHCP server:

```
# /etc/init.d/dhcp3-server start
# /etc/init.d/dhcp3-server stop
# /etc/init.d/dhcp3-server restart
```

maraDNS: DNS server

MaraDNS[24] is a package that implements the Domain Name Service (DNS). MaraDNS can function either as an authoritative DNS server or ”recursive” DNS cache that uses the DNS root. We use it as recursive DNS for our purpose.

Configuration of /etc/maradns/mararc:

```
bind_address = "172.168.1.1"
chroot_dir = "/etc/maradns"
upstream_servers["."] = "10.129.1.1"
```

Command on shell for DNS server: `#/etc/init.d/maradns start`

```
#/etc/init.d/maradns stop
#/etc/init.d/maradns restart
```

SPA3000: VoIP-PSTN gateway

SPA-3000 is PSTN-VoIP gateway which converts PSTN signals to VoIP signals and vice-versa. It has two interface: one connects to PSTN line using RJ-11 cable, other connects to Ethernet LAN using RJ-45 cable. It has web-based interface to configure its Settings. We keep it parallel to proxy and BS can directly communicate to it.

VoIP clients

SJphone is a VOIP softphone that allows you to speak with any other softphone running on a PC, any stand-alone IP-phone, or PSTN phone using VoIP-PSTN gateway. Softphone needs IP address of Asterisk PBX server, username and password using which it will communicate with other phones. Using IVRS system on server, one can establish a dial plan for all the clients. When ever someone from outside calls to WiFiRe system, we can redirect them using this IVRS system to desired client.

Apache: HTTP server

Apache is popular web-server used to serve static and dynamic web pages using HTTP protocol. Apache is developed and maintained by an open community and its developers. Apache is available for a wide variety of operating systems, including Linux. We used it to serve content to clients of WiFiRe which are connected to their respective ST. It usually operates on port 80 and serves requests from clients using TCP protocol. Following is the configuration detail of httpd.conf file.

```
Port 80
DocumentRoot /var/www
```

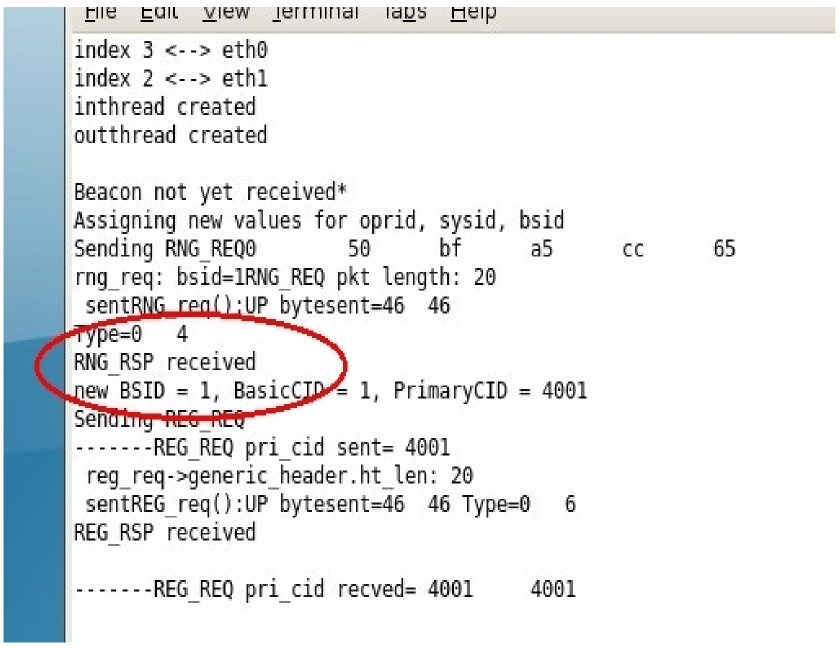
Following are the command to operate apache:

```
/etc/init.d/apache start
/etc/init.d/apache stop
/etc/init.d/apache restart
```

Appendix B

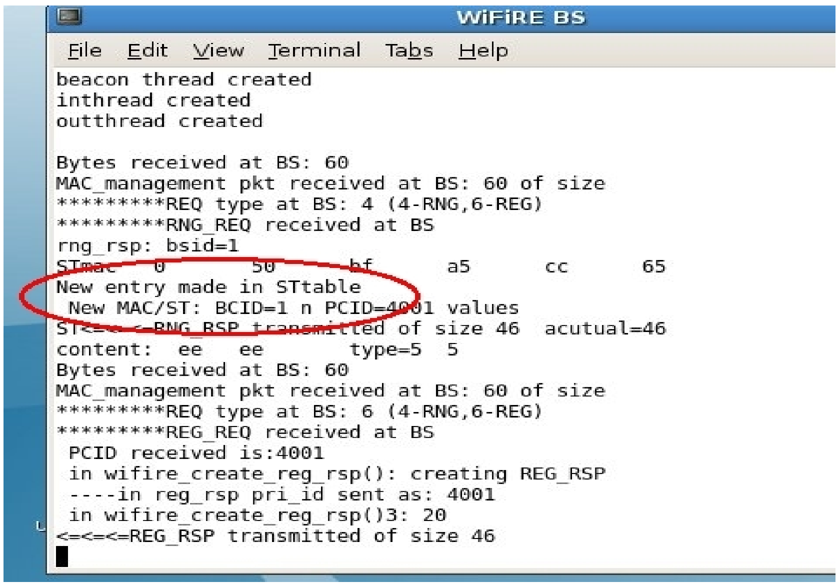
Additional Figures

Additional figures



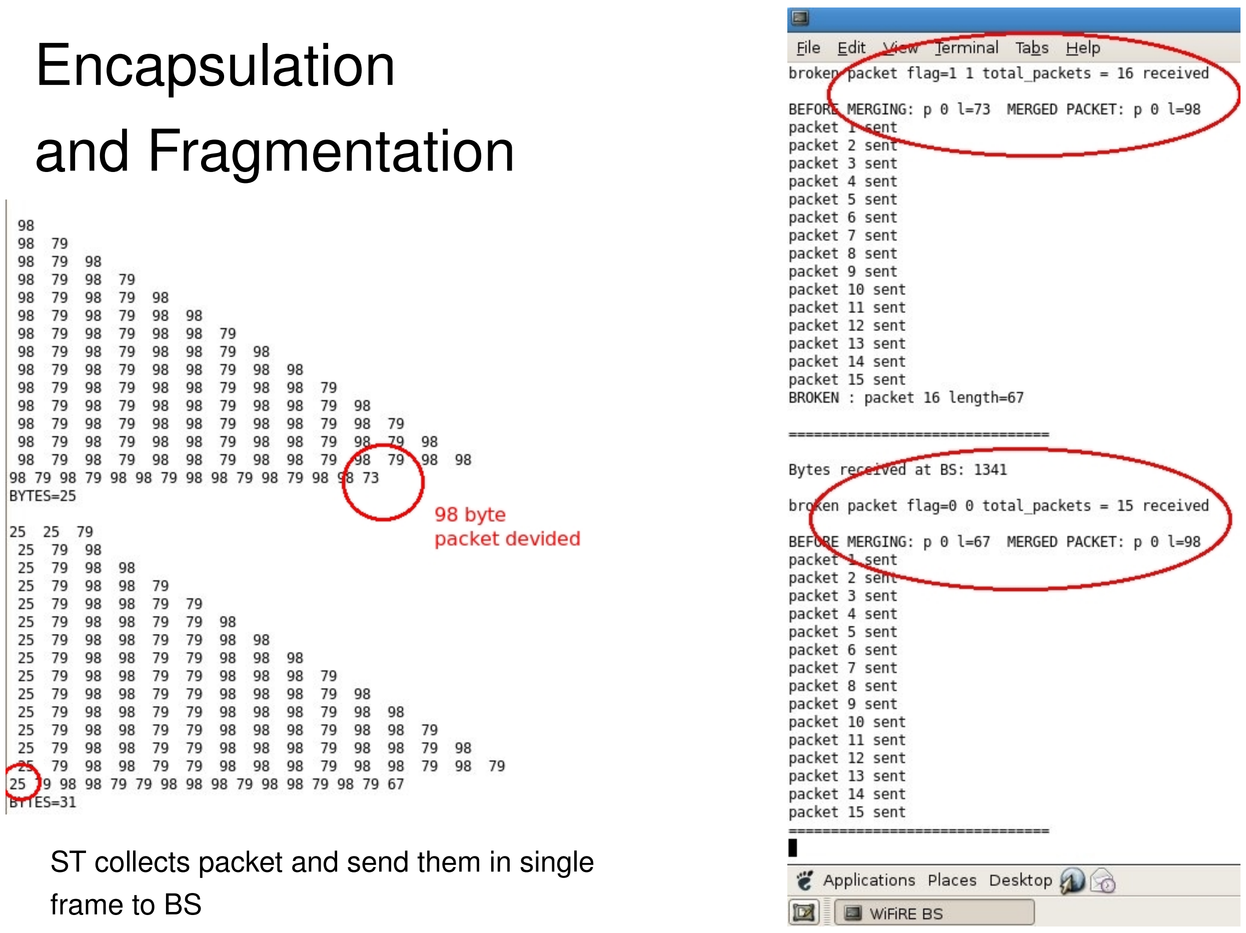
```
index 3 <--> eth0
index 2 <--> eth1
inthread created
outthread created

Beacon not yet received*
Assigning new values for oprid, sysid, bsid
Sending RNG_REQ0        50      bf      a5      cc      65
rng_req: bsid=1RNG_REQ pkt length: 20
 sentRNG_req():UP bytesent=46  46
Type=0   4
RNG_RSP received
new BSID = 1, BasicCID = 1, PrimaryCID = 4001
Sending REG_REQ
-------REG_REQ pri_cid sent= 4001
 reg_req->generic_header.ht_len: 20
 sentREG_req():UP bytesent=46  46 Type=0   6
REG_RSP received

-------REG_REQ pri_cid recved= 4001     4001
```

Figure B.1: Ranging steps at ST



```
beacon thread created
inthread created
outthread created

Bytes received at BS: 60
MAC_management pkt received at BS: 60 of size
*********REQ type at BS: 4 (4-RNG,6-REG)
*********RNG_REQ received at BS
rng_rsp: bsid=1
STmac    0        50       bf       a5       cc       65
New entry made in STtable
 New MAC/ST: BCID=1 n PCID=4001 values
ST<=<=<=RNG_RSP transmitted of size 46  acutual=46
content:  ee   ee       type=5  5
Bytes received at BS: 60
MAC_management pkt received at BS: 60 of size
*********REQ type at BS: 6 (4-RNG,6-REG)
*********REG_REQ received at BS
 PCID received is:4001
 in wifire_create_reg_rsp(): creating REG_RSP
 ----in reg_rsp pri_id sent as: 4001
 in wifire_create_reg_rsp()3: 20
<=<=<=REG_RSP transmitted of size 46
```

Figure B.2: Ranging steps at BS



Encapsulation

and Fragmentation

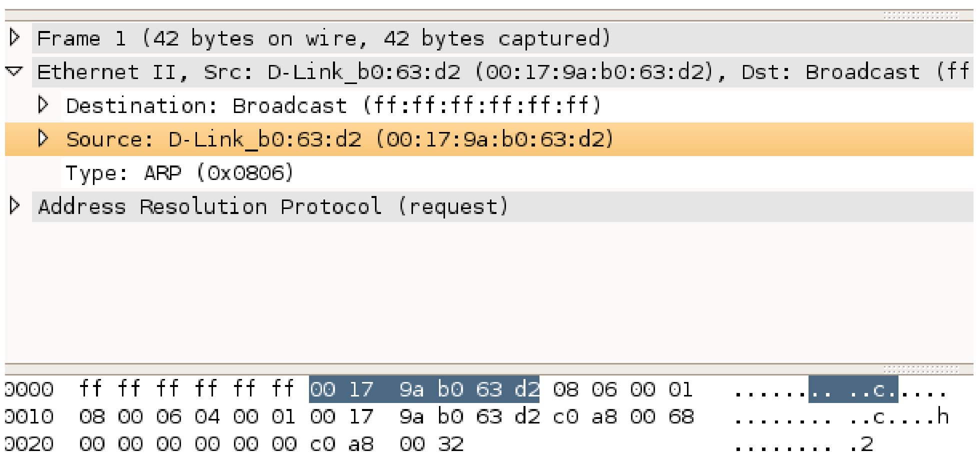
```
 98
 98   79
 98   79   98
 98   79   98   79
 98   79   98   79   98
 98   79   98   79   98   98
 98   79   98   79   98   98   79
 98   79   98   79   98   98   79   98
 98   79   98   79   98   98   79   98   98
 98   79   98   79   98   98   79   98   98   79
 98   79   98   79   98   98   79   98   98   79   98
 98   79   98   79   98   98   79   98   98   79   98   79
 98   79   98   79   98   98   79   98   98   79   98   79   98
 98   79   98   79   98   98   79   98   98   79   98   79   98   98
98 79 98 79 98 98 79 98 98 79 98 79 98 98 73
BYTES=25

25   25   79
 25   79   98
 25   79   98   98
 25   79   98   98   79
 25   79   98   98   79   79
 25   79   98   98   79   79   98
 25   79   98   98   79   79   98   98
 25   79   98   98   79   79   98   98   98
 25   79   98   98   79   79   98   98   98   79
 25   79   98   98   79   79   98   98   98   79   98
 25   79   98   98   79   79   98   98   98   79   98   98
 25   79   98   98   79   79   98   98   98   79   98   98   79
 25   79   98   98   79   79   98   98   98   79   98   98   79   98
 25   79   98   98   79   79   98   98   98   79   98   98   79   98   79
25 79 98 98 79 79 98 98 98 79 98 98 79 98 79 67
BYTES=31
```

98 byte
packet devided

ST collects packet and send them in single frame to BS

```
broken packet flag=1 1 total_packets = 16 received

BEFORE MERGING: p 0 l=73  MERGED PACKET: p 0 l=98
packet 1 sent
packet 2 sent
packet 3 sent
packet 4 sent
packet 5 sent
packet 6 sent
packet 7 sent
packet 8 sent
packet 9 sent
packet 10 sent
packet 11 sent
packet 12 sent
packet 13 sent
packet 14 sent
packet 15 sent
BROKEN : packet 16 length=67

===============================

Bytes received at BS: 1341

broken packet flag=0 0 total_packets = 15 received

BEFORE MERGING: p 0 l=67  MERGED PACKET: p 0 l=98
packet 1 sent
packet 2 sent
packet 3 sent
packet 4 sent
packet 5 sent
packet 6 sent
packet 7 sent
packet 8 sent
packet 9 sent
packet 10 sent
packet 11 sent
packet 12 sent
packet 13 sent
packet 14 sent
packet 15 sent
===============================
```

Figure B.3: Encapsulation and fragmentation



```
▷ Frame 1 (42 bytes on wire, 42 bytes captured)
▽ Ethernet II, Src: D-Link_b0:63:d2 (00:17:9a:b0:63:d2), Dst: Broadcast (ff
  ▷ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Source: D-Link_b0:63:d2 (00:17:9a:b0:63:d2)
    Type: ARP (0x0806)
▷ Address Resolution Protocol (request)

0000  ff ff ff ff ff ff 00 17  9a b0 63 d2 08 06 00 01   ........ ..c.....
0010  08 00 06 04 00 01 00 17  9a b0 63 d2 c0 a8 00 68   ........ ..c....h
0020  00 00 00 00 00 00 c0 a8  00 32                     ........ .2
```

Figure B.4: 802.11 MAC header in AP mode

Appendix C

Code snippets and project timeline

Code snippets

To create thread,

```
pthread_t thread[3];
ret_code = pthread_create(&thread[0], NULL, &beaconthread_handler,
NULL);
```

Here, thread is identified by thread[0] and used for any communication further. Beaconhandler is function where this thread will be activated.

Following is the way to create PF_SOCK packet:

```
sockfd=socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL))
strncpy(ifr.ifr_name, "eth0", sizeof(ifr.ifr_name));
```

Using above code, we get access to layer-2 of *eth0* NIC interface. We can send and receive data to this socket by following method:

```
n = sendto(sockfd,out_buffer,len,0..
ret = recvfrom(wifire_sockfd,in_buffer,2048,0,NULL,NULL);
```

It also suggests that NIC will accept whatever data being given to it. NIC will not verify whether its 802.3 frame or not.

To retrive packet from given frame,

```
memcpy(buffer,in_buffer+offset,packet_len);
```

To append given packet to outgoing buffer,

```
memcpy(out_buffer+offset,pack_buffer,packet_len);
```

If packet is fragmented in two parts and we want to join them back as single packet,

```
printf("BEFORE MERGING: p %d l=%d ",i,old_broken_pkt_len);
packet_length[0]+=old_broken_pkt_len;
merge_pkt_flag=0;
printf("MERGED PACKET: p %d l=%d ",i,packet_length[i]);
```

Project timeline

Month	Work
Aug-07	WiFiRe testbed started
Aug	Code walk-through
Sept	Basic encapsulation
Sept	DL-MAP and UL-MAP
Sept	NCC paper submission
Oct	Fragmentation support
Oct	GPSS mode
Nov	IIT Madras Visit
Nov	Registration and ranging
Dec	Wireless testbed
Jan-08	Second stage
Jan	Timer and beacon generation
Feb	Filters for BS and ST
March	10 mSec timer
March	Local traffic supported
April	Memory management
April	Packet queues for ST
May	VoIP server and clients
May	WiFiRe console
June	Log and config files
June	IIT Madras Visit

Table C.1: Project timeline with events at various stage

Appendix D

Abbreviations

ARP : Address Resolution Protocol

BE : Best Efforts

BS : Base Station

BSID : Base Station Identification

BWA : Broadband Wireless Access

CID : Connection Identifier

CRC : Cyclic Redundancy Code

CSMA : Carrier Sense Multiple Access

DCF : Distributed Co-ordination Function

DL : Down Link

DL-MAP : Down Link Slot Allocation Map

DQPSK : Differential Quadratic Phase Shift Keying

DSA : Dynamic Service Addtion

DL-TB : Down Link Transport Block

FTP : File Transfer Protocol

GPC : Grant Per Connection

HTTP : Hyper Text Transfer Protocol

ID : Identifier

IP : Internet Protocol

LAN : Local Area Network

LLC : Logical Link Control

LoS : Line of Sight

MAC : Medium Access Control layer

MTU : Maximum Transmission Unit

NIC : Network Interface Card

nrtPS : Nonreal Time Polling Service

PCF : Point Co-ordination Function

PDU : Protocol Data Unit

PHY : Physical Layer

PoP : Point of Presence

PS : Physical Slot

QoS : Quality of Service

rtPS : Real Time Polling Service

RF : Radio Frequency

Rx : Reception

S : System

SAP : Service Access Point

SS : Subscriber Station

ST : Subscriber Terminal

TCP : Transmission Control Protocol

TDD : Time Division Duplex

TDM : Time Division Multiplex

TDMA : Time Division Multiple Access

Tx : Transmission

UDP : User Datagram Protocol

UE : User Equipment

UGS : Unsolicited Grant Service

UL : Up Link

UL-MAP : Up Link Slot Allocation Map

UL-TB : Up Link Transport Block

VLAN : Virtual LAN

VoIP : Voice over IP

WiFi : Wireless Fidelity

WiFiRe: Wireless Fidelity for Rural Extension

WiMAX : Worldwide Interoperability for Microwave Access

Appendix E

Publications

- J. Chandarana, R. Madalapu, S. Kurkure, S. Hullur, A. Sahoo and S. Iyer, Emulation of WiFiRe protocol on LAN, *National Communication Conference*, Mumbai, 2008
- J. Chandarana, K. Sravana , S. Perur, R. Rangarajan, S. Sahasrabuddhe and S. Iyer, VoIP-based Intra-village Teleconnectivity: An Architecture and Case Study, *WISARD, COMSWARE*, Bangalore, 2007
- A. Gumaste, J. Chandarana, P. Bafna, N. Ghani and V. Sharma, On Control Plane for Service Provisioning in Light-trail WDM Optical Networks, *42nd IEEE International Conference on Communication (ICC)*, Glasgow, UK, 2007