

Design, Implementation, and Evaluation of new MAC Protocols for Long Distance 802.11 Networks

by

S Pavan Kumar

A thesis

submitted to the Indian Institute of Technology, Kanpur

in partial fulfillment of the

requirements for the degrees of

Bachelor of Technology and Master of Technology

in

Computer Science and Engineering



Kanpur, Uttar Pradesh, India, May 2006

©S Pavan Kumar, 2006

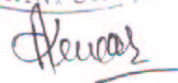
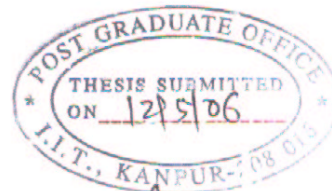
Certificate

This is to certify that the work contained in the thesis entitled "**Design, Implementation, and Evaluation of new MAC Protocols for Long Distance 802.11 Networks**", by *S Pavan Kumar* has been carried out under the supervision of *Dr. Bhaskaran Raman* and that this work has not been submitted elsewhere for a degree.

May, 2006



Dr. Bhaskaran Raman
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur-208016.



Abstract

802.11 MAC is based on *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) and was designed to be used in an indoor environment where a small number of wireless nodes share a common channel. It has been shown in *Digital Gangetic Plains* project that 802.11 hardware can be used beyond its intended use of WLANs by connecting long distance wireless links spread over tens of kilometers. These links have been used to operate voice (telephone through Voice over IP) and video (telemedicine through Video over IP) applications. But the existing MAC protocols like CSMA/CA that operate on 802.11 hardware do not work well in these outdoor wireless networks because of hidden node problem, huge *Round Trip Time* (RTT) and unnecessary contention.

In this thesis, we therefore motivate the need for design and development of new MAC protocols to operate on point-to-point and point-to-multipoint long distance links to obtain better performance and flexibility compared to CSMA/CA. This report describes the implementation of two new MAC protocols **2P** and **SRAWAN** that are suited for these long distance 802.11 networks. **SRAWAN** is designed from scratch. Even though the design of 2P exists, a significant challenge here is the implementation of these protocols on the off-the-shelf 802.11 Atheros hardware to preserve cost benefits. We have implemented **2P** and **SRAWAN** by exploiting the flexibilities in the *HAL* code and *Madwifi* driver of Atheros AR5212 chipsets. The report then presents experimental evaluation of **2P** and **SRAWAN**. The outdoor experimental results on a network of two links show that **2P** achieves significant performance (throughput) improvement over CSMA/CA (by more than 45%) for point-to-point links and **SRAWAN** outperforms CSMA/CA by more than 40% for point-to-multipoint long distance links. On larger networks, we expect an even higher throughput improvement. We expect the protocols and the implementation to be deployed on *Ashwini* network in Bhimavaram, Andhra Pradesh for providing internet connectivity to remote rural villages.

Acknowledgments

I take this opportunity to express my sincere gratitude towards my thesis supervisor Dr. Bhaskaran Raman for his invaluable guidance and suggestions which made my thesis successful. I would also like to thank him for providing me an opportunity to work with Zazu Networks on problems which can lead to direct impact in the society. I consider myself extremely fortunate to have had a chance to work under his supervision. It has been a very enlightening and enjoyable experience to work with him and I express heart-felt thanks to him. I would also like to thank Dr. Kameswari Chebrolu for her valuable suggestions in the course of my work. Last but not least, I would like to thank Bhaskar and Kameswari for all the gettogethers, fun and a memorable Goa trip.

I wish to thank all the faculty members of IIT Kanpur for the invaluable knowledge they have imparted me in most exciting and enjoyable way. I also wish to thank Ravi Puvvala, CEO of Zazu Networks and Pratik Sinha of Zazu Networks for providing me an opportunity to work with them and also for all the help with my thesis work. I extend my thanks to Media Lab Asia, Kanpur for providing all the equipment needed for experimenting my work.

I wish to thank Sukanto (who would come even on weekends to help me with my work), Anurag, Ramchand, Ranjit, Yogesh and A K Singh for helping me conduct the experiments outdoors. This acknowledgment would be incomplete without thanking my friend and thesis partner Puli Narasimha Reddy who has been very patient to bear with my annoying ideas and has supported and encouraged me throughout my work.

I am forever thankful to my parents and my sisters for their love and encouragement. I wish to thank Manider Pal Singh for his guidance and without his support I would not have made it into IIT.

I would like to conclude by extending my thanks to all my friends with whom I shared beautiful and memorable experiences during my stay at IIT Kanpur.

Contents

1	Introduction	1
2	2P: Design and Implementation	9
2.1	Architecture and Terminology	9
2.2	Achieving SynOp	10
2.3	2P operation on single link	12
2.4	2P operation on two links	14
2.5	Implementation	16
3	SRAWAN: Design and Implementation	20
3.1	Architecture	20
3.2	Functional Specification	21
3.2.1	Frame Structure	24
3.2.2	Ranging	26
3.2.3	Registration	27
3.2.4	Connection Formation	28
3.2.5	Authentication and Security	29
3.2.6	Packing	29
3.2.7	ARQ	29
3.2.8	Summary of the MAC Management Messages	30
3.2.9	Limitations of the scope of SRAWAN	31
3.3	Implementation	31
3.3.1	Scheduling	34
3.3.2	Implementation parameters and Throughput Estimation	35

4	Experiment Results	38
4.1	Software and Hardware	38
4.2	Results of disabling NAV and CCA	39
4.3	Performance evaluation of CSMA/CA and 2P on a point-to-point network	40
4.4	Performance analysis of 2P in SynTx and SynRx phases	45
4.5	Performance of CSMA/CA and SRAWAN on two links	46
4.6	Performance of CSMA/CA, SRAWAN, and 2P on single link	48
5	Conclusions	51
A	MAC PDU and Header Formats in SRAWAN	53
A.1	Generic MAC Header	54
A.1.1	Bandwidth Request Header	54
A.2	MAC Subheaders and Special Payloads	56
A.2.1	Grant Management sub header	57
A.2.2	Packing subheader	58
A.2.3	ARQ subheader	58
B	Transmit and Receive Calibration Plots	60
B.1	Transmit Power Calibration	60
B.2	Receive Power Calibration	62

List of Tables

4.1	TCP throughput for testing CCA and NAV	40
4.2	Signal levels	41
4.3	Performance of CSMA/CA on two p2p links operated on channels 1 and 6	42
4.4	Performance of CSMA/CA on two p2p links both operated on channel 1	43
4.5	Throughput results of 2P on two p2p links operated on channels 1 and 6	43
4.6	Throughput results of 2P on two p2p links operated on channel 6	43
4.7	Signal levels when two interfaces are mounted on same WRAP board	44
4.8	SynTx and SynRx throughput results	45
4.9	Throughput results of CSMA/CA on two links	48
4.10	Throughput results of SRAWAN on two links	48
4.11	Throughput results of CSMA/CA on single link	49
4.12	Throughput results of 2P on single link	49
4.13	Throughput results of SRAWAN on single link	49
A.1	Description of the fields in the generic MAC header	55
A.2	Description of Bandwidth Request header fields	56
A.3	Type Encodings	57
B.1	Tx calibration data	61

List of Figures

1.1	Hidden node in point-to-point scenario	2
1.2	Hidden node in point-to-multipoint scenario	3
1.3	SynRx, SynTx and Mix-Rx-Tx	4
1.4	Point-to-multipoint link	6
2.1	The two link point-to-point system used for 2P experimentation	10
2.2	2P state machine	13
2.3	Link synch/recovery process	14
2.4	Master Slave operation in 2P	17
3.1	A point-to-multipoint system	21
3.2	SRAWAN frame structure	22
3.3	FrameStructure	25
3.4	Time synchronization during Ranging	27
3.5	ARQ sub header	30
3.6	Frame structure showing various implementation parameters	35
4.1	Experiment setup to test NAV and CCA disabling	39
4.2	The two link point-to-point system used for 2P experimentation	41
4.3	Experimental setup for shared memory coordination scheme	45
4.4	Point-to-multipoint link with splitter	47
4.5	FBTop - Mohanpur point-to-point link	49
A.1	MAC PDU Format	53
A.2	Generic MAC header Format	54
A.3	Bandwidth Request Header	54
A.4	Grant Management sub-header for UGS connections	58

A.5	Grant Managment sub-header for rtPS, nrtPS or BE connections	58
A.6	Packing sub header	58
A.7	ARQ sub header	59
B.1	Experimental Setup for Calibrating Ubiquiti SR2 cards	61
B.2	Receive Calibration Plot	62

Chapter 1

Introduction

IEEE 802.11 [5] has been originally designed to connect multiple wireless nodes in the vicinity of an access point to wired LAN/Internet. But it has been used beyond its intended use of WLANs like in [8], [9]. Of specific interest in this thesis work are networks which involve point-to-point and point-to-multipoint long-distance links, envisioned for providing low cost Internet connectivity to rural villages.

70% of the Indian population lives in villages and many of these villages are deprived of basic amenities like education and medical facilities. One of the many reasons for this condition is that the villages are remote and are not connected to cities or towns. So if we can connect these villages to nearby cities or towns with the help of long distance wireless links, we can provide telemedicine, teleschool and telephone facilities through Voice over IP (VOIP) and Video over IP applications. One such attempt to alleviate the digital divide is the *Ashwini* network in Bhimavaram, Andhra Pradesh. The network is built by an NGO called *Byrraju Foundation* for providing telemedicine and telephone facilities to rural villages from district headquarters. The reasons for choosing 802.11 wireless technology are as follows.

- Inexpensive 802.11 hardware because of its mass production and interoperability
- Deregulation of 2.4 Ghz spectrum in India
- Deep penetration of 802.11 in industry and market

The DGP project [9] has shown that 802.11 hardware can be used for connecting links spread over tens of kilometers. But 802.11 MAC is based on CSMA/CA and was designed

to resolve contention in indoor environment where a small number of wireless nodes share a single channel. This contention based channel access mechanism is inefficient in long distance 802.11 WiFi networks. This is because the interference may not be between the adjacent nodes. The interference pattern differs among various MAC protocols and also depends on the kind of antenna used at various nodes in the network. Since the round trip time is huge (air propagation delay is the major contributor), there is a need for adaption of ack timeout and slot time to long distance point-to-point links [1].

Further, the case of a node with multiple point-to-point links or a single point-to-multipoint link is a classic case of hidden nodes. For example consider the Figure 1.1. All the wireless devices A, B, C and D are connected to highly directional parabolic grid antennae thereby forming 2 point-to-point links AB and CD. When D is transmitting to C, A does not know of this communication and may try to communicate with B. This might interfere with the communication at C if the transmit power of A is high. Another example in case of point-to-multipoint scenario is depicted in Figure 1.2. Wireless device A is connected to a sector antenna and wireless devices B and C are connected to parabolic grid antennae and are in the coverage of A. Now when B is transmitting to A, C does not know of this communication and may transmit to A which will lead to interference at A. Use of RTS/CTS to solve these problems is very inefficient because of the huge round trip (around $100\mu\text{s}$ for a 15 km link).

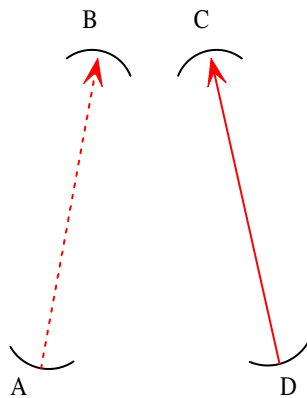


Figure 1.1: Hidden node in point-to-point scenario

Generic contention resolution mechanisms are not required for the network models in consideration since we are not dealing with arbitrary contention. Adding to this, all the nodes in the network are stationary. A time-division multiple access (TDMA) scheme is

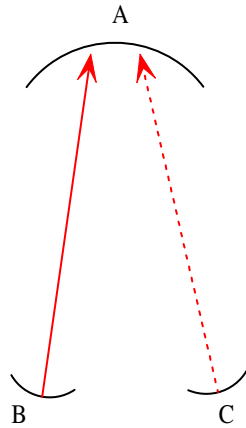


Figure 1.2: Hidden node in point-to-multipoint scenario

more appropriate. Apart from this, 802.11b/g defines eleven channels with a maximum of only 3 non-overlapping frequencies. So allocation of channels to all the links in the network may not be possible and many links will have to share a common channel. So new TDMA MAC protocols are needed which take these factors into account to efficiently utilize the scarce frequency spectrum and maximize the overall throughput of the network.

Background and Problem Statement

In the context of long distance WiFi mesh networks, prior work related to the MAC protocols are as follows. Sreekanth Garigala in his thesis [3] showed and experimentally proved the following. For a wireless node with 2 or more point-to-point links, it is possible to:

- simultaneously transmit on all the point-to-point links (SynTx)
- simultaneously receive on all the point-to-point links (SynRx)

But for SynTx/SynRx (also called as SynOp - Simultaneous Operation) to happen, the transmit power settings of all the interfaces on these point-to-point links have to be carefully set as described in [2]. It has also been experimentally shown that simultaneous transmission on some links and reception on some links cannot happen (Mix Rx-Tx). However CSMA/CA and 802.11 DCF do not allow SynOp as explained below.

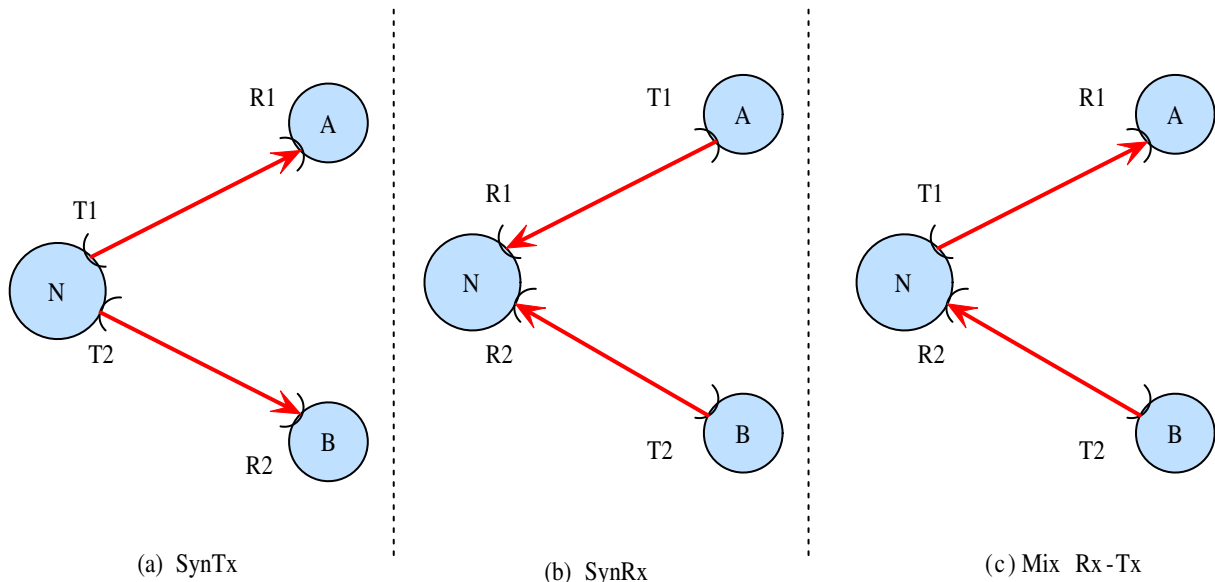


Figure 1.3: SynRx, SynTx and Mix-Rx-Tx

Consider the Figure 1.3. In case of SynRx, let's assume that R1 finishes reception of packet from T1 first. Then R1 will send an immediate MAC level ack to T1 after SIFS duration. This will lead to Mix-Rx-Tx situation which should not occur as reception at R2 will get corrupted. In case of SynTx, let's assume that T1 starts transmission first. Then T2 carrier senses the channel to be busy and backoffs according to exponential backoff scheme in 802.11 DCF. These problems are because of the side lobes of the external antennae.

Dr. Bhaskaran Raman and Dr. Kameswari Chebrolu designed 2P MAC protocol to utilize the synchronous operation of the wireless point-to-point links by scheduling all the links in the network and thereby achieve maximal throughput of the network [2]. It operates in 2 phases (SynRx and SynTx). When a node is in SynTx phase, all its neighbours will be in SynRx phase. After a specified period of time, it switches to SynRx phase and all its neighbours will switch to SynTx phase and this goes on. This requires that the network should be bipartite where some of the nodes will be in SynTx phase and the remaining nodes will be in SynRx phase. If the network is not bipartite, then a part of the network can be operated using 2P on a single channel and the remaining nodes of the network can be operated on a different channel/protocol.

Venkat in his thesis [4] implemented a Master-Slave protocol on the top of 802.11 DCF to support real time services in point-to-multipoint networks. It works like a wireless device in PCF mode suppressing the DCF mode of operation. The Master node controls the network and decides when the slaves can transmit. While PCF operates in the NIC (Network Interface Card), this protocol operates at the driver level giving more flexibility for implementing packet aggregation and various other schemes. However both the schemes (PCF and Master-Slave) are not good enough for long distance point-to-multipoint networks. Firstly 802.11 cards operating in PCF mode are not available in the market and are also very costly. The above mechanism of implementing PCF at driver level by suppressing DCF incurs a lot of overhead which is not suitable for providing Internet connectivity. Instead the Master-Slave protocol is aimed at achieving lower jitter values and reducing packet losses so that it can support VoIP kind of delay-intolerant traffic. There are also a lot of TDMA based implementations on 802.11 PHY [19]. However there are no open source designs of TDMA based MAC and all of the implementations are proprietary and their performance is not known.

This report essentially presents design, implementation and evaluation of two TDMA based protocols which address the above mentioned issues.

1. First is the implementation of 2P from scratch on the top of 802.11 PHY. Although 2P has been designed [2] and evaluated on a network simulator, it has not been evaluated outdoors. The major challenge is the implementation of 2P on the off-the-shelf 802.11 Atheros chipsets. This protocol is designed to operate on networks with the following characteristics.
 - Multiple radios per node
 - Multiple long distance point-to-point links at a single node
 - Use of high gain directional antennae
 - Use of a single channel on all the links
 - The network should be bipartite
2. Second is a bottom-up approach of implementing the TDMA-TDD based MAC from scratch on the top of 802.11 PHY to address the issue of hidden node problem and also to yield better throughput than CSMA/CA. The MAC design called SRAWAN (Sectorized Rural Area Wireless Access Network) is based on the WiMAX MAC [6].

The challenge here is to intelligently select minimal functionality from WiMAX that will suit the applications for rural networks. At the same time, the design should also use the flexibility and functionality of 802.11 hardware (Atheros) to ease the implementation process. The reason for choosing 802.11 PHY to implement WiMAX based MAC is because of the huge cost of WiMAX chipsets in the market. On the other hand a WiFi chipset is very cheap (costs around 25-50\$) because of its mass production and interoperability. This protocol is designed to operate on networks with the following characteristics.

- Point-to-multipoint links at a node
- Use of directional and sector antennae

Point-to-multipoint links comprises of a central wireless device C equipped with sector antenna and several wireless clients equipped with directional/omni antennae which are in the coverage area of the central device. All the wireless clients are connected to the network through the central device C. Figure 1.4 is an example of outdoor point-to-multipoint wireless link.

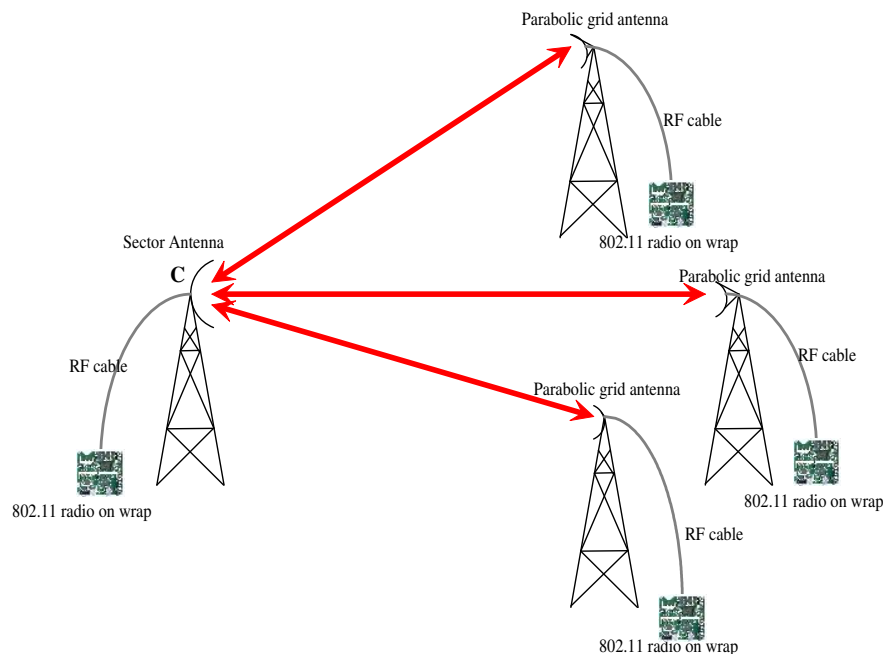


Figure 1.4: Point-to-multipoint link

In both of the scenarios described above, there will be a single node which has wired connectivity to Internet (landline node) and all the other nodes in the network are connected to Internet through this node. A detailed description of the protocols is given in the subsequent chapters.

Finally the significant questions that this thesis seek to answer are as follows.

- How to achieve 2P and SRAWAN on off-the-shelf hardware to preserve cost benefits?
- Is SRAWAN feasible outdoors? Is it possible to achieve tight time synchronization that is required for SRAWAN?
- Is 2P feasible outdoors? How to achieve 2P without tight time synchronization?
- What is the performance of SRAWAN and 2P in comparison to CSMA/CA (on 2 links)?
- What is the performance of SRAWAN in comparison to 2P (on single link) to contrast the time synchronization mechanisms (loose vs tight)?

We have implemented both the protocols at the driver level on the off-the-shelf 802.11 AR5212 Atheros chipsets (Ubiquiti SR2 and CM9 cards). We modified the existing Atheros driver *Madwifi* [11]. To implement these, we had to disable immediate MAC level acknowledgments, disable RTS/CTS, disable exponential backoff, disable virtual carrier sensing (NAV) and disable physical carrier sensing (CCA). We achieved these through appropriate modifications to the HAL (Hardware Abstraction Layer) of the Atheros chipsets some of which will be explained in the subsequent chapters.

The performance evaluation of the protocols (CSMA/CA, 2P and SRAWAN) show that 2P outperforms CSMA/CA by 45% and SRAWAN achieves significant performance (throughput) improvement over CSMA/CA by 40%. On a network of 2 links (4 interfaces), the resultant average throughput that can be achieved in 2P is 8.4 Mbps whereas for CSMA/CA, the average throughput is 6 Mbps. In case of SRAWAN operation on a network of 3 nodes (1 BS and 2 SSs), the average throughput is 5.8 Mbps while for CSMA/CA, the average throughput is 4.1 Mbps. On a single link, all the protocols achieve a throughput of around 6 Mbps. So the results favour the new protocols. The detailed analysis of these results is given in the subsequent chapters.

Thesis Organization

The thesis report is organized as follows. Chapter 2 gives the description of 2P protocol operation on single link and two links and then explains the protocol variant implemented in *Madwifi* driver. It also presents a theoretical estimate of the expected throughput for 2P operation on single link. Chapter 3 describes the functional specification and architecture of SRAWAN. It also gives a theoretical estimate of expected throughput of SRAWAN on single and two links. This is then followed by performance evaluation of 2P and SRAWAN in chapter 4. It also analysis the throughput results of 2P and SRAWAN in comparison to CSMA/CA. Chapter 5 concludes the thesis report by summarizing the two protocols and then gives some ideas for future work to extend the protocols 2P and SRAWAN for multi-hop and multi-sector networks respectively. Finally Appendix A consists of MAC PDU and header formats of SRAWAN. Appendix B shows the transmit and receive calibration data for Atheros AR5212 based ubiquiti SR2 cards.

Chapter 2

2P: Design and Implementation

2P is a TDMA based MAC designed to operate on top of 802.11 PHY. Its primary motivation is to utilize the synchronous operation (SynTx and SynRx) to achieve better throughput than CSMA/CA and provide low cost connectivity to rural areas. 2P is designed to operate as a point-to-point MAC protocol. The high level logic of the protocol is given in [2]. In this thesis, we laid emphasis on the implementation of this protocol in *Madwifi* driver to be operated on the off-the-shelf Atheros AR5212 hardware. Later we evaluated the performance of 2P on a network of two links in *DGP* testbed. This chapter gives an overview of the architecture, functional specification, state machine and detailed implementation design of 2P. The significant question that this chapter would try to answer is how to achieve 2P on the top of off-the-shelf Atheros AR5212 802.11 PHY.

2.1 Architecture and Terminology

2P can be operated on any networks that are bipartite in nature. But the analysis for the thesis is done on just 3 nodes A, B and C where B and C have single interface each and A has 2 interfaces N_1 and N_2 as shown in Figure 2.1.

Each end of the point-to-point links is called an interface *ifa*. The ends of the point-to-point link are link neighbors *link-nbrs*. The interfaces at the same node are called interface neighbours *ifa-nbrs*. For example in the Figure 2.1, N_1 , N_2 , B and C are *ifas*, N_1 and B as well as N_2 and C are *link-nbrs*, N_1 and N_2 are *ifa-nbrs*.

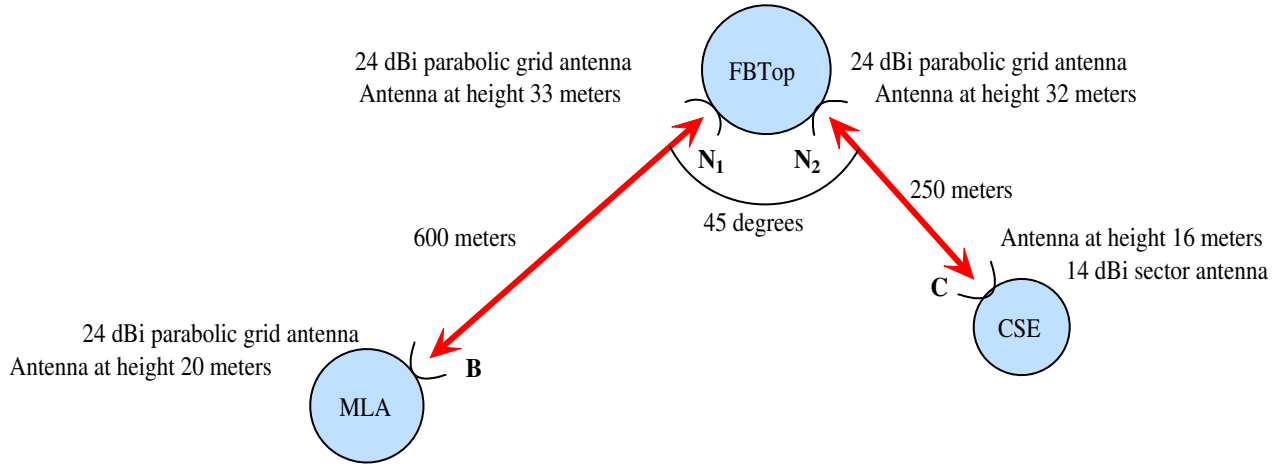


Figure 2.1: The two link point-to-point system used for 2P experimentation

2.2 Achieving SynOp

We have discussed in earlier chapters that 2P's objective is to utilize the SynOp and achieve better throughput than CSMA/CA. CSMA/CA does not allow SynRx and SynTx because of its immediate MAC level acknowledgments and carrier sensing. The paper [2] shows how these issues can be tackled at the driver level in the following ways.

- Immediate ACKs can be turned off by:
 1. use of IBSS mode (which is available in prism2 chipsets) of operation for all the interfaces operating on 2P with a separate SSID for each link and
 2. converting IP unicast packets to MAC broadcast packets at the driver level (because the receiver of the packets will not send an immediate ACK for MAC broadcast packets in IBSS mode of operation)
- Carrier-sense backoff can be taken care as follows. A wireless interface has 2 connectors (RIGHT and LEFT) to external antenna. Only one of the connector is connected to external antenna which should be used both for transmission and reception (lets assume it to be RIGHT). HostAP driver for prism chipsets provides the flexibility of changing the receive antenna at run time using an *antsel_rx* command. So when the interface is about to transmit, set its receive antenna to LEFT. The 802.11 hardware does carrier sensing, but finds the channel to be free because the LEFT connector is

not connected to an external antenna. The LEFT connector only sees noise. When the interface wishes to receive packets, it should switch back the receive antenna to RIGHT.

The drawbacks of this implementation are as follows:

1. If any wireless device A operating 802.11 DCF is in the vicinity of the wireless device B operating 2P, then B will backoff because of virtual carrier sensing (NAV - Network Allocation Vector) and physical carrier sensing (CCA - Clear Channel Assessment) eventhough B is in SynTx phase. This may lead to some disturbance in the synchronization because of timeouts at the other end of the 2P link. Ideally B should continue its transmission (packets will get corrupted) irrespective of whether the medium is free or not as this is a TDMA MAC and there should be no contention.
2. When the interface is in SynTx phase, it will use the CSMA/CA exponential backoff mechanism to transmit packets. So between every two packet transmissions, there will be a wastage of *DIFS + some random value* amount of time. This is not required for 2P as the protocol is not contention based and there will not be any collisions as the transmission schedule is inherent in the way the protocol operates i.e. an interface can transmit only in SynTx phase.
3. There is also a lot of overhead ($140\mu\text{s}$) in switching the receive antenna between RIGHT and LEFT at run time.

So there is an obvious scope for improving the performance of the 2P protocol if the above drawbacks can be handled properly. We utilize the flexibility in the Atheros *Madwifi* driver and AR5212 *HAL* code to achieve the following primary tasks for the successful operation of SynRx and SynTx.

1. Disabling immediate MAC level ACKs - The sender should not wait for ack after the transmission of the packet and the receiver should not send an ack after reception of the packet. Since the protocol is TDMA based, there is no need for MAC level acks which will interfere with the ongoing packet transmission.
2. Disabling physical carrier sensing i.e. CCA (Clear Channel Assessment) - CCA and NAV are incorporated in 802.11 hardware to support CSMA/CA operation which is contention based. 2P being TDMA based no longer needs these carrier sensing

mechanisms. The interfaces should transmit irrespective of whether the medium is free or not.

3. Disabling virtual carrier sensing i.e. NAV (Network Allocation Vector) - The reason for this feature is same as given for disabling of CCA.
4. Disabling the exponential backoff feature - We no longer need backoff mechanism of CSMA/CA as the MAC protocol is no longer contention based.
5. Elimination of RTS/CTS exchange - These are not required for the TDMA-based 2P protocol.
6. Nullifying SIFS, DIFS, EIFS and slot duration.

We have been able to achieve all the above mentioned tasks except CCA. We have not been able to disable CCA successfully. Disabling CCA lead to some random packet losses at the transmitter. Instead we set CCA threshold very high so that it virtually ignores any energy in the channel. We have implemented 2P in accordance with the functional specification as explained in [2]. The next few sections give a brief overview of its operation and state machine.

2.3 2P operation on single link

A detailed description of 2P protocol operation on single link is given in [2]. This section gives a summary of that description. In later sections, we will give a detailed description of the protocol scheme we implemented on Atheros 802.11 chipsets. The state machine for 2P operation is depicted in Figure 2.2. For this section consider only unshaded states that are represented in circles and rectangles. There are only 2 phases in 2P (SynTx and SynRx). The interfaces operate by switching between these phases periodically. Now consider the link N_1B in Figure 2.1. In steady state if *ifa* N_1 is in SynTx phase, then *ifa* B will be in SynRx phase and vice versa. Assume that N_1 is in SynTx phase. So after completion of transmission for SynTx duration, N_1 will send a *marker* packet to B and switches to SynRx phase. The purpose of the *marker* packet is to inform its *link-nbr* B in SynRx phase to switch its phase to SynTx. After reception of *marker* packet, B will switch to SynTx phase and start transmission if any or will be idle for SynTx duration. The *ifas* keep on switching the phases periodically. The *marker* packet is the only means

by which an *ifa* in SynTx phase tells its *link-nbr* in SynRx phase to switch phases. The *marker* packet will make sure that only one link is transmitting at any given instant. The two *ifas* are said to be in *loose synchrony* under steady state conditions.

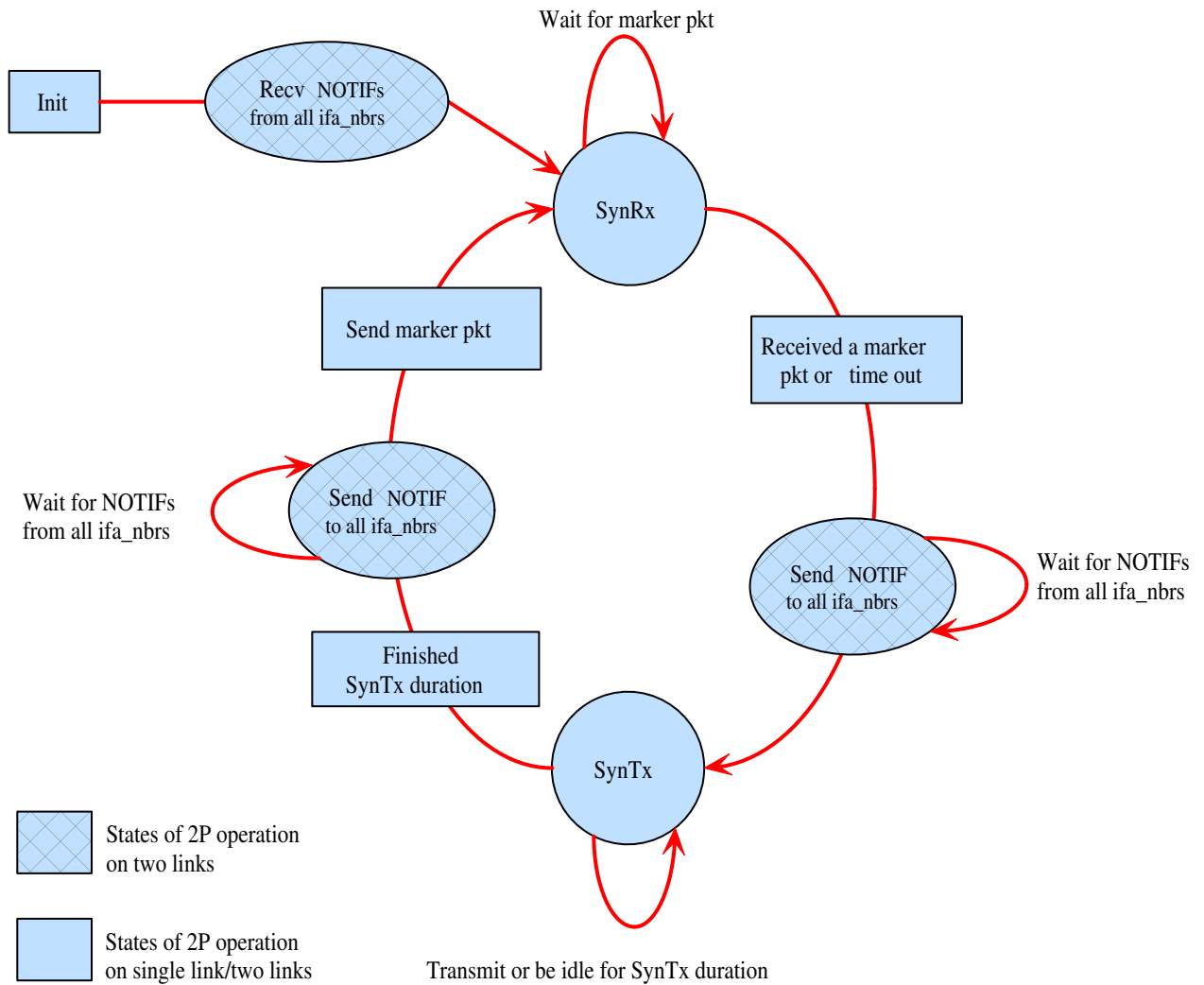


Figure 2.2: 2P state machine

The only cases that have to be handled for 2P operation on single links are as follows.

1. temporary loss of *synchrony* due to marker packet losses
2. link initialization or link recovery after a failure

The paper [2] explains how to handle these cases by using timeouts for 2P states. Every *ifa* starts a timer on entering SynRx phase. If it successfully finishes the SynRx phase, then *ifa* will delete the timer before entering the SynTx phase. To handle marker packet losses, consider the Figure 2.3. *ifa* N_1 started a timer at t_0 after entering SynRx phase. *ifa* B transmitted a *marker* packet at time t_1 , but *ifa* B did not receive it till the timeout event occurred at t_2 . Then N_1 will enter to SynTx phase and start transmissions. After reception of packets from N_1 , B will continue to be in SynRx phase and restarts its timer and this process goes on. In the same way link initialization and recovery can be handled.

It may happen that both the interfaces may undergo repeated timeouts because of coincidence of timeouts. [2] handles this by introducing some randomness in the timeout period. We did not use this explicit randomness in our implementation because the kernel's scheduling has high jitter values which itself adds some randomness.

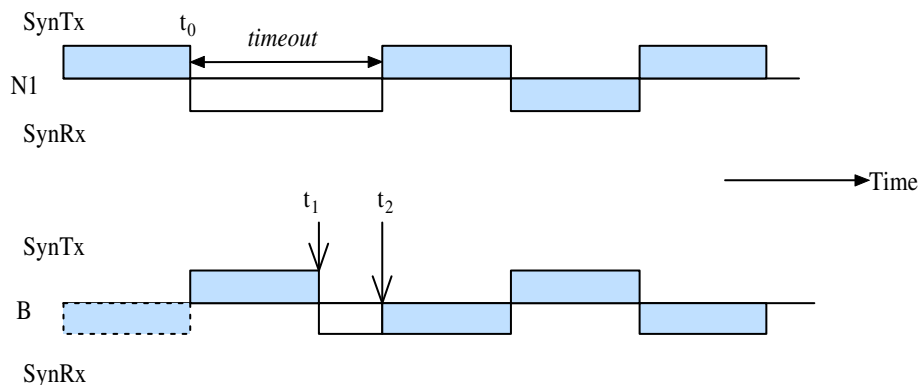


Figure 2.3: Link synch/recovery process

From the above discussion we can observe that SynTx and SynRx durations and the timeout values are the important parameters that will have an effect on the performance of the protocol. A detailed discussion on these parameters will be given in the section 2.5.

2.4 2P operation on two links

Consider the Figure 2.1. For 2P to achieve maximal throughput, both the interfaces at A i.e. N_1 and N_2 should simultaneously transmit and simultaneously receive. In order to achieve this, both the interfaces have to coordinate with each other to be in synchrony

apart from the synchrony maintained with their respective *link-nbrs*. This coordination can be achieved in many ways as given below.

1. Shared memory - If both the interfaces are operating on same machine/environment, then this is the best possible alternative.
2. Exchange of ethernet messages - Both the interfaces can maintain synchrony through the exchange of ethernet messages if they are operating on different machines.

We have experimentally seen that both the interfaces cannot be operated in same environment (WRAP board) because of the huge channel interference between themselves. The results of this experiment will be given in the next subsequent chapters. So we choose the second approach for coordination. The coordination scheme should make sure that a Mix-Rx-Tx situation never occurs.

The Figure 2.2 gives an overview of the state diagram of 2P operation on two links. The *ifa-nbrs* coordinate with the exchange of NOTIF ethernet messages. This is done

1. when the interfaces shift from SynRx to SynTx and
2. before transmission of *marker* packet at the end of SynTx phase

The need for coordination between *ifa-nbrs* is described as follows. In the Figure 2.1, assume that N_1 and N_2 are in SynRx phase and N_1 receives *marker* packet from B . If N_1 shifts to SynTx phase without consulting N_2 , then this will result in Mix-Rx-Tx situation and therefore N_2 will receive either corrupted packets or noise because of the side lobes at N_1 . Similarly Mix-Rx-Tx situation arises when N_2 receives *marker* packet before N_1 . So when the interfaces shift from SynRx to SynTx, the *ifa-nbrs* should coordinate and shift to SynTx phase at the same time. In the same way when the interfaces finish their SynTx phase, they have to coordinate with their *ifa-nbrs* before sending *marker* packet. Without loss of generality assume that N_1 and N_2 are in SynTx phase and N_1 finishes SynTx phase first. Suppose it sends marker packet immediately without consulting its *ifa-nbrs*, then B will shift to SynTx phase and might start transmitting packets to N_1 . But since N_2 did not complete its SynTx phase, this will lead to a Mix-Rx-Tx situation and N_1 's reception will be corrupted. So there is a need for coordination between *ifa-nbrs* in the above two cases.

In both the cases, the interface sends a NOTIF message to all its *ifa-nbrs* and waits for NOTIFs from all other *ifas* before proceeding to the next state as shown in state diagram in Figure 2.2.

Now the only case that is left out is the initialization of this coordination. All the *ifas* maintain the state of all their *ifa-nbrs* as IFA_UP or IFA_DOWN. By default the start state is IFA_DOWN. So when an *ifa* boots up, it sends an HELLO broadcast message on its ethernet connection. This is repeated at regular intervals if no *ifa* responds. Upon receiving a HELLO or NOTIF message from an *ifa*, the state of this particular *ifa* is changed to IFA_UP and thereupon NOTIFs are sent to this *ifa*. There is a very minute possibility of these ethernet packet losses and these can be handled by timeouts as follows. If a timeout of ethernet message from *ifa1* occurs at *ifa2*, then the *ifa2* will change *ifa1*'s state to IFA_DOWN from IFA_UP. And *ifa2* will change this state to IFA_UP upon receiving a NOTIF or HELLO message from *ifa1*. *ifa* will send NOTIFs to all its *ifa-nbrs* whose state is IFA_UP.

In our implementation we used a variant of the scheme explained above even though the logic is same. The next section will give a detailed explanation of the same.

2.5 Implementation

The implementation of 2P relied on some of the flexibilities in *Madwifi* driver and *HAL* code of AR5212 Atheros chipsets to achieve the tasks that are mentioned in the above sections. In short we disabled immediate ACKs, NAV, RTS/CTS, DIFS, SIFS, AIFS, SLOT, exponential backoff and set CCA threshold to the highest possible value. AR5212 Atheros hardware has the capability to give micro second resolution timer which we used for timeouts. As explained in the above section, we implemented a variant of the interface coordination scheme mentioned in that section. This is explained in Figure 2.4.

We define *Master* and *Slave* interfaces. This is done for the ease of implementation and will not affect the performance of the protocol. Without loss of generality assume that N_1 is *Master* and N_2 is *Slave* and both the interfaces N_1 and N_2 are in SynTx phase. When master completes SynTx, it sends a NOTIF to slave and waits for NOTIF_ACK from slave. After the slave finishes its SynTx duration and receives a NOTIF from master, it sends a NOTIF_ACK to master and also a *marker* packet to its *link-nbr*. After receiving NOTIF_ACK, master also sends a *marker* packet to its *link-nbr*. Then both the interfaces

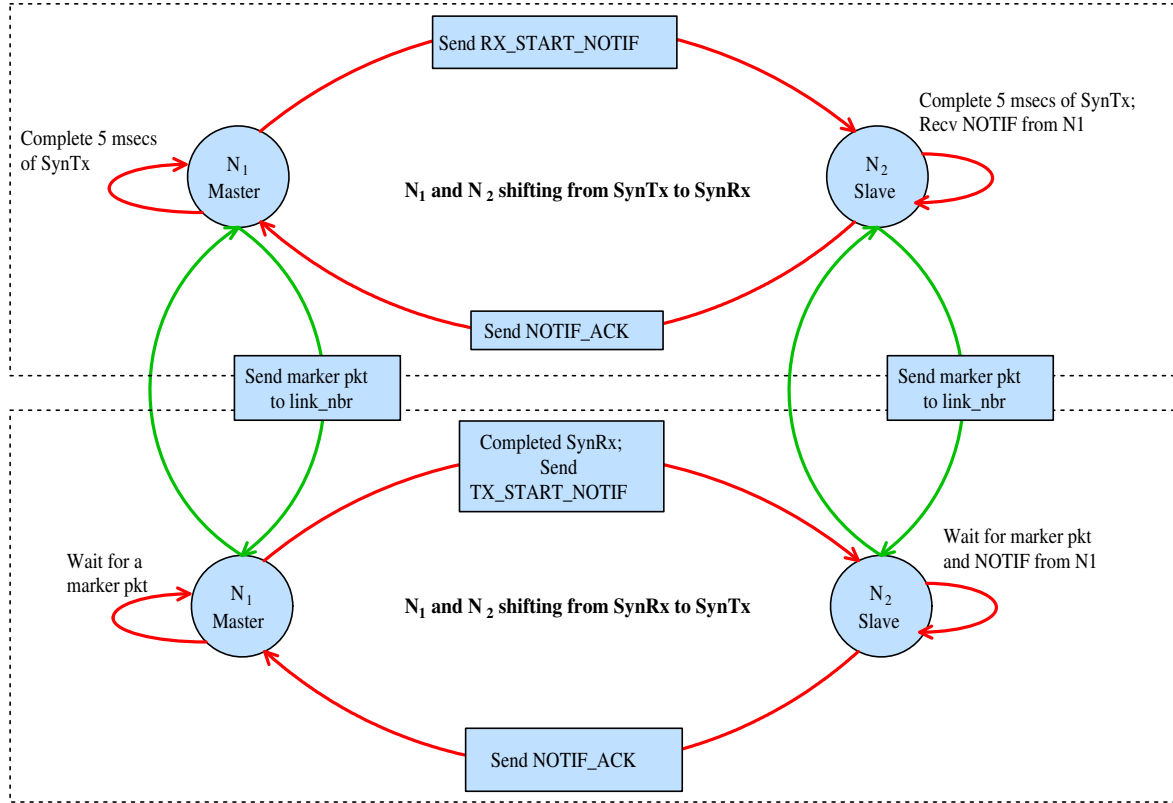


Figure 2.4: Master Slave operation in 2P

shift to SynRx. When master receives a *marker* from its *link-nbr*, it sends a NOTIF to slave and waits for NOTIF_ACK. After the slave receives a *marker* from its *link-nbr* and also receive a NOTIF from master, it sends a NOTIF_ACK to master. Then both the *ifas* shift to SynTx phase. So the difference from the scheme explained in the above section is that the master always takes the initiative to send NOTIF messages.

If the number of *ifas* is more than two, then every *ifa* should send NOTIF to every other *ifa* and all the *ifas* can shift phase only after receiving NOTIFs from all of their *ifa-nbrs*. The *ifa* which finishes its SynTx phase or which receives *marker* packet should immediately send NOTIF to all its *ifa-nbrs* irrespective of whether its *ifa-nbrs* sent a NOTIF or not. So in case of multiple interfaces, there is no use of NOTIF_ACKs.

Implementation parameters and Throughput Estimation

- Duration of SynTx and SynRx = 5 ms (Since it is single hop network, the ratio of SynTx and SynRx can be anything. The larger their values are, the lesser will be the overhead and higher will be the throughput. But the larger values will lead to an increase in response time)
- Timeout value for SynRx = 25 ms
- Timeout value for ethernet messages = 100 ms
- MTU size = 1500 bytes (this is the default value)
- Transmission rate = 11Mbps for data packets and 1Mbps for marker packet (Marker packets are sent at 1Mbps to decrease the possibility of their losses)

Ideally speaking the only overheads are *marker* packets and ethernet messages. The estimated throughput of 2P on single link is calculated as follows.

- SynTx + SynRx duration = 10 ms
- Size of a typical packet = 1400 + 60 (TCP and IP hdrs) + 24 (802.11 hdr) = 1484 bytes
- Time taken to transmit 1484 byte packet at 11Mbps = $1484 * 8 / 11 + 96(PHY preamble)$ usec = 1175 usec
- Size of marker packet = 1 (packet payload) + 24 (802.11 hdr) = 25 bytes
- Time taken to transmit marker packet at 1Mbps = $25 * 8 + 96(PHY preamble)$ = 296 usec
- Number of packets that can be transmitted during SynTx in 4700 usec = 3 (Even though 4700/1175 is 4, ideally there will be some jitter/delays in the handling of packet transmissions and so only 3 packets can be transmitted and the remaining time will get wasted)
- Similarly number of packets that can be received in SynRx phase = 3

- So 6 packets can be transmitted/received in 10 msec. The net throughput at the application layer will therefore be $6 * 1400 * 8/10 = 6.72$ Mbps

For 2P operation on 2 links, there should be an additional delay of few microseconds for coordination between the *ifa-nbrs*. You can observe from the above calculation that there is a huge amount of wastage which should compensate for all overheads that incur for synchronization among *ifa-nbrs*. So the net throughput should be 6.72 Mbps.

Chapter 3

SRAWAN: Design and Implementation

SRAWAN (Sectorized Rural Area Wireless Access Network) is a MAC based on IEEE 802.16, designed to operate on top of 802.11 PHY. Its primary motivation is to provide low-cost Internet access to rural areas over tens of kilometers. SRAWAN is designed to operate as a point-to-multipoint MAC protocol. Bharani is an implementation of SRAWAN MAC. While SRAWAN is an open specification, Bharani is a proprietary implementation developed by IIT Kanpur in collaboration with Zazu Networks. This chapter gives an overview of the architecture, functional specification of SRAWAN, packet and frame structure. We also estimate the performance (throughput) of SRAWAN on single link and two link network.

3.1 Architecture

The network model as shown in Figure 3.1 is similar to that of 802.16 (WiMAX). We have a point-to-multipoint scenario. The central node is called Base Station (BS) and several Subscriber Stations (SSs) are connected to this central node through long distance links. The BS controls the traffic to and from the SSs. Typically a BS employs a sector antenna and the SSs employ directional antenna facing towards the BS.

SRAWAN is a simplified version of WiMAX and specifies the communication between the BS and the SSs in the configuration above. It is potentially envisaged to be used in the multi-sector environment as well, albeit with the same specification. Time Division

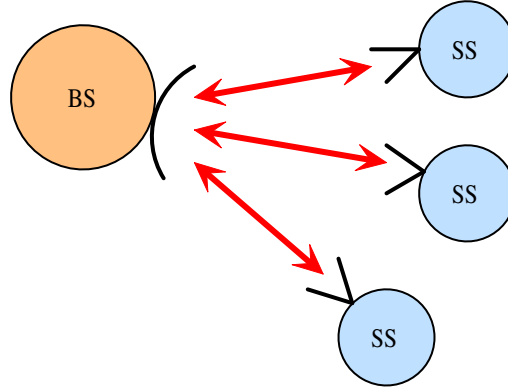


Figure 3.1: A point-to-multipoint system

Duplexing (TDD) is used to support the traffic from both sides i.e. BS to SSs and SSs to BS. Time Division Multiple Access (TDMA) and Demand-Assigned-Multiple-Access (DAMA) are used to support multiple SSs. SRAWAN relies on tight time synchronization to support TDD i.e. the SS clock has to be in sync with the BS clock throughout its operation. WiMAX also specifies Frequency Division Duplexing (FDD) as well which is not considered in the SRAWAN specification because it is not possible to switch between frequencies dynamically on 802.11 hardware. WiMAX itself is not chosen because of its complexity. WiMAX provides lots of functionality which is of no use for the scenarios we are dealing with.

3.2 Functional Specification

SRAWAN frame structure is shown in Figure 3.2. It consists of downlink (DL) followed by uplink (UL). DL starts with a beacon and then it is followed by downlink packets called PDUs (Protocol Data Units). The BS announces its presence to potential SS, through periodic *beacons*. An SS on coming up, listens to the beacons, and determines which BS is appropriate for it to get service from. It first has to synchronize with the BS. This is done through a *ranging* process. Then it has to *register* with the BS. Subsequently, there is a notion of *connections* between an SS and the BS. There can be one more connections between the BS and a single SS. Connections are setup after the registration process. In this section we discuss these various management phases an SS should go through in the process of establishing a connection with a BS.

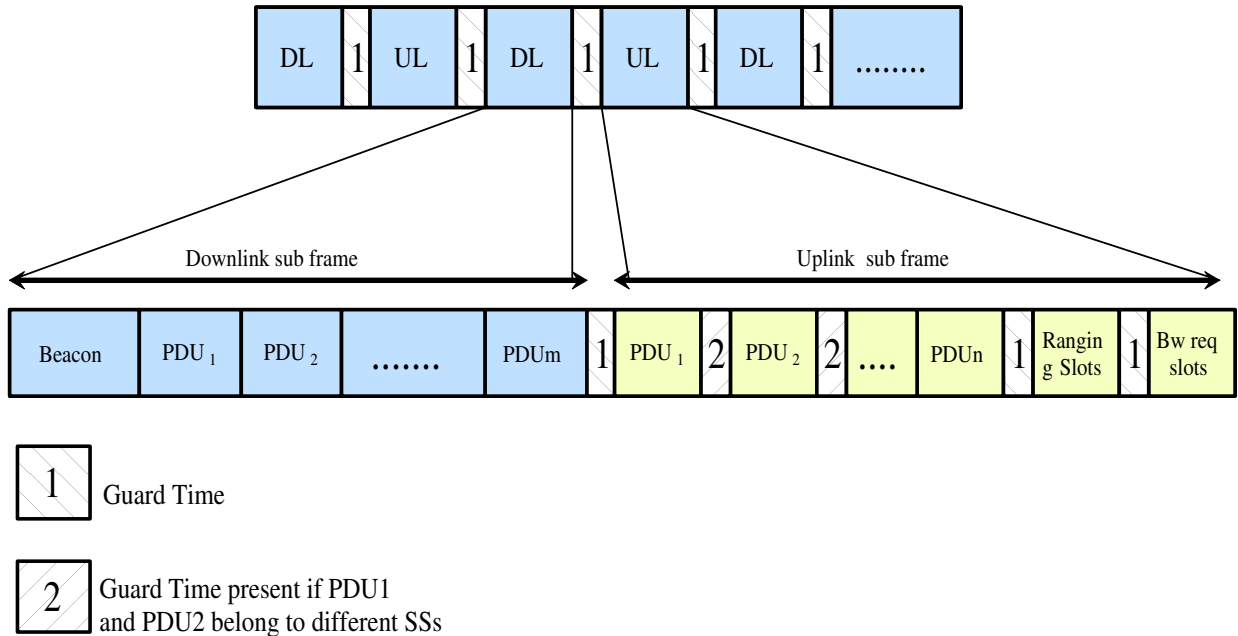


Figure 3.2: SRAWAN frame structure

1. **Ranging:** In this procedure, the SS performs time synchronization with the BS. This is important especially since the links can extend over several kilometers (RF propagation latency can be in the order of 50-100 microseconds and also the processing delay at BS and SS is of the order of 100 microseconds). Ranging is performed periodically so that the SS is kept in sync with the BS. SS has to be in sync with the BS because BS puts ULMAP. The BS periodically transmits a beacon. An SS first listens for a beacon and then sends a ranging request. The BS then sends a ranging response. In the ranging response to the first ranging request, the BS assigns the SS two connectionIDs (CIDs) called the primary CID and the basic CID. *The primary CID is used for further exchange of management messages except periodic ranging messages (see below), while the basic CID is used for further exchange of periodic ranging messages.* The need for two CIDs instead of one is explained in subsection 3.2.2. The SS and BS exchange their clock time in the ranging messages to perform time synchronization. Initial ranging request message should be sent in the ranging contention slots in the uplink and periodic ranging messages should be sent in the uplink slots allotted by the BS for the respective SS. Unlike SRAWAN, WiMAX defines ranging for SS to synchronize power, frequency and time with the BS. On the

other hand SRAWAN defines ranging as a mechanism to synchronize time among SSs and BS. Here operating frequency and power of the WiFi card has to be preassigned.

2. **Registration:** Through this procedure, the SS informs the BS that it is entering the set of SS serviced by the BS. The link between the BS and SS is connection oriented: one or more connections can be established for data exchange. The registration process is required prior to any connection formation. The process involves a registration request from the SS, followed by a registration response from the BS.
3. **Connection formation:** After registration, the SS can request for any number of further connections. A connection request from an SS to the BS elicits a connection response from the BS to the SS. The number of connections may be restricted by the BS in an implementation specific fashion. Connection request messages have to be sent in the bandwidth contention slots in the uplink of the frame. *In WiMAX, the SSs and BS exchange information regarding their capabilities, type of ARQ feedback mechanism (cumulative, selective, cumulative with selective etc) and also the type of connection (Best Effort, Unsolicited Grant Services (UGS), Real Time etc) whereas in SRAWAN only information regarding ARQ (to be enabled or not) is exchanged. The type of ARQ feedback mechanism is always cumulative with selective acknowledgment.*
4. **Authentication and Security:** The 802.1x authentication and security mechanisms shall be used. The primary CID will be used for any such exchange.
5. **Packing:** Multiple MAC SDUs may be packed into a single PDU (provided the PDU size is less than Maximum Transmit Unit size), for improved efficiency. If each SDU is sent separately, a lot of overhead is incurred due to PHY preamble and MAC headers for each SDU. Packing multiple SDUs in a single PDU will reduce these overheads and thereby increase the efficiency. The format to do this is similar to that in 802.16. However, SRAWAN does not define MAC level fragmentation. That is, we do not expect that a MAC SDU would have to be fragmented among more than two successive MAC PDUs.
6. **Automatic Repeat reQuest (ARQ):** The MAC (technically the Link Layer Control) defines an ARQ mechanism for link layer reliability. Much of the complexity of 802.16 ARQ is done away with in SRAWAN. SRAWAN supports only ARQ at the

granularity of MAC PDUs. SRAWAN uses a sliding window based ARQ mechanism to achieve link layer reliability.

The next few sections will elucidate each of these management phases in detail and also point out some of the drawbacks of SRAWAN.

3.2.1 Frame Structure

The SRAWAN MAC frame structure is depicted in Figure 3.3. A frame is divided into a downlink (DL) subframe and uplink (UL) subframe. [DL: BS to SS, UL: SS to BS]. Each subframe consists of one or more MAC Protocol Data Units (MPDUs). The first PDU in the DL subframe is a *beacon*. This is because the *beacon* contains the control information for the DL and UL and all the SSs use this control information to transmit their uplink traffic. This control information is called as Frame Control Header (FCH) and is added as a payload in the *beacon*. So the functionality of the beacon is to provide this control information and also provide the details of the DL and UL channels. The portion of the FCH that describes the PHY characteristics of the DL and UL channels is called Downlink Channel Descriptor (DCD) and Uplink Channel Descriptor (UCD) respectively. And the portion of FCH that gives information to each SS about their respective time slots for scheduling transmission on the UL subframe is called Uplink Map (ULMAP).

FCH has ULMAP in the beginning and then followed by DCD and UCD. The reason for this is as follows. DCD and UCD are used by an SS only during ranging and connection establishment process whereas ULMAP has to be processed every time a beacon is received to find out whether the SS is given any uplink slots or not.

A beacon is followed by one or more DL PDUs destined for various SSs. *The difference from WiMAX is that a beacon in 802.16 also contains Downlink Map (DLMAP). DLMAP contains information regarding the DL packet schedule to various SSs. So SSs can use this information to go to sleep in the DL time slots in which there are no packets destined to them. SSs sleep to save power. So DLMAP is used by SSs to save power in the DL.* We have not included DLMAP in SRAWAN. So every SS must receive and check each DL PDU to find out whether the packet is destined to it or not. This design decision has been taken because achieving tight time synchronization is a very difficult task even when all the SSs are awake. So if some of the SSs sleep, then there is a possibility that the SSs clock get skewed and might result in packet losses and collisions on its wakeup. And also we are not concerned about power consumption at this stage.

After the DL duration is completed, the UL starts with transmission of UL PDUs from various SSs to BS. This is then followed by contention based ranging slots and then contention based connection request slots. So an UL may contain any of the following set of packets.

- Ranging packets from SS to BS for performing ranging in contention based ranging time slots.
- Bandwidth request packets from SS to request for bandwidth beyond what BS may have assigned it in the ULMAP in contention based connection request time slot.
- UL PDUs from SS to BS in intervals allocated for individual SS (as specified in the ULMAP in beacon).

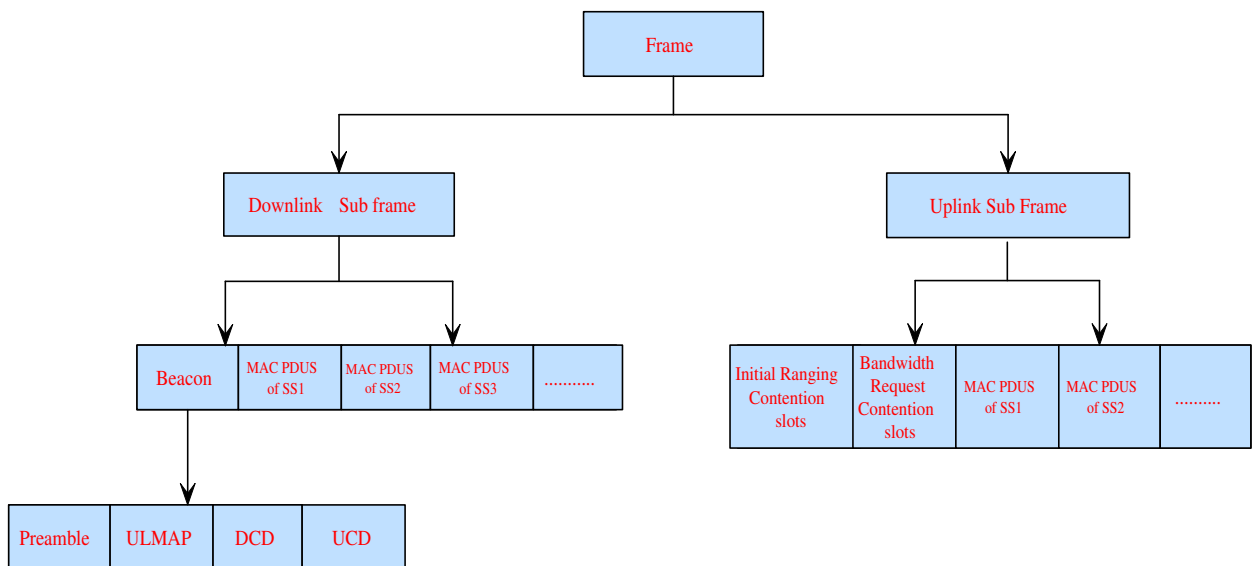


Figure 3.3: FrameStructure

Any of these slots in UL (data slots, ranging slots, connection request slots) may be present in any given frame. They may occur in any order and any quantity within the frame, at the discretion of the BS uplink scheduler as indicated by the ULMAP. An SS may transmit more than one MAC PDU if the BS allocates enough time slots for it in the ULMAP. Figure 3.2 shows that there is a guard time between DL and UL and also before the beginning of ranging slots and connection request slots. The guard time is also

present between two UL PDUs if both of them belong to different SSs. There is possibility of clock skew at various SSs even after tight time synchronization. This guard time helps to prevent any possible collisions between packets due to clock skew.

3.2.2 Ranging

The DCD, ULMAP and the BS timestamp in the beacon are the three fields which are used during the ranging phase. The DCD has the following fields: BSSID and CwMin/CwMax (contention window min and max) for ranging. The BSSID is the identifier that represents the network in the coverage of the BS. CwMin and CwMax are used for generating a random number which represents the number of slots an SS has to backoff before transmitting the ranging request. The BS also puts its time stamp in the first 8 bytes of the beacon payload. The SS is configured with a frequency and power value prior to booting. An SS after booting starts in receive mode and waits for the beacon from BS. After receiving the first beacon, SS reads the timestamp from the beacon payload and then sets its clock to that timestamp. So the SS's clock is slow by d where d is the air propagation delay. The SS then processes ULMAP and reads the initial ranging contention slots as allocated by the BS in the uplink subframe. The SS then generates a random number between 0 and 2^x where x is initialized to CwMin. SS backoffs for that many slots and then transmits the Initial Ranging Request (IRR) in the ranging slots. If the SS does not receive the Initial Ranging Response (IRRe) within a specified period, it increases x by 1 and then restarts the random number generation and backoff process. This goes on until $x < CwMax$. The subscriber time stamp T_{ss} is set in the first 8 bytes of the IRR payload. T_{ss} is the clock value at the SS at the time of transmitting IRR.

The BS will wait for the next downlink subframe to send the Initial Ranging Response (IRRe). As described above, the SSs clock is slow by air propagation delay between SS and BS. The BS calculates the air propagation delay using the T_{ss} in IRR and the time of reception of IRR and then sends this value in IRRe. The SS then corrects its clock by incrementing its clock with this value. The air propagation delay is calculated as $(Time\ of\ reception\ of\ IRR\ at\ BS - T_{ss}\ in\ IRR)/2$ (this is equal to $RTT/2$) and is added in the payload of the IRRe. The IRRe also consists of SS MAC address, the T_{ss} in IRR, basic CID and primary CID. SS periodically does ranging with the BS to be in sync with it (to handle clock skew). BS informs SS to send a periodic ranging request by allotting a slot in ULMAP for its basic CID. BS allots slots for primary CID to inform

SS to transmit either a registration request or a connection ack depending on whether the SS is in registration phase or connection establishment phase. The basic CID is used for exchanging these periodic *ranging* messages and is the only means by which BS can ask SS to send a periodic ranging request. So the primary CID is used for transmission of other management packets.

The time synchronization process during ranging is depicted in the Figure 3.4. The SS receives the IRRe from the BS. After the initial error checking is done, it checks the management type to confirm that its an IRRe packet. The SS then reads the T_{ss} in the IRRe packet to figure out which IRR is this IRRe in response to, in case the SS had sent out more than one IRR. If it is not in response to the last sent IRR, then the packet is rejected and the SS goes back to the synchronization phase. If it is the IRRe to the last sent IRR, then the packet is accepted and processed. The SS clock is synchronized to BS clock by setting its clock to $T_{bs} + RTT/2$.

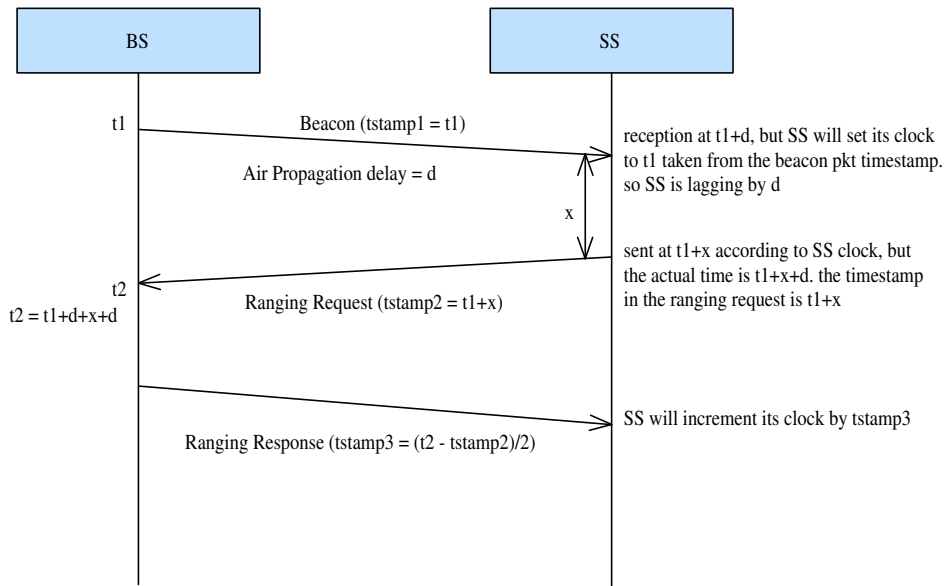


Figure 3.4: Time synchronization during Ranging

3.2.3 Registration

Once the ranging process is completed, SS has to register with the BS before establishing connections. So registration is a precursor to formation of any connections for data ex-

change. SS might do ranging with multiple BSs, but it can register with only one BS. The primary CID is used for the exchange of registration request (SS to BS) and registration response (BS to SS).

After transmitting the ranging response to an SS, the BS will allocate slots in the uplink to that SS for registration in the next few frames. The slots will be allocated in the 4 subsequent frames following the transmission of ranging response. If the SS does not send a registration request in those 4 frames, then the BS discards all the information regarding that SS and clears the CIDs allotted to that SS. If the SS sends a registration request, then the BS will send a registration response after which the SS can establish connections for data exchange.

3.2.4 Connection Formation

An SS sends connection request on the primary CID and receives a connection response from the BS on the same CID and then sends a connection ack. So connection establishment is a 3-way handshake similar to that in TCP. Any data exchange can happen only after connection formation. An SS may form more than one connection with the BS (at the discretion of the BS). There may be an implementation specific limit on the number of connections supported by a BS.

A connection is identified by a 16bit connectionID (CID). The CID is assigned by the BS. A connection is bidirectional. The same CID is used for communication from BS to SS, as well as from SS to BS.

In the case of multiple connections between an SS and a BS, SRAWAN does not specify how traffic may be split across the various connections. The installation of such filters is beyond the scope of SRAWAN. There are at least two possible ways to do this. If the point-to-multipoint setup is considered as a routed layer3 network, then the BS or SS may present each of the connections as a separate interface to the routing layer. In such a case, the installation of routing entries will automatically define which packets go on which connection. An alternate possibility arises when the point-to-multipoint setup is considered as a bridged layer2 LAN segment. Then the BS or SS may present each of the SRAWAN connections as a separate port to the bridging layer. In this scenario, the forwarding entries in the bridged network would define which packets are sent on which connection.

3.2.5 Authentication and Security

SRAWAN does not define any new authentication or security mechanism. 802.1x authentication and security mechanisms are used in the point-to-multipoint setup, just as in a 802.11 infrastructure mode of deployment. SRAWAN does not impose any restrictions on what to send in the packets (payload can contain anything) and also how to send the packets (packet payload can be encrypted). It only restricts when to send a packet. 802.1x authentication and security mechanisms rely on exchange of challenge-response messages and it does not lay any restrictions on when to send a response for a challenge (but the response should come in a limited period of time). So 802.1x security mechanisms can be used in SRAWAN for authentication and security.

3.2.6 Packing

Multiple MAC SDUs may be packed onto a single MAC PDU for efficiency reasons. If each SDU is sent separately, a lot of overhead is incurred due to PHY preamble and MAC headers for each SDU. Packing multiple SDUs in a single PDU will reduce these overheads and thereby increase the efficiency. Such packing happens only within a connection. SRAWAN also defines concatenation of multiple MAC PDUs (all belonging to same CID) to be sent with a single PHY layer overhead. This is similar to the packing scheme in WiMAX. WiMAX also defines concatenation of multiple MAC PDUs belonging to different CIDs.

3.2.7 ARQ

SRAWAN uses a per connection ARQ. An independent ARQ state machine is used for either direction of the connection.

SRAWAN significantly reduces the complexity of WiMAX ARQ mechanisms. WiMAX ARQ mechanism provides the flexibility of choosing ARQ scheme (selective ack, cumulative ack, cumulative with selective ack etc) at the time of connection establishment. The ARQ feedback is sent as a subheader in the payload of the MPDU. The granularity of retransmission is a block which is negotiated during connection establishment phase. All the packets need not contain ARQ feedback. On the other hand SRAWAN uses retransmissions at the granularity of MAC PDUs. This implies that every MAC PDU needs to have an independent CRC. Every MAC PDU has a 8 bit sequence number for ARQ purposes. A sliding window based ARQ with selective ACKs is used in SRAWAN. The

maximum window size is 16 (1/4th of the sequence number space). That is, there can be a maximum difference of 16 in the sequence numbers of (a) the PDU to be sent, and (b) the last acknowledged PDU. The MAC header in case of ARQ enabled connections is shown in Figure 3.5. The description of the various fields is given in A.2.3.

HT	EC	TYPE (6)	Version	CI	Rsvd	LEN MSB (4)
LEN LSB (8)			SEQ (8)			
CID MSB (8)			CID LSB (8)			
SACK MAP (16)						
ACK SEQ (8)			HCS (8)			

Figure 3.5: ARQ sub header

The sender shall retransmit a packet for a maximum of NUM_ARQ_RETRIES times. The receiver shall wait for a maximum of NUM_ARQ_RETRIES tries of the sender, after which it shall assume that the PDU has been lost, and shall proceed as though the PDU has been received and sent up the network stack.

3.2.8 Summary of the MAC Management Messages

SRAWAN management messages are sent on either *basic* or *primary* connection similar to that in WiMAX. Basic CID is used for exchanging only periodic *ranging* messages while primary CID is used for transmission of other management packets. SRAWAN defines and uses a very small subset of MAC management messages defined in WiMAX as given below.

1. **Beacon:** This is broadcasted periodically by the BS on broadcast connection in the beginning of the downlink. This contains ULMAP which defines the uplink timeslots for various Ss in the subsequent uplink.
2. **Ranging Request:** After the SS chooses which BS to associate with, it will send a ranging request to that BS on initial ranging CID for the purpose of ranging. Ss also send ranging requests periodically to be in sync with BS clock.

3. **Ranging Response:** The BS responds to SSs initial Ranging Request with a Ranging Response which contains a basic CID, primary CID. This is sent on initial ranging CID. All ranging responses for periodic ranging requests are sent on basic CID.
4. **Registration Request and Response:** The SS and the BS exchange these messages respectively for the purpose of registration.
5. **Connection Request, Response and Ack:** These messages are used for the creation of a connection.

3.2.9 Limitations of the scope of SRAWAN

SRAWAN does not specify:

- How the various SS may be placed or directed (to reduce interference)
- What the transmit powers of the BS and SS may be (again, to reduce interference)
- How various BS may operate in physical proximity of one another (they may use various frequencies of operation, or may adjust their transmit power to allow such operation)
- How a BS may schedule transmission intervals for the various SS

The specification of the above are beyond the scope of SRAWAN.

3.3 Implementation

This section shows how we achieved SRAWAN on the top of Atheros AR5212 802.11 hardware and also some of the features of the implemented driver. The implementation is a combined effort of the author, Narasimha Puli Reddy of MTech 2004, and Pratik Sinha of Zazu Networks.

Commodity 802.11 hardware typically divides the functionality of the MAC between hardware and software/driver running on that driver. So the flexibility of the WiFi systems varies greatly between manufacturers. Of particular interest in my work are the AR5212 chipsets designed by Atheros corporation and the proprietary *Multiband Atheros Driver for WiFi* (MADWIFI) driver. Atheros provides a *Hardware Abstraction Layer* (HAL) which is

an interface to the hardware. We used the flexibilities in the HAL to enable/disable many features of 802.11 to build SRAWAN on AR5212 Atheros chipsets. This section gives an overview of the SRAWAN implementation.

The following are the primary tasks that are needed for the SRAWAN implementation.

1. Overriding the 802.11 MAC PDU - In the implementation we have decided to encapsulate the SRAWAN MPDU in 802.11 payload. The reason for this design decision is as follows. A part of the MAC (packet filtering based on BSSID and filters, sending immediate ACK on reception of packets etc) is incorporated in the 802.11 hardware. This functionality provided by the 802.11 hardware can be used only if the received packet format complies with the 802.11 standard.
2. Disabling immediate MAC level acks - The sender should not wait for ack after the transmission of the packet and the receiver should not send an ack after reception of the packet. Since the protocol is TDMA-TDD based, there is no need for MAC level acks which will interfere with the ongoing packet transmission.
3. Disabling physical carrier sensing i.e. CCA (Clear Channel Assessment) - CCA and NAV are incorporated in 802.11 hardware to support CSMA/CA operation which is contention based. SRAWAN being TDMA based no longer needs these carrier sensing mechanisms. The nodes should transmit irrespective of whether the medium is free or not. Otherwise the Ss will loose their synchrony with the BS (because the nodes will do backoff if some legacy 802.11 node is in their vicinity) and will lead to packet collisions.
4. Disabling virtual carrier sensing i.e. NAV (Network Allocation Vector) - The reason for this feature is same as given for disabling of CCA.
5. Disabling the backoff feature - We no longer need backoff mechanism of CSMA/CA as the MAC protocol is no longer contention based.
6. Elimination of RTS/CTS exchange - These are not required for the TDMA-based SRAWAN protocol.
7. Nullifying SIFS, DIFS, EIFS and slot duration because SRAWAN is TDMA based.
8. A microsecond granular timer and interrupt generator is required for tight time synchronization

9. Using short preamble instead of long preamble. This is not a necessity, but would surely help in the increase of throughput performance. 802.11 PHY uses a preamble size of 24 bytes transmitted at 1Mbps which takes 192 microseconds. Atheros hardware provides a flexibility of sending short preamble which takes only 96 microseconds.
10. Introducing two new modes called as BS and SS in the *Madwifi* driver which have their own state machine that goes through the various phases explained in the above sections

Performing each of these tasks require modifying various portions of the Madwifi driver and the HAL. This section outlines these modifications in the driver. Tasks 2 to 7 and 9 required enabling/disabling certain bits in some PHY/MAC registers of the hardware. We are not successful in disabling the CCA altogether. So we set the CCA threshold high enough so that it virtually ignores any energy in the channel. With respect to the MPDUs, we had encapsulated the SRAWAN MPDUs in the 802.11 MPDU as the payload and the sender and the receiver code are modified appropriately to handle the transmission and reception of the packets.

The driver is based on interrupt driven model. The following are some of the features that are implemented in the driver apart from the tasks described above.

1. Operates in one of the two modes: SS or BS
2. The BS will provide UGS (unsolicited grant service) on each of the connections. For this reason, there shall be no bandwidth request contention slots in the uplink sub-frame. Instead we have connection request contention slots in place of bandwidth request contention slots. By this we mean that the bandwidth allotted to a connection is decided at the time of connection establishment and cannot be increased or decreased later on. The reasons for implementing only UGS are as follows. The subscribers do not know the amount of bandwidth they require apriori. The implementation is for providing internet connectivity to subscribers who pay for a limited amount of bandwidth support. So during the connection establishment, the SS requests BS to allot a specified amount of bandwidth and the BS accepts the connection request only if it can entertain the request.
3. The beacon sent by the BS is at the lowest possible PHY bitrate: 6Mbps for 802.11a/g and 1Mbps for 802.11b. This is to ensure that all SS can hear the BS reliably.

4. The txpower used by the BS and each SS shall be the maximum possible txpower allowed by regulations within the country/region of operation.
5. For now SS does not scan all the available channels to select a BS. The SS has to be configured with channel, transmit power and transmit rate before it is started.
6. The BS uses round-robin mechanism to schedule packet transmissions at the MAC layer both for uplink and downlink. So the bandwidth is equally divided among all the connections. The packets from the higher layer are queued up in the software queues for the respective connections and then the control over the packets is transferred to hardware when the packets are scheduled by the driver.
7. ARQ (link layer retransmission) mechanism is implemented to handle packet losses at MAC layer. We implemented a sliding window based per-connection ARQ. The MAC header of the ARQ enabled connections consists of both the selective and cumulative ACK. A detailed description of the header format and the ack scheme is given in subsection A.2.3 of Appendix. The sender buffers the packets in an ARQ buffer and removes it from the buffer only after receiving an ack. It retransmits the packet only for NUM_ARQ_RETRIES times in case of loss and then removes it from the buffer and slides the window appropriately.

3.3.1 Scheduling

This subsection gives a brief overview of the packet scheduling mechanism incorporated at the BS. Lets assume that each connection requires $x/2$ kbps downlink bandwidth and $x/2$ kbps uplink bandwidth, where x is the maximum bandwidth to be allotted for a connection. We also know the amount of time taken to transmit a packet (transmit rates are fixed apriori). So we can compute the Packet Transmission Slots (PTS) that have to be assigned to a connection over a period of time (say 1 sec) so that it gets $x/2$ kbps downlink bandwidth and $x/2$ uplink bandwidth. So we will maintain a counter (for PTS) for every connection at the BS (for downlink and for uplink) which is initialized to PTS. During the downlink the BS will schedule the packets in a round robin way. If a connection has no packets to send, it is skipped. If a connection has packet to send, then it is transmitted and the counter value is reduced by 1. Once the counter value reaches 0, no more packets are transmitted for that connection until the counter values are reset. The counter values are periodically

reset to the initial values (PTS) at a regular interval (say 1 sec). During the uplink, the BS will allot the slots in a round robin way for all the connections. This is very inefficient. This can be easily modified by allowing Ss to request for additional bandwidth after the connection establishment.

3.3.2 Implementation parameters and Throughput Estimation

This subsection analyzes and estimates the expected throughput that SRAWAN can yield when operated on 802.11b. It also outlines some of the parameters we have fixed for evaluating the performance of SRAWAN as shown in Figure 3.6.

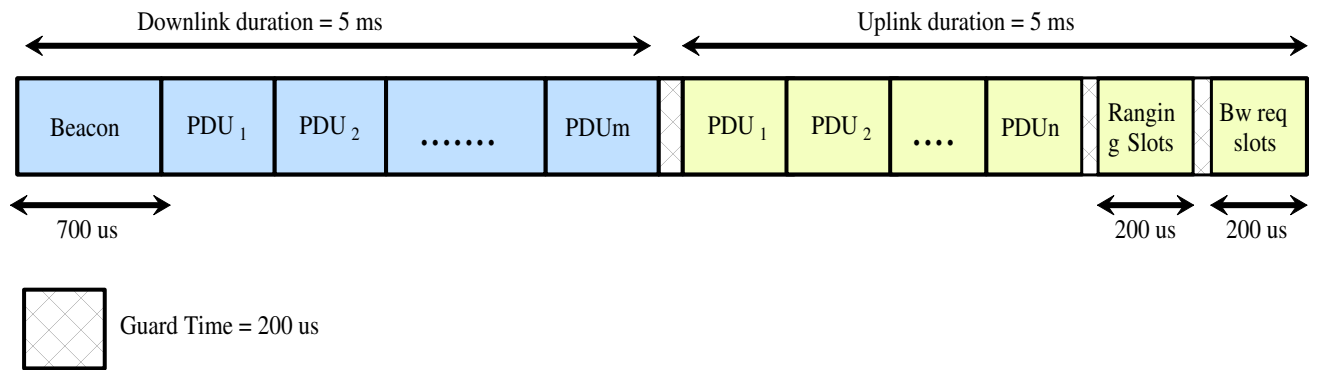


Figure 3.6: Frame structure showing various implementation parameters

- Frame duration = 10 msec. Very small frame durations will lead to lower throughput because of overheads incurred due to beacon transmission and ranging and connection request contention slots. On the other hand very large frame duration will lead to better throughput but higher response times. We have verified this with various frame durations like 7 ms, 10 ms and 15 ms. We found out that as frame duration increases, the throughput increases slightly. This is because the overhead per frame (ranging and connection request slots) remains constant irrespective of frame duration.
- Uplink time : Downlink time = 1:1 (This will not effect the net throughput of the network. But in reality, generally the uplink traffic will be lower downlink traffic and hence it would be better to have larger downlink duration)

- Downlink duration = 5 msec (700 microseconds of DL is used by beacon and 100 microseconds is used as the guard time at the end of DL)
- Uplink duration = 4 msec
- Guard time = 200 microseconds. Guard time is so high because we experienced a time skew of about 150 to 200 micro seconds even after time synchronization. So in order to prevent collisions between packets at the end of DL and at the beginning of UL or between different SSs in the UL, this guard time is chosen.
- MTU size = 1500 bytes (default MTU size)
- Ranging slots = 200 usec (Since at most 10 clients are expected to be in the coverage of a single BS and not all the SSs will boot at the same time to send the ranging requests, we have kept such low ranging slot duration. Larger durations will result in a lot of overhead)
- Bandwidth Contention slots = 200 usec (The reasons for this low duration value is same as given for ranging slots)
- Total guard time in UL = 600 μ s (200 μ s between UL and ranging slots, 200 μ s between ranging slots and connection request slots and 200 μ s between connection request slots and beginning of next frame)
- NUM_ARQ_RETRIES = 4. We choose this value because it is large enough to handle retransmissions for errored packets and small enough to interact negatively with TCP. If the number of retries is large, then it may happen that the TCP might also retransmit the same packet. At the MAC layer, the sequence numbers for both the packets are different and so the MAC transmits the same packet many times without its knowledge.
- Size of a typical packet = 1500 - 20 (ethernet header) + 6 (srawan header) + 24 (802.11 hdr) = 1510 bytes
- Time taken for transmission of 1510 byte packet at 11 Mbps = $1510 * 8/11$ usec = 1098 usec
- Total transmission time = 1098 + 96 (PHY short preamble) = 1194 usec

- So number of downlink packets that could be transmitted in a frame = 3 i.e. $3 \times 1194 = 3582$ usec
- Time taken for the Synch packet + FCH + ULMAP transmission at 1Mbps ≤ 700 usec (this could also include the slots assigned for registration responses and connection establishment responses)
- Similarly number of up link packets of size 1500 bytes that could be transmitted = 3 i.e. $3 \times 1194 = 3582$ usec
- So typically we can send/recv 3 + 3 packet of size 1510bytes in a frame duration i.e. 10msec.
- Effective throughput at the MAC layer is $6 \times 1510 / 10 \text{Mbps} = 7.25 \text{Mbps}$
- So effective throughput at the transport layer is $6 \times (1510 - 26 - 6 - 20 - 8) / 10 \text{Mbps} = 6.96 \text{Mbps}$ (MAC header is 26 bytes long, SRAWAN header takes 6 bytes, IP header takes 20 bytes and 8 bytes is consumed by UDP header)
- If the MTU size is reduced to 1000, a higher throughput can be achieved as 10 packets of size 1000 bytes can be transmitted in 10msec. As can be seen when MTU size is 1500, around $400 \mu\text{s}$ is being wasted in both DL and UL. This wastage can be avoided by reducing the MTU size to 1000.
- Effective throughput becomes $10 \times 1000 / 10 \text{Mbps} = 8 \text{Mbps}$
- So effective throughput at the transport layer is $10 \times (1000 - 6 - 24 - 40 - 20) / 10 \text{Mbps} = 7.28 \text{Mbps}$

Chapter 4

Experiment Results

In the previous chapters we have seen how to incorporate 2P and SRAWAN on the top of off-the-shelf Atheros AR5212 802.11 chipsets. In this chapter we evaluate the performance of SRAWAN and 2P based on our implementation and answer the following significant questions.

- Is 2P feasible outdoors? If yes, what is the performance of 2P in comparison to CSMA/CA on a network of 2 links (4 interfaces)?
- Is SRAWAN feasible outdoors? Is it possible to achieve tight time synchronization for the successful operation of multiple SSs within the coverage of a single BS?
- What is the performance of SRAWAN in comparison to CSMA/CA on a network of 3 nodes (one BS, two SSs)?

4.1 Software and Hardware

All the experiments are based on two drivers. They are open source *Madwifi* VR1406 [11] and *HostAP* V0.3.7 [12]. 2P and SRAWAN have been incorporated in proprietary version of the *Madwifi* driver which also has the *HAL* code.

Experiments involving *Madwifi* driver (either open source or proprietary) have been conducted on *Linux* V2.6.10 platform in *Pebble* [10] environment. We used Atheros AR5212 chipset based *Ubiquiti SR2* [16] and *Wistron CM9* [14] 802.11 miniPCI cards for experiments. Single board computers namely *WRAP* boards [13] and *Soekris* boards [18] have

ports for miniPCI 802.11 cards and also has a port for removable flash. These boards are used for operating 802.11 cards.

Experiments involving *HostAP* driver have been conducted on *Linux* V2.6.5 platform in *Fedora Core V2* environment. We used *Prism2* chipset based *Senao* [17] 802.11 PCMCIA cards for experiments. The experiments are conducted on laptops.

The other equipment that has been used in the experiments are senao pigtails, RF cables, 2.4 Ghz antennae, RF splitter and RF attenuator.

4.2 Results of disabling NAV and CCA

As explained in the previous chapters, successful operation of 2P and SRAWAN requires disabling of virtual carrier sensing (NAV - Network Allocation Vector) and physical carrier sensing (CCA - Clear Channel Assessment). We do not require these because the protocols are not contention based. So this section outlines some of the experiments that are conducted to test whether disabling of NAV and CCA can be achieved or not.

NAV can be disabled by setting a particular bit in one of the hardware MAC register. CCA can also be disabled in the same way as NAV. But we observed that disabling CCA resulted in random packet losses due to some error in the firmware. So we tried to work around this problem by setting CCA threshold high enough such that it virtually ignores any energy in the channel. The following experiment shows the effect of this setting.

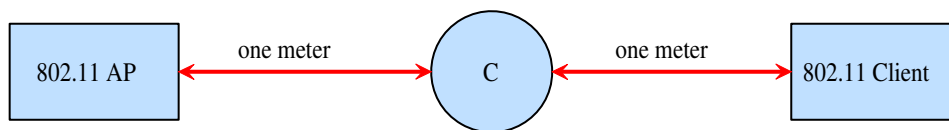


Figure 4.1: Experiment setup to test NAV and CCA disabling

The experiment setup consists of two legacy nodes (802.11b AP and 802.11b client). Another node C is also placed in the same room. The three nodes are almost collinear with C in between AP and 802.11 client as shown in Figure 4.1. C is a modified version of *Madwifi* driver. We have disabled immediate ACKs, RTS/CTS at C. The beacon interval is set to 50ms. After booting C, the MAC (driver) keeps on adding junk data packets of size 500 bytes to the hardware queue. The hardware in turn transmits these data packets whenever it finds the medium to be free. Depending upon whether CCA and NAV are

disabled or not, it might follow 802.11 backoff mechanism for transmitting packets. So in a way C is intended to hog the bandwidth. We collected the TCP throughput from AP to client in various cases shown in the Table 4.1.

Is C operating	NAV disabled at C	CCA threshold set at C	TCP/SCP throughput from AP to client (KBPS)
No	No	No	850
Yes	No	No	450
Yes	Yes	No	235
Yes	No	Yes	200
Yes	Yes	Yes	185

Table 4.1: TCP throughput for testing CCA and NAV

The summary of the results shown in Table 4.1 is as follows. In normal mode of operation, the TCP throughput between AP and its client is 850 KBPS and the throughput reduces to half after C is started. This is because C and the AP are sharing the same channel and CSMA/CA makes sure that the channel is shared fairly. Disabling of NAV at C resulted in reduction of throughput from AP to client by four times. Setting of a high CCA threshold also resulted in a huge decrease of throughput from AP to its client. This is because C takes control of the medium as soon as it gets free whereas legacy AP follows 802.11 DCF by backing off for atleast DIFS amount of time when the transmission of a packet is finished. Ideally the throughput from AP to client should be 0. But the final result in Table 4.1 shows that C also does backoff sometimes which is the only way by which legacy AP gets control over the medium to transmit to its client.

4.3 Performance evaluation of CSMA/CA and 2P on a point-to-point network

The experimental setup is described in Figure 4.2. We have 2 point-to-point links at FBTop. Interface N_1 at FBTop (Faculty Building Top) is connected to interface B at MLA (Media Lab Asia, IIT Kanpur) and interface N_2 is connected to interface C at CSE (Computer Science and Engineering Department, IIT Kanpur). Interfaces N_1 , N_2 and B

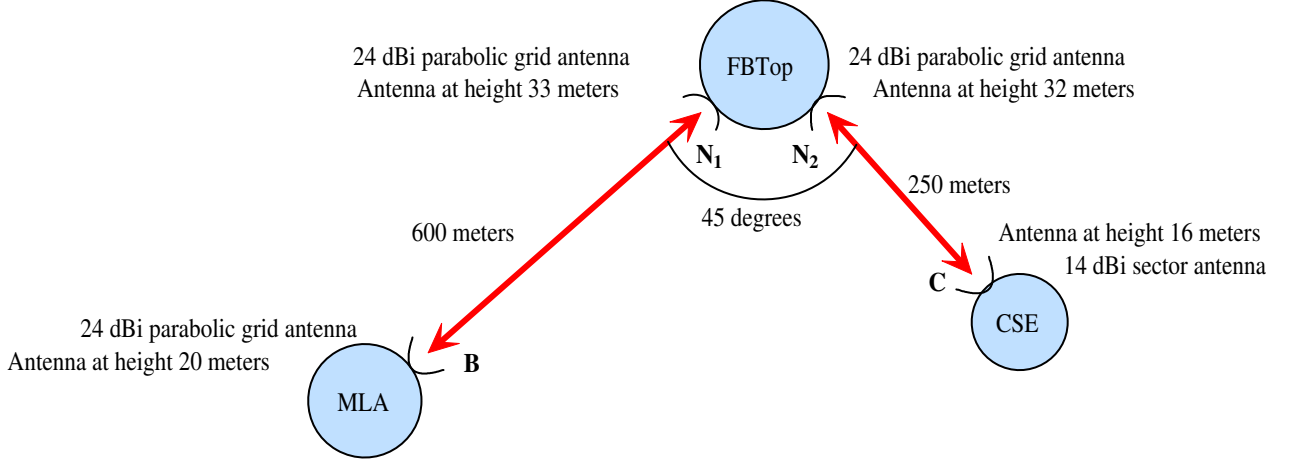


Figure 4.2: The two link point-to-point system used for 2P experimentation

are connected to 24dBi *parabolic grid antennae* and interface C is connected to 14dBi *sector antenna*. Depending upon the driver used, we used appropriate hardware (as mentioned in section 4.1) to conduct the experiments. The 802.11 cards are connected to the antennae with the help of *Senao Pigtailes* and RF cables. The tower heights at all the ends and the angle of separation between the two point-to-point links are shown in the figure. Interfaces N_1 and N_2 will interfere with each other if links N_1B and N_2C operate on same channel. Both the links are made symmetrical by the use of sufficient amount of attenuation at interfaces B and C as given in the Table 4.2.

Interface	Attenuation	Signal level
N_1	0	-66 dB (from B)
N_2	0	-66 dB (from C)
B	5 dB	-66 dB (from N_1)
C	23 dB	-66 dB (from N_2)

Table 4.2: Signal levels

We had saturated bidirectional UDP traffic on each link with 1400 byte packets being transmitted every $1ms$. The time interval is good enough to saturate the links with UDP packets. With this, we observed an average throughput of 6 Mbps for *HostAP* and 7.2 Mbps for *Madwifi*. This is close to the maximum transport layer throughput that can be achieved for 11Mbps raw transmission after deducting the overheads due to PHY preamble,

carrier sense based backoff, MAC, IP and UDP headers. The expected overhead per packet and the expected throughput for CSMA/CA is calculated as follows.

- Time taken to transmit 24 byte PHY preamble = $192 \mu s$
- Time taken to transmit 34 byte MAC header at 11Mbps = $34 * 8/11 \mu s = 24.72 \mu s$
- Time taken to transmit 20 byte IP header and 8 byte UDP header = $28 * 8/11 \mu s = 20.36 \mu s$
- With $20 \mu s$ slot time and a minimum contention window CW_{min} of 32, average backoff period per packet = $320 \mu s$
- Therefore time taken to transmit 1400 byte UDP packet = $192 + 24.72 + 20.36 + 320 + 1400 * 8/11 = 1575 \mu s$
- Therefore average expected throughput = $1400 * 8/1575 \text{ Mbps} = 7.1 \text{ Mbps}$

With *Madwifi* driver, we have been able to achieve the estimated throughput on outdoor links. But with *HostAP* we have been able to achieve a maximum of 6Mbps throughput on outdoor links. The reason for this is because *HostAP* and the *Prism* chipset based cards do not provide support for DMA.

Now lets consider the CSMA/CA operation on the above mentioned network of 2 links. Table 4.3 gives the throughput results of 802.11b CSMA/CA operation on this network when the two links are operated on different channels. Similarly Table 4.4 gives the throughput results of CSMA/CA when both the links are operated on same channel. We tested with *Madwifi* and *HostAP* open source drivers.

Driver	Throughput on link N_1B (Mbps)	Throughput on link N_2C (Mbps)
Madwifi	7.5	6.9
HostAP	6.1	5.8

Table 4.3: Performance of CSMA/CA on two p2p links operated on channels 1 and 6

The data shows that when the two links are operated on same channel, they tend to share the channel. One can observe a slight decrease of throughput in case of same channel operation. This is because of collisions due to hidden node problem as explained in the introduction. After several runs in the experiment, we observed that the distribution of

Driver	Throughput on link N_1B (Mbps)	Throughput on link N_2C (Mbps)	Total
Madwifi	4.1	2.9	7.0 Mbps
HostAP	3.0	2.8	5.8 Mbps

Table 4.4: Performance of CSMA/CA on two p2p links both operated on channel 1

the bandwidth on links N_1B and N_2C differ for every run. This is because CSMA/CA does not ensure fairness among the links.

Table 4.5 and Table 4.6 gives the throughput results of 2P operation on the above mentioned network when operated on different channels and same channel respectively. These measurements help us to evaluate the feasibility and performance of 2P implementation outdoors. The single channel 2P experiment has been conducted in channel 6 because it was found to be the only free channel on our testbed.

Exp	Throughput on link N_1B (Mbps)	Throughput on link N_2C (Mbps)
Indoor	6.6	6.4
Outdoor	6.3	6.23

Table 4.5: Throughput results of 2P on two p2p links operated on channels 1 and 6

Exp	Throughput on link N_1B (Mbps)	Throughput on link N_2C (Mbps)	Total
Outdoor	4.3	4.1	8.4 Mbps

Table 4.6: Throughput results of 2P on two p2p links operated on channel 6

The results show that 2P outperforms CSMA/CA and is able to achieve a significant amount of increase in throughput. If we can set the power settings accurately satisfying the power equations given in [2], we can achieve higher throughput improvement using 2P. Some amount of overhead is because of the exchange of ethernet messages between the two interfaces at FBTop for synchronization (to simultaneously transmit or receive). We observed an average time gap of around $700\mu\text{s}$ between NOTIF from *Master* and NOTIF_ACK from *Slave*. Ideally this time gap should be the sum of transmission delays of NOTIF and NOTIF_ACK ethernet packets. The reason for this overhead is because of the transmission and reception of these ethernet messages using *NetFilter hooks* in *Linux* kernel on low processing power WRAP boards.

Earlier we tried to handle this problem by operating the two interfaces on the same WRAP board and synchronizing them using shared memory (WRAP board has two minipci slots). Since the interfaces are very near to each other, the resultant signal level at each interface from its neighbor interface is around $-30dBm$. Because of this huge signal level, only one of the interfaces is able to transmit while the other interface backoffs (CCA). Even after setting the CCA threshold to the maximum possible value, we observed that one interface takes control of the channel while the other interface repeatedly backoffs. This shows that setting the maximum possible CCA threshold will not result in complete disabling of CCA.

Table 4.7 shows the resultant signal levels of one interface’s transmission at the other interface and vice versa. We have observed this for the *Wistron CM9* cards [14], [15] which transmit at much lower power ($0dB$) compared to *Ubiquiti SR2* cards [16] ($26dB$). The leakage from the pigtailed was so high that it resulted in a signal level of $-30dB$ at the neighbor interface on the same WRAP board.

Card	Experiment	Signal level at interface 1 because of interface 2	Signal level at interface 2 because of interface 1
CM9	Interfaces connected to no external antennae	-77 dBm	-75 dBm
CM9	Interfaces connected to external antennae with 3 dBi gain	-33 dBm	-32 dBm
CM9	Interfaces connected to netgear pigtailed	-30 dBm	-30 dBm

Table 4.7: Signal levels when two interfaces are mounted on same WRAP board

This was observed even when the pigtailed are connected to RF cables as explained in the experiment below. The experimental setup is shown in Figure 4.3. The WRAP board with two miniPCI 802.11b cards is placed in room R_3 . Each of the interfaces is connected to a different terminator (interface in *Monitor* mode) with the help of pigtailed and RF cables. The terminators are placed in rooms R_1 and R_2 . Both the terminators are kept far apart from WRAP board such that they do not hear any leakages from the interface to which they are not connected to. In this setup also, we observed that the resultant signal level at each interface from its neighbor interface is around $-50dBm$. This shows that

using shared memory scheme for coordination among *ifa-nbrs* is not possible.

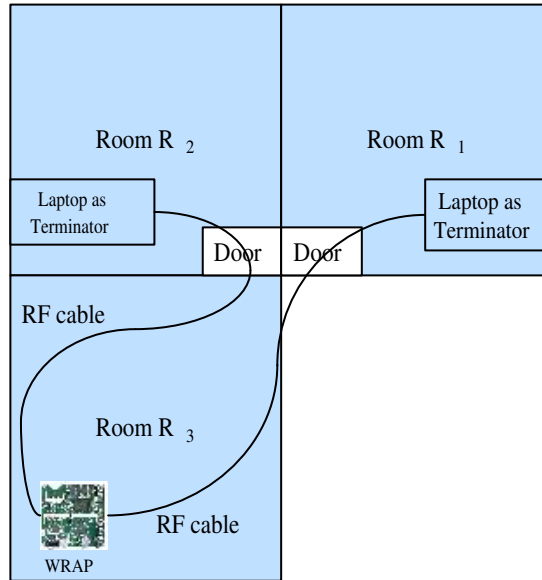


Figure 4.3: Experimental setup for shared memory coordination scheme

4.4 Performance analysis of 2P in SynTx and SynRx phases

While conducting the experiments mentioned in the above section, we observed that the throughput in SynRx phase is lesser than throughput in SynTx phase by a significant amount. This section tries to seek the reason for this behavior.

Phase	Throughput on link N_1B	Throughput on link N_2C
SynTx	2.3 (from N_1 to B)	2.6 (from N_2 to C)
SynRx	2.0 (from B to N_1)	1.5 (from C to N_2)

Table 4.8: SynTx and SynRx throughput results

The experimental setup is same as given in Figure 4.2. We used saturated bidirectional UDP traffic (1400 byte packets at 1ms interval) for measuring throughput. Table 4.8 shows the results of throughput from/at interfaces N_1 and N_2 when they are in SynTx and

SynRx phases. The net throughput during SynRx phase is 1.4 Mbps lower than during SynTx phase. The throughput is especially lower at interface N_2 on link N_2C . During the experiments we have observed that interface B 's transmission to N_1 interferes with the ongoing reception at N_2 from C whereas interface C 's transmission to N_2 does not interfere with the ongoing reception at N_1 . This interference at N_2 is the cause for low throughput at N_2 during SynRx. To prove this we conducted an experiment on this network as described below.

Firstly start interface B in AP mode and start interfaces N_1 and N_2 as clients of B . We observed that B induced a signal level of $-66dBm$ and $-85dBm$ at N_1 and N_2 respectively. Now conduct the same experiment with AP at C and clients at N_2 and N_1 . We observed that C induced a signal level of $-66dBm$ at N_2 and N_1 was not able to associate to C because it could not see any packets from C . This is due a concrete wall that covers N_1 from C . So this experiment clearly shows that B is interfering with the ongoing reception at N_2 .

We re conducted the SynTx and SynRx experiment twice. But this time the transmission power at B is reduced by $5dB$ for first experiment and by $10dB$ for second experiment. This is done to reduce the interference caused by B at N_2 . We noticed an improvement in throughput during SynRx phase. But at the same time there was a reduction in the throughput during SynTx phase. The net throughput also decreased to 7.5 Mbps from the earlier 8.4 Mbps ($2.3 + 2.6 + 2.0 + 1.5$).

4.5 Performance of CSMA/CA and SRAWAN on two links

In the previous sections we evaluated the performance of 2P in comparison with CSMA/CA on a point-to-point network of 2 links. This section explains the feasibility of SRAWAN outdoors and evaluates the performance of SRAWAN on a point-to-multipoint network of 2 links with one BS and two SSs.

The experimental setup is described in Figure 4.4. We have 2 point-to-point links at FBTop. Antenna N_1 at FBTop is aligned to antenna B at MLA and antenna N_2 is aligned to antenna C at CSE. N_1 , N_2 and B are 24dBi *parabolic grid antennae* and C is a 14dBi *sector antenna*. N_1 and N_2 are connected to the same interface (802.11b based ubiquiti SR2 card) with the help of splitter, senao pigtails and RF cables as shown in the figure. A

splitter is an RF device which splits the RF signal into two equal signals. So this network essentially works as a point-to-multipoint link. We used pebble [10] platform installed on WRAP boards to operate the cards. Other details like the tower heights at all the ends and the angle of separation between the two point-to-point links are shown in the figure. Both the links are made symmetrical by the use of sufficient amount of attenuation at MLA and CSE ends. The traffic pattern used is similar to that described in section 4.3. We used saturated bidirectional UDP traffic by generating 1400 byte packet at $1ms$ interval.

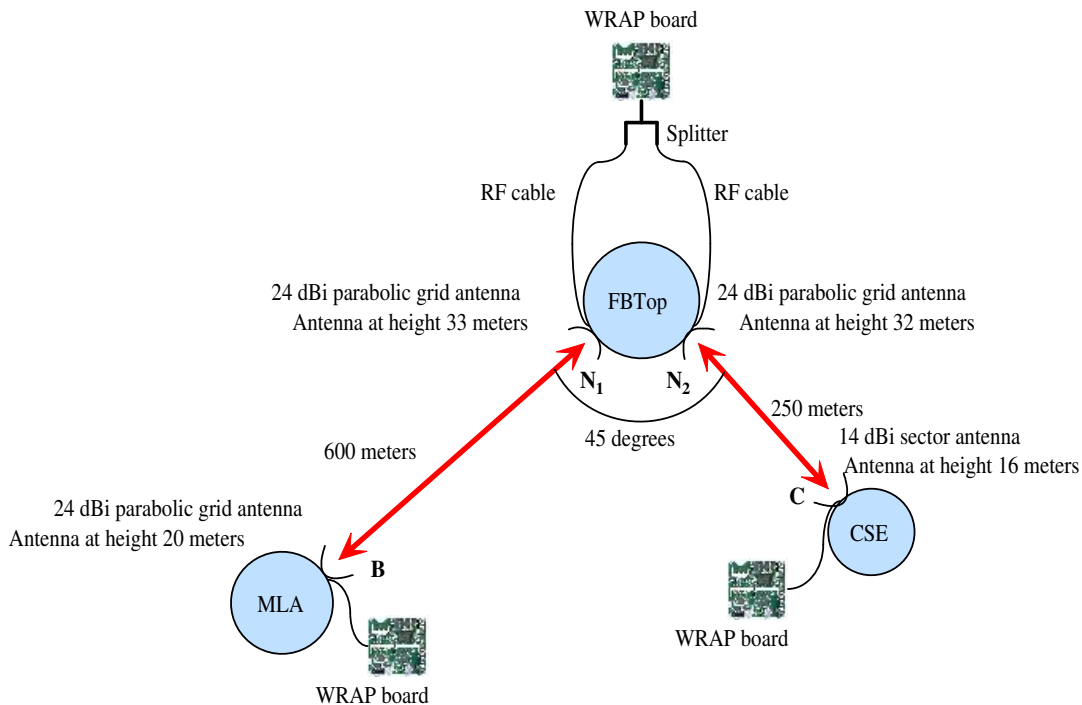


Figure 4.4: Point-to-multipoint link with splitter

Table 4.9 shows the performance of CSMA/CA on this point-to-multipoint network. The experiment has been conducted twice once with RTS/CTS disabled and then with RTS/CTS enabled. Similarly Table 4.10 shows the throughput results of SRAWAN operation on the same network. It can be seen that the throughput of CSMA/CA drops from 7.1Mbps (max possible throughput as calculated in 4.3) to 6.3Mbps in case of *Madwifi* and from 6Mbps (max possible for *HostAP*) to 4.1Mbps in case of *HostAP*. Hidden nodes are the reason for this drop. Use of RTS/CTS to handle hidden nodes does not help and instead resulted in even lower throughputs. The reasons for this behavior are not known.

On the other hand, SRAWAN gives uniform throughput of 6Mbps for both single link operation and two link operation. A slight decrease in throughput is because of the use of guard time ($200\mu\text{s}$) to prevent collisions between multiple clients in the uplink.

Driver	RTS/CTS	Effective throughput (link1, link2 in Mbps)
Madwifi	disabled	6.3 (2.31, 3.97)
Madwifi	enabled	5.7 (2.5, 3.2)
HostAP	disabled	4.1 (2, 2.1)
HostAP	enabled	3 (1.5, 1.5)

Table 4.9: Throughput results of CSMA/CA on two links

Exp	Effective throughput (link1, link2 in Mbps)
Indoor	5.78 (2.95, 2.83)
Outdoor	5.70 (3.00, 2.70)

Table 4.10: Throughput results of SRAWAN on two links

4.6 Performance of CSMA/CA, SRAWAN, and 2P on single link

We have evaluated the performance of 2P and SRAWAN in comparison to CSMA/CA on a network of two links. This section evaluates the performance of 2P and SRAWAN in comparison to CSMA/CA on single link. We also contrast the time synchronization mechanisms of 2P (loose) and SRAWAN (tight) in this section. The experimental setup is described in Figure 4.5. We have a point-to-point link from FBTop to Mohanpur (a village near IIT Kanpur). All the details like link length, tower heights and the antennae type are given in the figure. Depending upon the driver, we used appropriate hardware as mentioned in section 4.1. The traffic pattern is a saturated bidirectional UDP flow similar to that given in section 4.3. Table 4.11, Table 4.12, and Table 4.13 give the throughput results of the single link operation of CSMA/CA, 2P and SRAWAN respectively.

All the three protocols more or less give same amount of throughput on a single link. For all the protocols, we observe that the throughput in outdoor environments is slightly lesser

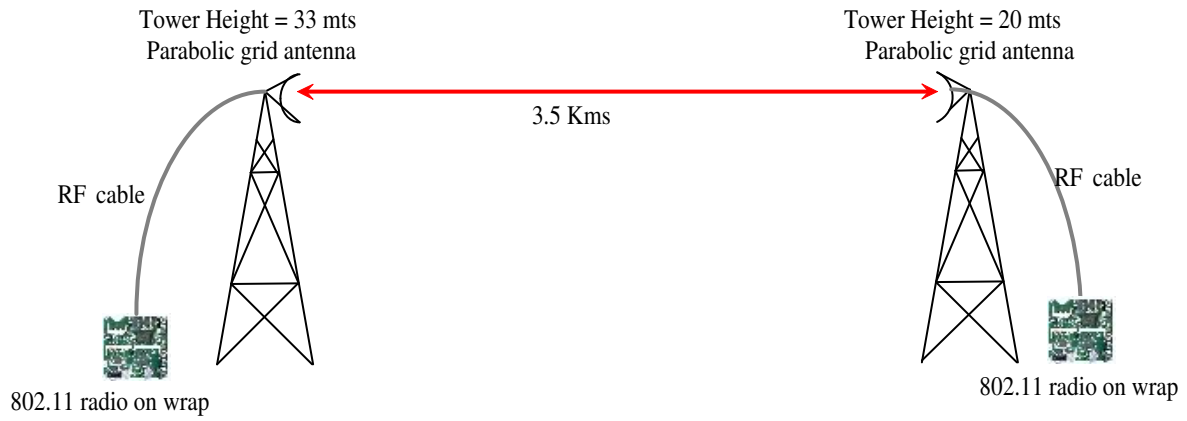


Figure 4.5: FBTop - Mohanpur point-to-point link

Driver	Exp	Average throughput (Mbps)
Madwifi	Indoor	7.70
Madwifi	Outdoor	7.42
HostAP	Indoor	6.35
HostAP	Outdoor	5.95

Table 4.11: Throughput results of CSMA/CA on single link

Exp	Average throughput (Mbps)
Indoor	6.86
Outdoor	6.33

Table 4.12: Throughput results of 2P on single link

Exp	Average throughput (Mbps)
Indoor	6.28
Outdoor	5.95

Table 4.13: Throughput results of SRAWAN on single link

than in indoor environments. This might be due to CRC errors in the received packets. The theoretical estimated throughput for CSMA/CA, 2P and SRAWAN are 7.1 Mbps (section 4.3), 6.72 Mbps (section 2.5) and 6.96 Mbps (section 3.3.2) respectively. So the experimentally observed throughput values in all the cases except CSMA/CA operation on *Madwifi* driver are lesser than the theoretical estimations. While calculating the theoretical estimations, we did not take into account the processing delays incurred in the transmit and receive path of the packet in the *Linux* kernel, 802.11 MAC driver and the 802.11 hardware. This delay is even higher when the platform used is WRAP board because of its low processing power. These processing delays are the reason for the slight decrease in throughput values in experiments. In case of CSMA/CA operation in *Madwifi* driver, the experimental value is slightly higher than theoretical estimation. The reason for this is not known.

The throughput can be increased for SRAWAN if it has to operate on a single link by removing the ranging and bandwidth contention slots of 1 msec out of 10 msec long frame. So the throughput is expected to increase to 7.4Mbps. However we might still need a guard time of around 200 micro seconds between downlink and uplink to avoid any possible packet collisions.

So for operation on a single link, it is preferable to use tight time synchronization as it results in better throughput (7.4 Mbps). On the other hand, loose time synchronization results in some amount overhead because of exchange of control packets for synchronization. The amount of overhead per 10ms because of these control packets is $600\mu s$ as shown in section 2.5. In the experiments conducted on the above mentioned setup Figure 4.5, we observed no control packet losses. So there were no 2P timeouts. This is because the control packets are sent at the lowest possible data rates (1 Mbps). If the link is lossy, then there is a possibility of loss of control packets and the nodes might go out of synchrony (in case of 2P). But this will not happen in SRAWAN because if a beacon (which contains all the control data regarding when uplink should start) is lost, then the SS will remain idle. So for lossy links, tight time synchronization is the best possible option. For single link operation, TDMA based protocols are suited only if there is a continuous traffic from both ends. For bursty traffic, CSMA/CA is the better option.

Chapter 5

Conclusions

802.11 is a cost-effective solution to provide internet connectivity to rural villages from district headquarters or nearby towns. In this thesis, we described the design, implementation and experimental evaluation of two MAC protocols SRAWAN and 2P that achieve better performance than CSMA/CA over this cost-effective technology. SRAWAN and 2P are replacement for 802.11 CSMA/CA for point-to-multipoint and point-to-point long distance networks respectively. While SRAWAN requires tight time synchronization to achieve maximal throughput, 2P does it with loose time synchronization. Even though achieving tight time synchronization in SRAWAN over long distance links is a very difficult task, we have been able to synchronize all the nodes in the network with the use of proper guard time. We have shown through outdoor experiments on *Digital Gangetic Plains* testbed that SRAWAN and 2P can perform several times better than CSMA/CA. This work also proves that TDMA based MAC protocols can be implemented on top of cost-effective 802.11 PHY.

802.11 is an open, inter operable standard for WLANs. The reason for cost effectiveness is the interoperability and mass production. The protocol is inefficient for outdoor long distance links. Currently the hardware that can support long distance links use proprietary protocols. They are very costly because of poor interoperability and low usage. We therefore expect the design and performance study of SRAWAN and 2P to help in the formulation of future standards for long distance wireless networks which will lead to better inter-operation between products of different vendors. The protocols and the implementation also have an immediate applicability in *Ashwini* network at Bhimavaram, Andhra Pradesh. *Ashwini* network is built by an NGO *Byrraju Foundation* to connect several

remote villages to district headquarters and thereby provide medical (telemedicine) and telephone facilities. We therefore conclude that SRAWAN and 2P together is a promising solution to operate on such networks.

The future work includes the extension of 2P for multi-hop point-to-point networks and extension of SRAWAN for multi-sector point-to-multipoint networks. The effect of ARQ on SRAWAN and 2P implementation also has to be tested and evaluated outdoors.

Appendix A

MAC PDU and Header Formats in SRAWAN

MAC PDU is the data unit exchanged between MAC layer of the peer entities. The format of the MAC PDU is illustrated in Figure A.1. Each PDU shall begin with a fixed length generic MAC header. The header is followed by the payload of the MAC PDU. The payload consist of zero or more subheaders and zero or more MAC SDUs and/or fragments thereof. The payload length is variable and so the MAC PDU is of variable size. A MAC PDU may also contain a CRC at the end.

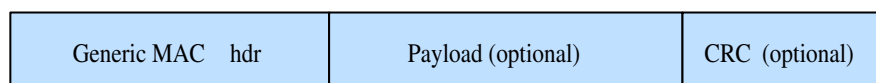


Figure A.1: MAC PDU Format

Similar to that of WiMAX, we define two types of MAC headers. The first is the generic MAC header that begins each MAC PDU containing either MAC management messages or transport data. The second is the bandwidth request header and is used to request additional bandwidth. The single bit Header Type (HT) field distinguishes the two header formats. HT is set to zero for generic MAC header and is set to one for bandwidth request header.

A.1 Generic MAC Header

Figure A.2 illustrates the generic MAC header structure. The various fields in the header are explained in the Table A.1.

HT=0 (1)	EC (1)	Type (6)	Version (2)	CI (1)	Rsvd (1)	LEN MSB (4)
LEN LSB (8)			CID MSB (8)			
CID LSB (8)			HCS (8)			

Figure A.2: Generic MAC header Format

A.1.1 Bandwidth Request Header

Bandwidth request PDU consists of bandwidth request header only and does not contain a payload similar to that in WiMAX. Figure A.3 illustrates this header format.

HT=1 (1)	EC =0 (1)	Type (3)	BR MSB (4)
BR LSB (8)			CID MSB (8)
CID LSB (8)			HCS (8)

Figure A.3: Bandwidth Request Header

Some of the features of bandwidth request header are:

	Name	Length (bits)	Description
1	HT	1	Header Type, 0 = generic MAC header and 1 = bandwidth request header
2	EC	1	Encryption Control, 0 = payload is not encrypted and 1 = payload is encrypted
3	Type	6	Indicates the Subheader and special payload types present in the message payload
4	Version	2	Protocol Version
5	CI	1	CRC Indicator, 1(0) \Rightarrow the PDU (does not) contain a CRC at the end of the payload
6	Rsvd	1	Reserved for future use
7	Length	12	Length of the MAC PDU including the MAC header and CRC in bytes
8	CID	16	Connection Identifier
9	HCS	8	Header Check Sequence is the checksum of the first 5 bytes in the MAC header

Table A.1: Description of the fields in the generic MAC header

- The length of the header is always 6 bytes.
- The EC bit is always set to zero, indicating no encryption.
- The CID indicates the connection for which uplink bandwidth is requested.
- The Bandwidth Request (BR) field indicate the number of bytes requested.
- The allowed types for bandwidth requests are "000" for incremental and "001" for aggregate.

A subscriber station can use a portion of the bandwidth allocated in response to a Bandwidth Request for a connection to send another Bandwidth Request rather than sending data. This is known as bandwidth stealing in WiMAX.

	Name	Length (bits)	Description
1	HT	1	Header Type, 0 = generic MAC header and 1 = bandwidth request header
2	EC	1	Encryption Control, 0 = payload is not encrypted and 1 = payload is encrypted. EC is set to zero for Bandwidth Request packets
3	Type	3	Indicates the type of bandwidth request header, 000 is used for incremental and 001 for aggregate
4	BR	19	Bandwidth Request is the number of bytes of uplink bandwidth requested by the SS. BR is for the CID and the request does not include any PHY overhead
5	CID	16	Connection Identifier
6	HCS	8	Header Check Sequence is the checksum of the first five bytes in the header

Table A.2: Description of Bandwidth Request header fields

A.2 MAC Subheaders and Special Payloads

This section outlines various types of sub headers and their formats. Table A.3 gives the values that are set in the type field of the generic MAC header corresponding to the inclusion of the following sub headers.

	Type bit	Value
1	5 (MSB)	Reserved as of now
2	4	ARQ subheader, 1 = present and 0 = absent
3	3	Reserved
4	2	Fragmentation subheader, 1= present and 0 = absent
5	1	Packing subheader, 1= present and 0 = absent
6	0 (LSB)	Uplink: Grant Management header, 1= present and 0 = absent

Table A.3: Type Encodings

Some of the sub headers are per-PDU and some are per-SDU. ARQ subheader is inserted immediately after the generic MAC header in case of ARQ enabled connections. The other per-PDU sub headers (Fragmentation and Grant Management) are inserted in MAC PDU immediately after the Generic MAC header and ARQ subheader if present. The only per-SDU sub header (Packing sub header) may be inserted before each MAC SDU. The fragmentation and packing sub headers will not be present with in the same MAC PDU.

A.2.1 Grant Management sub header

The grant management sub header is an alternative to bandwidth request header. This is a lightweight way by which an SS can request uplink bandwidth without having to create and transmit a complete MPDU with the overhead of MAC headers and CRCs. There are two types of grant management headers depending on the type of scheduling service of the connection, identified by the 16 bit CID in the generic MAC header. Figure A.4 shows the grant management header for CIDs using the unsolicited grant service (UGS). The slip indicator (SI) bit is used by the SS to inform the BS that the uplink buffer servicing the flow has filled up thereby asking the BS to increase the uplink grants. The poll me (PM) bit is set by SS to request a bandwidth poll. When an SS is polled individually, no explicit message is transmitted to poll the SS. Rather the SS is allocated in the ULMAP, bandwidth sufficient to respond with a bandwidth request.

Figure A.5 shows the grant management sub header for non-UGS connections. The 16 bit Piggyback Request field is the number of bytes of uplink bandwidth requested by the SS for a CID. The request does not include any PHY overhead and is incremental.

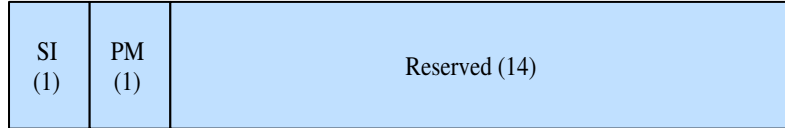


Figure A.4: Grant Management sub-header for UGS connections

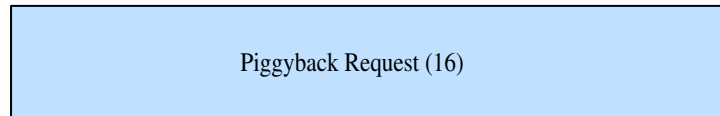


Figure A.5: Grant Management sub-header for rtPS, nrtPS or BE connections

A.2.2 Packing subheader

MAC-level packet aggregation is known as packing. An MPDU can be constructed from one or more MSDUs of a single connection. An MPDU can contain multiple packing sub headers, each followed by an MSDU. The channel can be efficiently used as a lot of overhead incurred due of various headers and PHY preamble can be reduced. The length field enables the receiver to identify where the next packing sub header begins in the MPDU payload. Packing subheader is a 16 bit header which is the length of the MSDU. Figure A.6 shows a packed MPDU with packing subheader.

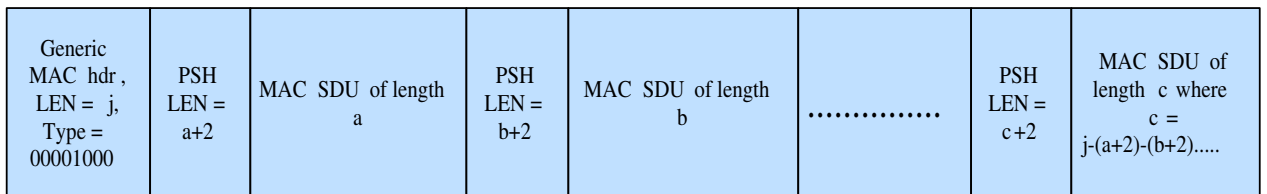


Figure A.6: Packing sub header

A.2.3 ARQ subheader

As explained in previous chapters SRAWAN uses a sliding window based ARQ scheme to handle packet losses at MAC layer. The receiver of the packet uses a set of selective acknowledgment map (SACK map) and a cumulative acknowledgment to send feedback to the sender. Figure A.7 shows the resultant MAC header after the inclusion of ARQ

subheader. The ACKSEQ field is the cumulative acknowledgment and the SACKMAP is the map of the reception of next 16 packets starting from MSB to LSB. For example, let the ACK SEQ be 12 and the SACKMAP be 0100100100000000. This means that packets till sequence number 11 are received and packets 13, 16 and 19 are also received. The SACKMAP will help to reduce retransmission of successfully received packets.

HT	EC	TYPE (6)	Version	CI	Rsv	LEN MSB (4)
LEN LSB (8)			SEQ (8)			
CID MSB (8)			CID LSB (8)			
SACK MAP (16)						
ACK SEQ (8)			HCS (8)			

Figure A.7: ARQ sub header

Appendix B

Transmit and Receive Calibration Plots

This chapter gives the transmit (tx) and receive (rx) calibration plots of Atheros AR5212 based *Ubiquiti SR2* 802.11 b/g card. The experimental setup is shown in the Figure B.1. A laptop with senao prism2 pcmcia card is placed in room R_1 . Open Source *HostAP* V0.3.7 is used for operating the senao card. An attenuator is placed in room R_2 . A WRAP board with an ubiquiti sr2 minipci card is placed in room R_3 . Open source *Madwifi* VR1406 is used for operating ubiquiti sr2 cards. All the three (senao card, attenuator and ubiquiti card) are kept far away from each other such that they hear the signal only through RF cables (and not from the leakages). Depending on the experiment (tx calibration or rx calibration), the senao card and the ubiquiti card are configured to be in either *Monitor* mode or *AP* mode. We already have the calibrated data of the prism2 cards [7] and are used for calibrating the ubiquiti sr2 cards. Those who are interested in the calibration data of prism2 cards can contact author of [7].

B.1 Transmit Power Calibration

The experimental setup is same as described above. The senao card is set in *Monitor* mode and the ubiquiti card is set in *AP* mode. We fixed the attenuation to $60dB$. Now we vary the transmit power (*txpower*) of the ubiquiti sr2 card using *iwconfig* command and try to find out the actual power at which the packets are transmitted by the card as follows. At the senao end, we get the RSSI value which corresponds to a receive power (*rxpower*). We

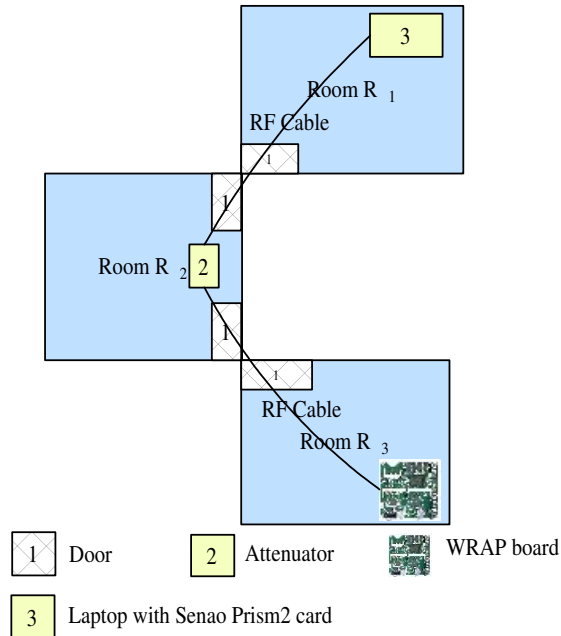


Figure B.1: Experimental Setup for Calibrating Ubiquiti SR2 cards

also know the attenuation value. So the actual transmit power is $rxpower + attenuation$. In all the experiments, the transmit rate is fixed to 11 Mbps. The experiment is conducted for packets of two types. One is the beacon and the other is UDP data packet of various sizes. Table B.1 shows the actual transmit power for various values of $txpower$ set at the transmitter. The data shows that for data packets of any size, the actual power at which the packets are transmitted is $24dBm$ for all the values of $txpower$ set at the transmitter using *iwconfig*. However for beacon packets, the actual transmit power seems to fluctuate over either 0 or 24 for all values of $txpower$ settings.

TxPower set in the card (dBm)	Actual Tx Power (dBm)	Packet
20, 15, 10, 5, 0	24	UDP
20, 15, 10, 5, 0	Fluctuates over 2 values - either 0 or 24	Beacon

Table B.1: Tx calibration data

B.2 Receive Power Calibration

The experimental setup is same as described above except that the senao card is set in *AP* mode and the ubiquiti card is set in *Monitor* mode. Here we try to find the power at which the packets are received by the ubiquiti cards. We fix the actual transmit power (*atxpower*) of the senao card to 20dB and the transmit rate is also fixed to 11Mbps . Now we vary the attenuation values from 90dB to 10dB and measure the RSSI value for these various attenuation values. The 802.11 firmware sends RSSI (Receiver Signal Strength Indicator) value along with the packet to the MAC driver for further processing. The receive power (*rxpower*) corresponding to a RSSI value can be calculated as $\text{rxpower} = \text{atxpower} - \text{attenuation for that RSSI reading}$. Figure B.2 shows the plot of reception power (RxPower) against the RSSI value.

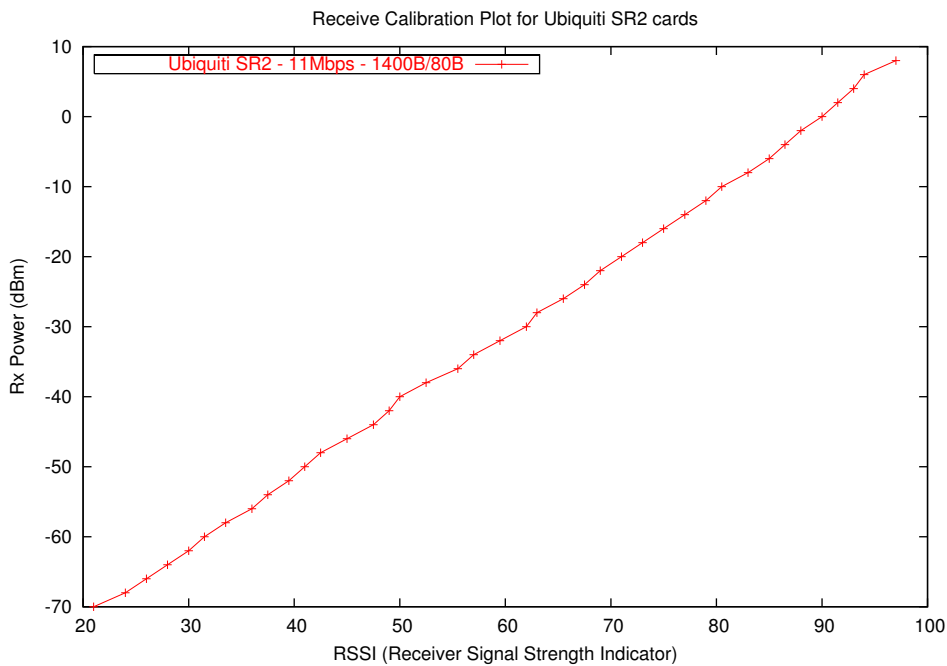


Figure B.2: Receive Calibration Plot

Bibliography

- [1] Pravin Bhagwat, Bhaskaran Raman and Dheeraj Sanghi, “Turning 802.11 Inside-Out”, Second Workshop on Hot Topics in Networks (HotNets-II), Nov 2003, Cambridge, MA, USA.
- [2] Bhaskaran Raman and Kameswari Chebrolu, “Design and Evaluation of a new MAC Protocol for Long-Distance 802.11 Mesh Networks”, 11th Annual International Conference on Mobile Computing and Networking paper (MOBICOM), Aug/Sep 2005, Cologne, Germany.
- [3] Sreekanth Garigala, “Experimental Validation of Simultaneous Operation in an 802.11 Multi-hop Mesh Network”, Master’s thesis, Indian Institute of Technology, Kanpur, 2004.
- [4] Venkat Rao Chimata, “Implementation and Performance Issues of VoIP in Long Distance 802.11b Networks”, Master’s thesis, Indian Institute of Technology, Kanpur, 2005.
- [5] Bob O’Hara and Al Petrick, “The IEEE 802.11 Handbook: A Designer’s Companion”, Standards Information Network, IEEE Press, January 1999.
- [6] Draft IEEE Standard for Local and metropolitan area networks, Air Interface for Fixed Broadband Wireless Access Systems, IEEE P802.16-REVd/D5-2004.
- [7] Bhaskaran Raman, “Digital Gangetic Plains: 802.11-based Low-Cost Networking for Rural Areas”, Status Report, April 2005.
- [8] “Roofnet”, <http://pdos.csail.mit.edu/roofnet/doku.php>
- [9] “Digital Gangetic Plains”, <http://www.iitk.ac.in/mladgp/>

- [10] “Pebble Linux”, <http://nycwireless.net/pebble/>
- [11] “Madwifi”, <http://madwifi.org/>
- [12] “HostAP”, <http://hostap.epitest.fi/>
- [13] “WRAP board”, <http://www.pcengines.ch/wrap.htm>
- [14] “Wistron CM9 cards”, <http://www.wneweb.com/wireless/wireless.htm>
- [15] “Wistron CM9 cards”, <http://pcengines.ch/cm9.htm>
- [16] “Ubiquiti SR2 cards”, http://www.ubnt.com/supper_range.php4
- [17] “Senao cards”, <http://www.seattlewireless.net/index.cgi/SenaoCard>
- [18] “Soekris board”, <http://www.soekris.com/>
- [19] “BelAir Networks”, <http://belairnetworks.com/>
- [20] “Article on WiMAX by Intel”, http://www.intel.com/technology/itj/2004/volume08issue03/art04_ieee80216mac/p03_maclayer.htm
- [21] “Article on WiMAX”, <http://www.commsdesign.com/printableArticle/?articleID=17500163>
- [22] “Article on estimating throughputs for CSMA/CA in 802.11a/b/g”, http://www.oreillynet.com/pub/a/wireless/2003/08/08/wireless_throughput.html