Cleveland State University
EngagedScholarship@CSU

ETD Archive

2009

Multihop Transmission Opportunistic Protocol on Software Radio

Sachin C. Hirve
Cleveland State University

Follow this and additional works at: https://engagedscholarship.csuohio.edu/etdarchive

 Part of the Electrical and Computer Engineering Commons

How does access to this work benefit you? Let us know!

Recommended Citation

Hirve, Sachin C., "Multihop Transmission Opportunistic Protocol on Software Radio" (2009). *ETD Archive*. 790.
https://engagedscholarship.csuohio.edu/etdarchive/790

This Thesis is brought to you for free and open access by EngagedScholarship@CSU. It has been accepted for inclusion in ETD Archive by an authorized administrator of EngagedScholarship@CSU. For more information, please contact library.es@csuohio.edu.

MULTIHOP TRANSMISSION OPPORTUNISTIC PROTOCOL ON SOFTWARE RADIO

SACHIN C. HIRVE

Bachelor of Electrical Engineering
Jiwaji University
June, 2000

Master of Technology in Electrical Engineering
Indian Institute of Technology, Mumbai
June, 2004

submitted in partial fulfillment of requirements for the degree
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
at the
CLEVELAND STATE UNIVERSITY
August, 2009

This thesis has been approved for the

Department of Electrical & Computer Engineering

and the College of Graduate Studies by

Thesis Committee Chairperson, Dr. Chansu Yu

Department & Date

Dr. Vijay Konangi

Department & Date

Dr. Wenbing Zhao

Department & Date

MULTIHOP TRANSMISSION OPPORTUNISTIC PROTOCOL ON SOFTWARE RADIO

SACHIN C. HIRVE

ABSTRACT

The need of high speed communication motivates us to use high bit rate communication to transmit more information in as little time as possible. However, MAC layer protocol overheads dominate the transmission capability particularly at high rates and hinder high speed transmission. Opportunistic transmission has been proposed to help to overcome this disadvantage by transmitting packets back-to-back without inter-packet delays. Though this approach alleviates the problem in single-hop wireless LAN scenario, it doesn't help in multi-hop networks. This thesis presents an approach for multi-hop wireless networks, which is named as Multi-hop Transmission OPportunity (MTOP). It achieves better performance by ensuring better end-to-end packet transmission by allowing back-to-back packet transmission over multiple hops rather than that by one node.

Recent developments in wireless communication research have fueled verification of new approaches on real-life systems rather than simulation. In this thesis, the MTOP approach has been implemented and verified on a widely known software radio platform i.e. USRP hardware and GNU Radio open source software framework. Software radio has emerged as one of the potential platforms for future wireless applications. The wide spread acceptance of software radio is due to the flexibility achieved by porting complex hardware functions to software. This not only speeds up the development, but also creates the multi-standard support for wireless applications. The results show that MTOP improves network performance in a multi-rate ad-hoc network as compared to the contemporary approaches.

TABLE OF CONTENTS

LIST of TABLES

LIST OF FIGURES

Chapter I

INTRODUCTION

Wireless networks have become ubiquitous and their presence and vital role played in our life is felt in various modes. Applications range from use of sensor networks to obtain information from war zone and hazardous locations in a factory installation, to highly common Wi-Fi networks at starbucks and airports. These networks are always in the need of performance boost to meet the demands of increasing user base. Security of the data may be one of critical needs, but the highly desired feature remains speed of data transmission. There are also some emerging areas of wireless applications such as vehicular ad-hoc networks for disaster management. Speed of information also becomes important from the aspect of these time-critical applications.

For achieving the higher speed of data transmission, an obvious solution is to transmit data at higher bit rate. As a result, the markets are swamped with new devices with higher transmission rate capabilities, while existing devices still support low rate data transmission. Though newer devices can transmit at higher speed, their transmission range gets smaller due to higher signal fading. Due to this, while the demand for higher data rate devices is increasing, there is still demand for low data rate communication devices due to longer communication range capability. This

effectively leads us to a multi-rate communication environment where both high and low data rate communication devices share the medium. This thesis presents a new approach towards improving the multi-hop communication scenario in multi-rate ad-hoc networks.

I.1 Motivation for Research

Proliferation of wireless LANs can be attributed to wide spread acceptance of IEEE 802.11 standards [1]. These standards have proven to be the back bone in the evolution of highly secure and faster communication. Though these standards have played a pivotal role for development of contemporary wireless networks and standards, there are some inherent issues which pose as a challenge in improving further network performance. IEEE 802.11 a/b/g standard supports various data rates such as 1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48 and 54 Mbps. Even though the high data rate support is provided, there exist rate-independent overheads. At higher bit rate, the overhead of MAC layer becomes dominant and proves to be the bottleneck in achieving fast speed communication without loss of data.

Similarly, in a multi-rate wireless network, slow data rate nodes eclipse the overall network performance. In other words, multi-rate transmission suffers from the dominance of slower rate communication over channel time leading to the reduced throughput of the network. An attempt had been made to address this issue by use of Opportunistic transmission approach [2]. This improvement has been seen working effectively especially in a single-hop wireless network. This approach achieves a better performance by allowing a node to transmit multiple back-to-back packets. Even though there is some benefit from this approach, Opportunistic Transmission still suffers in a wireless network setup with multi-hop communication needs. For every hop,

each data packet has to pass through channel contention, even for an intermediate hop node.

As it is obvious from the above mentioned reasons, there is a need for an approach to address the issue of reduced throughput in a multi-rate multi-hop wireless network; this thesis presents an enhanced Opportunistic Transmission approach. This improved approach has been named as Multi-hop Transmission OPportunity (MTOP). MTOP is further shown to improve the performance of wireless transmission by reducing inter-frame penalty by relaying a packet over multi-hops without inter-frame delay for high data rate communication. The hypothesis is further being proved using a widely known software radio test bench.

I.2 Choice of Testbed

Wireless research community highly relies on simulation results to support the new approaches and hypothesis behind it, but it has been shown that simulation results may not present the real-life performance of wireless networks [21]. In that case, it is highly encouraged that the new approaches and ideas should be substantiated with the experimental results. Though experimental results may not be able to capture all the complexities of real-life scenario, they still fairly represent the actual performance on a prototype.

As there are many different experimental testbenches available, selecting a particular testbench is a tedious work given the functionality support and capability of different hardware. We chose USRP (Universal Software Radio Peripheral) [3] as a hardware platform and GNU Radio (open source project) [4] for software support. Collectively this software/hardware platform is used as a tool to implement and verify

MTOP algorithm. Reasons for selecting USRP and GNU Radio are primarily flexibility provided by FPGA and ease of programming from PC, which is being used as the computing element in this setup, in place of a dedicated micro-controller as in other testbeds. Additionally, there were not enough modulation scheme implementations found on other contemporary hardware prototypes, which can suffice our needs.

Software defined radio (SDR) is finding an increased importance due to flexibility that it provides to implement radio functions. GNU Radio, an open source software tool, being a driving force to develop new wireless applications, is adding to multiple functionalities from same hardware. Universal Software Radio Peripheral (USRP) is the hardware device which is flexible enough to integrate with GNU radio. GNU radio and USRP together help in implementing Software Defined Radio functionalities. The base concept of SDR is to make radio functions hardware independent. The tasks like modulation, demodulation, encoding etc are implemented in software which eliminates the need of corresponding hardware. In essence, SDR leaves the hardware to do the basic functions like transmission/reception of signal and does all the complex signals processing on the general purpose processor thereby relieving the hardware from signal processing complexities.

I.3 Thesis Organization

This thesis illustrates the hypothesis behind the working of MTOP algorithm and describes proof of concept through the implementation and verification of MTOP over GNU Radio and USRP framework. MTOP approach of wireless communication in ad-hoc networks has been conceptualized by Dr. Chansu Yu and this thesis is an attempt to presents the idea and its experimental results. This thesis has been organized in six chapters. Chapter 1 begins with the introduction to the idea followed

by related work by other researchers in chapter 2. Chapter 3 and 4 describes the MTOP algorithm and testbench of Software Radio. In chapter 5, experiment results are presented. Chapter 6 summarizes the report followed by chapter 7 with the scope of future work and actions to be performed to achieve them. Python application code used in experimentation is listed in appendix for the reference.

Chapter II

RELATED WORK

Wireless networks are undergoing through a continuous research for performance enhancement. While original IEEE 802.11 standards supported only base rate communication [5], further standards such as IEE 802.11a/b have added multi-rate communication enhancements. In addition to the higher data rate transmission, various approaches were targeting the performance improvement through rate adaptive protocols for single hop and multi-hop wireless communication. In addition to these approaches to improve network performance and primarily throughput in it, there were also few opportunistic communication approaches being researched. These various approaches are investigated in detail in the following sections.

II.1 Multirate support in Wireless Networks

As per the physical layer (PHY) specifications [1] of IEEE 802.11, 2.4 GHz *Direct Sequence Spread Spectrum* (DSSS) is supported at data rates of 1 and 2 Mbps. With IEEE 802.11b standards, data rates of 5.5 and 11 Mbps were added to the same 2.4 GHz DSSS. With the IEEE 802.11a standard, 5 GHz *Orthogonal Frequency Division*

Multiplexing (OFDM) at multiple data rates of 6, 9, 12, 18, 24, 36, 48 and 54 Mbps was introduced. With a further addition to the IEEE standards i.e. IEEE 802.11g, same set of data rates and modulation schemes were supported at 2.4 GHz frequency.

With the multi-rate support, a data transmission takes place with *Physical Layer Convergence Procedure* (PLCP) frame using two data rates. PLCP frame consists of PLCP preamble, PLCP header and payload. The PLCP preamble and header is transmitted on lowest data rate i.e. 1 Mbps DBPSK modulation in 802.11b and 6 Mbps BPSK modulation in 802.11a. The payload is transmitted at higher data rate, information of which is specified in PLCP header.

II.2 Characteristics of Multirate Communication

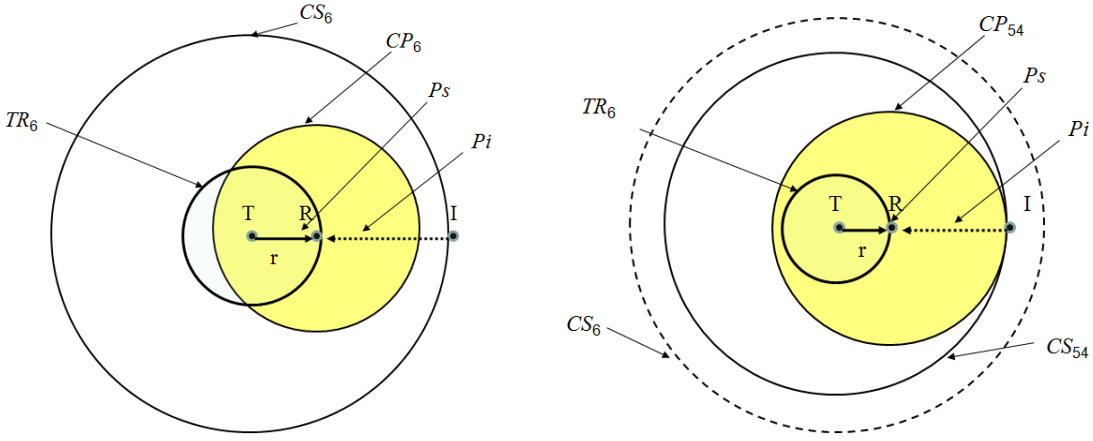
Wireless communication performance depends on two important parameters i.e. *receive sensitivity* and *capture threshold*, both of which vary with different data rates. Receive sensitivity defines the possibility of successful communication based on received signal power. If the received signal power is more than the receive sensitivity under path loss over distance, communication is deemed as successful. Further, this received signal power must be strong enough to overcome the noise and interference from the data transmissions of neighboring nodes. In other words, *Signal to Interference and Noise Ratio* (SINR) at the receiver should be higher than the capture threshold [6, 7].

Table I shows various parameter values at different data rates for a typical 802.11a radio device [18] and this data is used in the following analysis. Fig. 1(a) and 1(b) shows a communication environment where the transmitter (T), the receiver (R) and the interferer (I) are communicating at 6 and 54 Mbps data rates. The receiver (R)

Data rate (Mbps)	Receive sensitivity (dBm)	Capture threshold (dB)	Communication range (m)	Interference range (m)	CS range (m)
6	-82	6.02	238	337	575
9	-81	7.78	224	352	576
12	-79	9.03	200	336	536
18	-77	10.79	178	331	509
24	-74	17.04	150	400	550
36	-70	18.80	119	351	470
48	-66	24.05	95	389	484
54	-65	24.56	89	366	455

Table I: Characteristics of an 802.11a multi-rate radio (Transmit power: 6 dBm, radio propogation model: two-ray ground reflection, path loss exponent: 4.

receives two signals i.e. signal from transmitter with signal power P_s and signal from interferer with signal power P_i.



(a) Low rate (6 Mbps)

(b) High rate (54 Mbps)

Figure 1: Various data rate communication scenario

For a successful communication between T and R in Fig 1(a), signal power P_s should be higher than the receive sensitivity, -82 dBm, for corresponding data rate. Alternatively, distance between T & R should be less than communication range TR_6, 238m. Additionally, SIR should be higher than the capture threshold (6.02

dB) of receiver for 6 Mbps communication, or in other words, R-I distance should be greater than the interference range CP_6, 377m. Therefore there should not be any other simultaneous transmission within a distance of 377m from receiver. Since receiver can not dictate the transmission, this requirement is fulfilled by transmitter. Transmitter uses the carrier sense and ensures that there is no other simultaneous transmission happening within a distance of 575m (377 + 238), which is also known as carrier sense threshold CS_6.

Considering the scenario in fig. 1(b), due to increased receive sensitivity (-65 dBm) for a data rate of 54 Mbps, communication range TR_{54} is much smaller, 89m. Corresponding SIR, 24.56 dB, is higher than the previous case, which leads to a considerably higher interference range CP_{54}, 366m. Therefore CS range and required CS threshold comes as 455m and -93.3 dBm, which is 4 dB greater than 6 Mbps case. Since the CS range is based on lowest data rate transmission, this leaves an additional freedom for network improvement. MTOP utilizes this freedom by allowing a frame to travel 1-3 more hops without channel contention, but still leads to a collision free communication. MTOP is further explained in details in section 3.2.

II.3 Rate Adaptive approaches in WLANs

There have been a number of new approaches proposed for exploiting the multi-rate capability of wireless network. They can be primarily categorized as sender based and receiver based approaches.

Auto-Rate Fallback (ARF) [8] is a sender based algorithm which tries to optimize the application throughput in Wavelan II devices. It works on the principle of using higher data rates on consecutive successful transmissions and falling back on

lower data rates upon consecutive transmission failures. Here transmission success is judged on the basis of acknowledgement reception. There have also been some proposals of ARF variations which include *adaptive* ARF [9], *adaptive multi rate retry* (AMRR) [10, 9] and *estimated rate fallback* (ERF) [11]. Recently, it has been observed that ARF does not work well when there are many transmission failures because of collisions rather than channel errors [12]. Kim *et al.*, therefore, have proposed *Collision-Aware Rate Adaptation* (CARA), which differentiates the transmission failures happening from collision rather than by channel errors.

Receiver based multi rate approach, *Receiver-Based Auto Rate* (RBAR), proposed by Holland *et al.* uses RTS/CTS handshaking to estimate the channel quality which is based on SINR of the received RTS frame. This is further used to estimate the best data rate for the transmitter. This estimated data rate is then communicated to the transmitter through CTS piggybacks. *Opportunistic Auto Rate* (OAR) [5] improves the performance of wireless network by sending back-to-back multiple packet when high-quality channel conditions exists. In excellent channel conditions, higher data rate is used. Recently multi-rate adaptive approaches have gathered a lot of attention due to increased usage of wireless LANs and higher throughput demand.

II.4 Performance Anomaly in Multi-rate Networks

Multi-rate communication capability of IEEE 802.11a/b helps to improve the performance of wireless networks by permitting packet transmission at higher than the base data rate. Though high rate communication is effective, 802.11 standards itself inhibits the performance of wireless network by guaranteeing fair opportunity of channel access to every node irrespective of the operating data rate. Due to opportunity based fairness, chances of channel allotment to slow and fast data rate nodes are equal. This

leads to time wasted in competing for channel time after every packet transmission irrespective of capability of higher data rate node to transmit more information in its allotted time. Additionally, low rate nodes are liable to take more time to transmit as compared to high rate nodes. High data rates available in 802.11 radios seem to reduce the end-to-end latency, but the multi-rate control becomes complicated due to multi-hop nature of wireless network [13].

Performance of wireless networks does not improve linearly with data rate increase due to rate-independent overhead at the PHY (header and preamble) and MAC (DIFS and back-off) layers, imposed by 802.11 MAC protocol [1]. Moreover, this overhead becomes a dominant as data rate increases because the actual packet transmission time decreases proportionally. Equal share of channel time approach [27], adopted in IEEE 802.11e improves the overall communication throughput for single-hop networks though leaving low data rate communication unaffected. This approach, also known as transmission opportunity (TXOP), addresses the problem by sending back to back packets for high data rate communication nodes [2]. Though TXOP has been seen improving network performance, it only works better for single-hop networks. Other issues related to TXOP are (i) the source node may not always have multiple data frames to transmit back-to-back although given an opportunity and (ii) an intermediate node may get overloaded if it gets lesser opportunity than the predecessor.

With the emphasis of prevailing approaches on multi-rate communication and back-to-back packet transmission to achieve higher performance wireless networks, the throughput does not meet the expected behavior of WLAN.

Chapter III

PROPOSED APPROACH

As mentioned in the above discussion, there was a persistent emphasis on increasing the speed of communication to satisfy the need of increasing consumer base with increased demands. To accommodate these demands, enormous amount of efforts had been made to realize the devices which can support higher data rates. As a result of this, present day wireless networks support various data rates of communication making the communication as a multi-rate scenario in which both low rate and high rate communication devices share the medium. Though multi-rate wireless networks present their own peculiar challenges, they provide a boost to the speed of message transfer. But with the increased data rate, the communication range become smaller and leads to increased number of hops to relay the information to the destination nodes. Additionally, the performance anomaly [15, 27, 16] present in wireless networks restrict the ability to increase the amount of transmitted data. Even after increasing the payload data rate, the PLCP header is still transmitted at low rate (1 Mbps), reducing the overall performance. With the higher data rates, PLCP overhead becomes even more dominant leading to further performance degradation.

Opprtunistic transmission (TXOP) [2] has been studied and implemented to improve the network performance in single-hop networks. TXOP allows a high rate communication node to send back-to-back packets without contending for the channel. The number of packets allowed to be sent back-to-back depends on the data rate of transmitting node as well as the data rate of neighboring nodes. TXOP surely improves the network performance, but the benefit from it can be only be felt in single-hop wireless networks. In order to improve the overall performance of the multi-hop multi-rate domain, a new approach called as Multi-hop Transmission OPprtunistic protocol (MTOP) has been proposed. MTOP is planned to achieve higher performance by assuring faster data communication between the end nodes rather than sending bulk of data from one node to the next hop as in TXOP. MTOP approach is based on multi-rate margin available in a multi-rate ad-hoc network. Multi-rate margin and functioning of MTOP is explained in the following section.

III.1 Multi-Rate Margin

In a multi-rate wireless network, various data rate devices share the network medium. A typical wireless network is shown in figure (2) which is shown to be composed of devices transmitting data on 6 and 54 Mbps. Whenever a node needs to transmit a data packet, it contends for the medium and when the medium is free, it transmits the packet. While transmitting the packet, it needs to ensure that the packet reaches the destination node with least possibility of collision. For this reason, this communication is guarded by carrier sense range. By definition, carrier sense range is the distance at which received signal power is equal to the receive sensitivity at that rate. All the nodes inside the carrier sense range hold their transmission till the current transmission gets over.

In figure (2), node A is trying to send a data packet to node B. Node A, B, C and D communicate at higher data rate of 54 Mbps. Additionally in this multi-rate environment, nodes P, Q, R communicate at lower data rate of 6 Mbps. For node A, communication range, carrier sense range for 6Mbps and 54Mbps data rate is shown by TR_0, CS_6, and CS_{54} respectively. While transmitting, if node A tries to guard the carrier sense range of CS_{54}, it will be subject to packet collision from the low data rate communication between node Q and R. Additionally, as the data rate increases, attenuation of the signal becomes larger. Since node Q communicates at lower data rate, it has a bigger carrier sense range, thereby becoming a potential interferer for node A. Therefore, though Node B is inside the communication range of node A, due to presence of low rate communication nodes, a much bigger area is guarded for this communication to succeed. In this figure the guarded area is shown by carrier sense rage CS_6.

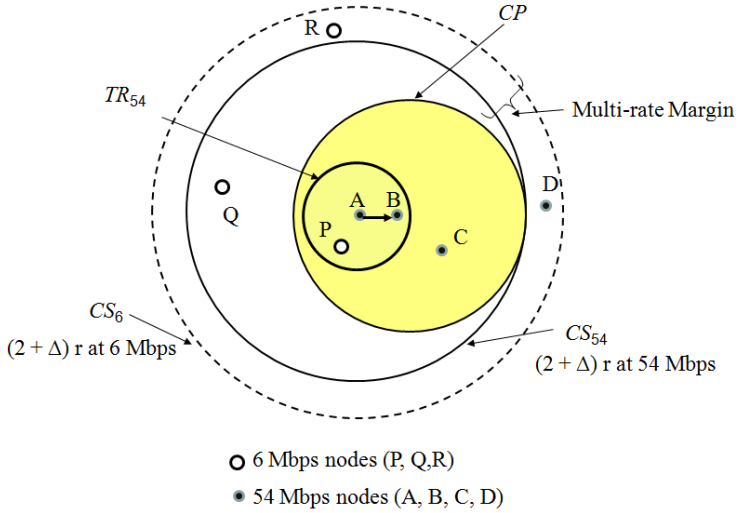


Figure 2: A typical wireless network with multi-rate transmission scenario

From the observation of the above scenario, it is obvious that in a multi-rate environment, carrier sense range is decided by the lowest data rate devices in the

network. The difference between the carrier sense range for high rate communication nodes and low rate communication nodes has been identified as Multi-Rate Margin. This multi-rate margin depends on the different data rates present in the wireless network. Higher the difference between the data rates of wireless nodes, higher will be the multi-rate margin.

This multi-rate margin can be harnessed to improve the network performance by increasing the throughput. It can be easily observed that a high data rate communication still guards a bigger area than required. The guarded area has enough opportunity for more hops of communication without introducing noticeable interference from the devices outside the carrier sense range. This methodology has been utilized in MTOP, which shows a simple change in data communication approach leading to higher performance gains. The next section describes the functioning of MTOP in detail.

III.2 Multihop Transmission OPportunity (MTOP)

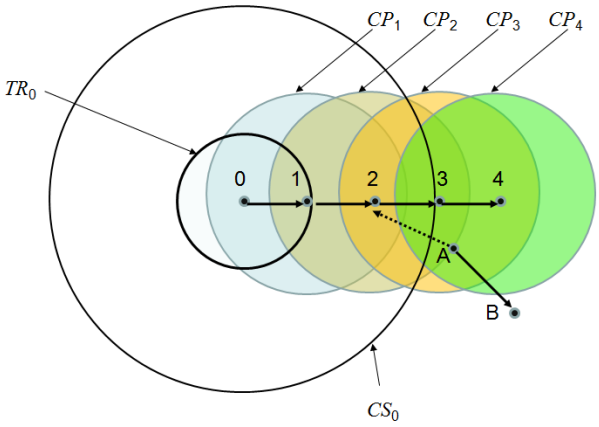
MTOP is primarily based on multi-rate margin and it tries to improve the performance of high data rate nodes even further. With the existing approaches in wireless networks, emphasis is more on achieving high speed transmission through fast rate and harnessing the advantage of back-to-back packet transmission. Though these approaches are effective, they do not perform consistently given the different communication scenarios. While fast rate transmission is dependent on winning the opportunity to transmit in case of multi rate transmission environment, there is still need of opportunistic algorithms to increase the performance of the network. Though the popular opportunistic algorithm, TXOP achieves higher performance for data intensive network, it fails to improve the performance in case wireless node needs

to send intermittent data to be sent over the multi-hops. Additionally TXOP does not improve end-to-end latency due to rate independent overheads i.e. fixed rate transmission of IEEE 802.11 frame preamble.

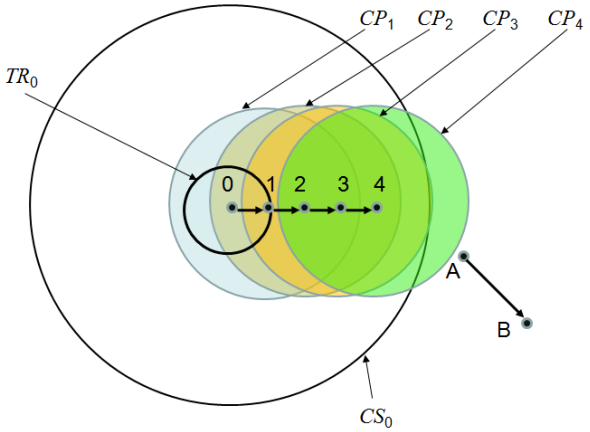
This brings us to a new idea which can transform TXOP mechanism to improve the multi-hop wireless communication. This approach relies on following assumptions.

- Fast rate transmission can achieve more data transmission if not subject to repetitive competition for transmission opportunity i.e. competing for transmission in contention period.
- Given the opportunity, if the inter-hop nodes forward the frame to the next hop node, other nodes will sense the carrier's presence and will backoff.
- End-to-end packet transmission speed improves the communication by asuring the transfer of information between far off nodes in relatively lower time.

The key idea to improve the wireless network performance is to extend the concept of TXOP to multi hop networks. Figure 3(a) shows node 0, 1, 2, 3 and 4 representing a multi-hop scenario. Assuming that the carrier sense (CS) zone of the sender node 0, which is denoted as CS_0, is the circular region in which if a wireless node observes the sender´s signal strength to be higher than the CS threshold, it defers its transmission. Further it is assumed that capture zone of the receiver node 1, denoted as CP_1, is such that any wireless node outside of CP_1 does not cause collisions for the 0 - 1 communication [17]. For a successful 0 - 1 communication, CP1 must be protected to ensure that the SNR (Signal-to-Noise Ratio) at the receiver node 1 is higher than the required SNR at that data rate. This is in fact enforced by carrier sensing in CS-based protocols such as IEEE 802.11 MAC.



(a) 6 Mbps



(b) 54 Mbps

Figure 3: Carrier Sense and Capture Threshold regions for low and high data rate communication

In multi-rate radio systems, the SNR requirement increases as data rate increases, making high-rate communications more vulnerable to interference and noise [18]. Therefore, although transmission range (TR) as well as the communication distance reduces, CP_j is approximately the same as shown in figs. 3(a) and 3(b). Fig. 3(b) exhibits that as data rate increases, CS_0 covers not only CP_1 but also CP_2, CP_3, and so on due to nearness of wireless nodes to meet the communication distance requirements. In other words, when a node transmits data at high data rates e.g. 54 Mbps, most of potential interferers for the current communication (0 - 1) as well

as the next hop communication (1 - 2) would be inhibited. The proposed multi-hop transmit opportunity (MTOP) exploits this opportunity to transmit packets in consecutive hops. MTOP allows a frame to travel multiple hops, 0 - 1 and 1 - 2 without having to compete for the medium between hops. This way it avoids the time required to compete for career which also depends on the probability of claiming the carrier. It helps reducing the multi-hop latency. MTOP is more beneficial for the high rate communication given the rate independent overhead of MAC layer. Based on our analysis to ensure better performance and less collisions, the number of hops allowed under MTOP for various data rate nodes is one for 6 and 9 Mbps nodes, two for 12 and 18 Mbps nodes, three for 24 and 36 Mbps nodes, and four for 48 and 54 Mbps nodes in a network of multi-rate scenario with 6 - 54 Mbps nodes.

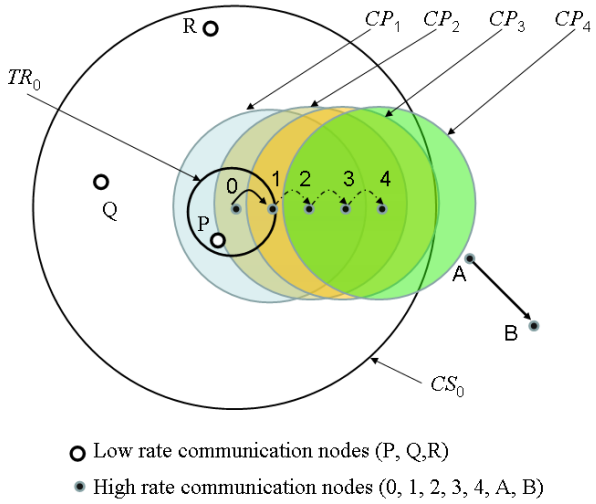
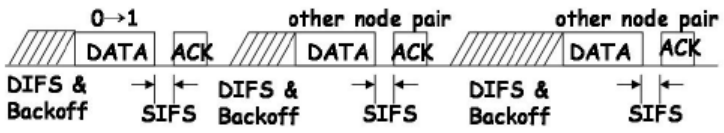


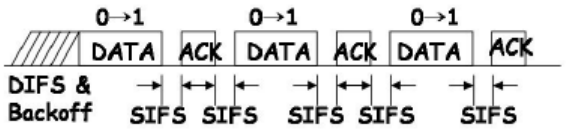
Figure 4: Example of MTOP implementation for 6 and 54 Mbps multirate network

Further observation of fig. 4, explains that the communication under MTOP is even secured from the out of carrier sense nodes. Let us assume node 0 wants to send a packet to node 4. Under MTOP approach, once node 0 finds the medium free, it sends the packet to node 1, which in turn forwards the packet to node 2 without

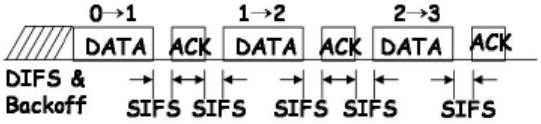
competing for channel again. In the similar way, node 3 receives the packet from node 2 and sends it to node 4. Since the communication is guarded by the carrier sense range of lowest data rate communication, node 4 is still able to receive the packet without collision. As in the fig. 4, node S from another set of wireless nodes sends a packet to node T. This communication does not affect the communication between node 0 and 4.



(a) DCF



(b) TXOP



(c) MTOP

Figure 5: Packet transmission in DCF, TXOP and MTOP

Analysis of contemporary approaches such as DCF and TXOP with respect to MTOP shows a considerable difference in the packet transmission characteristic. Fig. 5 explains the performance of each approach with time required for sending the equal information. DCF leads to the maximum time taken for transmitting the information as it follows the back off and DISF period before sending a new packet 5(a). TXOP improves the performance by avoiding the back-off and DISF time period and sending multiple frames back-to-back 5(b), but as discussed earlier advantage of TXOP lies

in multiple packet transmission requirement at the sender node. MTOP improves the performance of the wireless network similarly as TXOP by avoiding back off and DISF time period, but it emphasizes on end-to-end packets transmission up to four hops 5(c). Overall MTOP functions superior to other approaches, specifically in a network where nodes do not enough information to be sent in multiple frames. The claims of better performance by MTOP are supported by the experimental results achieved using software radio platform.

Chapter IV

SOFTWARE RADIO - A TESTBENCH

In a standard off-the-shelf wireless device, the computation intensive communication functionalities are handled by a dedicated microcontroller. Testing or implementing new communication approaches is practically not easy with these hardware components. Complex nature of low level communication algorithms, make it difficult to apply new approaches. There have been huge amount of efforts by industry and academic research to address this issue. A s a result of this, an experimental set-up has been which can facilitate easy programmability along with feasibility to test and implement new and innovative wireless communication approaches. Even though there are some wireless devices available to help academicians in research and experimentation such as mica2 motes, TelosB motes and CalRadio, we chose to use USRP and GNU Radio due to flexibility and ease of use.

Recent emergence of GNU Radio as a software platform and USRP as a hardware platform has provided a powerful tool for academic research community and amateurs to test the wireless communication application in software radio environment.

GNU Radio is an open source software development toolkit which enables runtime signal processing. Signal processing blocks in software facilitate implementation of software radio functionalities on a general purpose processor. GNU Radio connects to a radio front-end through USRP. USRP works as a low-cost and flexible hardware to accommodate the basic RF front end functionalities with additional flexibility for changing the function of hardware on demand.

IV.1 GNURadio - SDR Software Architecture

GNU Radio software architecture is composed of a layered structure. Top layer application programs are written in Python script language. The bottom layer contains computation intensive signal processing blocks written in C++ in the form of classes. These signal processing blocks replicate the complex communication blocks which are implemented in hardware for conventional devices. Each signal processing block represents a low level communication component. These individual processing blocks are connected together by application program in the form of a flow graph. Application programs and signal processing blocks, being written in two different programming languages, use SWIG to connect with each other. SWIG is a utility which works as a glue between C++ classes and Python language flow graph. In a simpler way, SWIG converts the C++ classes into analogous Python compatible classes. Therefore, GNU Radio framework makes use of the strong features of both programming languages.

While C++ provides compact code for signal processing block, Python is preferred because of its flexibility and ease of programming. Python code makes a flow graph of the signal processing blocks to channelize the information flow and its processing. Using Python and C++ in combination, the signal processing units are implemented on general purpose processor by GNU Radio. USRP is connected to the GNU Radio

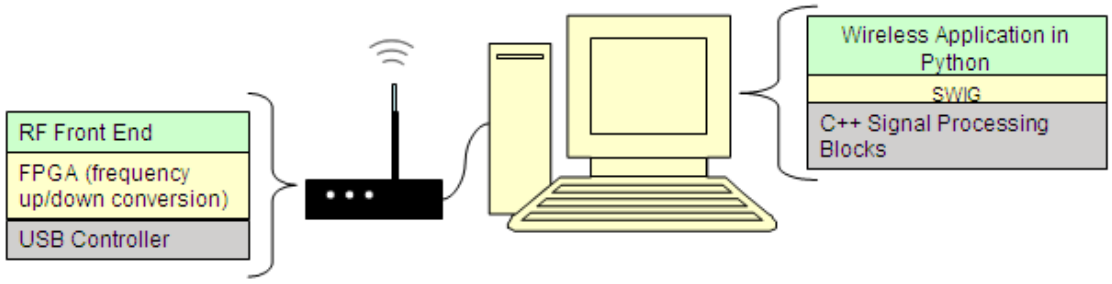


Figure 6: Schematic representation of GNU Radio and USRP.

on general purpose computer through a USB cable.

IV.2 USRP - SDR Hardware Platform

While a lot of complex communication components are dealt by GNU Radio, USRP provides a generic hardware platform for various frequency and bandwidth wireless signals. USRP has been developed by a team headed by Matt Ettus, who has USRP is composed of RF front end, FPGA, ADC/DAC and USB controller. USB cable connects USRP with the general purpose computer hosting GNU Radio application and it is used to transfer data from USRP to host computer for signal processing and vice a versa. FPGA is used to perform the frequency up/down conversion. In essence, FPGA converts the high sampling rate data to a rate which USB controller can handle. ADC/DAC converts the data from digital to analog format and vice-versa. Functionality of RF front end is taken care by detachable daughterboards. Depending on the target frequency band, daughterboards can be plugged, which enables USRP hardware to serve multiple signal bands and frequencies.

For the validation of MTOP approach, we are using Ubuntu 8.10 as the development platform. It is found to be one of the most easy to work with distributions

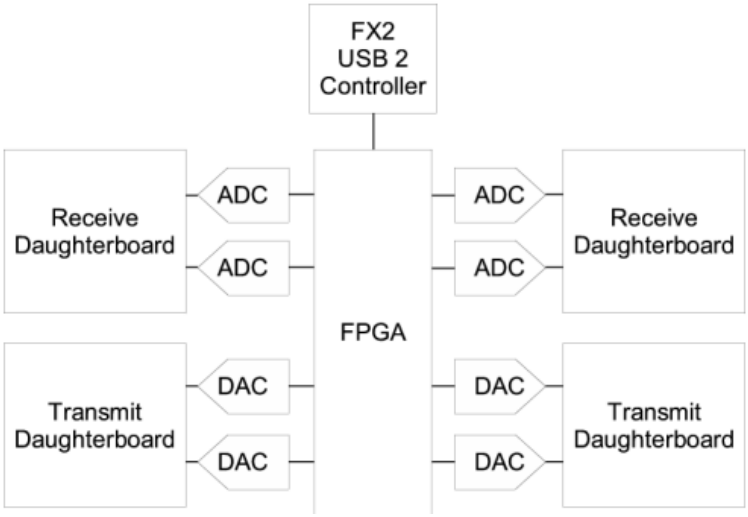
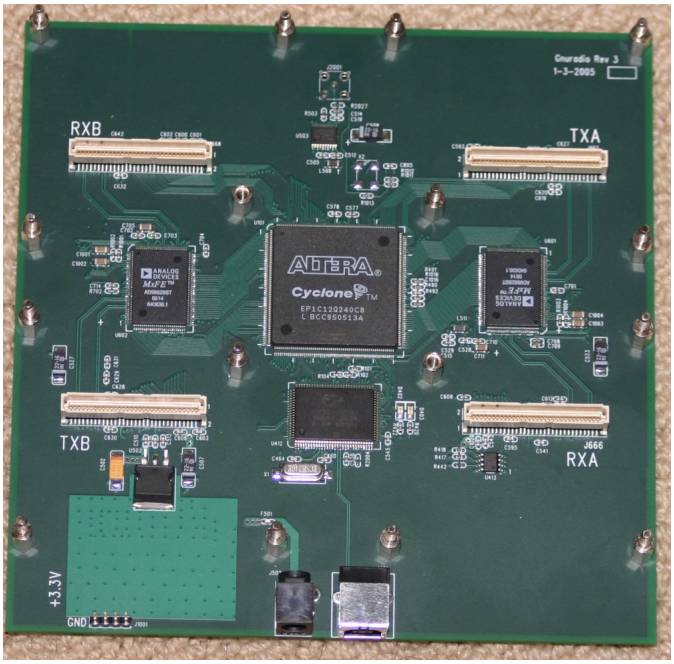


Figure 7: Universal Software Radio Peripheral Block Diagram. Figure taken from [28]

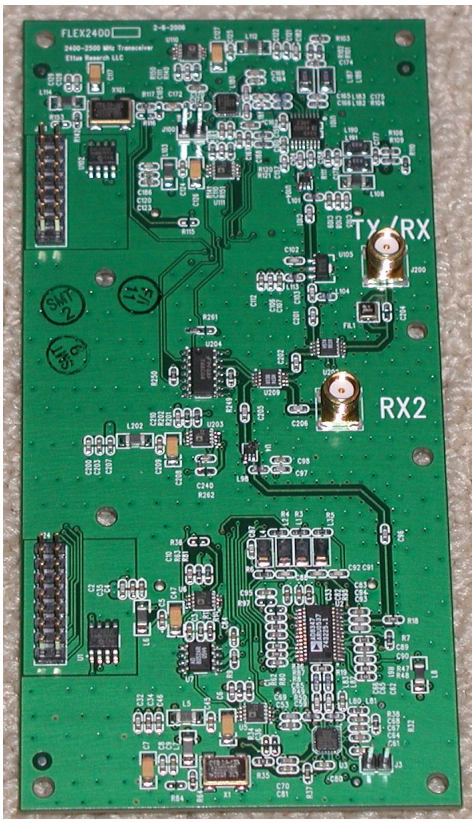
of Linux operating system with additional ease of installation of GNU Radio package. We have also used wingware 3.0 as a software development environment. For validation of MTOP, we are employing two modulation schemes to compare the performance known as DBPSK and DQPSK. While DBPSK provides good results for moderately high data rates, QPSK is known to be better performing for specifically high data rate communication.

IV.3 Limitations of USRP/GNU Radio for MAC functionality

The last few years have seen a tremendous development in implementation of various wireless applications on USRP/GNU Radio. Even after being a powerful tool for researchers to test new ideas of wireless applications, USRP/GNU Radio fails to meet timing requirements of certain low layer implementations i.e. MAC layer functionality. A couple of studies [19, 20] on USRP/GNU Radio have shown it to be unsuitable



(a) USRP motherboard



(b) USRP RF front end for 2400 MHz - RFX2400

Figure 8: USRP Hardware

for full feature MAC layer protocol implementation. The results from the studies show that there is a non-negligible latency between packet reception at the receiver antenna and arrival of packet at the GNU Radio application layer. This considerable delay makes it vulnerable to packet loss/collision during transmission under carrier sense.

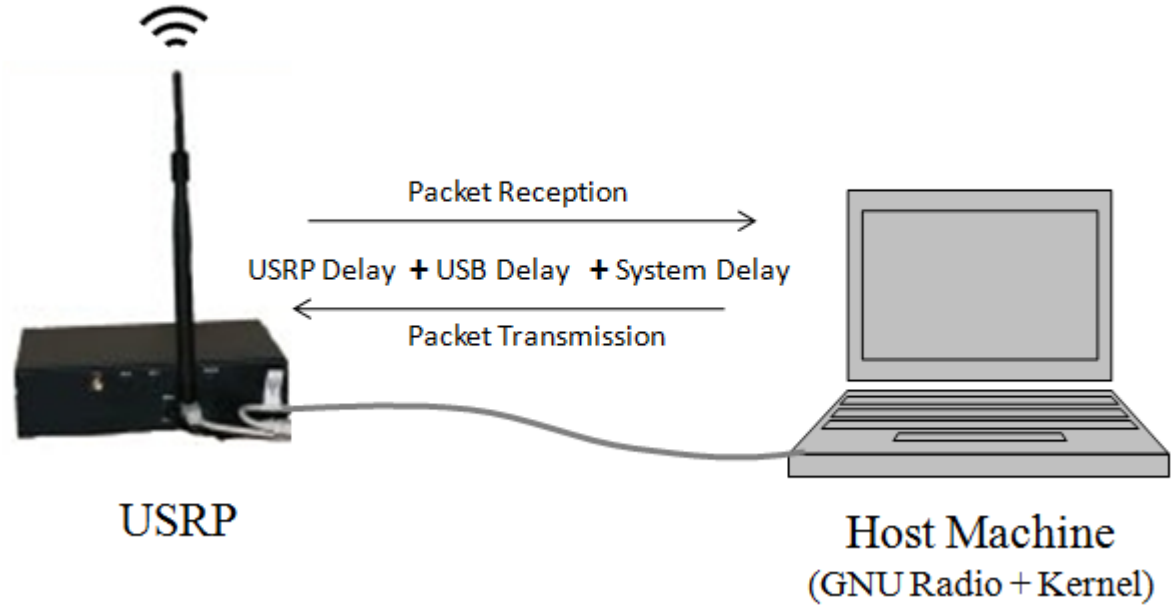


Figure 9: Path of packet transmission between Host PC and USRP

Schmidt et al. [19] demonstrated this blind spot being dependent on latency introduced by FPGA, USB bus and system latency. George et al. [20] have further expanded the delay measurement by considering kernel level latency. This non-negligible latency in packet processing path is found to be 2 - 32 msec [19] depending upon the system configuration and working load on the host machine. Conventional radio devices have the signal processing blocks implemented very close to the radio frontend (antenna). This results in the packet transmission latency of the order of tens of microseconds. The latency introduced by USRP/GNU Radio is obviously higher

than the latency of conventional NICs, resulting in degraded network performance for MAC layer services.

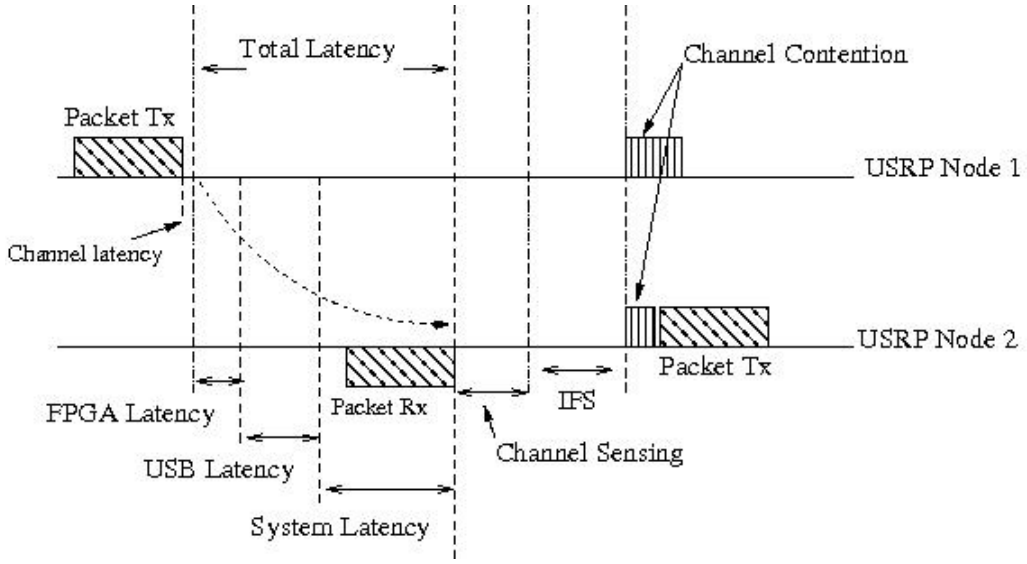


Figure 10: Latency in career sense and packet trasnmission in USRP

Apart from the latency, the architecture of GNU Radio presents a challenge to incorporate the MAC layer functionalities [21]. GNU Radio architecture is based on flow graphs. Both for transmission and reception, separate flow graphs run continuously in different threads. MAC functionalities require an implementation based on state machine and therefore needs to be implemented on separate threads with sufficient buffer to enqueue continuously incoming message frames [21, 22]. Results from the previous research to develop MAC functionalities in USRP/GNU Radio architecture demonstrates non-feasibility of incorporating full functional MAC [21].

In our experimental setup, due to increased complexity in implementing MAC in GNU Radio, we decided to use minimal functionality of MAC layer protocol (CSMA) using extended timing requirements to demonstrate the effectual working of MTOP on multi-hop wireless networks. As the blind spot duration is unpredictable, it is

difficult to find an optimal value of longer timing delays at MAC layer i.e. DIFS, IFS, SIFS. Repetitive experiments led us to relatively high values of IFS (1 ms) and DIFS (120-150 ms), which can also be attributed to the system configuration and relatively low data rates. The current implementation of CSMA is based on contention and random back-off and it does not include the exponential back-off on loss of packets. Indoor experiments have shown that relatively acceptable PDR (more than 90%) can be achieved using this extended IFS and DIFS. Though this setup does not meet the standards of conventional radio, it provides the basic functionality to prove our concept using USRP/GNU Radio.

Chapter V

EXPERIMENTAL RESULTS

As mentioned in previous sections, though simulation results are widely accepted to propound and verify a new approach in academic world, their results vary on multiple factors. Therefore, we chose a hardware prototype consisting of USRP and GNU Radio to implement MTOP. In this process, various steps were followed before actually implementing MTOP on hardware prototype. As the first step, communication range and carrier sense range were found out for the chosen scenario. Further MTOP was checked without using carrier sense capability in USRP as a proof of concept. In the final stage, MTOP was implemented using a 5 node scenario with multi-rate environment. Since this complete setup required a considerable amount of open space, initial experiments were done at Edgewater Park in Cleveland, Ohio. Later part of experiments was performed in the open space in front of Rhodes Tower at Cleveland State University.

Since experiments were done at remote locations, a portable power supply was needed to power laptops and USRP hardware. For this purpose, two "Power1500" portable power supplies from XANTREX were acquired. Complete experiment procedure required five laptops with equal number of USRPs along with RFX2400 (2.3-2.9

GHz) daughterboards and antennas. From the point of user friendliness, Ubuntu8.0 was chosen as the operating system for all the computing machines. Furthermore, GNU Radio version 3.1.3 with $USRP_1$ was used for the experiments. $USRP_1$ is the first version of USRPs which connects to host machine through USB cable for exchange of data packets. The new version $USRP_2$ incorporates Gigabit Ethernet cable to achieve the data exchange.

On software side, the modulation schemes used were DBPSK and DQPSK. Carrier frequency was set at 2.462GHz and a bandwidth of 200 KHz was selected. Corresponding maximum data rates for DBPSK and DQPSK were found to be 200Kbps and 400Kbps respectively. The reason for selecting such a smaller amount of bandwidth is due to the constraints imposed by relatively slow speed communication between host machine and USRP through USB connectivity. Since the default transmit power amplitude was too high and was requiring a huge open space for experiments, the transmission power amplitude was reduced from 12000 to 8000. By adjusting the power, communication range was brought down below 300 ft. Following subsections describes in detail the experiments performed with their significance.

V.1 Measurement of Communication & Carrier Sense Range

First part of the experiment involved finding the communication range for DBPSK and DQPSK modulation schemes with the respective data rates of 200Kbps and 400Kbps. For this purpose, two USRP systems were set up as transmitter (Tx) and receiver (Rx) nodes. The data set collected for finding the communication range were RSSI (Receiver Signal Strength Indicator), PDR (Packet Delivery Ratio) and distance. Keeping the receiver stationary, the transmitter was moved towards receiver

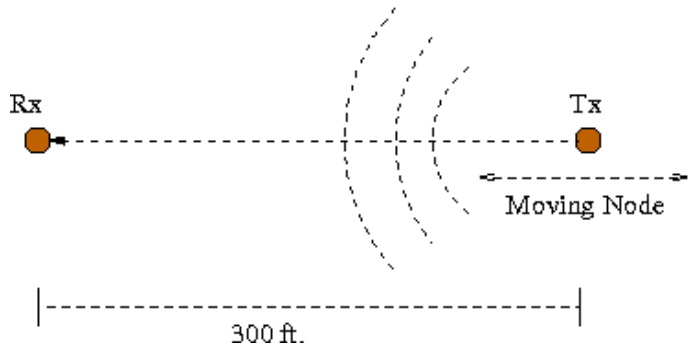
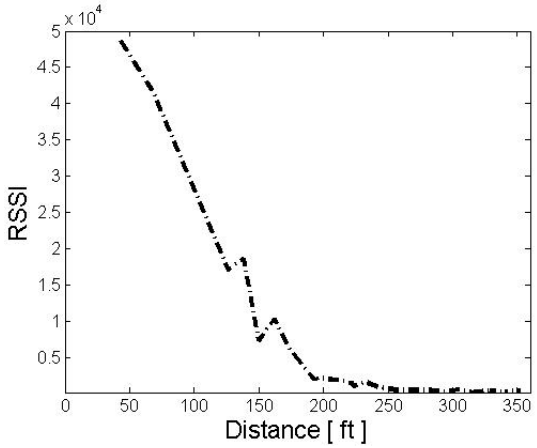
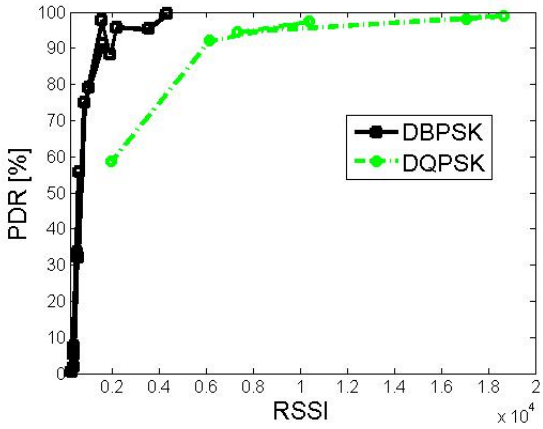


Figure 11: Experimental setup for finding communication range

starting from a distance of 300 ft.



(a) RSSI v/s Distance Plot



(b) PDR v/s RSSI Plot

Figure 12: RSSI, Distance and PDR plots for two node communication scenario

From this exercise, RSSI versus Distance and RSSI versus PDR relationship, as in fig. 12(a) and 12(b), were found out. At each position, multiple measurements were taken. Each measurement consisted of 3300 packets being sent by the transmitter to receiver and average RSSI was calculated over these multiple measurements.

Communication range was deduced from the two different relationships, as shown by fig. 12(a) and 12(b), among PDR, RSSI and distance. Communication range for DBPSK and DQPSK came as 215 ft and 150 ft respectively. It should be noted that

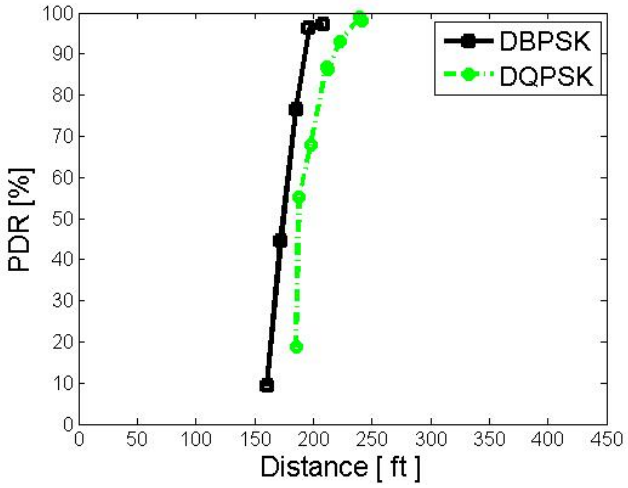


Figure 13: PDR v/s Distance Plot

estimation of communication range is based on more than 95% PDR.

The next step in this series of experiment was to find carrier sense range for DBPSK and DQPSK modulation schemes. In order to find the carrier sense range, we firstly measured the interference range. Interference range is a function of communication range (r) and is denoted as $((1 + \Delta)r)$. To find the interference range, three USRPs were set up, as one Transmitter (Tx), one Receiver (Rx) and one as an Interferer (I) in a straight line $(Tx - Rx - I)$. Tx and Rx were separated by 80% distance of communication range, i.e. 172 ft. and 120 ft. for DBPSK and DQPSK respectively, for an excellent PDR of more than 95%.

Keeping the Tx and Rx as stationary, Interferer (I) brought near to Rx with an intent to cause interference between Tx and Rx communication. As Interferer approaches Receiver, Receiver starts experiencing higher interference, resulting in increased number of collision, lower SIR and therefore lower PDR. As a result of this experiment, a relationship between RSSI versus PDR 15(a) and SIR versus PDR

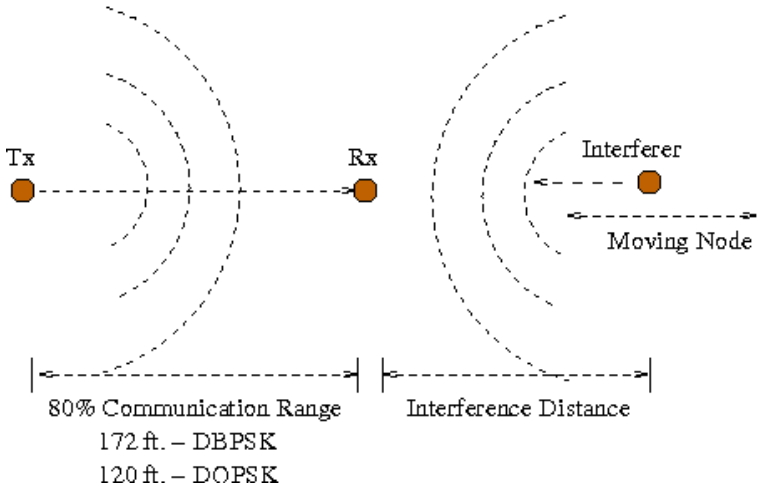
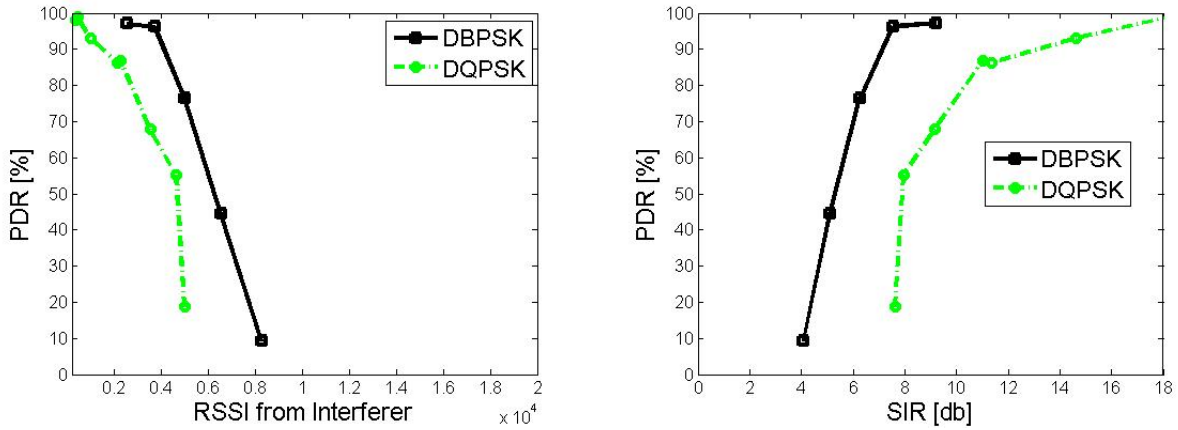


Figure 14: Experimental setup for finding interference range

15(b) is obtained. SIR at the receiver is calculated as the difference between RSSI from Transmitter and RSSI from Interferer.

From the results, Interference Range $((1+\Delta)r)$ is estimated to be 220 ft and 235 ft for DBPSK and DQPSK respectively. It should be noted that the degree of capture is quite significant for low data rate communication as in DBPSK, which coincide with the findings of CITE reference 19.

After finding the interference range$((1+\Delta)r)$, carrier sense range $((2+\Delta)r)$ was found to be 392 ft and 355 ft respectively for DBPSK and DQPSK. It can be observed that there is a margin (37 ft) existing between these two different data rate carrier sense ranges. This coincides with the multi-rate margin explained earlier. Since the data rates are too close in magnitude, the margin is not that significant as possible for 6 & 54 Mbps communication scenario.



(a) RSSI v/s Interferer PDR plot

(b) SIR v/s PDR Plot

Figure 15: Measurement plots showing effect of interferer on communication

	DBPSK	DQPSK
Communication Range	215 ft	155 ft
Interference Range	220 ft	235 ft
Carrier Sense Range	392 ft	355 ft
Multi-rate Margin	37 ft	

Table II: Table representing communication parameters for USRP

V.2 Verification of MTOP for PDR Sustainabilty

Current implementation of GNU Radio does packet transmission and reception through USRP based on streaming of packets i.e. there is no carrier sense mechanism implemented in the benchmark programs for packet transmission and reception. It was modified to include carrier sense algorithm [23]. Further this code was modified to include MTOP algorithm too. As explained earlier, under the MTOP algorithm, nodes supporting high data rate communication retransmit the packets without contending for medium, if that node is not the specified receiver.

Though the wireless nodes following MTOP, retransmits the packet to next hop without contending for the carrier, it does not lead to packet collision over the next

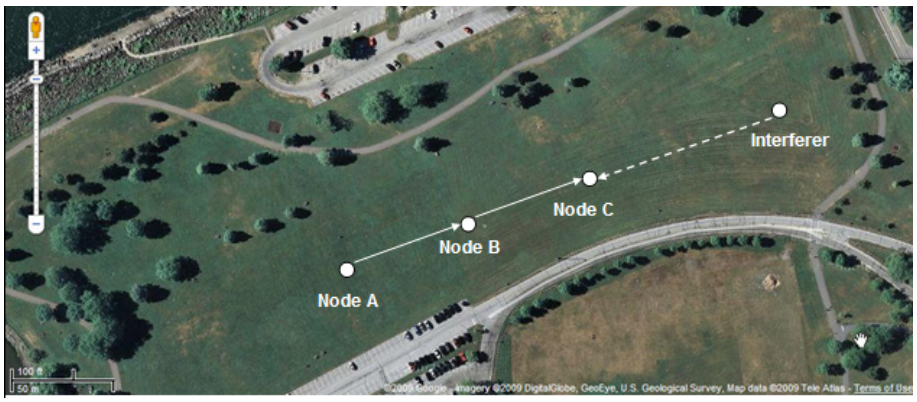


Figure 16: USRP nodes placement in Edgewater Park Cleveland

	PDR (%)	Throughput (Kbps)
Without MTOP	87.53	53.64
With MTOP	90.54	55.31

Table III: Table representing PDR and throughput improvement

hop. The following two sets of experiments were conducted to exhibit this property of MTOP. The test environment consisted of five sets of USRP (A, B, C, D & E) hardware and host machines. In both the sets of experiments, measurement matrix consisted of PDR (Packet Delivery Ratio) and throughput at receivers. Node A was used as transmitter, which was transmitting data packets destined to receiver node C. Node C was kept outside the communication range of node A, enabling node B to be the intermediate hop for data packets as shown in Fig. 16. The remaining two nodes (D & E) were kept outside the carrier sense range of node A. Node D was configured as the transmitter and node E as the receiver.

First set consisted of observing wireless communication among three nodes with CSMA and with/without MTOP algorithm. In the first part of this exercise, nodes A, B & C used DQPSK modulation scheme at 400 Kbps and interferer node D used DBPSK modulation at 200 Kbps. Since MTOP affects only the working of

intermediate hop, node B was selected for implementing MTOP for this prototype. Measurements were recorded for both the scenarios in which node B was working on CSMA with and without MTOP implementation. The goal of this experiment was to prove that the second hop of transmission does not get impacted by the presence of an interferer sitting outside the carrier sense range of transmitting node i.e. node A. In this exercise, firstly, PDR and throughput measurements for node C were recorded without MTOP implementation on intermediate hop node i.e. node B. In the second part of this exercise, measurements were recorded with MTOP implementation on node B.

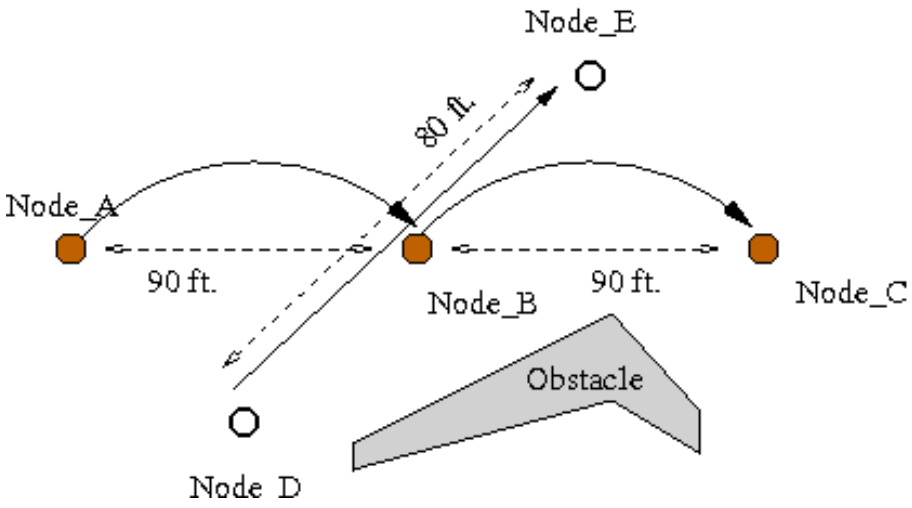


Figure 17: Test setup to observe the network performance improvement

Experimental observations III show a marginal improvement of 3.43% in PDR when nodes communicate with MTOP as compared to without MTOP implementation. It proves the argument of no adverse impact on wireless network performance when intermediate nodes send a packet without waiting for next contention period (after DIFS). Even the interferer, sitting outside the carrier sense range, could not interfere with the second hop communication. In addition to PDR, throughput with

	Node C		Node E	
	PDR (%)	Throughput (Kbps)	PDR (%)	Throughput (Kbps)
Without MTOP	90.22	48.37	71.94	38.83
With MTOP	93.15	54.15	76.62	44.54

Table IV: PDR and throughput recorded for node C and E for out of carrier range interference

MTOP in effect was also higher by 3.11% than without MTOP scenario, proving our argument of improvement in end-to-end communication scenario. It is expected that throughput improvement can be even higher for a sparse multi-rate environment i.e. 6 and 54 Mbps communication scenario.

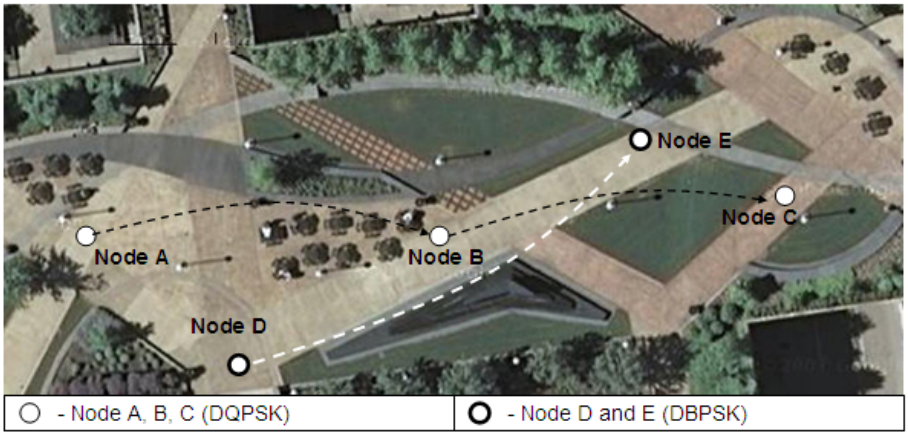


Figure 18: Real test setup in front of Rhodes Tower of Cleveland State University

The second set of experiments was done to observe the effect of MTOP on communication inside the multi-rate wireless network boundaries. Without MTOP algorithm, each node competes for the medium and transmits the packet on medium availability. This restricts the ability to transmit the data packet over multiple hops of communication and reach the destination in less time. Throughput improvement is the main objective of MTOP. Using MTOP, a packet can be transmitted much faster over a series of hops without possible packet collision.

	PDR (%)	Throughput (Kbps)
Without MTOP	81.08	43.59
With MTOP	84.88	49.34

Table V: Average PDR and throughput for out of carrier range interference

In this exercise, we wanted to prove that MTOP improves the network performance through increased throughput without increased collision. A set of PDR and throughput measurements was taken in this scenario with all the nodes inside the carrier sense range of node A. Node A, B & C were kept in a line with intermediate distance of 90 ft. between nearby nodes. Node D & E were kept in the vicinity of node B & C with a distance of 80 ft. in between them. The testing setup is shown in Fig. V.2 with a top view of the location, in Fig. V.2, where this experiment was actually performed. Firstly, the measurements were taken without MTOP implementation on node B followed by another set of measurements with MTOP implementation. Under the MTOP implementation, node B forwards the packets received from node A, to node C which is the packet destination node. Frame length of each packet transmitted was kept as 1000 bytes and each measurement was performed multiple times to exclude any effect of environmental disturbances i.e. wind, temperature etc. It was observed that throughput was improved by 13.19% after implementing MTOP in the wireless scenario, without any impact on PDR which was seen as almost constant. Further details of the measurements are listed in table IV and V.

Chapter VI

CONCLUSION

MTOP is an opportunistic transmission approach to improve the multi-hop communication in a multi-rate wireless environment. In this research project, a prototype implementation of MTOP on the software radio was achieved. For the proof of concept, MAC layer implementation with Career Sense Multiple Access (CSMA) was desired. Since MAC layer implementation for the targeted prototype hardware was not possible in entirety, a partial MAC layer implementation from BBN's Adroit project was taken. The timing requirements for the real wireless networks were seen to be very difficult with the given hardware platform. Therefore the timing requirements were relaxed to work around the communication blind spot due to high latency.

MTOP has been seen as performing better as compared to a simple ad-hoc network communication scenario. While the communication of second hop is shown to be unaffected by any interference from outside the carrier sense range of first hop, the network performance inside the network cluster is shown to be improved. Throughput improved by 13.19% with a partial improvement of 4.68% in PDR.

For a long time, there was an opportunity existed to improve the network performance further with a novel multi-hop communication approach in multi-rate net-

works. This report presents a novel approach investigated at Cleveland State University by Dr. Chansu Yu. With this thesis, an attempt is made to showcase the impact of MTOP on multi-rate wireless networks in the real-world through experimental data from a Software Radio prototype.

Chapter VII

FUTURE WORK

The implementation of MTOP on Software Radio platform consisting of USRP and GNU Radio has indeed shown the proof of concept. Though there is still a place for improvement in the experimental data derived from the experiments. With the new USRP hardware, which uses Ethernet to connect with the host PC, the latency introduced by USB communication is expected to be reduced drastically. Apparently this Ethernet connection works at Gigabit rate which is surely superior to the current USB communication rate of 480Mbps. Therefore as a subsequent step, experimentation of MTOP with USRP2 is a proposed scope for future work, which may give much better results as compared to USRP1.

Entire academic community relies highly on simulations for proof of concept till a right prototype can be selected and the new idea is implemented. Even though this research work has been able to verify and show the better performance achieved from MTOP, simulating MTOP on conventional simulation software NS-2 will greatly support the experimental findings. Though simulations themselves are prone to errors and are not able to replicate real world scenario, wide spread acceptability of NS-2 by research community makes it a perfect choice. I envisage that using these two

steps in future work can prove the performance improvement by MTOP over other conventional communication approaches.

BIBLIOGRAPHY

[1] *"IEEE 802.11-1999, Local and Metropolitan Area Network, Part 11: Wireless LAN Medium Access Control and Physical Layer Specifications".*

[2] *IEEE 802.11e, Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS)* Supplement to IEEE 802.11 Standard, Nov. 2005.

[3] *www.ettus.com*

[4] *www.gnuradio.org/trac*

[5] B. Sadeghi, V. Kanodia, A, Sabharwal, E. Knightly *"Opportunistic Media Access for Multirate Ad Hoc Networks",* Mobicom, September 2002.

[6] C. Yu, K. Shin, L. Song *Link Layer Salvagin for Making Routing Progress in Mobile Ad HOc Nerworks",* ACM MobiHoc, 2005.

[7] M. Zorzi, R. Rao *Capture and retransmission control in mobile radio",* IEEE Journal on Selected Areas in Communications, 12(8), 1994, 1289-1298.

[8] A. Kamerman, L. Monteban *"WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band",* Bell Labs Technical Journal, Summer 1997, 118-133.

[9] M. Lacage, M.H. Manshaei, T. Turletti *"IEEE 802.11 Rate Adaptation: A Practical Approach",* ACM MSWiM, 2004

[10] L. Iannone, S. Fdida *"Can Multi-rate Radios reduce end-to-end delay in mesh networks? A simulation case study",* Mesh Networking (Meshnets), 2005.

[11] A. Akella, G. Judd, P. Steenkiste, S. Seshan *"Self Management in Chaotic Wireless Deployments",* ACM MobiCom, 2005.

[12] J. Kim, S. Kim, S. Choi, D. Qiao *"CARA: Collision-Aware Rate Adaptationfor IEEE 802.11 WLANs",* IEEE INFOCOM, 2006.

[13] S. Kim, S.-J. Lee, S. Choi, *"The Impact of IEEE 802.11 MAC Strategies on Multi-Hop Wireless Mesh Networks",* IEEE WiMesh, 2006.

[14] G. Tan, J. Guttag *"Time-based Fairness Improves Performance in Multi-rate Wireless LANs",* The USENIX Annual Technical Conference, 2004.

[15] M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda *"Performance anomaly of 801.11b",* IEEE IFOCOM, 2003.

[16] I. Tinnirello, S. Choi *"Temporal fairness provisioning in multirate contention-based 802.11e WLANs",* IEEE WoWMoM, 2005.

[17] C. Yu, K. G. Shin, L. Song *"Maximizing Communication Concurrency via Link-Layer Packet Salvaging in Mobile Ad Hoc Networks",* IEEE Trans. Mobile Computing, 6(4), Apr. 2007.

[18] H. Zhai, Y. Fang, *"Physical Carrier Sensing and Spatial Reuse in Multirate and Multihop Wireless Ad Hoc Networks",* IEEE INFOCOM, 2006.

[19] Schmidt Thomas, Sekkat Oussama, Shrivastava Mani, *"An Experimental study of network performance impact of increased latency in software define radios",* 1st IEEE Workshop on Networking Technologies for Software Defined Radio Networks, 2006. SDR06.

[20] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, P. Steenkiste, *Enabling MAC Protocol Implementations on Software-Defined Radios",* CISDR Workshop, 2008.

[21] Dhar Rahul, George Gesly, Malani Amit, Steenkiste Peter, *"Supporting Integrated MAC and PHY Software Development for the USRP SDR",* 1st IEEE Workshop on Networking Technologies for Software Defined Radio Networks, 2006. SDR06.

[22] J.Corgan, Mail archive, http://www.ruby-forum.com/topic/172606.

[23] *http://acert.ir.bbn.com/projects/adroit/*

[24] *www.cs.uni-paderborn.de/en/research-group/research-group-computer-networks/projects/gsr.html*

[25] *http://www.nd.edu/ jnl/sdr/docs/tutorials/*

[26] Chen Ke-Yu, Chen Zhi-Feng, *"GNU Radio",* http://www.wu.ece.ufl.edu/projects /softwareRadio/documents/Project Report_James Chen.pdf

[27] G. Tan, J. Guttag, *"Time-based Fairness Improves Performance in Multi-rate Wireless LANs",* The USENIX Annual Technical Conference, 2004.

[28] *http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html*

[29] J. Polastre, R. Szewczyk, D. Culler *"Telos: Enabling Ultra-Low Power Wireless Research",*

[30] A. Jow, C. Schurgers, D. Palmer *"CalRadio: A Portable, Flexible 802.11 Wireless Research Platform",* MobiEval, June 2007.

[31] G. Holland, N. Vaidya, P. Bahl *"A Rate-Adaptive MAC Protocol for Multi-hop Wireless Networks",* ACM MobiCom, July 2001.

[32] Z. Ji, Y. Yang, J. Zhou, M. Takai, R. Bagrodia *"Exploiting Medium Access Diversity in Rate Adaptive Wireless LANs",* ACM MobiCom, 2004

APPENDIX

APPENDIX A
Program: Transmission without Carrier Sense

```
#!/usr/bin/env python
#
# GNU Radio Open Source License agreement ...
#

from gnuradio import gr, gru, modulation_utils
from gnuradio import usrp
from gnuradio import eng_notation
from gnuradio.eng_option import eng_option
from optparse import OptionParser

import random, time, struct, sys

# from current dir
from transmit_path import transmit_path
import fusb_options


class my_top_block(gr.top_block):
    def __init__(self, modulator, options):
        gr.top_block.__init__(self)
        self.txpath = transmit_path(modulator, options)
        self.connect(self.txpath)

# /////////////////////////////////////////////////////////////////////////
#                                   main
# /////////////////////////////////////////////////////////////////////////

def main():

    def send_pkt(payload='', eof=False):
        return tb.txpath.send_pkt(payload, eof)

    def rx_callback(ok, payload):
        print "ok = %r, payload = '%s'" % (ok, payload)

    mods = modulation_utils.type_1_mods()

    parser = OptionParser(option_class=eng_option,
                          conflict_handler="resolve")
    expert_grp = parser.add_option_group("Expert")
```

```
parser.add_option("-m", "--modulation", type="choice",
                  choices=mods.keys(), default='gmsk',
                  help="Select modulation from: %s [default=%%default]"
                        % (', '.join(mods.keys()),))
parser.add_option("-s", "--size", type="eng_float", default=1500,
                  help="set packet size [default=%default]")
parser.add_option("-M", "--megabytes", type="eng_float", default=1.0,
                  help="set megabytes to transmit [default=%default]")
parser.add_option("","--discontinuous", action="store_true",
                  default=False, help="enable discontinous \\
                  transmission (bursts of 5 packets)")
parser.add_option("","--from-file", default=None,
                  help="use file for packet contents")

transmit_path.add_options(parser, expert_grp)

for mod in mods.values():
    mod.add_options(expert_grp)

fusb_options.add_options(expert_grp)
(options, args) = parser.parse_args ()

if len(args) != 0:
    parser.print_help()
    sys.exit(1)

if options.tx_freq is None:
    sys.stderr.write("You must specify -f FREQ or --freq FREQ\n")
    parser.print_help(sys.stderr)
    sys.exit(1)

if options.from_file is not None:
    source_file = open(options.from_file, 'r')

# build the graph
tb = my_top_block(mods[options.modulation], options)

r = gr.enable_realtime_scheduling()
if r != gr.RT_OK:
    print "Warning: failed to enable realtime scheduling"

tb.start()                      # start flow graph

# generate and send packets
nbytes = int(1e6 * options.megabytes)
```

```
    n = 0
    pktno = 0
    pkt_size = int(options.size)

    while n < nbytes:
        if options.from_file is None:
        data = "A-node" + (pkt_size - 8) * chr(pktno & 0xff)
        else:
            data = source_file.read(pkt_size - 2)
            if data == '':
                break;

        payload = struct.pack('!H', pktno & 0xffff) + data
        send_pkt(payload)
        n += len(payload)
        sys.stderr.write('.')
        if options.discontinuous and pktno % 5 == 4:
            time.sleep(1)
        pktno += 1

    send_pkt(eof=True)

    tb.wait()                       # wait for it to finish

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        pass
```

APPENDIX B
Program: Reciever with PDR estimation

```
#!/usr/bin/env python
#
# Copyright 2005,2006,2007 Free Software Foundation, Inc.
#
# This file is part of GNU Radio
#
# GNU Radio is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3, or (at your option)
# any later version.
#
# GNU Radio is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GNU Radio; see the file COPYING.  If not, write to
# the Free Software Foundation, Inc., 51 Franklin Street,
# Boston, MA 02110-1301, USA.
#

from gnuradio import gr, gru, modulation_utils
from gnuradio import usrp
from gnuradio import eng_notation
from gnuradio.eng_option import eng_option
from optparse import OptionParser

import random
import struct
import sys

from receive_path import receive_path
import fusb_options


class my_top_block(gr.top_block):
    def __init__(self, demodulator, rx_callback, options):
        gr.top_block.__init__(self)
        self.rxpath = receive_path(demodulator, rx_callback, options)
        self.connect(self.rxpath)
```

```
# ////////////////////////////////////////////////////////////////////
#                                     main
# ////////////////////////////////////////////////////////////////////

global n_rcvd, n_right

def main():
    global pktno, n_rcvd, n_right, n_nodeApkt, n_nodeCpkt, pdr

    n_rcvd = 0
    n_right = 0
    n_nodeApkt = 0
    n_nodeCpkt = 0

    def rx_callback(ok, payload):
        global pktno, n_rcvd, n_right, n_nodeApkt, n_nodeCpkt
        (pktno,) = struct.unpack('!H', payload[0:2])
        n_rcvd += 1

        if ok:
        n_right += 1
        if payload[2:8] == "A-node":
        n_nodeApkt += 1

    demods = modulation_utils.type_1_demods()

    # Create Options Parser:
    parser = OptionParser(option_class=eng_option,
               conflict_handler="resolve")
    expert_grp = parser.add_option_group("Expert")

    parser.add_option("-m", "--modulation", type="choice",
              choices=demods.keys(), default='gmsk',
                      help="Select modulation from: %s [default=%%default]"
                            % (', '.join(demods.keys()),))

    receive_path.add_options(parser, expert_grp)

    for mod in demods.values():
        mod.add_options(expert_grp)

    fusb_options.add_options(expert_grp)
    (options, args) = parser.parse_args ()

    if len(args) != 0:
```

```
        parser.print_help(sys.stderr)
        sys.exit(1)

    if options.rx_freq is None:
        sys.stderr.write("You must specify -f FREQ or --freq FREQ\n")
        parser.print_help(sys.stderr)
        sys.exit(1)

    # build the graph
    tb = my_top_block(demods[options.modulation], rx_callback, options)

    r = gr.enable_realtime_scheduling()
    if r != gr.RT_OK:
        print "Warning: Failed to enable realtime scheduling."

    tb.start()        # start flow graph
    tb.wait()         # wait for it to finish
    print 'Packet Received = %s' %(n_nodeCpkt)
    pdr = (n_nodeCpkt)/(500)

    print 'PDR = %s' % (pdr)

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        pass
```

APPENDIX C

Program: Packet forwarding with Carrier Sense and without MTOP

```
#!/usr/bin/env python
#
# Copyright 2005, 2006, 2007 Free Software Foundation, Inc.
#
# This file is part of GNU Radio
#
# GNU Radio is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3, or (at your option)
# any later version.
#
# GNU Radio is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GNU Radio; see the file COPYING.  If not, write to
# the Free Software Foundation, Inc., 51 Franklin Street,
# Boston, MA 02110-1301, USA.
#

from gnuradio import gr, gru, modulation_utils
from gnuradio import usrp
from gnuradio import eng_notation
from gnuradio.eng_option import eng_option
from optparse import OptionParser
from transmit_path import transmit_path
from receive_path import receive_path

import random, time, struct, sys
import fusb_options


class my_top_block(gr.top_block):
    def __init__(self, modulator, demodulator, rx_callback, options):
        gr.top_block.__init__(self)
        self.txpath = transmit_path(modulator, options)
        self.rxpath = receive_path(demodulator, rx_callback, options)
        self.connect(self.txpath)
        self.connect(self.rxpath)
```

```
# ////////////////////////////////////////////////////////////////////
#                                     main
# ////////////////////////////////////////////////////////////////////
def main():

    def send_pkt(payload='', eof=False):
        return tb.txpath.send_pkt(payload, eof)

    def rx_callback(ok, payload):
        global n_rcvd, n_right, n_Apkt
        global rcv_flag
        n_rcvd = 0
        n_right = 0
        n_Apkt = 0
        print "ok = %r" % (ok)
        (pktno,) = struct.unpack('!H', payload[0:2])
        n_rcvd += 1
        if ok:
            n_right += 1
            if payload[2:8] == "A-node":
                n_Apkt += 1
                payload[2:8] = "B-node"

                while 1:
                if carrier_sensed():
                        continue
                    else:
                        time.sleep(min_delay)    #IFS
                        if carrier_sensed():
                            continue
                        else:
                            rand_num = random.randint(1,2)
                            #slot_time * random
                            time.sleep(min_delay * rand_num)
                            if carrier_sensed():
                                continue
                            else:
                                break
                send_pkt(payload)

    def carrier_sensed():
        """  Return True if the receive path thinks there's carrier """
        return tb.rxpath.carrier_sensed()
```

```
demods = modulation_utils.type_1_demods()
mods = modulation_utils.type_1_mods()

parser = OptionParser(option_class=eng_option,
                      conflict_handler="resolve")
expert_grp = parser.add_option_group("Expert")

parser.add_option("-m", "--modulation", type="choice",
                  choices=mods.keys(), default='gmsk',
                  help="Select modulation from: %s [default=%%default]"
                        % (', '.join(mods.keys()),))
parser.add_option("-s", "--size", type="eng_float", default=1500,
                  help="set packet size [default=%default]")
parser.add_option("-M", "--megabytes", type="eng_float", default=1.0,
                  help="set megabytes to transmit [default=%default]")
parser.add_option("","--discontinuous", action="store_true",
                  default=False, help="enable discontinous \\
                  transmission (bursts of 5 packets)")
parser.add_option("","--from-file", default=None,
                  help="use file for packet contents")

receive_path.add_options(parser, expert_grp)
transmit_path.add_options(parser, expert_grp)

for mod in mods.values():
    mod.add_options(expert_grp)

fusb_options.add_options(expert_grp)
(options, args) = parser.parse_args ()

if len(args) != 0:
    parser.print_help()
    sys.exit(1)

if options.tx_freq is None:
    sys.stderr.write("You must specify -f FREQ or --freq FREQ\n")
    parser.print_help(sys.stderr)
    sys.exit(1)

# build the graph
tb = my_top_block(mods[options.modulation],
                  demods[options.modulation], rx_callback, options)
r = gr.enable_realtime_scheduling()
if r != gr.RT_OK:
    print "Warning: failed to enable realtime scheduling"
```

```
    tb.start()                      # start flow graph
    tb.wait()                       # wait for it to finish

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        pass
```

APPENDIX D

Program: Packet forwarding with Carrier Sense and MTOP

```
#!/usr/bin/env python
#
# Copyright 2005, 2006, 2007 Free Software Foundation, Inc.
#
# This file is part of GNU Radio
#
# GNU Radio is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3, or (at your option)
# any later version.
#
# GNU Radio is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GNU Radio; see the file COPYING.  If not, write to
# the Free Software Foundation, Inc., 51 Franklin Street,
# Boston, MA 02110-1301, USA.
#

from gnuradio import gr, gru, modulation_utils
from gnuradio import usrp
from gnuradio import eng_notation
from gnuradio.eng_option import eng_option
from optparse import OptionParser

import random, time, struct, sys

from transmit_path import transmit_path
from receive_path import receive_path
import fusb_options

class my_top_block(gr.top_block):
    def __init__(self, modulator, demodulator, rx_callback, options):
        gr.top_block.__init__(self)
        self.txpath = transmit_path(modulator, options)
    self.rxpath = receive_path(demodulator, rx_callback, options)
        self.connect(self.txpath)
    self.connect(self.rxpath)
```

```
# ////////////////////////////////////////////////////////////////////
#                                     main
# ////////////////////////////////////////////////////////////////////

def main():

    def send_pkt(payload='', eof=False):
        return tb.txpath.send_pkt(payload, eof)

    def rx_callback(ok, payload):
        global n_rcvd, n_right, n_Apkt
        global rcv_flag
    n_rcvd = 0
    n_right = 0
    n_Apkt = 0
        print "ok = %r" % (ok)

        (pktno,) = struct.unpack('!H', payload[0:2])
        n_rcvd += 1
        if ok:
            n_right += 1
            if payload[2:8] == "A-node":
        n_Apkt += 1
        payload = struct.pack('!H', pktno & 0xffff) + "A-node" + \\
                              (pkt_size - 8) * chr(pktno & 0xff)
        send_pkt(payload)

    def carrier_sensed():
        """  Return True if the receive path thinks there's carrier """
        return tb.rxpath.carrier_sensed()

    demods = modulation_utils.type_1_demods()
    mods = modulation_utils.type_1_mods()

    parser = OptionParser(option_class=eng_option,
                          conflict_handler="resolve")
    expert_grp = parser.add_option_group("Expert")


    parser.add_option("-m", "--modulation", type="choice",
                      choices=mods.keys(), default='gmsk',
                      help="Select modulation from: %s [default=%%default]"
                            % (', '.join(mods.keys()),))
    parser.add_option("-s", "--size", type="eng_float", default=1500,
                      help="set packet size [default=%default]")
```

```
    parser.add_option("-M", "--megabytes", type="eng_float", default=1.0,
                      help="set megabytes to transmit [default=%default]")
    parser.add_option("","--discontinuous", action="store_true",
                      default=False, help="enable discontinous \\
                      transmission (bursts of 5 packets)")
    parser.add_option("","--from-file", default=None,
                      help="use file for packet contents")

    receive_path.add_options(parser, expert_grp)
    transmit_path.add_options(parser, expert_grp)

    for mod in mods.values():
        mod.add_options(expert_grp)

    fusb_options.add_options(expert_grp)
    (options, args) = parser.parse_args ()

    if len(args) != 0:
        parser.print_help()
        sys.exit(1)

    if options.tx_freq is None:
        sys.stderr.write("You must specify -f FREQ or --freq FREQ\n")
        parser.print_help(sys.stderr)
        sys.exit(1)

    # build the graph
    tb = my_top_block(mods[options.modulation], demods[options.modulation],
                      rx_callback, options)

    r = gr.enable_realtime_scheduling()
    if r != gr.RT_OK:
        print "Warning: failed to enable realtime scheduling"

    tb.start()                       # start flow graph
    tb.wait()                       # wait for it to finish

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        pass
```

APPENDIX E

Program: Reciever with throughput estimation

```
#!/usr/bin/env python
#
# Copyright 2005,2006,2007 Free Software Foundation, Inc.
#
# This file is part of GNU Radio
#
# GNU Radio is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3, or (at your option)
# any later version.
#
# GNU Radio is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GNU Radio; see the file COPYING.  If not, write to
# the Free Software Foundation, Inc., 51 Franklin Street,
# Boston, MA 02110-1301, USA.

from gnuradio import gr, gru, modulation_utils
from gnuradio import usrp
from gnuradio import eng_notation
from gnuradio.eng_option import eng_option
from optparse import OptionParser
from receive_path import receive_path

import random
import struct
import sys
import time
import fusb_options


class my_top_block(gr.top_block):
    def __init__(self, demodulator, rx_callback, options):
        gr.top_block.__init__(self)
        self.rxpath = receive_path(demodulator, rx_callback, options)
        self.connect(self.rxpath)
```

```
# ////////////////////////////////////////////////////////////////////////
#                                     main
# ////////////////////////////////////////////////////////////////////////
global n_rcvd, n_right, time_start, time_stop

def main():
    global pktno, n_rcvd, n_right, n_nodeApkt, time_start, time_stop
    n_rcvd = 0
    n_right = 0
    n_nodeApkt = 0
    time_start = 0
    time_stop = 0

    def rx_callback(ok, payload):
        global pktno, n_rcvd, n_right, n_nodeApkt, time_start, time_stop
        (pktno,) = struct.unpack('!H', payload[0:2])
        n_rcvd += 1

        if ok:
        n_right += 1
        if payload[2:8] == "A-node":
        if n_nodeApkt == 1:
            time_start = time.time()
        if n_nodeApkt == 400:
            time_stop = time.time()
        n_nodeApkt += 1

    demods = modulation_utils.type_1_demods()

    # Create Options Parser:
    parser = OptionParser (option_class=eng_option,
                           conflict_handler="resolve")
    expert_grp = parser.add_option_group("Expert")

    parser.add_option("-m", "--modulation", type="choice",
                      choices=demods.keys(), default='gmsk',
                      help="Select modulation from: %s [default=%%default]"
                            % (', '.join(demods.keys()),))

    receive_path.add_options(parser, expert_grp)
    for mod in demods.values():
        mod.add_options(expert_grp)

    fusb_options.add_options(expert_grp)
    (options, args) = parser.parse_args ()
```

```
    if len(args) != 0:
        parser.print_help(sys.stderr)
        sys.exit(1)

    if options.rx_freq is None:
        sys.stderr.write("You must specify -f FREQ or --freq FREQ\n")
        parser.print_help(sys.stderr)
        sys.exit(1)

    # build the graph
    tb = my_top_block(demods[options.modulation], rx_callback, options)

    r = gr.enable_realtime_scheduling()
    if r != gr.RT_OK:
        print "Warning: Failed to enable realtime scheduling."

    tb.start()        # start flow graph
    tb.wait()         # wait for it to finish
    print 'Packet Received = %s, StartTime = %s, StopTime = %s'
            %(n_nodeApkt, time_start, time_stop)
    time1 = time_stop - time_start

    print 'Throughput for first 400 packets = %s' % (400/time1)

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        pass
```