Design and Implementation of *WiFiRE* MAC Layer Protocol

Dissertation

submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

by

Sameer Kurkure
(Roll no. 05329025)

under the guidance of

Prof. Anirudha Sahoo

and

Prof. Sridhar Iyer

Department of Computer Science & Engineering (KReSIT)

Indian Institute of Technology Bombay

2007

Dissertation Approval Sheet

This is to certify that the dissertation entitled

Design and Implementation of *WiFiRE* MAC Layer Protocol

by

Sameer Kurkure

(Roll no. 05329025)

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

____

Prof. Anirudha Sahoo

(Supervisor)

____

Prof. Sridhar Iyer

(Co-Supervisor)

____

Prof. Purushottam Kulkarni

(Internal)

____

Dr. Vijay Raisinghani

(External)

____

Prof. V M Gadre

(Chairperson)

Date: ______________

Place: ______________

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY
CERTIFICATE OF COURSE WORK

This is to certify that **Mr. Sameer Kurkure** 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.	IT601	Mobile Computing	6
2.	HS699	Communication and Presentation Skills (P/NP)	4
3.	IT608	Data Warehousing and Data Mining	6
4.	IT619	IT Foundation Laboratory	8
5.	IT623	Foundation course of IT - Part II	6
6.	IT653	Network Security	6
		Semester 2 (Jan – Apr 2006)	
7.	IT614	Internet Technologies	6
8.	HS618	Introduction to Indian Astronomy (Institute Elective)	6
9.	IT630	Principles and Practices of Distributed Computing	6
10.	IT694	Seminar	4
11.	IT680	Systems Lab.	6
		Semester 3 (Jul – Nov 2006)	
12.	CS601	Algorithms and Complexity	6
		M.Tech. Project	
13.	IT696	M.Tech. Project Stage - I (Jul 2006)	18
14.	IT697	M.Tech. Project Stage - II (Jan 2007)	30
15.	IT698	M.Tech. Project Stage - III (Jul 2007)	42

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

Dated:

Acknowledgements

I take this opportunity to express my sincere gratitude for **Prof. Anirudha Sahoo** and **Prof. Sridhar Iyer** for their constant support and encouragement. Their excellent guidance has been instrumental in making this project work a success.

I would like to thank **Janak Chandarana** for his constant help throughout the project. I would also like to thank my colleagues **Shravan** and **Ranjith** for helpful discussions in the initial part of my project, **Kushal** and **Paresh** for being a supportive friend and the KReSIT department for providing me world class computing infrastructure.

I would also like to thank my **family** and **friends** especially the entire **MTech Batch**, who have been a source of encouragement and inspiration throughout the duration of the project.

Last but not the least, I would like to thank the entire KReSIT family for making my stay at IIT Bombay a memorable one.

Sameer Kurkure

I. I. T. Bombay

July 16th, 2007

Abstract

Wireless Fidelity for Rural Extension (WiFiRE) is a MAC layer protocol especially designed to provide internet broadband facility in rural area in India. It supports long ranged communication using inherited 802.16d MAC and 802.11b PHY layer. WiFiRE uses low cost chip sets with capability to communicate on unlicensed spectrum to transmit over 15-20km. The report covers the background knowledge of WiFiRe protocol with basic working of the protocol and its implementation in C. Problems associated with design and implementation and their plausible solutions are covered as a part of report. Additionally, it also comprises of sequence diagrams, flow diagrams, state diagrams etc. of working components of WiFiRE along with design model in C sockets and describes the issues and challenges involving implementation of the projects.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

ARP	: Address Resolution Protocol
BE	: Best Efforts
BS	: Base Station
BSID	: Base Station Identification
BWA	: Broadband Wireless Access
CFI	: Canonical format indicator
CID	: Connection Identifier
CRC	: Cyclic Redundancy Code
CS	: Carrier Sense
CRA	: Contention Resolution Algorithm
CSA	: Contention Slot Allocator
CSMA	: Carrier Sense Multiple Access
DCF	: Distributed Co-ordination Function
DCID	: Data Connection Identifier
DBPSK	: Differential Binary Phase Shift Keying
DL	: Down Link
DL-MAP	: Down Link Slot Allocation Map
DLL	: Data Link Layer
DQPSK	: Differential Quadratic Phase Shift Keying
DSA	: Dynamic Service Addtion
DSSS	: Direct Sequence Spread Spectrum
DL-TB	: Down Link Transport Block
FCC	: Federal Communications Commission
FTP	: File Transfer Protocol

GPC	: Grant Per Connection
GPSF	: Grant Per Service Flow Type
GPST	: Grant Per Subscriber Terminal
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
MAN	: Metropolitan Area Network
MTU	: Maximum Transmission Unit
NIC	: Network Interface Card
nrtPS	: Non-real Time Polling Service
PCF	: Point Co-ordination Function
PDU	: Protocol Data Unit
PHY	: Physical Layer
PoP	: Point of Presence
PS	: Physical Slot
PSH	: Packing Sub-Header
QoS	: Quality of Service
rtPS	: Real Time Polling Service
RF	: Radio Frequency
Rx	: Reception
S	: System
SAP	: Service Access Point
SDU	: Service Data Unit
SF	: Service Flow
SS	: Subscriber Station
ST	: Subscriber Terminal
TCI	: Tag Control Information
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
URL	:	Universal Resource Locater
VID	:	VLAN Identification
VLAN	:	Virtual LAN
VoIP	:	Voice over IP
WAN	:	Wide Area Network
WiFi	:	Wireless Fidelity
WiFiRE	:	Wireless Fidelity for Rural Extension
WiMAX	:	Worldwide Interoperability for Microwave Access

Chapter 1

Introduction and Motivation

Nowadays the use of Internet and mobile communication has grown to such a large extent that it become mandatory for daily usage in life. The statistics in India shows that there are more than 100 million mobile users in India [as per June 10th of 2006] which shows its importance in daily routine. Major population in India resides in remote areas where access to basic amenities like telephony, internet etc. are difficult to provide. Broadband wireless access (BWA) can become the best way to meet escalating business demand for rapid Internet connection and integrated data, voice and video services. BWA can extend fiber optic networks and provide more capacity than cable networks or digital subscriber lines (DSL). But deployment of BWA (WiMAX) compatible devices are much complex and costlier.

Rural areas are sparsely populated and their distances varies in few kilometers, unlike urban areas. Installation of more base stations will probably not solve this problem, which also costs more. Wireless Fidelity - Rural Extension (WiFiRE) introduces the concept of wireless communication over WiFi IEEE 802.11b physical layer (PHY) and WiMAX IEEE 802.16 MAC layer using low cost chip sets. 802.11b PHY has better availability of low cost chip sets which can operate on unlicensed 2.4GHz frequency band and WiMAX has potential to work over larger distances of 30-40km.

Almost every rural area can avail fixed phone lines, but mobile communication and broadband are difficult to deploy. For this, *WiFiRE* can provide a very good solution. *WiFiRE* uses WiFi PHY which has got a free license band spectrum (IEEE 802.11b, 2.4 GHz Band), the easy availability of WiFi chip sets, and good QoS features of WiMAX, which makes it suitable to provide long range communications for rural areas. *WiFiRE* uses a star topology network, in which main station (S) will be connected to set of Base Stations (BS) which in turn connected to sectorized antennas through which a Subscriber

Terminal (ST) will communicate.

Other approaches to solve the problem such as WiMAX, Optical Networks, DSL etc. are not cost effective and did not proved to provide affordable services to rural environment. The concept of *WiFiRE* seems to be good solution for this scenario and can satisfy bandwidth need at proper price that suits rural people.

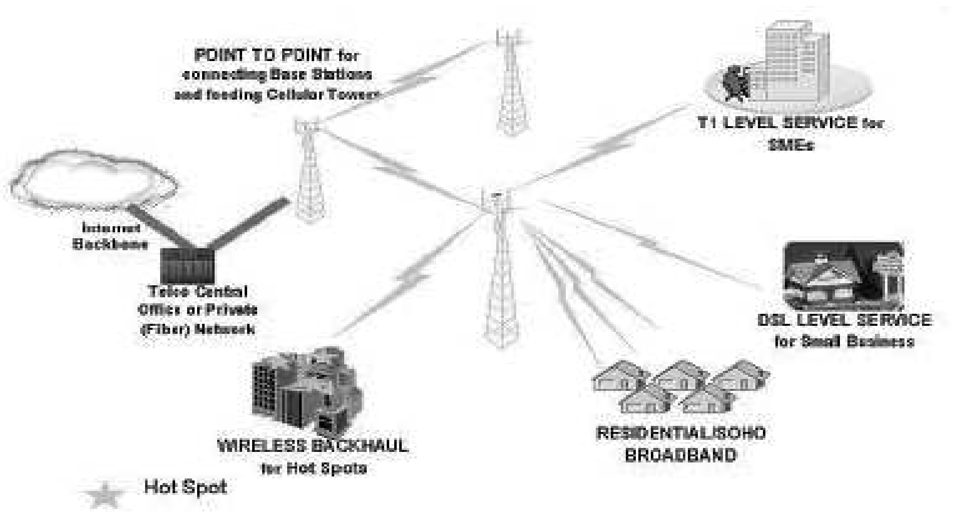

POINT TO POINT for connecting Base Stations and feeding Cellular Towers
T1 LEVEL SERVICE for SMEs
Internet Backbone
Telco Central Office or Private (Fiber) Network
DSL LEVEL SERVICE for Small Business
WIRELESS BACKHAUL for Hot Spots
RESIDENTIAL/SOHO BROADBAND
Hot Spot

Figure 1.1: WiFiRE Overview

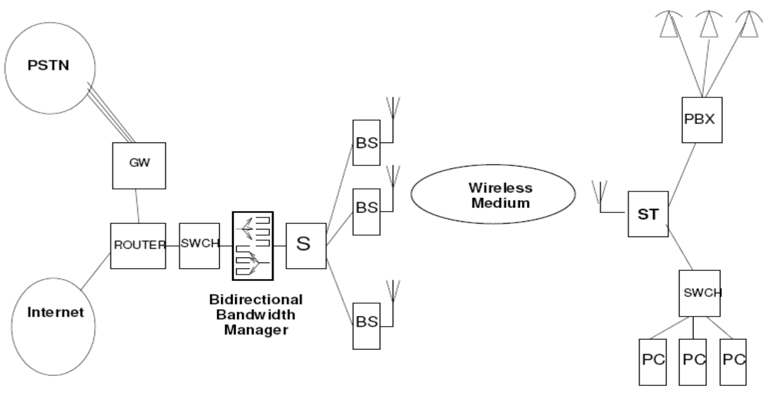


Figure 1.2: WiFiRE System Architecture

adapted from [1]

1.1 Architecture

WiFiRE broadband is basically for establishing connectivity between fixed stations in rural areas within 18-20Km around a fiber point of presence (PoP). It comprises of Base Stations (BS) which is WiFi 802.11b PHY enabled transceiver connected with a System (S) shared by multiple Subscriber Terminals (ST). Both BS and ST are fixed where as users with in ST (e.g. building, house, small campus etc) can be either fixed or mobile depending upon the internal network being used. ST is linked to various end users with different broadband bandwidth requirement and communicates with BS through exterior antennas, which is finally connected with public network like fiber PoP. Figure 1.1 gives a general architecture of WiFiRE.

The system uses a frequency band of 2.4GHz (unlicensed) for both line of sight (LOS) and non LOS communication with help of directional antennas for different sectors to communicate with ST. One base station setup can serve an area of about 20Km of radius which could cover few hundreds of villages.

1.2 Physical Layer

The PHY specification is basically targeted for operation in the 2.4GHz frequency band, which requires a base station of about 40m high tower connected with back haul service and a 12m high pole at ST to maintain the system gain of about 150dB for signal to travel around 30Km with better quality of reception. Figure 1.3 [1] shows PHY of WiFiRE shared by different directional sectorized antennas for signal to cover a larger distance.

Directional antennas covering each sector divides the cell into multiple PHYs and can operate independently from the other. But an ST may receive signals from multiple BS, in order to prevent interference due to side lobes of antennas no two adjacent antennas are permitted to transmit/receive at same time. BSs in the system (S) are configured as star topology which can operate alternatively or diagonally opposite BS for non-overlapping transmission. WiFiRE supports time division duplex (TDD) over single channel with multi-sector TDM (MSTDM) mechanism, which supports about 25Mbps (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 with synchronization with each other. Basic frame structure in case of TDD is shown in Figure

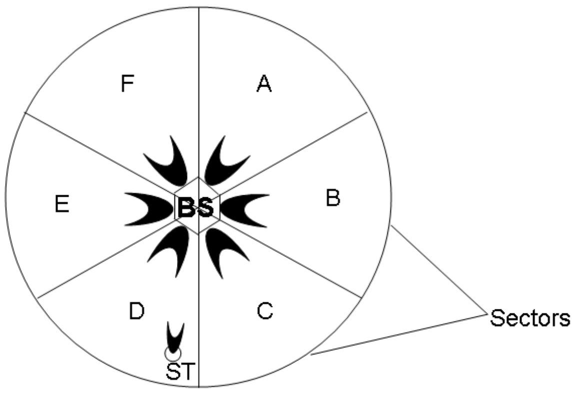


Figure 1.3: WiFiRE PHY Network Configuration
adapted from [1]

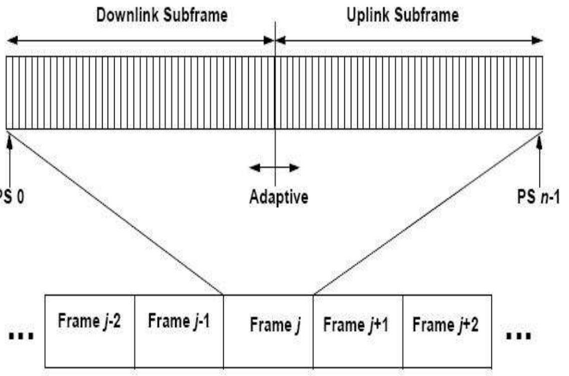


Figure 1.4: TDD frame structure
adapted from [2]

1.4 [2].

1.3 Motivation

There are alternatives present for broadband wireless access but most of them are not cost effective. WiMAX-d (IEEE 802.16d), can provide an alternate solution as it has got high gain and a good spectral efficiency, which can carry 80Mbps over-the-air per base station with a 20MHz allocation. The main drawback is deploying It requires complex and costly hardware that is not available easily. WiFi (IEEE 802.11b) can provide for short distances communication of about few meters but not for long distances. In 802.11 based networks, contention algorithm like Distributed Coordination Function (DCF) mechanism does not provide any delay guarantees and are more distributed in nature, while the Point Coordination Function (PCF) mechanism is efficient only for small number of nodes. 802.11 based Mesh Network [3], where it doesn't use the existing CSMA/CA technology in 802.11, instead it uses 2-phase TDMA based protocol. But the problem with current approach is MAC of 802.11b is its non capability of providing any quality of service except PCF. The outdoor long-distance use of 802.11 requires a revisit to the protocols at various layers of the OSI stack, as well as various system design issues. And finally, our Mobile cellular technologies cannot provide broadband services with high bandwidth need.

1.4 Problem Statement

WiFiRE made the mid way out which uses the WiFi IEEE 802.11b physical layer (PHY) and inherited WiMAX IEEE 802.16 MAC layer. 802.11b PHY has better availability of low cost chip sets which operates on unlicensed 2.4GHz frequency band. WiMAX systems are more complex than their WiFi counterparts and thus a WiMAX Base Station will cost more than its equivalent WiFi node. The support of 802.16 MAC layer helps WiFiRE to perform on larger area with support for greater number of subscriber terminal (ST). Also 802.16 MAC has more provisions for Quality of Service (QoS) and various synchronization mechanism to perform successful communication between distant subscriber terminals.

WiFiRE is a large project carried out among three organizations (IITB, IITM and IISc). PHY layer hardware (low cost chip set) fabrication and implementation of Scheduler at system S are taken care by other WiFiRE teams. Designing appropriate MAC and implementing it to provide access capabilities between base station (BS) and subscriber terminals (ST) comes out to be the major task to accomplish this project for our team.

1.5 Thesis Outline

WiFiRE works positively as a MAC layer protocol between base station (BS) and subscriber terminal (ST) providing various access capabilities like best effort and nrtPS.

The major contribution of this work are :

- An insight into the problem of communication over WiFiRE MAC.

- An approach to implement MAC layer using C sockets, Signals handling and Multi threading.
- A demonstration of the implemented MAC on a test bed to prove the effectiveness of the approach.

In this thesis, chapter 2 discusses existing protocol available for long range communication. It also discusses the details of PHY used for WiFiRE protocol which make the protocol cost effective.

In chapter 3, contains the detail description of protocol to be implemented and finally execution and demonstration of working of protocol in described in chapter 4 which includes the testing scenarios along with valid assumptions. Finally we present conclusions and future work in chapter 5.

Chapter 2

Literature Survey

This chapter covers related work of WiFiRE MAC layer protocol which includes study of WiMAX [2] and other Wireless Broadband technologies, WiFi PHY, C sockets along with some implementation issues.

2.1 Broadband Wireless Access Technology

WiMAX technology offers higher bandwidth and greater range as compared to WiFi (Wireless Fidelity - IEEE 802.11 Standard) based wireless systems. It is a wireless alternative to many existing wired back haul and last mile coverage deployments such as Cable Modems, Digital Subscriber Line (DSL), T and E-Carrier systems and Optical carrier technologies.

The IEEE 802.16 Working Group has developed standards for broadband wireless access that cater mainly to two types of usage models.

- **IEEE 802.16** - Fixed users (Residential, SOHO, Enterprises)
- **IEEE 802.16e** - Portable users (Nomadic, Mobile)

2.1.1 Architecture

A fixed BWA system has at least one common Base Station (BS) shared by multiple Subscriber Stations (SS). WiFiRE network topology is inherited from WiMAX architecture and follows similar MAC mechanism for communication between BS and ST which will further explained in detail in the next chapter. Subscriber Station is linked to various end users with different broadband bandwidth requirement and communicates with the BS through exterior antennas, which is finally connected with the public network.

The system uses a frequency band of 10-66GHz, for which line of sight (LOS) communication become the practical necessity due to shorter wavelength. This may be one of the reasons of SS being stationary for better reception of signal. BS uses directional antennas for different sectors to follow the line of sight communication to SS. Other 2-11GHz frequency band (both licensed and unlicensed) are being used for non-line of sight (NLOS) communication and are addressed in IEEE Project 802.16a.

2.1.2 MAC Mechanism

Each SS is periodically granted transmission opportunities by the BS. The BS accepts bandwidth requests from the SSs and grants them time-slots on the uplink channel. These grants are made based on the service agreements, which are negotiated during connection setup. The BS may also reserve certain time slots on the uplink that are available to all SSs for contention. The SSs may use these slots to transfer data or to request for dedicated transmission opportunities.

The uplink channel is divided into a stream of mini-slots (refer figure 1.4). A subscriber station that desires to transmit on the uplink requests transmission opportunities in units of mini-slots. The BS accepts requests over a period of time and compiles an allocation map (MAP) message describing the channel allocation for a certain period into the future called the MAP time. The MAP is then broadcast on the downlink to all subscriber stations. In addition to dedicated transmission opportunities for individual subscriber stations, a MAP message may allocate a certain number of open slots for contention based transmission. These transmission opportunities are prone to collisions.

The downlink sub frame includes a frame control section that contains the downlink MAP (DL-MAP) for the current downlink frame as well as the uplink MAP (UL-MAP) for a frame in future. The DL-MAP informs all SSs when to listen for transmissions destined for them in the current frame. The UL-MAP informs SSs of their transmission opportunities as a response to their dynamic bandwidth requests, or on the basis of prior service agreements. Due to the dynamic allocation of bandwidth according to the demands for the variety of services that may be active, duration of burst profiles and the presence or absence of a TDMA portion (used for synchronization of BS and SS) vary dynamically from frame to frame.

2.1.3 General QoS Architecture

Major component for the architecture [4] includes traffic classifier, SS's upstream (access) scheduler, BS's upstream (grant) scheduler and downstream scheduler.

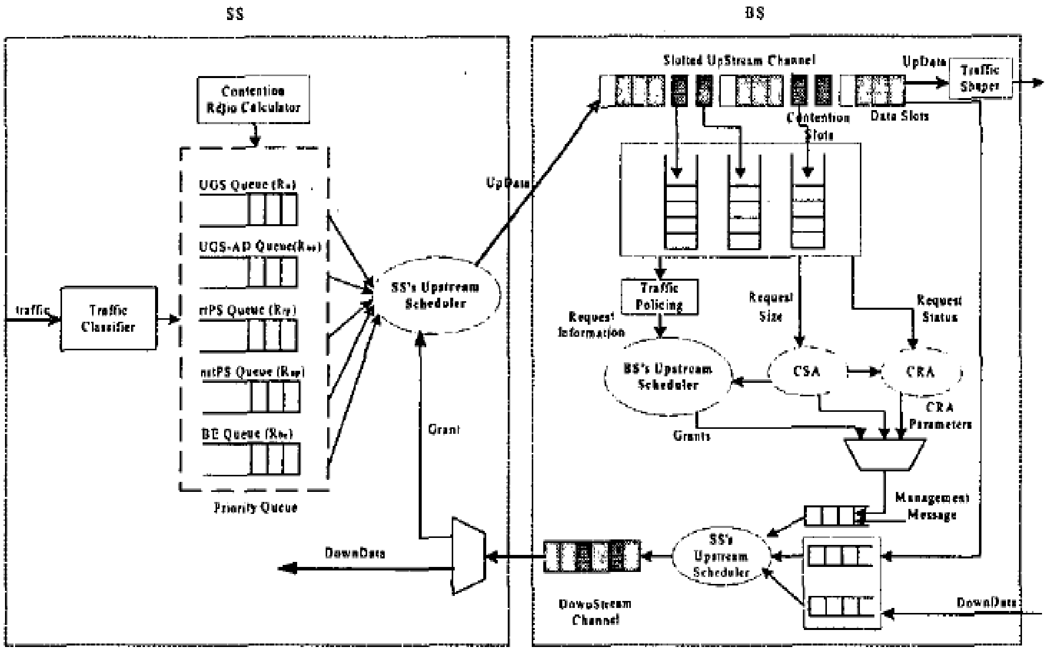


Figure 2.1: QoS Architecture for IEEE 802.16 MAC protocol
(adapted from [4])

Each SS is employed with an additional upstream scheduler to allocate the granted upstream slots to their respective connections. In GPSS mode, only the information of total bandwidth requirement of the SS has to be sent to BS's upstream scheduler. In this way updated information is sent to BS reduces the workload upstream scheduler at base station. Referring to Figure 2.1, when traffic is generated at SS, traffic classifier divides the traffic according to the various QoS levels and associated them to the corresponding queue like UGS queue, rtPS queue, nrtPS queue and BE queue. This whole comprises of Contention Ratio Calculator (CRC) which dynamically assign different ratio parameters R_u, R_{rp}, R_{np} and R_{be} to UGS, rtPS, nrtPS and BE queue respectively. Relation among these parameters must be:

$$R_u + R_{rp} + R_{np} + R_{be} = 100\%$$

To ensure that the connection to SS matches with its traffic parameters, traffic policing technique is introduced at the BS which efficiently monitors the incoming traffic behavior i.e to prevent SS from using more upstream bandwidth than what is specified. Contention Slot Allocator (CSA) is used for dynamically adjusting the number for usage of contention and reservation slots. And Contention Resolution Algorithm (CRA) defines the utilization of contention slots and constraints to resolve the contention issues for nrtPS and BE service flows. CRA [5] improves the maximum achievable stable throughout of random access protocol, which basically assign retransmission times after collisions deterministically to a subgroup of users to avoid idle channel periods due to random retransmission.

Priority	**Service Flows**	**Queuing Policy**
1	UGS	WPS
2	rtPS	WPS
3	nrtPS	WRR Scheduler
4	BE	FIFO

Table 2.1: SS's Scheduling Strategy

According to the information of request from the SS, BS's upstream scheduler schedules grants to the SS, then SS's upstream scheduler selects appropriate data packets from the various queue and send them through upstream data slot granted by the BS's upstream scheduler. SS's scheduler can use various priority based techniques like MPFQ (Multiple Priority Fair Queuing), WFQ (Wireless Fair Queuing) for high priorities, WRR (Weighted Round) scheduling for middle priorities and FIFO for lower priorities. With this approach traffic streams requiring short delays like UGS and rtPS can satisfy their delay guarantee. Table 2.1 [4] shows the priority of services flows and their queuing policies.

2.1.4 Admission control based on the scheduling services characteristics

In this scheme [6] BS decides whether to accept or reject the user's request for bandwidth allocation. The decision is made on the basis of long term bandwidth requirements of the connections and current network state i.e. there become a trade off between accepting a request for connection (may cause QoS degradation of existing connections) and rejecting a request for connection in order to sustain QoS of existing connections at certain level. Now it depends on the type of service flows that are block on for satisfying other bandwidth requirements. Blocking UGS flow could be serious issue for an end user, compared to blocking a non-UGS flow. UGS flows are given higher priorities than any other non-UGS flows like rtPS, nrtPS and BE. So every request for UGS connection is accepted without any restriction (if bandwidth is available) where as a non-UGS request is accepted only when total used bandwidth is considerably less than the predetermined value.

Suppose total bandwidth allocated for a SS is B, then its predetermined value becomes $B - U$ where U is bandwidth reserved for UGS traffic flow. Let bandwidth required by UGS, rtPS, and nrtPS service flows be b_{UGS}, b_{rtPS}, and b_{nrtPS}, where b_{nrtPS} can vary between [b^{min}_{nrtPS}, b^{max}_{nrtPS}].

As the request for the connections increases, some bandwidth from nrtPS flow is given to the new connections to have more UGS, rtPS, and nrtPS connections. We gradually degrade the assigned bandwidth for existing nrtPS connection to certain acceptable level. Let δ be the amount of bandwidth degraded at each step and l^{n}_{nrtPS} be the degradation level then the current reserved bandwidth for each nrtPS connection will be $b^{max}_{nrtPS} - l^{n}_{nrtPS}\delta$ which must satisfy $b^{max}_{nrtPS} - l^{n}_{nrtPS}\delta \geq b^{min}_{nrtPS}$. This model as a whole, is considered as Degradation Model.

Degradation Model

- A new UGS connection is accepted when the total bandwidth currently used for existing connections plus b_{UGS} is less than B otherwise the request is rejected.

- A new rtPS connection is accepted when the total bandwidth currently used for ongoing connections plus b_{rtPS} is less than $B-U$ and BS reserves specific bandwidth for the connection along b_{rtPS} otherwise BS degrades the bandwidth of all existing nrtPS connections until total bandwidth currently used plus b_{rtPS} is less than equal to $B - U$, if this figure is still greater than $B - U$ and further degradation is not possible in order to satisfy the minimum bandwidth requirement of nrtPS flow (i.e. degradation till l^{max}_{nrtPS} level has been done) then the request for the connection is rejected.

- Like rtPS, when a request for new nrtPS connections arrives at the BS, the total bandwidth allocated for existing connections plus $b_{nrtPS}^{max} - l_{nrtPS}^{n}\delta$ must be less than or equal to $B-U$ otherwise BS degrades the bandwidth until the total bandwidth for ongoing connections plus bandwidth allocated for new nrtPS is less than $B-U$. The new bandwidth for all the nrtPS connections including new nrtPS becomes $b_{nrtPS}^{max} - l_{nrtPS}^{n'}\delta$ where $l_{nrtPS}^{n'}$ ($l_{nrtPS}^{n} \leq l_{nrtPS}^{n'} \leq l_{nrtPS}^{max}$) becomes the updated degradation level for all nrtPS connections.

- The request for new BE connection is admitted but BS does not degrade bandwidth of any existing connections, rather it records the request and grant access when other services do not transmit.

2.2 WiFi PHY layer

As with other 802.11 Physical layers, 802.11b includes Physical Layer Convergence Procedure (PLCP) and Physical Medium Dependent (PMD) sub-layers. These are somewhat sophisticated terms that the standard uses to divide the major functions that occur within the Physical Layer. The PLCP prepares 802.11 frames for transmission and directs the PMD to actually transmit signals, change radio channels, receive signals, and so on.

2.2.1 PLCP Frame Fields

The PLCP takes each 802.11 frame that a station wishes to transmit and forms what the 802.11 standard refers to as a PLCP protocol data unit (PPDU). The resulting PPDU includes the following fields in addition to the frame fields imposed by the MAC Layer:

- Sync. This field consists of alternating 0s and 1s, alerting the receiver that a receivable signal is present. The receiver begins synchronizing with the incoming signal after detecting the Sync.

- Start Frame Delimiter. This field is always 1111001110100000 and defines the beginning of a frame.

- Signal. This field identifies the data rate of the 802.11 frame, with its binary value equal to the data rate divided by 100Kbps. For example, the field contains the value of 00001010 for 1Mbps, 00010100 for 2Mbps, and so on. The PLCP fields, however, are always sent at the lowest rate, which is 1Mbps. This ensures that the receiver is initially uses the correct demodulation mechanism, which changes with different data rates.

- Service. This field is always set to 00000000, and the 802.11 standard reserves it for future use.
- Length. This field represents the number of microseconds that it takes to transmit the contents of the PPDU, and the receiver uses this information to determine the end of the frame.
- Frame Check Sequence. In order to detect possible errors in the Physical Layer header, the standard defines this field for containing 16-bit cyclic redundancy check (CRC) result. The MAC Layer also performs error detection functions on the PPDU contents as well.
- PSDU. The PSDU, which stands for Physical Layer Service Data Unit, is a fancy name that represents the contents of the PPDU (i.e., the actual 802.11 frame being sent).

2.2.2 DSSS Spreading Function

802.11b uses DSSS to disperse the data frame signal over a relatively wide (approximately 30MHz) portion of the 2.4GHz frequency band. This results in greater immunity to radio frequency (RF) interference as compared to narrow band signaling, which is why the Federal Communications Commission (FCC) deems the operation of spread spectrum systems as license free.

Because of the relatively wide band DSSS signal, you must set 802.11b access points to specific channels to avoid channel overlap, which can cause reductions in performance. Refer to a previous tutorial for more details on setting 802.11b access point channels.

In order to actually spread the signal, an 802.11 transmitter combines the PPDU with a spreading sequence through the use of a binary adder. The spreading sequence is a binary code. For 1Mbps and 2Mbps operation, the spreading code is the 11-chip Barker sequence, which is 10110111000. The binary adder effectively multiplies the length of the binary stream by the length of the sequence, which is 11. This increases the signaling rate and makes the signal span a greater amount of frequency bandwidth.

5.5Mbps and 11Mbps operation of 802.11b doesn't use the Barker sequence. Instead, 802.11b uses complementary code keying (CCK) (define) to provide the spreading sequences at these higher data rates. CCK derives a different spreading code based on fairly complex functions depending on the pattern of bits being sent. The modulator simply refers to a table for the spreading sequence that corresponds to the pattern of data bits being sent. This is necessary to obtain the most efficient processing of the data in order to achieve the higher data rates. DSSS Modulation

The modulator converts the spread binary signal into an analog waveform through the use of different modulation types, depending on which data rate is chosen. For example with 1Mbps operation, the PMD uses differential binary phase shift keying (DBPSK). This isn't really as complex as it sounds. The modulator merely shifts the phase of the center transmit frequency to distinguish a binary 1 from a binary 0 within the data stream.

For 2Mbps transmission, the PMD uses differential quadrature phase shift keying (DQPSK), which is similar to DBPSK except that there are four possible phase shifts that represents every two data bits. This is a clever process that enables the data stream to be sent at 2Mbps while using the same amount of bandwidth as the one sent at 1Mbps. The modulator uses similar methods for the higher, 5.5Mbps and 11Mbps data rates.

2.2.3 Transmit Frequencies

The transmitter's modulator translates the spread signal into an analog form with a center frequency corresponding to the radio channel chosen by the user. The following identifies the center frequency of each channel:

Various countries limit the use of these channels. For example, the U.S. only allows the use of channels 1 through 11, and the U.K. can use channels 1 through 13. Japan, however, authorizes the use all 14 channels. This complicates matters when designing international public wireless LANs. In that case, you need to choose channels with the least common denominator.

After RF amplification takes place based on the transmit power you've chosen (100mW maximum for the U.S.), the transmitter outputs the modulated DSSS signal to the an-

Channel	**Frequency (GHz)**
1	2.412
2	2.417
3	2.422
4	2.427
5	2.432
6	2.437
7	2.442
8	2.447
2	2.452
10	2.457
11	2.462
12	2.467
13	2.472
14	2.484

Table 2.2: Transmit Frequencies

tenna in order to propagate the signal to the destination. The trip in route to the destination will significantly attenuate the signal, but the receiver at the destination will detect the incoming Physical Layer header and reverse (demodulate and despread) the process implemented by the transmitter.

2.3 Programming components

Implementation of MAC layer protocol involves in depth knowledge of a programming language which can handle system level jobs and has ability to access specific hardware addresses and to ”pun” types to match externally imposed data access requirements, and low runtime demand on system resources. Here we prefer language C to code almost all the modules of the project.

2.3.1 C Sockets

A socket is a generalized interprocess communication channel. Like a pipe, a socket is represented as a file descriptor. But, unlike pipes, sockets support communication between unrelated processes, and even between processes running on different machines that communicate over a network. Sockets are the primary means of communicating with other machines; telnet, rlogin, ftp, talk, and the other familiar network programs use sockets. Not all operating systems support sockets. In the GNU library, the header file `sys/socket.h` exists regardless of the distribution type, and the socket functions always exist, but if the system does not really support sockets, these functions always fail.

Concept of sockets generally comes above the TCP layer in the protocol stack. The data sent or receive from the socket is comes after processing from MAC, IP and TCP layers. But it also depends on the way socket are created and handled. Some socket can directly sent desired bytes on PHY and receive directly from network interface card (NIC) for further execution.

The packet's trip proceeds on different paths depending on the socket's current state; if the connection is already established, the packet will be passed to `tcp_rcv_established()` function. This one has the important task of dealing with the complex TCP acknowledgment mechanisms and header processing, which of course are not very relevant here. The only interesting line is the call to the `data_ready()` function belonging to the current sock (sk), commonly pointing to `sock_def_readable()`, which awakens the receiving process (the one that was receiving on the socket) with `wake_up_interruptible()`. But the `PF_PACKET` family deserves a special handling. Packets must be sent directly to the application's socket without being processed by the network stack. This is one the reason why we part from path of using `libpcap` packet capturing library to `PF_PACKET` family `RAW` sockets. The `libpcap` library gives us the functionality to sniff the packet at NIC, read and alter them accordingly but a copy of the packet thus arrived get processed and is forwarded to the concern layer. But in sockets the communication is held between socket to socket of two applications running on different machines, which gives us added advantage of having full control over packet contents and comparatively fast sent/receive of data as it by passes network layer processing which could help us to write our own desired set of protocols.

2.3.2 Signal Handling

A signal is a software interrupt delivered to a process. The operating system uses signals to report exceptional situations to an executing program. Some signals report errors such as references to invalid memory addresses; others report asynchronous events, user generated interrupts etc.

If an event is anticipated that can cause signals, we can define a handler function and tell the operating system to run it when that particular type of signal arrives. Also one process can send a signal to another process; this allows a parent process to abort a child, or two related processes to communicate and synchronize.

2.3.2.1 Basic concepts of Signal

This section explains basic concepts of how signals are generated, what happens after a signal is delivered, and how programs can handle signals. A signal reports the occurrence of an exceptional event. These are some of the events that can cause (or generate, or raise) a signal:

- A program error such as dividing by zero or issuing an address outside the valid range.
- A user request to interrupt or terminate the program. Most environments are set up to let a user suspend the program by typing Ctl-z, or terminate it with Ctl-c. Whatever key sequence is used, the operating system sends the proper signal to interrupt the process.
- The termination of a child process.
- Expiration of a timer or alarm.
- A call to `kill` or `raise` by the same process.
- A call to kill from another process. Signals are a limited but useful form of inter-process communication.

In general, the events that generate signals fall into three major categories: errors, external events, and explicit requests. An error means that a program has done something invalid and cannot continue execution. But not all kinds of errors generate signals–in fact, most do not. For example, opening a nonexistent file is an error, but it does not raise a signal. The errors which raise signals are those which can happen anywhere in the program, not just in library calls. These include division by zero and invalid memory addresses. An external event generally has to do with I/O or other processes. These include the arrival of input, the expiration of a timer, and the termination of a child process. An explicit request means the use of a library function such as kill whose purpose is specifically to generate a signal.

Signals may be generated synchronously or asynchronously. A synchronous signal pertains to a specific action in the program, and is delivered (unless blocked) during that action. Errors generate signals synchronously, and so do explicit requests by a process to generate a signal for that same process. Asynchronous signals are generated by events outside the control of the process that receives them. These signals arrive at unpredictable times during execution. External events generate signals asynchronously, and so do explicit requests that apply to some other process.

2.3.2.2 How Signals are delivered

When a signal is generated, it becomes pending. Normally it remains pending for just a short period of time and then is delivered to the process that was signaled. However, if that kind of signal is currently blocked, it may remain pending indefinitely–until signals of that kind are unblocked. Once unblocked, it will be delivered immediately.

When the signal is delivered, whether right away or after a long delay, the specified action for that signal is taken. For certain signals, such as `SIGKILL` and `SIGSTOP`, the action is fixed, but for most signals, the program has a choice: ignore the signal, specify a handler function, or accept the default action for that kind of signal. The program specifies its choice using functions such as signal or sigaction

If the specified action for a kind of signal is to ignore it, then any such signal which is generated is discarded immediately. This happens even if the signal is also blocked at the time. A signal discarded in this way will never be delivered, not even if the program subsequently specifies a different action for that kind of signal and then unblocks it.

If a signal arrives which the program has neither handled nor ignored, its default action takes place. Each kind of signal has its own default action. For most kinds of signals, the default action is to terminate the process. For certain kinds of signals that represent "harmless" events, the default action is to do nothing.

When a signal terminates a process, its parent process can determine the cause of termination by examining the termination status code reported by the wait or `waitpid` functions. The information it can get includes the fact that termination was due to a signal, and the kind of signal involved. If a program you run from a shell is terminated by a signal, the shell typically prints some kind of error message.

The signals that normally represent program errors have a special property: when one of these signals terminates the process, it also writes a core dump file which records the state of the process at the time of termination. You can examine the core dump with a debugger to investigate what caused the error.

2.3.2.3 Standard Signals

This section lists the names for various standard kinds of signals and describes what kind of event they mean. Each signal name is a macro which stands for a positive integer–*the signal number* for that kind of signal. Table 2.3 lists the signals which are directly or indirectly related to the implementation.

Type of Signals	**Signals**
Program Error Signals	`SIGFPE`, `SIGILL`, `SIGSEGV`, `SIGBUS`, `SIGABRT`
Termination Signals	`SIGINT`, `SIGQUIT`, `SIGTERM`, `SIGKILL`
Alarm Signals	`SIGALRM`, `SIGVTALRM`, `SIGPROF`
Asynchronous I/O Signals	`SIGIO`, `SIGURG`

Table 2.3: Type of Signals

Masking Signals

One of the problems that might occur when handling a signal, is the occurrence of a second signal while the signal handler function executes. Such a signal might be of a different type than the one being handled, or even of the same type. Thus, we should take some precautions inside the signal handler function, to avoid races. This can be achieved by masking signals. Following are the masking bits used in our project for smooth running of the protocol:

- `SA_INTERRUPT`: If a reentrant system call is being executed when the signal is received, do not automatically restart it. A `SA_INTERRUPT` bit that must be set in the `sigaction` structure to force system calls to be interrupted. `SIGALRM` signal is `SA_INTERRUPT` masked so it can be interrupted by other succeeding `SIGALRM` signal to proceed precise running of our timers.

- `SA_RESTART`: This flag controls what happens when a signal is delivered during certain primitives (such as `open`, `read` or `write`), and the signal handler returns normally. There are two alternatives: the library function can resume, or it can return failure with error code `EINTR`. The choice is controlled by the `SA_RESTART` flag for the particular kind of signal that was delivered. If the flag is set, returning from a handler resumes the library function. If the flag is clear, returning from a handler makes the function fail. All other signals, except `SIGARLM`, are `SA_RESTART` masked so as to continue their interrupted functionality.

2.3.3 Parallel Executions

Parallel processing in C can be acheived with the use of POSIX Threads. The implementation of the protocol involves use of threads for various reasons like Beacon Broadcast mechanism, thread for each upcoming ST at BS[1], threads for sending/receiving data at ST etc.

Threads

Technically, a thread is defined as an independent stream of instructions that can be scheduled to run as such by the operating system. Thread operations include thread creation, termination, synchronization (joins,blocking), scheduling, data management and process interaction. All threads within a process share the same address space. This is the reason for using `pthreads` instead of `fork()`, as `fork()` make exact copy of process including environmental variables, file descriptor table etc. The forked process does not retain values of common variables. Threads in the same process share:

- Process instructions

[1]Subjected to change during precise implementation

- Most data
- Open files (descriptors)
- Signals and signal handlers
- Current working directory
- User and group id

In the UNIX environment a thread:

- Exists within a process and uses the process resources
- Has its own independent flow of control as long as its parent process exists.
- Duplicates only the essential resources it needs to be independently scheduled.
- May share the process resources with other threads that act equally independently (and dependently)
- Dies if the parent process dies - or something similar
- Is "lightweight" because most of the overhead has already been accomplished through the creation of its process.

Thread has unique -

- Thread ID
- Set of registers, stack pointer
- Stack for local variables, return addresses
- Signal mask
- Priority
- Return value: errno

Compiling a thread program

```
$ gcc -o wifire.o wifire.c -lpthread
```

Most of the threads in the implementation do not need synchronization as they are performing isolated tasks and mostly does read operations on shared data.

Terminating Threads

- There are several ways in which a Pthread may be terminated:
  - The thread returns from its starting routine (the main routine for the initial thread).
  - The thread makes a call to the `pthread_exit` subroutine.
  - The thread is canceled by another thread via the `pthread_cancel` routine.
  - The entire process is terminated due to a call to either the `exec` or `exit` subroutines.
- `pthread_exit` is used to explicitly exit a thread.
- If `main()` finishes before the threads it has created, and exits with `pthread_exit()`, the other threads will continue to execute otherwise, they will be automatically terminated when `main()` finishes.
- The programmer may optionally specify a termination status, which is stored as a void pointer for any thread that may join the calling thread.

Chapter 3

WiFiRE MAC Layer Protocol

3.1 Description of WiFiRE protocol

The basic design of WiFiRE comprises of a single operator Station (S) which have licensed bandwidth like dedicated lines, fiber PoP etc. This operator provides the communication base for the outside world to rural environment. The total area is being sectored and each sector will be having Base Station (BS), which is a sectorized antenna of height around 40m that lies near point of presence (PoP). BS are arranged such that they can simultaneously able to transmit or receive within the sectors. There are Subscriber Terminals (ST) situated at the villages which have 10-12m directional antennas. Both BS and ST are fixed where as users with in ST (e.g. building, house, small campus etc) can be either fixed or mobile depending upon the internal network being used.

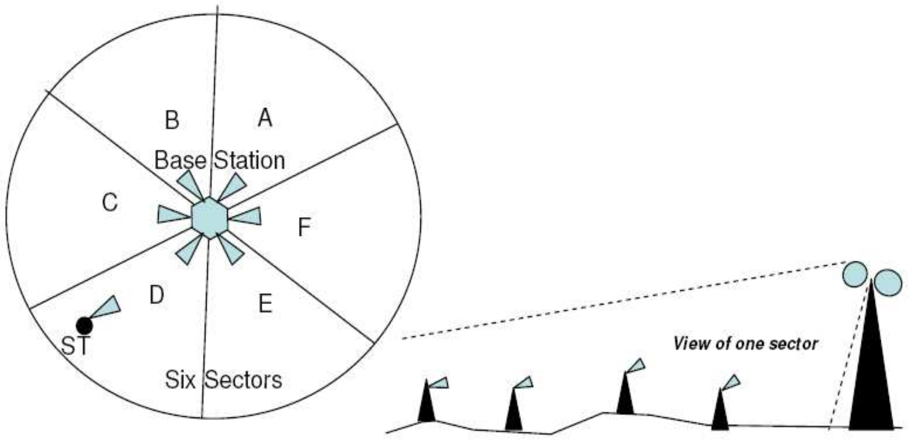


Figure 3.1: WiFiRE Topology

adapted from [1]

These are the basic points for the villages from where people will be able to communicate with the outer world. These ST's should be at a height so as to maintain a system gain of 150dB. Users may connect to these ST's using wired or wireless means of communication. The System will be of star topology. The network topology will be as shown in the following figure 3.1.

Each BS can cover up to 15-20km range, covering around 100 villages. Each BS will be responsible for all the communication that takes place in its sector range. Each ST will be connected to voice and data terminals in the village by a local area network. As mentioned earlier these ST will be directional and will be connected to corresponding BS covering the sector, thus providing reliable data transfer. Chances of interference with the other transceivers can be solved by locking up ST with the BS with highest signal strength. BSs in the system (S) are configured to operate alternatively or diagonally opposite BS for non-overlapping transmission. WiFiRE supports time division duplex (TDD) over single channel with multi-sector TDM (MSTDM) mechanism, which supports about 25Mbps (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 with synchronization with each other. Time is divided into frames, which is further divided into DownLink (DL) and UpLink (UL) segments, which may not be of equal time intervals. In each DL slot one or zero transmissions can take place in each sector. Multiple BS antennas can transmit simultaneously provided they do so in a non-interfering manner.

Figure 3.2 is sequence diagram for basic working of WiFiRE protocol. Beacons are being transmitted at the start of each DL segment, which contains information for time synchronization of the ST(s) in that sector, information regarding the DL and UL slots allocations (which are called DL and UL maps respectively) for that frame, and other control information. These DL and UL maps are computed online because there may be site dependent or installation dependent losses and different time varying requirements at each point of time.
The basic assumptions for working for WiFi-Re protocol are stated as:

- Wireless links in the system are fixed, single hop, with a star topology. Mobility and multi-hop wireless links are not considered.

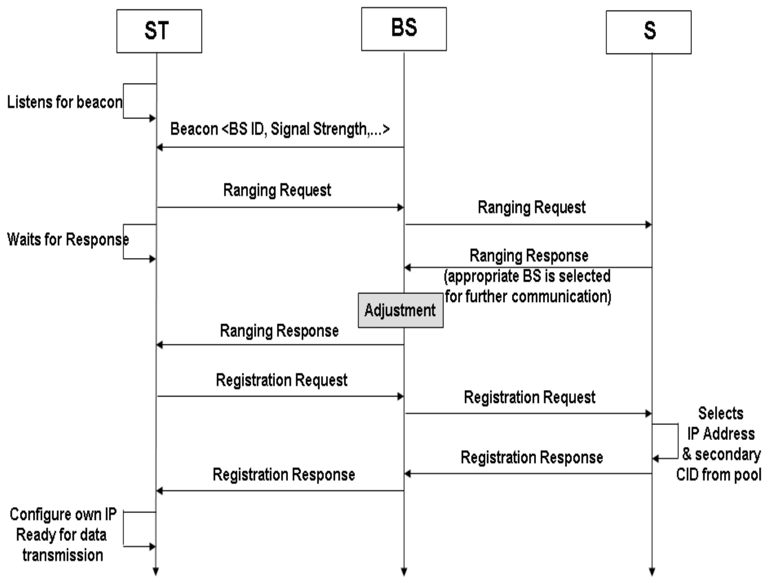


Figure 3.2: Basic communication sequence diagram

- Fixed carrier frequency and WiFi radios operating at 11Mbps, except PHY operating at 1 or 2 Mbps.
- Various components in the system will be having unique IP addresses.
- About 20MHz(1 carrier) of conditionally licensed spectrum is available for niche/rural areas.
- All nodes in the system are operated by a single operator who owns the conditional license.
- The availability of unlicensed or free spectrum in the 2.4GHz band.
- The existence of point of presence (PoP) every 25km or so, for backbone connectivity.

3.2 Basic MAC Mechanism and Framing

WiFiRE MAC is time division duplexed, where time is divided into frames and each frame comprises of uplink (UL) sub-frame and downlink (DL) sub-frame. Each sub-frame is again divided in time slots called mini-slots. The downlink communication is from BS to ST happens on point to multi-point basis and the uplink communication is from ST to BS happens on point to point basis. DL sub-frame is generally greater than UL sub-frame, their partition point is adaptive but DL to UL ratio should be fixed during the time of initialization (default 2:1).

Each ST is associated to some BS of system S statically and is periodically granted transmission opportunities by the BS. The BS accepts bandwidth requests from the STs and grants them time-slots on the uplink channel. These grants are made based on the service agreements, which are negotiated during connection setup. The BS may also reserve certain time slots on the uplink that are available to all STs for contention. The STs may use these slots to transfer data or to request for dedicated transmission opportunities.

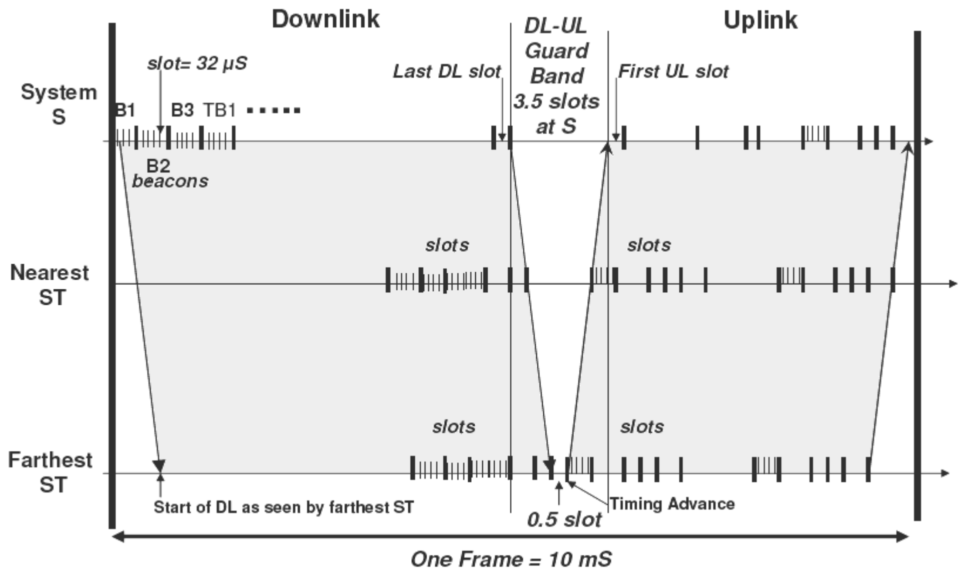


Figure 3.3: WiFiRE Timing Diagram
adapted from [1]

The downlink sub-frame includes a frame control section that contains the beacon (Figure 3.3 [1]) which include Operator ID, System ID, BS ID, downlink MAP (DL-MAP) for the current downlink frame as well as the uplink MAP (UL-MAP) for a frame in future. The DL-MAP informs all STs when to listen for transmissions destined for them in the current frame. The UL-MAP informs STs of their transmission opportunities as a response to their dynamic bandwidth requests, or on the basis of prior service agreements. Due to the dynamic allocation of bandwidth according to the demands for the variety of services that may be active, duration of burst profiles and the presence or absence of a TDMA portion (used for synchronization of BS and SS) vary dynamically from frame to frame.

A subscriber terminal that desires to transmit on the uplink, requests transmission opportunities in units of mini-slots. The BS accepts requests over a period of time and compiles an allocation map (MAP) message describing the channel allocation for a certain period into the future called the MAP time. The MAP is then broadcast on the downlink to all subscriber terminals. In addition to dedicated transmission opportunities for individual subscriber terminal, a MAP message may allocate a certain number of open slots for contention based transmission. These transmission opportunities are prone to collisions.

3.2.1 Working of protocol

Whenever a subscriber terminal powers on following steps are performed by BS and ST in order to start successful data communications:

1. ST fetches Operator ID and System ID from configuration file installed at the time of establishment of ST.

2. ST listens for beacons messages.

3. ST notes BS ID, its signal strength and ranging slots (allocated in ULMAP) for each beacon received.

4. ST transmit a ranging request to the BS with beacon received with highest signal strength.

5. ST waits for start of ranging slot in the corresponding uplink subframe.

6. Further ST transmits the ranging request message in the ranging slot prescribed in ULMAP.

7. ST waits for ranging response and monitors DLMAP in all beacons of the subsequent frames. (ST can initiate backoff mechanism and again repeats from step 5 in case of no response. This process is cyclic in nature.)

8. System S receives the ranging request and selects an appropriate BS for further communication with the ST.

9. S construct ranging response and scheduler allocates it into next or other subsequent downlink sub-frame and makes an entry in DLMAP. S transmits ranging response in appropriate DL slot through the selected BS.

10. After getting the ranging response from S, ST determines basic connection ID and primary connection ID for further communication. Here it should be noted that the ranging response is received only from the associated BS. (Here the BS and ST are in synchronization with each other in terms of PHY and Time)

11. ST constructs a registration request and transmit in UL slot (if allocated by S) or in contention slots and wait for response. (ST can initiate back off mechanism and again repeats registration process in case of no response.)

12. On reception of registration request, S constructs registration response and assigns IP address and secondary connection ID for further data communication.

13. Now ST has its own IP address and can dynamically request a connection to S. ST or S issues MAC_CREATE_CONNECTION.request in the form of Dynamic Service Addition Request message to S or ST respectively.

14. At the received end S constructs MAC_CREATE_CONNECTION.response in the form of Dynamic Service Addition Response message to ST.

15. Upon receiving response message from S, ST performs MAC_DATA transmission/reception. As ST can only communicate with S (and not directly with any other ST) it continuously monitors DLMAP and ULMAP for reception and transmission of data respectively. In between ST or S can generate MAC_CHANGE_CONNECTION.request/response in form of Dynamic Service Change Request/Response message.

16. MAC_TERMINATION_CONNECTION.request is transmitted in the form of Dynamic Service Deletion Request message after completion of data transfer.

17. MAC_TERMINATION_CONNECTION.response is given back as a Dynamic Service Deletion Response message to requester.

The basic working of the protocol is divided into three major procedures: (1) *Network Initialization* which includes ranging and registration of ST with system S. Basic

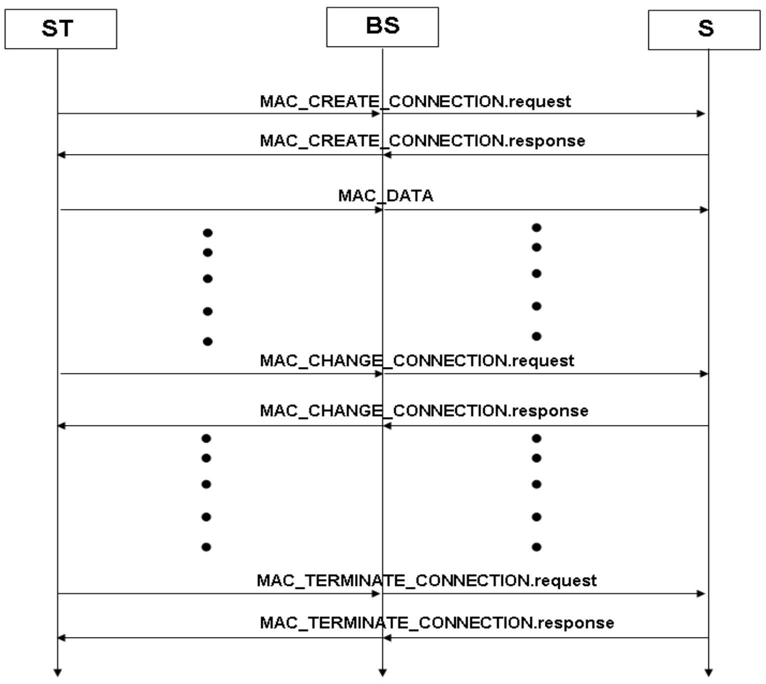

ST
BS
S
MAC_CREATE_CONNECTION.request
MAC_CREATE_CONNECTION.response
MAC_DATA
MAC_CHANGE_CONNECTION.request
MAC_CHANGE_CONNECTION.response
MAC_TERMINATE_CONNECTION.request
MAC_TERMINATE_CONNECTION.response

Figure 3.4: Connection Provision and Data Delivery Services

communication parameters are exchanged like basic CID and primary CID (explained in later part) to initiate further data communication and register an ST as an authenticated terminal. Network initialization along with message exchanges are shown in Figure 3.2. (2) *Connection Management* procedure includes messages in form of dynamic service requests like creation, change and deletion of a connection and associating this connection with various service flows. (3) *Data Transport* handles the data transfers between S and ST with functionality of packing/unpacking MAC SDUs/PDUs respectively. Figure 3.4. shows the exchange of the messages above types.

3.2.2 Addressing and Connection Identification

In a setup an ST is uniquely addressed by a 48 bit MAC address at medium access layer which is used during registration for ST with system S. It is also used as part of the authentication process by which the BS and ST verify each other's identity.

Connections established between S and ST are identified by 16 bit Connection Identifier (CID) within each downlink and uplink. At ST initialization, two management connections, basic CID and primary CID, are establish between BS and ST. Basic CID is used by BS MAC and ST MAC to exchange shorter urgent MAC management messages (e.g. ranging). Primary CID shared by BS MAC and ST MAC is used to exchange comparatively longer delay tolerant MAC management messages (e.g. creation of data connections). Whenever a request for data connection is made (from higher layer), a data CID is assigned to that connection by S. The format of the CID is shown in Figure 3.5 [1].

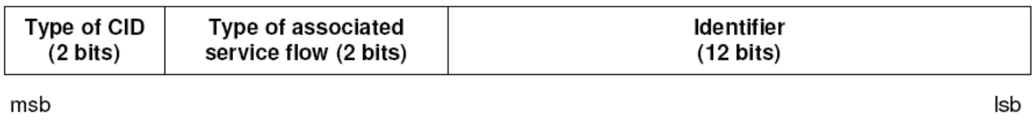


Figure 3.5: CID format

Code	Type of CID
00	Basic CID
01	Primary CID
10	Data CID
11	Data CID

Table 3.1: Connection Type codes

Code	Type of Service Flow
00	UGS
01	rtPS
10	nrtPS
11	BE

Table 3.2: Service Flow codes

3.2.3 Framing Structures

3.2.3.1 Basic MAC PDU

A MAC PDU starts with a fixed length generic MAC header followed by payload followed by CRC (Cyclic Redundancy Check). The payload information is variable length and can hold zero or more sub-headers and zero or more MAC SDUs (Service Data Units). MAC

PDU is bounded by maximum size payload accepted by WiFiRE in which fragmentation is not supported. MAC PDU format is shown in Figure 3.6 [1].

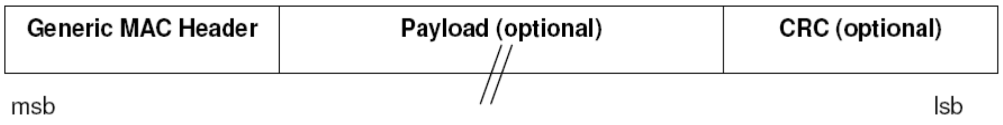


Figure 3.6: MAC PDU format

3.2.3.2 MAC Headers

MAC headers are defined in following formats:

1. *Generic MAC Header* consist of *HT* field for header type and is set to 0 for generic header, *Len* field of 15 bits to represent the length of whole MAC PDU including the header length, *Type* field 1 byte long defines the type of message being contained by PDU (such as Ranging, Registration, Dynamic Service and data payload). Further details for specific codes are given in [1]. Generic MAC header format is shown in Figure 3.7.

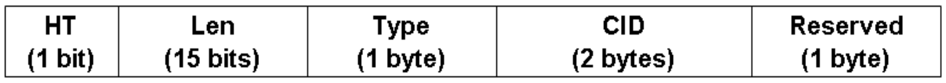


Figure 3.7: Generic MAC header

2. *Beacon Header* is comparatively small in size and transmitted whenever a beacon is generated at BS. It has *HT* field for header type set to 1 for beacon header followed by *Len* field of 15 bits to represent the length of MAC PDU. Figure 3.8 gives the notion of beacon header.

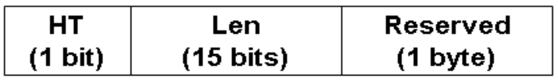


Figure 3.8: Beacon header

One byte is kept reserved for future use in both headers.

3.3 Bandwidth Grant Mechanism

3.3.1 Scheduling Services

In order to support the QoS for different services by scheduling the uplink access opportunity, different scheduling services corresponding to uplink scheduler policy can be categorized as:

- *Unsolicited Grant Service (UGS):* UGS is designed to support real-time service flows that generate fixed size data packets on a periodic basis, such as Voice over IP. The service offers fixed size unsolicited data grants (transmission opportunities) on a periodic basis. BS allocates fixed sized grants to the UGS at periodic intervals without any explicit request from SS. The BS can dynamically allocate additional bandwidth to the SS when the backlog transmission queue is formed.

- *Real Time Polling Service (rtPS):* The rtPS is designed to support real-time service flows that generate variable size data packets on periodic basis, such as MPEG video. The BS provides periodic dedicated request opportunities for ST to satisfy flow's demand comparatively more request overhead than UGS. ST is prohibited using any contention request opportunities and is allowed to use only unicast requests issued by BS.

- *Non-real time Polling Service (nrtPS):* The nrtPS is designed to support non-real-time service flows that require variable size data grants on a regular basis, such as high bandwidth FTP. The nrtPS connections uses random access transmit opportunities for sending bandwidth request.

- *Best Effort (BE):* The BE service flow is designed to support data streams for which no minimum transmission rate is required. The ST uses contention and piggyback requests opportunities for BE service flows like HTTP. Unlike rtPS, ST does not receive periodic polls or periodic data grants from BS.

3.3.2 Bandwidth Allocation and Request Mechanism

STs use to request bandwidth requirement to BS for the need of upstream bandwidth allocation. For receiving grant of bandwidth requested, it can operate in three modes:

- *Grant per Connection (GPC):* The decision of allocating the bandwidth depends on the request of type of connection from ST. Mostly suitable for few numbers of connections per ST.
- *Grant per Subscriber Terminal mode (GPST):* BS grants bandwidth to STs according to the current number of requests made, ST has to take care of the QoS among its connections and is responsible for division of the bandwidth among them (maintaining QoS and fairness). The size of UL-MAP is comparatively small and all subsequent scheduling decision are taken by the ST. Complexity and processing time at BS is reduced in this case.
- *Grant per Service Flow (GPSF):* It is an intermediate between GPS and GPST in which bandwidth is collectively granted to all the connections of a service flow type to an ST. This reduces the overhead of sending detailed UL-MAP and complexity of scheduler at ST.

Chapter 4

Implementation of WiFiRE

This chapter mainly discusses design and implementation of the protocol, components needed for the protocol and describes the problems and solutions related to handling basic network protocols and connection classification.

4.1 Implementation Details

Implementation of WiFiRE protocol begins with designing the implementation model from the actual scenario, with a successful emulation over LAN it get refined to perform nearly similar to that of actual MAC layer functionality.

4.1.1 Real System Components

WiFiRE project is collaborative project which is distributed among multiple groups of people at different locations. Figure 4.1 well describes the actual scenario of the project and corresponding responsibilities in abstract form at System 'S'[1]. It comprises of design and fabrication of PHY chip set and implementation of MAC layer above it. Here S and BS can be interchangeably used for centralized operating component.

WiFiRE mechanism starts just above the ethernet of system S, it captures the packet from the layer above embeds it into ethernet frame and delivers to the layer 2 device (designed by IITM). The device reads the packet, fetches the WiFiRE MPDU and maps it to the corresponding interface through CIDs and finally transmits over the wireless link. Layer 2 device is still in the process of designing and fabrication so anticipating the working of the device and implementing the model become more complex. It might come up with some unexpected results. So decision has been made to merge the functionality of

[1]Similar diagram also qualifies for ST

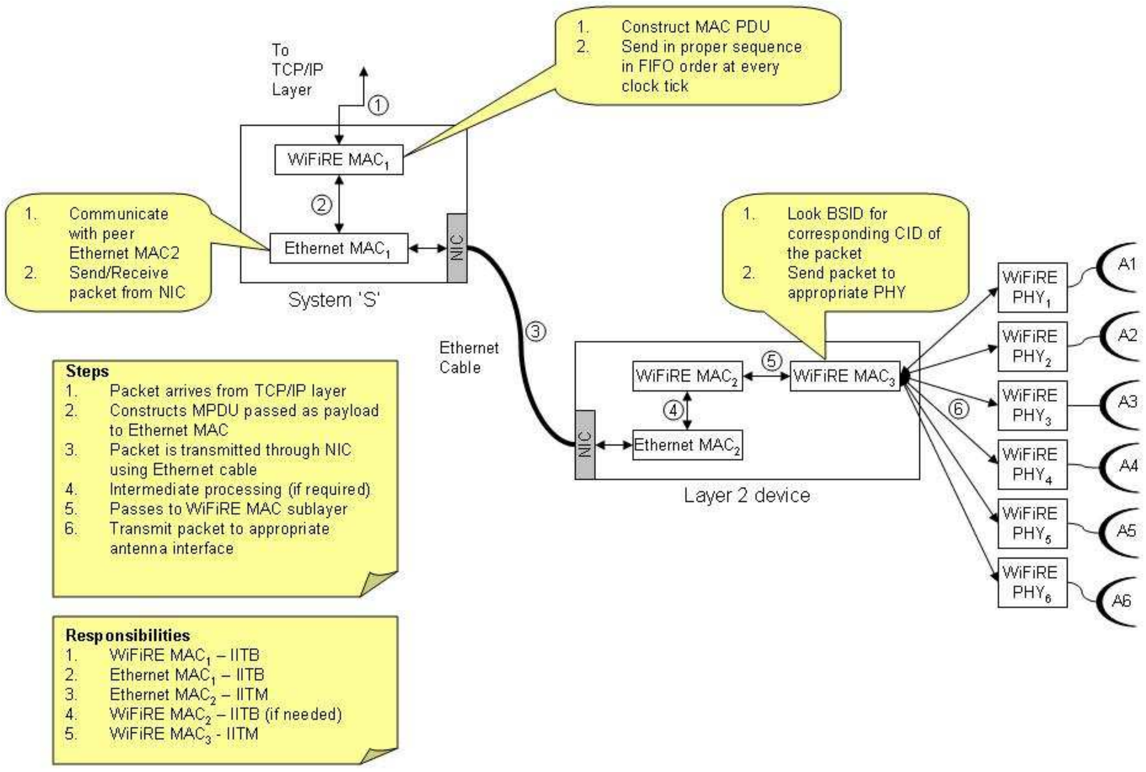

To TCP/IP Layer
1
1. Construct MAC PDU
2. Send in proper sequence in FIFO order at every clock tick
WiFiRE MAC$_1$
2
1. Communicate with peer Ethernet MAC2
2. Send/Receive packet from NIC
Ethernet MAC$_1$
NIC
System 'S'
1. Look BSID for corresponding CID of the packet
2. Send packet to appropriate PHY
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
Steps
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
Responsibilities
1. WiFiRE MAC$_1$ – IITB
2. Ethernet MAC$_1$ – IITB
3. Ethernet MAC$_2$ – IITM
4. WiFiRE MAC$_2$ – IITB (if needed)
5. WiFiRE MAC$_3$ - IITM

Figure 4.1: Real System Components

the device to the MAC implementation so as to run a successful demo. We have made our own test bed, a set of connected debian based machines, on which currently our protocol is being rigorously tested and executed to perform WiFiRE MAC layer functionalities. Figure 4.2 shows various components involved in the test bed.

The test bed includes a BS with 2 NICs, an ST with 2 NICs, a Server which acts as ftp, proxy web server etc. for the clients and multiple clients connected to ST through a ethernet hub. The doted lined blocks are the components which are planned to be bypassed for this implementation. Instead BS and ST are connected through a cross cable for collision free communication of constant propagation delay. The major components like BS and ST can be zoomed in through implementation of view. Figure 4.3 and figure 4.4 shows significantly working components of BS and ST respectively.

Here we assume that all the entities involved in WiFiRE architecture are in same subnet. Same subnet concept helps Client not to worry about the WiFiRE hop of ST and BS, as it can directly sent connection requests (HTTP, FTP, TELNET etc) to proxy server which is easy to handle at ST and further on BS. Here BS and ST are having at least 2 NICs with same subnet IPs, which leaves IP table with multiple entry for

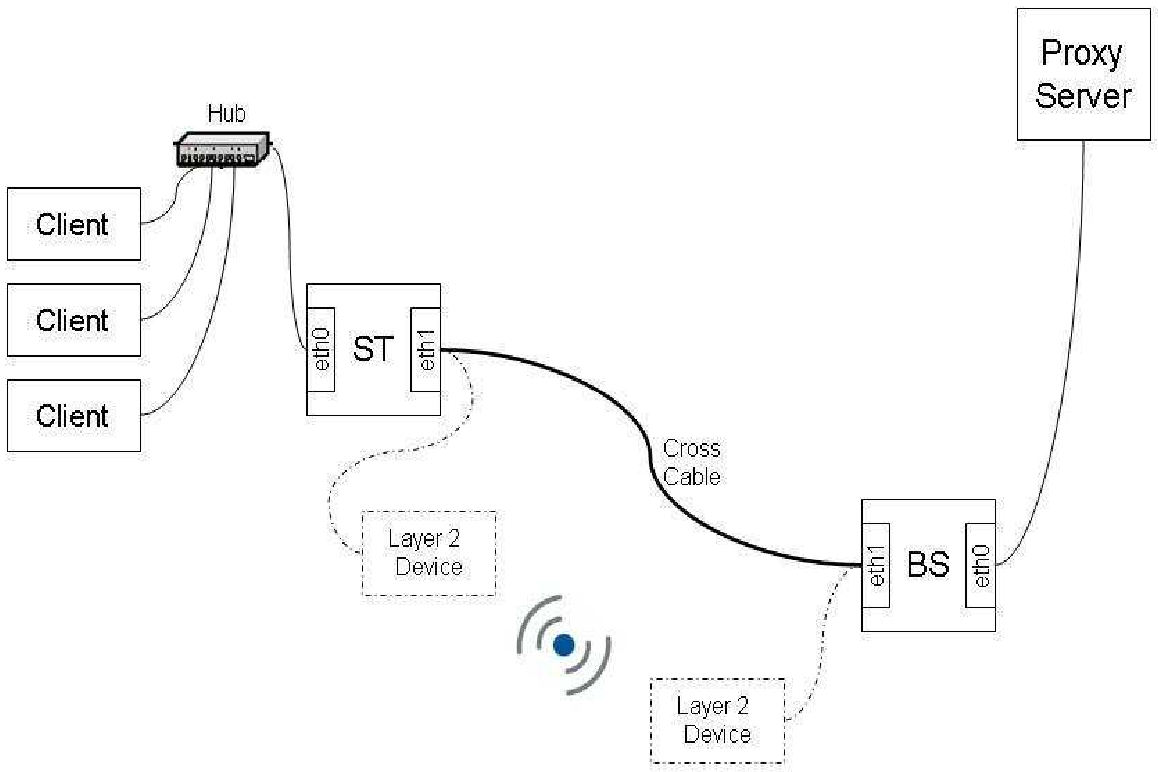


Figure 4.2: WiFiRE Test bed

same destination network address. Behavior for basic communication protocols like ARP, ICMP etc. changes as they depend on the metric value in the IP table maintained. To update ARP cache entries, Client or Server do transfer ARP messages to their virtually neighboring machine. Running of such protocols between client and server are avoided, but a provision has been made to forward such frames through designated CID. (All 1s in 2 byte CID value).

4.1.2 Assumptions

Our assumptions for this implementation are:

- All the machines involving in the test bed shares same subnet IP.
- IP addresses and ARP cache entries are assumed to be statically assigned.
- Phenomenon of ranging is superficially done as this does not involve actual wireless link.
- No special authentication method for upcoming ST is being deployed on BS.

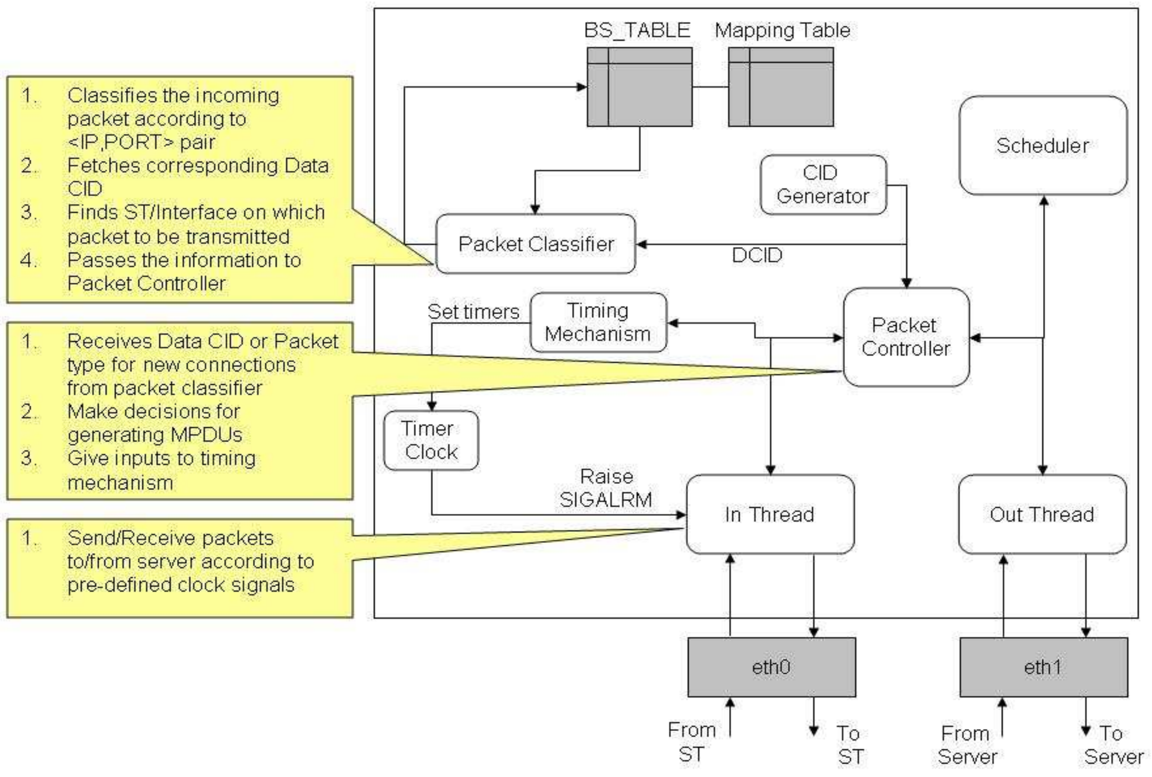

BS_TABLE
Mapping Table
1. Classifies the incoming packet according to <IP,PORT> pair
2. Fetches corresponding Data CID
3. Finds ST/Interface on which packet to be transmitted
4. Passes the information to Packet Controller
1. Receives Data CID or Packet type for new connections from packet classifier
2. Make decisions for generating MPDUs
3. Give inputs to timing mechanism
1. Send/Receive packets to/from server according to pre-defined clock signals
Scheduler
CID Generator
Packet Classifier
DCID
Set timers
Timing Mechanism
Packet Controller
Timer Clock
Raise SIGALRM
In Thread
Out Thread
eth0
eth1
From ST
To ST
From Server
To Server

Figure 4.3: Block diagram for BS

- Bandwidth alteration request is currently not handled. This involves creating and processing of `DSC_REQ` and `DSC_RSP`.

- Client is unaware of the service provider and underlying MAC layer protocol between BS and ST. It also does not enables any QoS mechanism explicitly.

- Clients are allowed to access services only when ST is in registered (REGD) state.

4.1.3 LAN Emulation

WiFiRE protocol is emulated over LAN using C sockets. A virtual WiFiRE MAC is made working over a TCP sockets where actual WiFiRE MAC PDUs are created, processed and executed in the traditional manner. Emulation is done using monotonous service flows which helped us to understand working of WiFiRE protocol. So it become comparatively easy to handle hurdles before actually going into implementation.

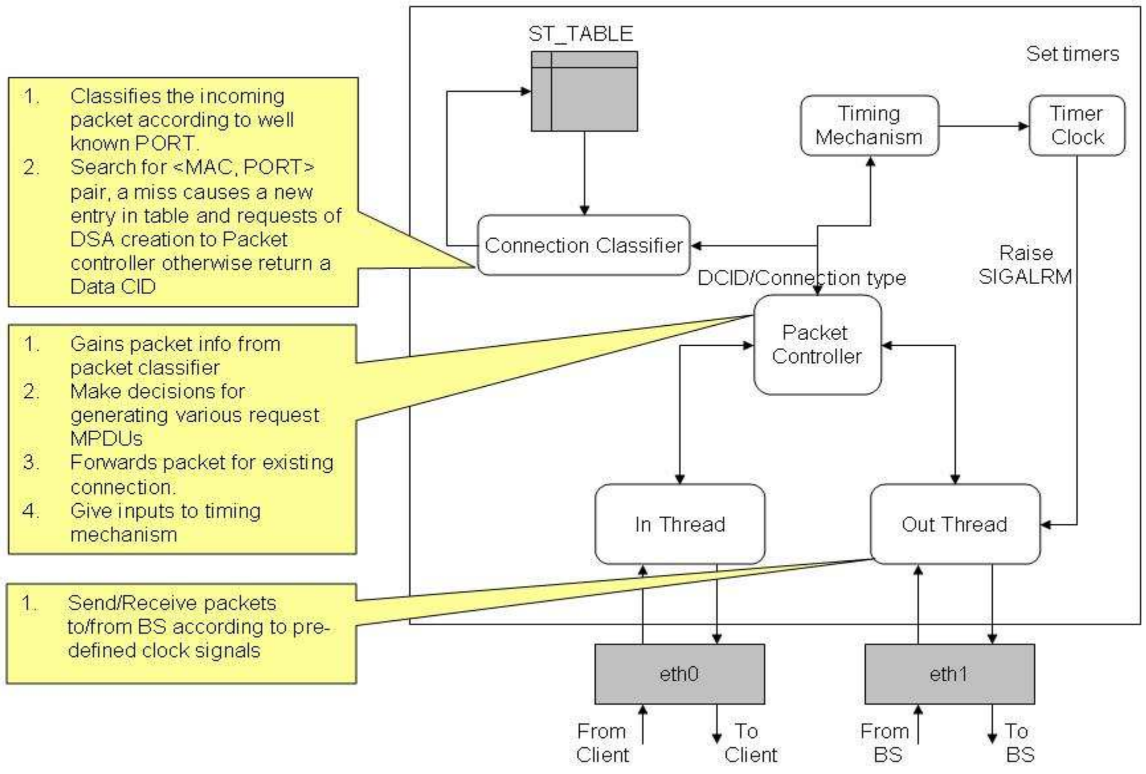

1. Classifies the incoming packet according to well known PORT.
2. Search for <MAC, PORT> pair, a miss causes a new entry in table and requests of DSA creation to Packet controller otherwise return a Data CID
1. Gains packet info from packet classifier
2. Make decisions for generating various request MPDUs
3. Forwards packet for existing connection.
4. Give inputs to timing mechanism
1. Send/Receive packets to/from BS according to pre-defined clock signals
ST_TABLE
Set timers
Timing Mechanism
Timer Clock
Connection Classifier
DCID/Connection type
Raise SIGALRM
Packet Controller
In Thread
Out Thread
eth0
eth1
From Client
To Client
From BS
To BS

Figure 4.4: Block diagram for ST

4.1.4 Why Sockets and not Kernel

The implementation model emulated over LAN is reexamined and refined to perform according to the protocol specification. It uses `PF_PACKET` family `RAW` socket, to communicate with the peer application, to send actual WiFiRE MPDUs over PHY. Following are the reason behind the implementation on sockets instead of kernel:

- A user level communication component which provides direct access to low level communication mechanisms bypasses operating systems in critical paths of communication.
- Overheads of kernel traps and memory copies along with various dependencies between user space and kernel space are avoided, a user level communication system may deliver higher performance than a kernel level communication system.
- To accomplish the implementation of volatile design of protocol model into working example in limited span of time.
- It became easier to make frequent desirable changes in the implementation with

quick debugging.

4.1.5 Data structures and Operations

To decrease the processing time and provide the functionality of absent hardware static char buffers are used to hold and process ethernet packets. Lookup tables are maintained both at BS and ST using array of structures which contain MAC address (at ST) or IP address (at BS) which shares same prefixes. Sequential search with reserve match pattern helps in early rejection of the entry inside a table. For example, multiple NICs with same manufacture ID differs only in their device ID and multiple IP addresses of same subnet differs only in last octet (in our case) and searching any MAC/IP address inside a table using reverse match pattern makes it faster. All the comparisons are done using bitwise operators (instead of logical operators) to speed up all operations.

4.2 Protocol Execution

Protocol skeleton is based on C sockets (`PF_PACKET RAW`), signal handling and multi threading. C sockets of family `PF_PACKET` gives the feel of working on wireless interface as it does not require any binding address and listening port. Signal handling is helpful in time synchronization and handling abrupt behavior of the protocol modules. And finally multiple threads are useful in parallel processing of `IN_THREAD`s and `OUT_THREAD`s which captures/injects data packets on network interfaces.

4.2.1 Flow of Packet

Initially BS comes up and it starts executing its routine procedure of beacon broadcast which is handled by Packet Controller (refer to figure 4.3). Whenever an ST comes up, its in IDLE state (refer figure 4.5), and starts listening for beacon. After acquiring beacon it changes it states to BCNR and extract its timer ticks according to the slots allocated to it or in contentions slots.

Packet Controller forwards the beacon message always to the *Timing Mechanism* at ST which generates sequence of time (a cumulative figures) from DL/UL MAP constructed by *Scheduler*. These sequence is fed to Clock timer to raise `SIGALRM` at the beginning of designated slots. This helps ST to wakes up or becomes ready at required slot intervals. Here due to absence of real hardware clock, software timers are used in implementation which guarantees precision till milliseconds.

The implementation does not perform real ranging procedure as the test bed uses ethernet cable where is propagation delay is not variable. Here ranging is done mere to assign and transfer Basic and Primary CIDs. After ranging procedure ST finally get registered and remains in REGD state. ST is now ready to serve its client's requests, initially which are not entertained. Description of ST states can be referred from Appendix table B.2.

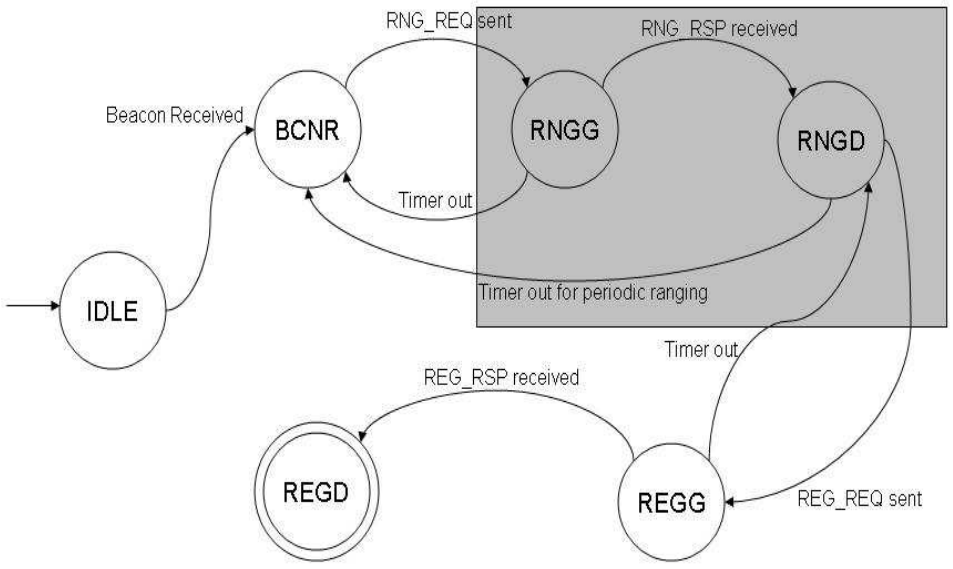


Figure 4.5: State diagram for ST

Client are now allowed to access services like web, ftp, telnet, VoIP etc. provided by the server. Whenever clients try to access any service the initial request packet is captured by ST and handed over to *Connection Classifier*, which in turns recognizes the packet and categorize it into UGS, rtPS, nrtPS and BE service flows. A new connection request from the client qualifies an entry in the ST_TABLE (a data structure used to keep track of existing active service flows) and DSA request to BS. Data CID is assigned in response from BS and table entry is made complete. ST_TABLE holds information about clients MAC address and Data CID. Here we are removing ethernet header to preserve bytes as payload greater than that of MTU (1500 for ethernet) of network interface fails to send/receive bytes from ethernet link. Further incoming packets to ST which belongs to an active connection are forwarded to BS in designated slots with appropriate WiFiRE headers.

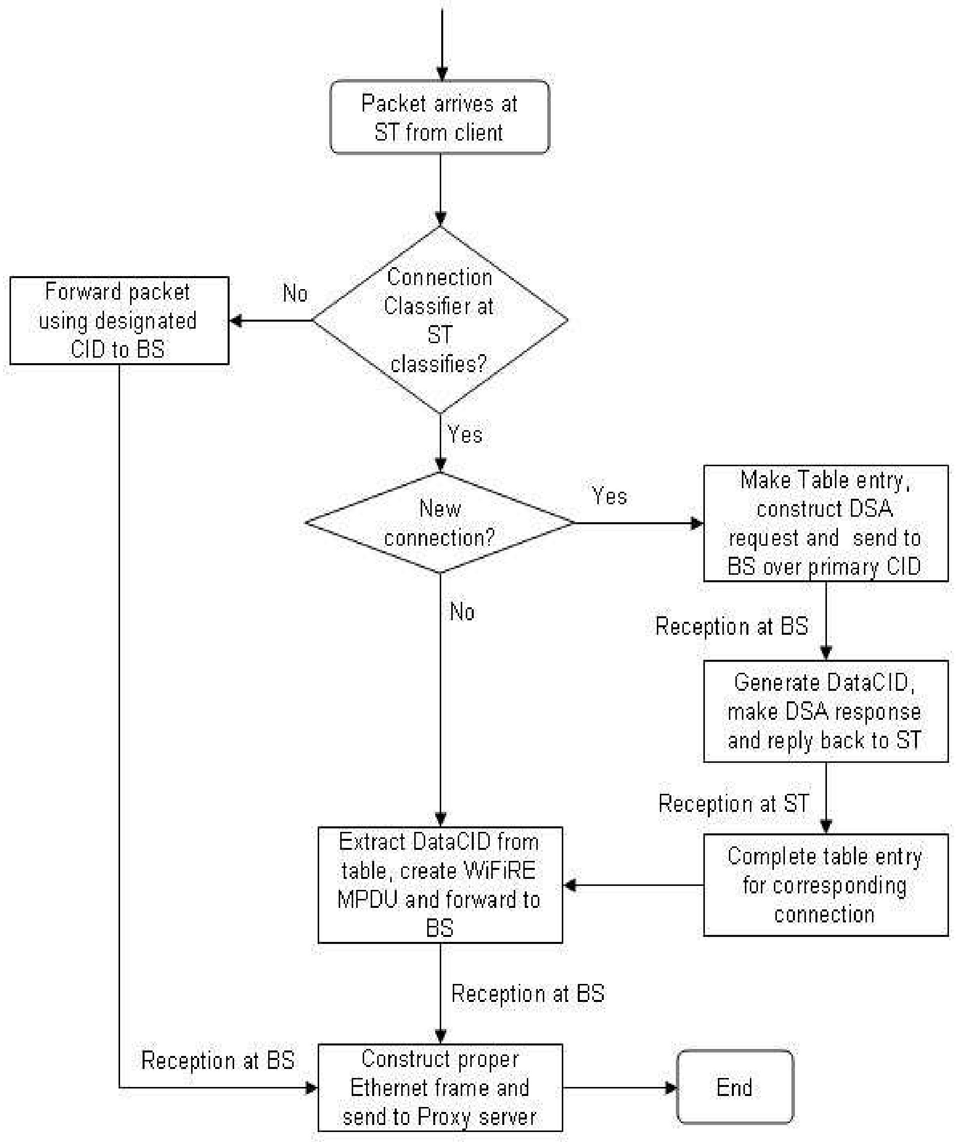

Packet arrives at ST from client
Forward packet using designated CID to BS
No
Connection Classifier at ST classifies?
Yes
New connection?
Yes
Make Table entry, construct DSA request and send to BS over primary CID
No
Reception at BS
Generate DataCID, make DSA response and reply back to ST
Reception at ST
Extract DataCID from table, create WiFiRE MPDU and forward to BS
Complete table entry for corresponding connection
Reception at BS
Reception at BS
Construct proper Ethernet frame and send to Proxy server
End

Figure 4.6: Packet flow: Client request to Server

BS forwards all packets by making an ethernet frame from WiFiRE MPDU and placing source MAC address of it own and destination MAC address of server (proxy). All control packets are sent over primary CID and every DSA request received at BS makes an entry in BS_TABLE (<IP,DATA CID>pair) with corresponding entry in mapping table (<DATA CID, INTERFACE ID>pair, interface id like `eth3` corresponds to some sector). Both tables can be merged into one but for future use, they are kept parted. BS sends back DSA response and assigns a Data CID to the new connection. All new CIDs (primary and data)

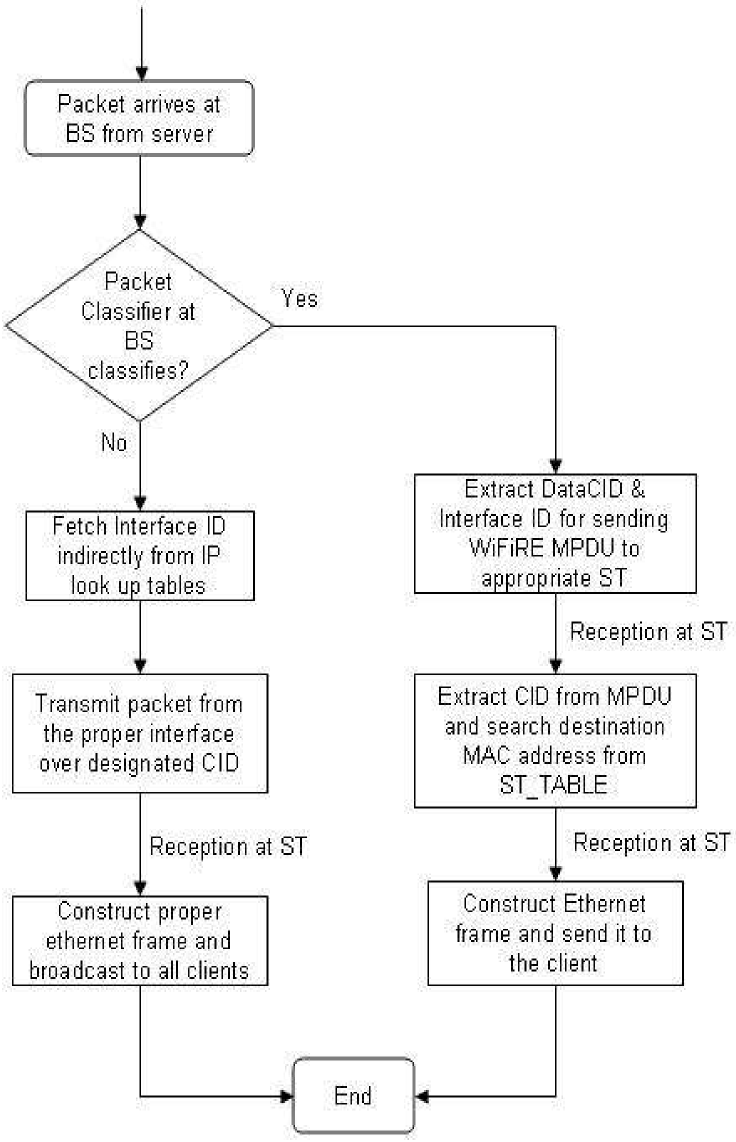

Packet arrives at BS from server
Packet Classifier at BS classifies?
Yes
No
Fetch Interface ID indirectly from IP look up tables
Extract DataCID & Interface ID for sending WiFiRE MPDU to appropriate ST
Reception at ST
Transmit packet from the proper interface over designated CID
Extract CID from MPDU and search destination MAC address from ST_TABLE
Reception at ST
Reception at ST
Construct proper ethernet frame and broadcast to all clients
Construct Ethernet frame and send it to the client
End

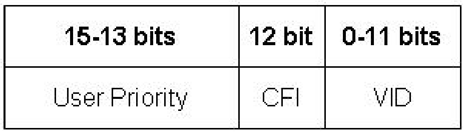
Figure 4.7: Packet flow: Server response to Client

are given by *CID generator*. A response from proxy server executes a IP lookup procedure by the *Packet Classifier* to fetch corresponding Data CID. *Packet Controller* then creates appropriate header to forward the response to the ST. Snapshots of ST_TABLE and BS_TABLE are shown in appendix table B.3 and table B.4.

ST now extracts the Data CID and looks up for the corresponding MAC address in ST_TABLE (entry made while creating new connection) and reform normal ethernet frame placing destination address as fetched MAC address and source address as ST's MAC address. In this way a client receives proper response from the above mechanism. Figure 4.6 and figure 4.7 describes the flow of packet from client to server (proxy) and back to client.

4.2.2 IEEE 802.1Q feature

Here we believe on client's innocence of not being enabling any QoS parameter explicitly in their own machines. A machine with QoS parameter enabled start communicating with an ethernet frame with extended MAC header which comprises of type field as 802.1Q (`0x8100`) and an addition 2 bytes revealing information about priority. IEEE 802.1Q (also known as VLAN Tagging) used to develop a mechanism to allow multiple bridged networks to transparently share the same physical network link without leakage of information between networks (i.e. trunking). 802.1Q does not actually encapsulate the original frame. Instead, for Ethernet frames using Ethernet II framing (Ethernet v2 framing interprets the 2-octet field following the destination and source addresses as an EtherType that immediately identifies an upper-layer protocol.), it sets the EtherType value in the Ethernet header to Tag Protocol ID (TPID) `0x8100`, identifying this frame as an 802.1Q frame. It then inserts an extra two-bytes of Tag Control Information (TCI) after the TPID, followed by another two bytes containing the frame's original EtherType. Together the four bytes of TPID and TCI are called the VLAN Tag. The format of TCI is shown in 4.8



15-13 bits	12 bit	0-11 bits
User Priority	CFI	VID

Figure 4.8: Tag Control Information

- **User priority**: a 3-bit field storing the priority level for the frame.
- **Canonical format indicator (CFI)**: a 1-bit indicator that is always set to zero for Ethernet switches. CFI is used for compatibility between Ethernet and Token Ring networks.
- **VLAN ID (VID)**: a 12-bit field specifying the VLAN to which the frame belongs.

It become easier to map 8 priority levels (3 out of 16 bits) into four service flows without actually looking into the contents of ethernet payload. But we assume client's terminal to be dump and are not smart enough to engage in QoS functionality.

4.3 Modifications/Additions in WiFiRE release draft

Following are some of the modifications/addition should be made in the WiFiRE draft:

- Length field in the WiFiRE header should be increased from existing 7 to 15 bits to address 2318 bytes of packet length.
- STID is considered to be MAC address of ST so the field length holding STID should be from from 4 byte to 6 bytes.
- Add designated CIDs to carry non classified packets which may contain useful information of other basic network communication protocol. Assign `11` in Type of CID field for designated CID instead for redundant data CID.
- Include Flow diagrams involving registration and ranging at BS.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The WiFiRE implementation over the test bed comes up with satisfactory results. WiFiRE protocol performs up to the mark as designed for communication between BS and ST. WiFiRE packets can be captured over the link between BS and ST verifies the factual working of the protocol. Probably an ethereal WiFiRE patch could made our task easier. Actual hardware chip set (layer 2 devices) would have made a difference to the implementation but merging its functionality to corresponding entities (BS and ST) pools up the gap and gave full control over the link on obvious cost of speed.

To perform better in terms of packet delivery and time synchronization we propose some extensions in the future work section.

5.2 Future Work

This project can be enhanced by performing following tasks:

- Use the implemented MAC layer code with slight changes for multiple NICs at BS (for multiple sector execution).
- Fragment the appropriate functionality of layer 2 device from the protocol and embed it into actual fabricated chip sets will speed up the table look up and decreases complexities at BS and ST.
- Use VoIP header compression techniques to reduces the size of the packet and decrease in number of slots required for sending it.
- Implement hardware clock for ticking signals at more precision.

- Add more features to protocol, like complete ranging procedure (for wireless link), DSC mechanism etc. to make it flexible and fault tolerant.
- From operating system's point of view the protocol still executes in user space and it requires efforts to implement it as a kernel patch or a device driver.

A combination of above extensions and with possible improvement in classifier can make this protocol work really fast, flexible and fault tolerant.

Chapter 6

Appendix

Appendix A

Figures

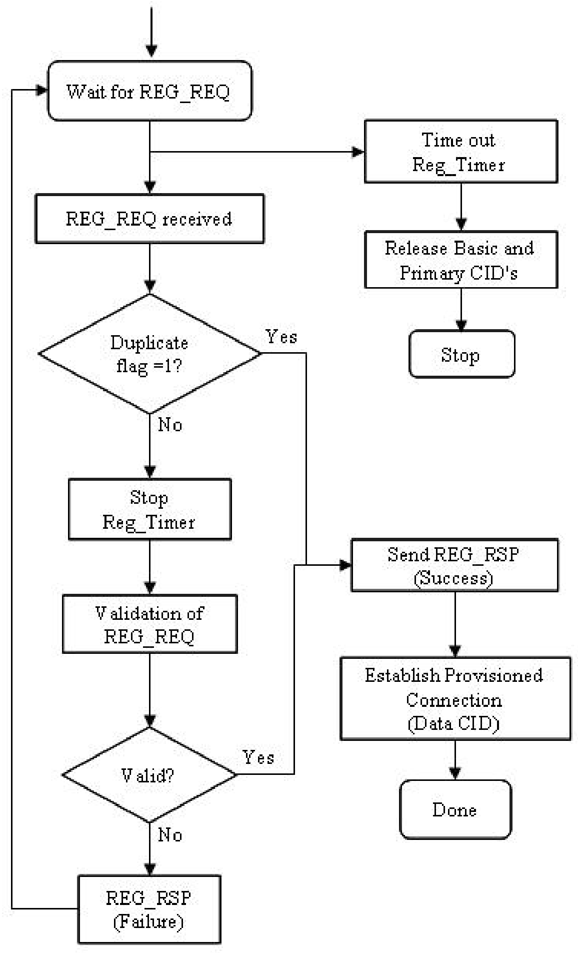


Figure A.1: Flow Diagram: Registration at BS

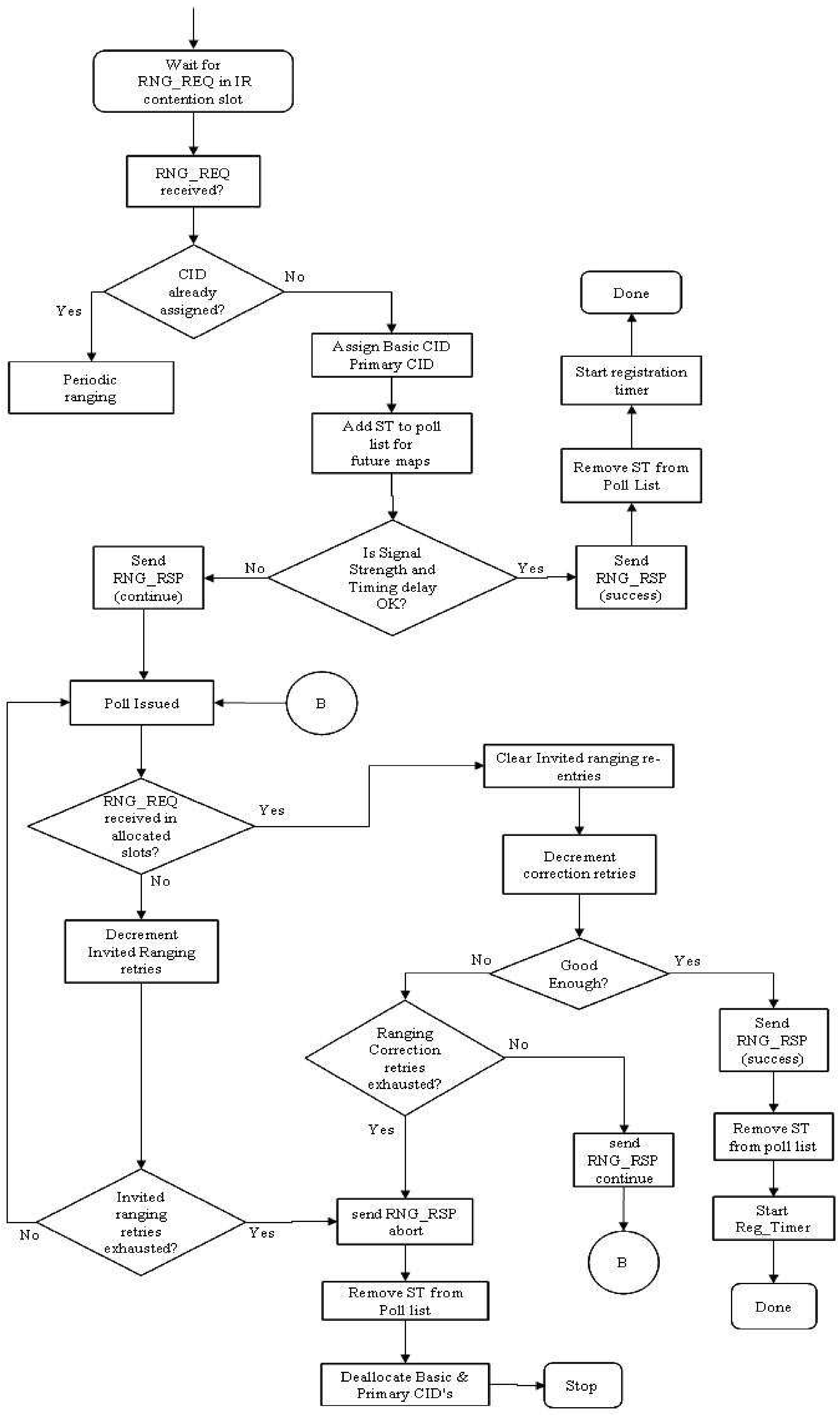
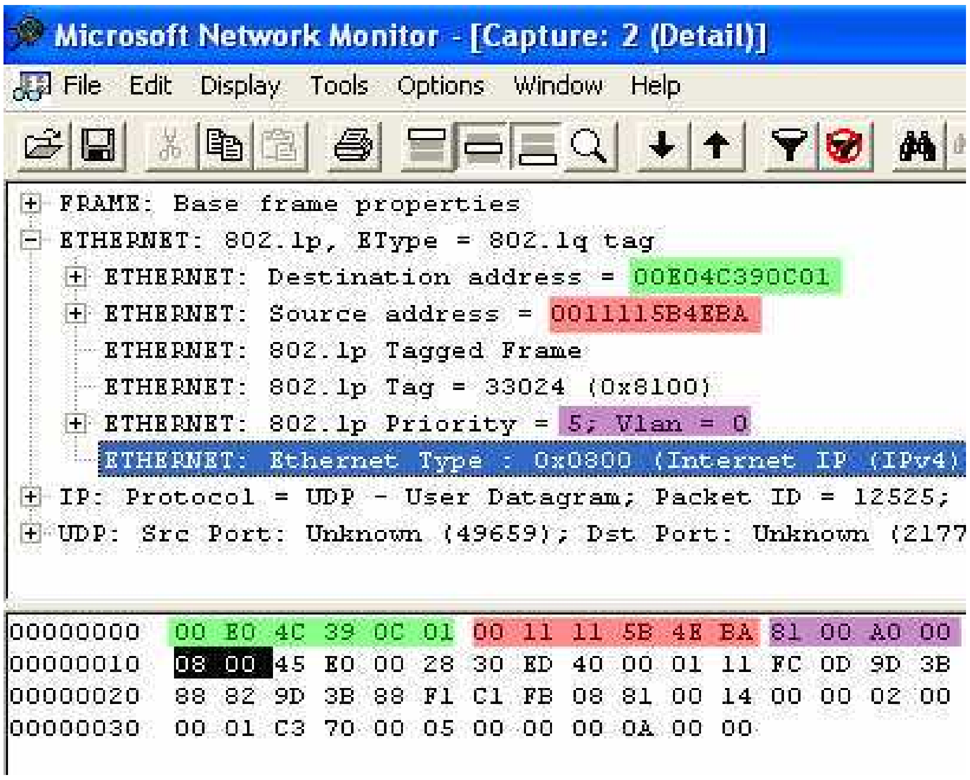

Wait for RNG_REQ in IR contention slot
RNG_REQ received?
CID already assigned?
Yes
No
Periodic ranging
Assign Basic CID Primary CID
Add ST to poll list for future maps
Is Signal Strength and Timing delay OK?
No
Yes
Send RNG_RSP (continue)
Send RNG_RSP (success)
Remove ST from Poll List
Start registration timer
Done
Poll Issued
B
RNG_REQ received in allocated slots?
Yes
No
Clear Invited ranging re-entries
Decrement correction retries
Good Enough?
No
Yes
Decrement Invited Ranging retries
Ranging Correction retries exhausted?
No
Yes
send RNG_RSP continue
B
Send RNG_RSP (success)
Remove ST from poll list
Start Reg_Timer
Done
Invited ranging retries exhausted?
No
Yes
send RNG_RSP abort
Remove ST from Poll list
Deallocate Basic & Primary CID's
Stop

Figure A.2: Flow Diagram: Ranging at BS


Microsoft Network Monitor - [Capture: 2 (Detail)]
File Edit Display Tools Options Window Help

```
+ FRAME: Base frame properties
- ETHERNET: 802.1p, EType = 802.1q tag
   + ETHERNET: Destination address = 00E04C390C01
   + ETHERNET: Source address = 0011115B4EBA
     ETHERNET: 802.1p Tagged Frame
     ETHERNET: 802.1p Tag = 33024 (0x8100)
   + ETHERNET: 802.1p Priority = 5; Vlan = 0
     ETHERNET: Ethernet Type : 0x0800 (Internet IP (IPv4)
+ IP: Protocol = UDP - User Datagram; Packet ID = 12525;
+ UDP: Src Port: Unknown (49659); Dst Port: Unknown (2177

00000000  00 E0 4C 39 0C 01 00 11 11 5B 4E BA 81 00 A0 00
00000010  08 00 45 E0 00 28 30 ED 40 00 01 11 FC 0D 9D 3B
00000020  88 82 9D 3B 88 F1 C1 FB 08 81 00 14 00 00 02 00
00000030  00 01 C3 70 00 05 00 00 00 0A 00 00
```

Figure A.3: Snapshot of 802.1Q tagged ethernet frame

Appendix B

Tables

Code	**Type of CID**
`00`	Basic CID
`01`	Primary CID
`10`	Data CID
`11`	Designated CID

Table B.1: Type of Connection codes

State	**Description**
$IDLE$	Idle state
$BCNR$	Beacon Received state
$RNGG$	Ranging state‘
$RNGD$	Ranged state
$REGG$	Registering state
$REGD$	Registered state

Table B.2: States of ST

Data CID	**MAC Address**	**Type**
1000 0000 0000 0001	00:BF:0D:24:46:1F	*UGS*
1000 0000 0000 0010	00:D2:A3:16:22:3C	*UGS*
1011 0000 0000 1101	00:AD:13:87:C2:31	*BE*
1010 0000 0000 0001	00:BF:0D:24:46:1F	*nrtPS*
1011 0000 0000 0101	00:D2:A3:16:22:3C	*BE*
1000 0000 0000 1001	00:AD:13:87:C2:31	*UGS*
.	.	.
.	.	.
.	.	.
.	.	.

Table B.3: Snapshot of lookup table at ST

IP Address	**Data CID**	**Type**
172.16.1.6	1000 0000 0000 0001	*UGS*
172.16.1.15	1000 0000 0000 0010	*UGS*
172.16.1.7	1011 0000 0000 1101	*BE*
172.16.1.6	1010 0000 0000 0001	*nrtPS*
172.16.1.15	1011 0000 0000 0101	*BE*
172.16.1.7	1000 0000 0000 1001	*UGS*
.	.	.
.	.	.
.	.	.
.	.	.

Table B.4: Snapshot of lookup table at BS

User Priority	Acronyms	Traffic Type
1	BK	Background
2	-	Spare
0	BE	Best Effort
3	EE	Excellent Effort
4	BK	Controlled Load
5	VI	Video, < 100 ms latency and jitter
6	VO	Voice, < 10 ms latency and jitter
7	NC	Network Control

Table B.5: Traffic type acronyms for 802.1Q priority levels

Debug Level	Purpose
0	No Output
1	Event And High-Level Information Messages
2	Track The Flow Of Command Processing
3	For Inner Loops, Table Traversals, Etc
4	I/O Debug - Mover Messages Traces
5	Trace-Level Debug
6	Redundant Information

Table B.6: Debug Levels

In order to keep track on the execution of the protocols, DEBUG LEVELS has been introduced to get output information at user defined level of depth. Table B.6 defines the levels and their purpose of execution.

Bibliography

[1] WiFiRe: Medium Access Layer (MAC) and Physical Layer (PHY) Specification Center of Excellence for Wireless Technology (CEWiT) June 2006.

[2] IEEE 802.16. IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems 2002.

[3] Design and Evaluation of a new MAC Protocol for Long-Distance 802.11 Mesh Networks, Bhaskaran Raman and Kameswari Chebrolu, 11th Annual International Conference on Mobile Computing and Networking paper (MOBICOM), Aug/Sep 2005, Cologne, Germany

[4] GuoSong Chu; Deng Wang; Shunliang Mei A QoS architecture for the MAC protocol of IEEE 802.16 BWA system, *IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions,* pages 435- 439 vol.1, 29 June-1 July 2002.

[5] Broadband Wireless Internet Forum, Media Access Protocols: DOCSIS, Document Number WP-2_TG-1 Version 1.1, December 5, 2000.

[6] Wang, H.; Li, W.; Agrawal, D.P. Dynamic admission control and QoS for 802.16 wireless MAN. *Wireless Telecommunications Symposium, 2005* pages 60- 66, April 2005.

[7] Hawa, M.; Petr, D.W. Quality of service scheduling in cable and broadband wireless access systems, Tenth IEEE International Workshop on *Quality of Service.* pages 247- 255, 2002.

[8] LAN/MAN standards Committee, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEESA Standards Board, June 2003.

[9] W Richard Stevens UNIX Network Programming, Volume 1, Second Edition, Prentice Hall, 1998

[10] Inside the Linux Packet Filter: The packet's journey through the kernel, `http://delivery.acm.org/10.1145/520000/513092/5617s1.html`

[11] The Linux Socket Filter: Sniffing Bytes over the Network, in Linux Journal By Gianluca Insolvibile `http://www.linuxjournal.com` June 2001

[12] GNU C Library: Signal Handling, `http://www.cs.utah.edu/dept/old/texinfo/glibc-manual-0.02/library_21.html`

[13] Introduction To Unix Signals Programming, `http://users.actcom.co.il/ choo/lupg/tutorials`