AN AD HOC NETWORKING TESTBED USING SOFTWARE-DEFINED RADIOS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

NEGIN TABRIZI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2019

Approval of the thesis:

AN AD HOC NETWORKING TESTBED USING SOFTWARE-DEFINED RADIOS

submitted by **NEGIN TABRIZI** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

ABSTRACT

AN AD HOC NETWORKING TESTBED USING SOFTWARE-DEFINED RADIOS

Tabrizi, Negin
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Ertan Onur

September 2019, 62 pages

For establishing communication infrastructures in the aftermath of man-made or natural disasters, ad hoc networking architectures seem to be more adequate. There are various simulators for studying ad hoc networks. However, simulators cannot accurately model all the factors of the physical environment. By using a test-bed, the significant gap between simulations and real-life implementations can be reduced. In this work, an ad hoc indoor testbed has been developed using software-defined radios (SDR). By using SDRs, we can bring reconfigurability and flexibility to our system. In order to develop and implement the physical and link layers, GNU Radio blocks are used. By abstracting the link-layer details through a virtual Ethernet interface, any legacy application can be run on the developed testbed. As an example, an open-source implementation of the optimized link state routing (OLSR) protocol and Babel routing protocol are used at the network layer in our experiments. We made the ad hoc networking testbed highly-redefinable by employing SDR and legacy software that could run on any legacy operating system or hardware. We report extensive evaluation results of the developed testbed.

Keywords: Ad hoc testbed, SDR platform, USRP devices, GNU Radio software, OLSR protocol, Babel routing protocol

ÖZ

YAZILIM TABANLI TELSİZLER KULLANARAK TASARSIZ AĞ TEST ORTAMI OLUŞTURULMASI

Tabrizi, Negin
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Ertan Onur

Eylül 2019 , 62 sayfa

İnsan kaynaklı veya doğal oluşan afetler sonrasında iletişim altyapılarını tekrar oluşturabilmek için tasarız ağ yapıları uygun görülmektedir. Bu ağları incelemek için birçok simülatör mevcuttur. Bununla birlikte, simülatörler fiziksel çevrenin tüm faktörlerini doğru bir şekilde modelleyemezler. Bir test ortamı kullanılarak, simülasyonlar ve gerçek hayattaki uygulamalar arasındaki farklılıklar önemli ölçüde azaltılabilir. Bu çalışmada, yazılım tabanlı radyoları (SDR) kullanarak bir tasarsız iç mekan test ortamı geliştirdik. SDR'leri kullanarak sistemimize yeniden yapılandırılabilirlik ve esneklik getirebiliriz. Fiziksel ve bağ katmanlarını geliştirmek ve uygulamak için GNU Radio bloklarını kullandık. Bağlantı katmanının ayrıntılarını sanal bir Ethernet arayüzü kullanarak geliştirilen bu test platformunda, herhangi bir uygulama çalıştırılabilir. Örnek olarak, deneylerimizde OLSR ve Babel yönlendirme protokollerinin açık kaynaklı bir uygulaması kullanılmıştır. Tasarsız ağ testlerini, herhangi bir işletim sisteminde veya donanımda çalışabilecek SDR yazılımları kullanarak rahatlıkla yeniden tanımlanabilir hale getirdik. Bu tezde, geliştirilen bu test ortamının kapsamlı değerlendirme sonuçlarını sunuyoruz.

Anahtar Kelimeler: Tasarsız Ağ Test Ortamı, SDR platformu, USRP cihazları, GNU Radio yazılımı, OLSR protokolü, Babel protokolü

To my family

ACKNOWLEDGMENTS

In the beginning, I would like to thank my supervisor, Dr.Ertan Onur, for his guidance and support in my research. I also would like to thank my beloved parents who have always encouraged me and give me confidence and courage. Especial thanks to my brother, who has always supported me in my life and also thanks to my friends who helped me during though moments and when I felt frustrated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

B.A.T.M.A.N	Better Approach To Mobile Ad hoc Network
CRC32	cyclic redundancy check
CSMA	Carrier Sense Multiple Access
DSDV	Destination-Sequenced Distance Vector routing
DSR	Dynamic Source Routing
EIGRRP	Enhanced Interior Gateway Routing Protocol
ETT	Expected Transmission Time
ETX	Expected Transmission Count
FANET	Flying Ad hoc Network
FPGA	Field Programmable Gate Array
GMSK	Gaussian Minimum Shift Keying
GRC	GNU Radio Companion
IF	Intermediate Frequency
LQ	Link Quality
MANETs	Mobile Ad hoc Networks
Msk	Minimum Shift Keying
NLQ	Neighbor Link Quality
OLSR	Optimized Link State Routing
RF	Radio Frequency
RFIC	Radio Frequency Integrated Circuit
RX	Receiver
SDR	Software Defined Radio
TVWS	TV Whitespaces

TX	Transmitter
UAVs	Unmanned Aerial Vehicles
UHD	USRP Hardware Driver
USRP	Universal Software Radio Peripheral
vEth	Virtual Ethernet
VNI	Visual Networking Index

CHAPTER 1

INTRODUCTION

Nowadays there is a tremendous use of network and one can see the use of network and sharing the data between different devices in everyone's life. The current trend toward wireless networks is much higher compared to wire connections mainly due to the increasing number of users who are rapidly growing and making the wire connection impractical. The amount of data produced by users and devices are increasing dramatically furthermore the quality of service, time and cost have become important issues. Based on the Cisco Visual Networking Index (VNI) report, there is a growing trend in the use of wireless networks. The report forecasted that the monthly global mobile data traffic will reach 49 Exabytes by 2021 and the global mobile data traffic will increase seven-fold between 2016 and 2021 [3]. Moreover, it was also suggested connecting to the Internet and have access to the Internet will become a significant goal for a rapidly increasing amount of users and applications.

The wireless network can be categorized into two modes as illustrated in Figure 1.1: infrastructure mode, such as Cellular Networks where devices communicate with each other through the access point, router or base station; or the second one, infrastructure-less networks where devices communicate with each other directly. For instance, multi-hop wireless ad hoc networks which are more flexible, low cost and rapidly deployed [4].

In the infrastructure network, the gateway is wired and fixed whereas, in ad hoc, each node has two functionalities, it can operate as a router or as a communication endpoint. The nodes are connected by the links. The nodes can be a wide range of devices that have equal status on the network. The nodes can move freely [5].

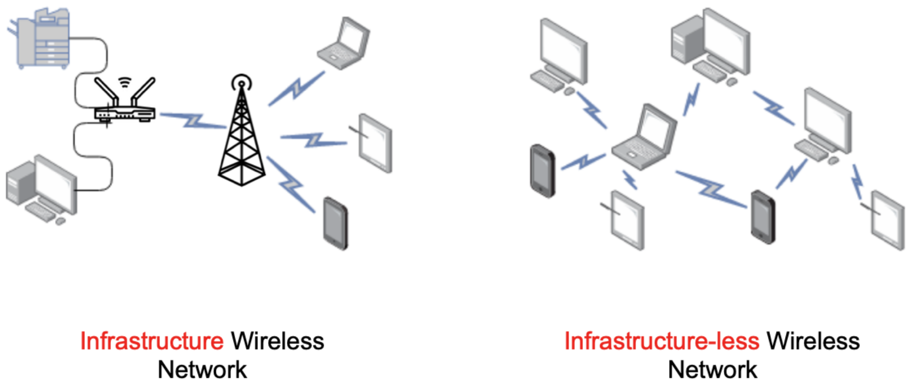

Infrastructure Wireless Network
Infrastructure-less Wireless Network

Figure 1.1: Schematic design of a wireless network: Infrastructure and Infrastructure-less network. In an infrastructure network, there are a base station, access point or router to make a connection while in infrastructure-less network devices communicate to each other directly.

1.1 Motivation and Problem Definition

There are various usage of ad hoc networks and the interest in this type of network is continuously increasing. However, most researches in this area usually have used simulators [6] [7]. By using a simulator, the development of new architectures and network protocols can be studied. The simulation brings flexibility and high level of scaling like a real network or even up to many thousands of nodes with the low cost. Nevertheless, we cannot consider some other facts such as the influence of hardware delays such as encoding/decoding time on our devices or the resource usage and the interoperability between different protocols and also signaling issues like timing and synchronization [8] [9]. As a result, although simulation is useful, in some cases it is not compatible with reality. In order to address this issue, a software-defined radio (SDR) testbed in ad hoc networks is proposed and implemented by using universal Software Radio Peripheral (USRP) devices and signal propagation and communication in this thesis. It can be said that in this thesis, due to the need of an ad hoc testbed, an indoor testbed is implemented which is reproducible and moreover, it can be used for other researches since it is flexible and configurable.

1.2 Contributions and Methodology

In designing our ad hoc network testbed some features are considered. The setup and the results are reproducible. In most cases, the running setup is a one-time experiment and rebuilding the setup is impossible. And for some other cases, the testbed is limited just for specific usage. However, our implemented testbed provides an ad hoc network that can be used for a variety of situations and testing and it is not just limited and specified just for this research. This indoor ad hoc network testbed provides a common platform for further researches. By making an indoor testbed, the more complex and the real environment is provided especially if the intended applications are indoor. The main contributions of this thesis can be categorized in the following order.

- Indoor ad hoc testbed is implemented rather than simulation to fill the gap between simulations and real-world experiments.
- Software-defined Radio (SDR) platform is used for bringing reconfigurability and flexibility to our system.
- TCP/IP communication is made between the nodes in the system.
- On top of this platform, Optimized Link State Routing Protocol (OLSR) and also Babel routing protocol are used and compared for discovering the nodes in our ad hoc testbed.

1.3 Outline

The proposed thesis is composed and organized in the following order. In Chapter 2 background information, definitions, the software and protocols which are used for creating the testbed are provided. Additionally, previous related work is reviewed. In Chapter 3, the ad hoc testbed which has been implemented and developed is presented. The experimental results from the prepared testbed are obtained and presented in Chapter 4. Finally, the conclusion and potential future research work are given in Chapter 5.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Software Defined Radio

Software Defined Radio (SDR) is a radio communication system and its concept introduced great flexibility and adaptability to networks by replacing radio hardware components by configurable and programmable software [10]. Therefore it brings reconfigurability to the system since radio functions are implemented in software and for various use cases there is no need to redesign the hardware. Instead of redesigning and reproducing the hardware, it can be simply reconfigured in software for supporting new functionality. Moreover, with the rapid evolution in wireless protocols, reconfigurability has an important role. In general, SDR brings creation and application development in the whole system, from network to service [11].

The SDR architectures are used for research and development purposes. However, implementing an optimized SDR system is costly, complex and time-consuming. Hence, in an increasing number of cases for implementing more rapidly and reduce the implementation time, already developed testbeds are used [12]. Although a simpler and more affordable testbed is desired for academic research.

2.2 The Universal Software Radio Peripheral

The Universal Software Radio Peripheral (USRP) is reconfigurable hardware developed by Ettus Reseach and by connecting it to the computer, it can be operated as a basic SDR platform [13]. The purpose of designing the USRP was to make a general computer works as a high bandwidth software-defined radios functions. The general architecture of USRPs is all the same while they provide an appropriate platform for conducting research in wireless ad hoc networks [14][15].

The USRP performs high-speed operations such as passband processing [16], the digital baseband to Intermediate Frequency (IF) and from IF to Radio Frequency (RF). Whereas all the waveform-specific digital processing such as modulation and demodulation are done on the host computer [17]. There is a diverse sort of USRPs and each has its specifications and characteristics.

2.2.1 USRP B210

The USRP B210 is a single board device that could produce fully integrated and continuous frequency coverage from 70 MHz to 6 GHz. A single-chip direct-conversion transceiver, the AD9361 RFIC, which is embedded in USRP B210 provides a real-time bandwidth up to 56 MHz. For signal processing and controlling the AD9361, Spatan6 FPGA is onboard. USRP B210 gets connected to the host computer through the SuperSpeed USB 3.0 connection. The board of USRP B210 is illustrated in Figure 2.1 and also the steel enclosure kit are illustrated in Figure 2.2. As can be seen, there are two different part on USRP B210 (A and B side). Since the USRP B210, is a two-channel USRP device, each side can transmit and receive frequency [18].

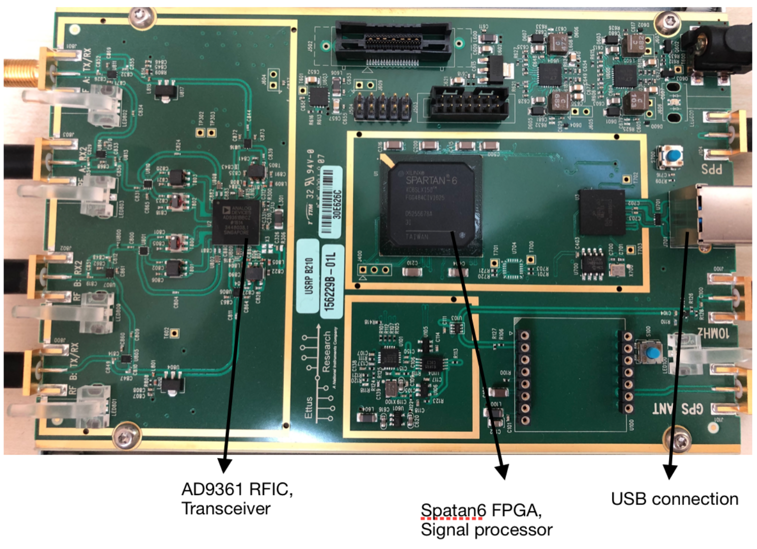


Figure 2.1: The USRP B210 board.

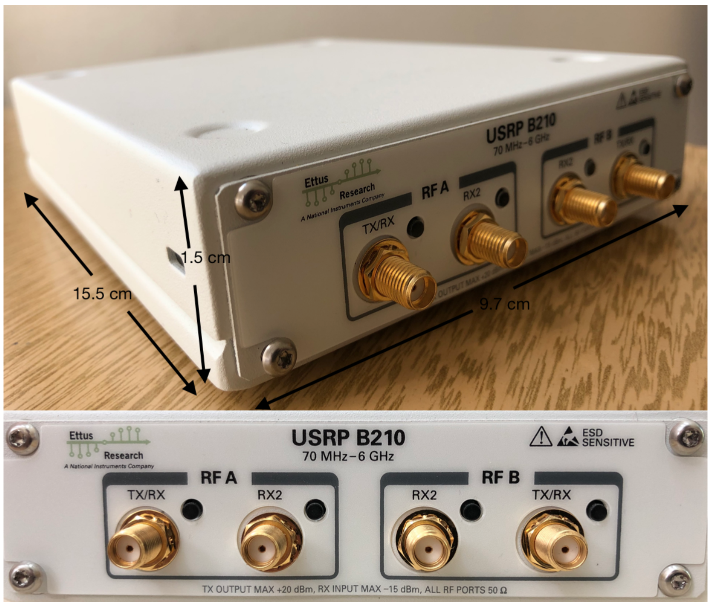

1.5 cm
15.5 cm
9.7 cm
USRP B210
70 MHz–6 GHz
Ettus Research
A National Instruments Company
RF A
RF B
TX/RX
RX2
ESD SENSITIVE
TX OUTPUT MAX +20 dBm, RX INPUT MAX -15 dBm, ALL RF PORTS 50 Ω

Figure 2.2: The USRP B210 with steel kit. There are one transmit and one receive channel on "$RF\ A$" and another transmit and receive channels on "$RF\ B$".

2.3 USRP Hardware Driver

USRP B210 is fully supported by the USRP Hardware Driver (UHD) software. UHD is an open-source driver that allows you to port your design and application to a higher-performance USRPs. A common Application Programming Interface (API) is provided by UHD that allows applications to be developed. Moreover, UHD is a cross-platform that can work by different software frameworks and development environments such as GNU Radio. With the GNU Radio, a system can be developed to process signals and implement a more sophisticated SDR platform [19]. Figure 2.3 shows the relation between the UHD and the GNU Radio.

2.4 GNU Radio

GNU Radio is free software for processing the signals and also it is open-source. Therefore, it is used widely in academic experiments to implement a variety of real-world radio systems. Although it was established for Linux platform, it is supported by other operating systems such as MAC and Windows platform [20]. It can be

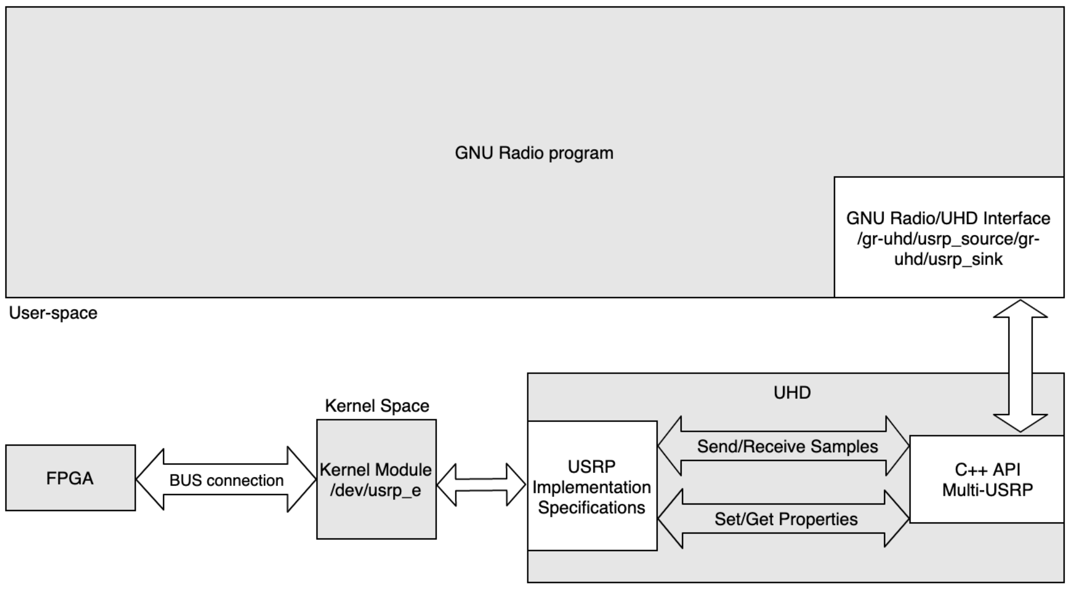

GNU Radio program
GNU Radio/UHD Interface /gr-uhd/usrp_source/gr-uhd/usrp_sink
User-space
Kernel Space
UHD
FPGA
BUS connection
Kernel Module /dev/usrp_e
USRP Implementation Specifications
Send/Receive Samples
Set/Get Properties
C++ API Multi-USRP

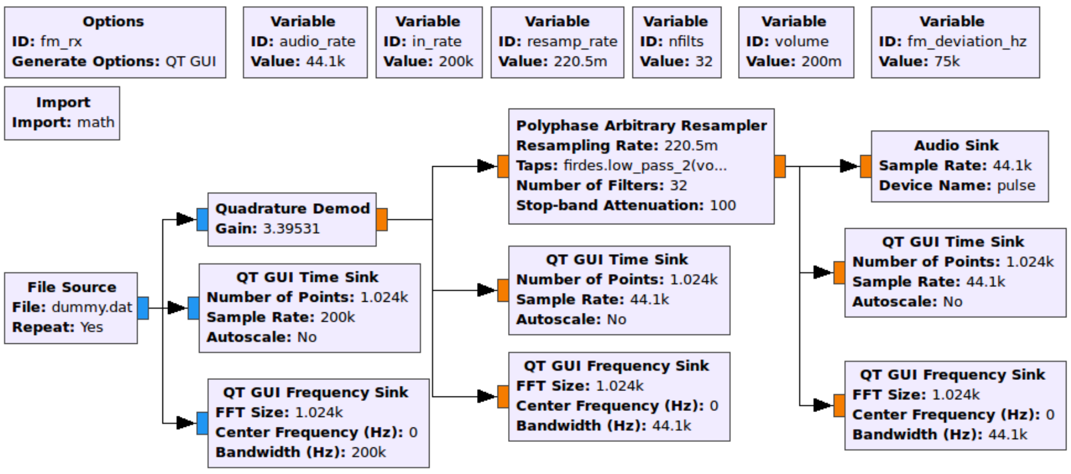
Figure 2.3: The UHD in GNU Radio/USRP SDR Platforms [1].

used for simulation environments without hardware or as an SDR platform if it is connected to the RF hardware like the USRP that has been explained in section2.2.

The target of GNU Radio was to bring the code to the antenna and do the other processing in software rather than hardware [21]. In GNU Radio, there are various building blocks. Each block is a piece of software for implementing specific signal processing functions like filtering, coding/decoding, modulation/demodulation. While the blocks are written in the $C++$ programming language, the data flows that connect these blocks and the GNU Radio applications are written in $Python$ language to simplify the building and implementations [22].

GNU Radio Companion (GRC) offers a graphical environment for GNU Radio users. In GRC, instead of writing the codes, the blocks can be dragged-and-dropped and the components can be connected and in general, the signal flow diagram can be generated. It is useful for rapidly building and implementing a simple radio system [23].

In Figure 2.4, the graphical environment of GNU Radio is presented. This a simple FM receiver sample which has been implemented in GNU Radio Companion and the flowgraphs are created in the graphical interface. It uses the file source block to read the file, then it sends them to FM demodulator (Quadrature Demod block). The signal is demodulated and converted the complex FM signal to a float signal by using the Quadrature Demod block.



Options
ID: fm_rx
Generate Options: QT GUI

Variable
ID: audio_rate
Value: 44.1k

Variable
ID: in_rate
Value: 200k

Variable
ID: resamp_rate
Value: 220.5m

Variable
ID: nfilts
Value: 32

Variable
ID: volume
Value: 200m

Variable
ID: fm_deviation_hz
Value: 75k

Import
Import: math

File Source
File: dummy.dat
Repeat: Yes

Quadrature Demod
Gain: 3.39531

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 200k
Autoscale: No

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 200k

Polyphase Arbitrary Resampler
Resampling Rate: 220.5m
Taps: firdes.low_pass_2(vo...
Number of Filters: 32
Stop-band Attenuation: 100

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 44.1k
Autoscale: No

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 44.1k

Audio Sink
Sample Rate: 44.1k
Device Name: pulse

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 44.1k
Autoscale: No

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 44.1k

Figure 2.4: An FM receiver example in GNU Radio graphical environment [2].

2.5 Optimized Link State Routing Protocol

As wireless networks communication becomes more advanced there is a need for a stable, scalable and robust routing protocol in ad hoc networking. Optimized Link State Routing (OLSR) protocol and Ad Hoc On-Demand Distance Vector (AODV) routing protocol, are two main routing protocols for mesh networks. Whereas the OLSR is configured more easily [24].

The purpose of designing the OLSR was to be able to work on any network equipment. It is a proactive protocol, therefore it reconfigures and reacts to network failures even before they happen. OSLRd is released under a BSD license and it can be integrated into other software projects and it is one of the most widely deployed open-source projects [25] [26].

OLSRd runs on the network layer and it can be portable to various platforms such as Windows, Linux, OS X, Google phone, NetBSD, etc. As it is mentioned in its name, it focuses on optimization. As a result, it is very fast while it has a low consuming tool. In OLSRd, each entity replies on any entry in the routing table.

OLSRv2 which is called as OLSRD2 outperforms previous OLSR version while it can work in larger networks [27]. It is a new version with much more features. Two features of OLSR and OLSRv2 for measuring the quality of a link are explained in the following:

- Expected Transmission Count (ETX) metric measures the number of the lost packet on each link and later uses the loss to define the cost for the link. However, the link speed and available bandwidth are not considered in the cost. The costs for the total path are minimized and also the OLSRD avoids unstable links. The ETX can be calculated as $ETX = \frac{1}{LQ \times NLQ}$ where LQ stands for link quality while the NLQ is the neighbour link quality [28].
- Expected Transmission Time (ETX) is another metric that works on ETX. It tries to estimate the transmission time for sending a packet over a link and it can be computed as $ETT = ETX \times \frac{S}{B}$ which multiplies the ETX with transmitted packet size (S) and then divides that by the link capacity (B) [28] [29].

2.6 Babel Routing Protocol

While OLSR is a link-state routing protocol, Babel, on the other hand, is an open-source proactive robust distance vector routing protocol. It uses the same ideas which are used in the three other main distance vector routing protocol as (AODV), Destination-Sequenced Distance Vector routing (DSDV) and Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP). However, in selecting the route it uses a different technique in order to avoid loops. Like OLSR, Babel uses the ETX metrics and it can be used for both IPv4 and IPv6 while it works for wired and wireless networks [30].

$Feasible\ Distance$ is the best metric towards a specific destination. The $Reported\ Distance$ is a metric that shows the distance to reach the target network from the neighbor node and when it comes to verifying a route, Babel uses $Feasibility\ Condition$ like EIGRP which means that a route would be accepted just in case that the Reported Distance is equal or less than the best path's Feasible Distance. Moreover, Babel uses history-sensitive routing. It means that if there are more than one routes to the destination, the one which is stable and had been used before would be selected if the link qualities and costs are all the same.

2.7 Related Work

As it was mentioned in Chapter 1, there is a transition from infrastructure networks into the infrastructure-less networks due to flexibility, dynamic and self-healing and self-configuring features in these type of networks. Hence, there has been a vast number of researches about the infrastructure-less network field. Although most of these researches used the simulators for generating a scalable environment. Nevertheless, in all existed simulators, the noise is defined by the users while in the real world there are some noises and interferences in the environment that we are not aware of, such as the number of base stations or the WiFi modems around the test environment that could cause unknown interferences. In other words, we can say that the simulation's environment is quite deterministic while the testbed provides a practical environment. Moreover, the impairments in the environment is increasing eventually as the test is running due to variety of reasons, for instance, the power could be lost in the device's circuit or voltage changes, and many other unknown reasons while in the simulators those impairments does not exist unless implemented. Additionally, there is a delay attribute in a point to point link for the propagation delay or transmission delay while the hardware delay is not considered in simulations.

In some other research like [31], a connection between simulation and the real network interfaces is made. By bridging the simulator to the real network devices both reality and scalability can be considered in a more advanced way. However, the gap between real-world and simulators could not be considered completely. As they mentioned in their paper due to the reasons that the data packets are represented differently in simulators and the real networks, and also they wanted to consider the real interferences, they connected the Omnet ++ simulator to the virtual Ethernet Interfaces. As a result, they found out that the processing delays are different from reality to simulators.

There are some ad hoc testbeds to capture more realistic results compared to simulators. Simulations are more limited since hardware delays, radio propagations and many other physical interferences are not considered. For some of these researches, outdoor testbeds are implemented and developed [32] [33] [34] [35]. Outdoor testbeds research usually tries to cover mobility issues. For instance in [32], a Flying Ad hoc Network (FANET) testbed is implemented. They focused on mobility issues and the communication of Unmanned Airborne Vehicles (UAVs) while they use WiFi connections between UAVs and the station. In [33], they compare the throughput of Mobile Ad hoc Networks (MANETs) in different scenarios by OPNET simulators while increasing the number of nodes. A MANETs testbed is implemented in research [34], to evaluate the ad hoc communication protocols in a disaster scenario. The authors in [35], provides a wireless mobile ad hoc network testbed while they use combinations of nodes. For mobile nodes on the ground, they use nodes on the vehicles and in the air, they use nodes on UAVs. In addition, some fixed nodes are used. They used the Dynamic Source Routing (DSR) protocol. As a result, the DSR protocol was limited to 5-hop diameter and increasing the nodes would cause throughput reduction. Although they wanted to provide a reproducible testbed, the same scenario throughput and latency results can only be captured again if the placement of the nodes were ideal and identical. For web browsing, there is a delay similar to a dial-up connection and for end-to-end voice communication, it can happen only for a network with a diameter of 3 hops. The main problem in [35], was the DSR protocol that could not make a routing table for highly mobile nodes.

In some other researches, indoor testbeds are provided which can be useful especially if they are going to be used for indoor applications [1] [36] [37] [38] [39]. These testbeds are developed for various purposes like energy efficiency [38], latency [1] [40], analysis and evaluation [39], and improvements [37]. In "Implementation of Hybrid Ad-Hoc Routing Protocol", Wi-Fi connection is used for making the ad hoc testbed [36]. In this thesis, SDR platforms were used for making radio connections. While SDR systems have the advantages to be reconfigurable and flexible, it faces some disadvantages like low throughput and high latency. There are several works for measuring the latency in SDR/USRP platforms [1] [40]. The authors in [1], try to find and analyze the source of the latency.

The thing that distinguishes our work from other studies is that in our study, we focus on the low throughput and how to increase that by evaluating different parameters while our testbed and results are reproducible whereas our ad hoc testbed is implemented by using SDR systems.

In [41], a TCP/IP communication is established by using USRPs. Their goal was to implement a TV Whitespaces (TVWS) detector with energy detection. Whereas the number of USRPs are limited to just two. In this thesis, a TCP/IP connection is made between two up to six different USRPs.

There are some researches about different routing protocols and the comparisons between them [42] [43]. In this thesis, we compare the OLSR and Babel routing protocol while a testbed is developed and prepared for further research on the various open-source routing protocols.

CHAPTER 3

SDR AD-HOC TESTBED IMPLEMENTATION AND DEVELOPMENT

The evaluation of wireless communication can be so challenging due to the high interferences that can affect the system. Nowadays, many simulation software is used for research in ad-hoc wireless networking since they are more flexible and scalable like the real world. However, in simulation, many simplifying assumptions are made. In multipath propagation, there are interferences and fading that we can classify as reflection, diffraction, and scattering [44]. In most software simulators, interferences, signal propagations, and the physical environment hardware features are not considered. In this thesis, instead of the simulation environment, a testbed experiment is made, to compare the results and see the impact of the real physical environment.

For setting up our indoor test-bed, some preparations and software have been used. Several USRP B210 devices that have been explained in Section 2.2.1, are connected to a single server as a node of an ad-hoc system via SuperSpeed USB 3.0 connections. Network namespaces are defined to separate the USRP devices. In this way, we can be sure that although they are on the same server, the only way to communicate with each other is through radio communication.

For radio communication between these nodes, SDR is used since it is more reconfigurable and it can be used in an increasing number of cases. However, to create the SDR system, GNU Radio is used along with USRPs. With GNU Radio we can reconfigure and control the parameters of the USRP through the USRP Hardware Driver (UHD) that all are presented in Chapter 2. Later the Tunnel block in GNU Radio is used for making a TCP/IP communication between nodes. This block creates a Tap interface in the kernel to tunnelling Ethernet frames since only virtual Ethernet (vEth) interfaces can be assigned to a network namespace.

3.1 TUN/TAP Interface

TUN/TAP are virtual network kernel interfaces without physical media associated and it can be used for point-to-point or Ethernet devices and instead of physical devices, it transmits and receives packets from user space programs and in while, the raw network traffics can be observed and analyzed. TUN simulates a network layer device which is layer three or IP layer, whereas TAP simulates a link layer device which is layer two or Ethernet layer [45].

The GNU Radio tunnel block provides a framework for building your MAC. It creates a virtual interface that needs to be configured with an IP address in the kernel and transceive Ethernet frames through that. After making a TCP/IP connection between nodes, OLSRD protocol which has been explained in Section 2.5 and Babel routing protocol which has been presented in Section 2.6 are used for finding the neighbour nodes and making the routing table. In our radio communication system for frequency modulation, Gaussian Minimum Shift Keying (GMSK) modulation is used since it works better than other frequency modulations [46] [47].

3.2 GMSK Modulation

GMSK modulation is a frequency shift keying modulation and it is more powerful compare to the other frequency shift keying like Minimum Shift Keying (MSK). Since in MSK, there is an interference between adjacent signals. Whereas in GMSK, there are no phase discontinuities and it uses the spectrum efficiently.

3.3 CRC32 Algorithms

In the receiving section, the received payloads are checked by cyclic redundancy check (CRC32) algorithms for controlling and error detection. The CRC is an error detection algorithm that is used normally in the digital network since it can detect common errors caused by noise and it is simple to implement and analyze [48]. The CRC algorithm works as follows: First, the data is treated as a binary number. Therefore, the message is interpreted as a long binary bitstream and then it is divided by another fix binary number which is called the polynomial. The remainder of this division is the CRC checksum. In the end, the fixed-length checksum is added to the message for encoding. In the receiving side, the receiver divides the message, by the same polynomial and if the result of this division is zero, then it shows there is no error and the transmission was successful [49].

3.4 Testbed Setup

The testbed setup for making an indoor SDR ad hoc testbed is illustrated in Figure 3.1.

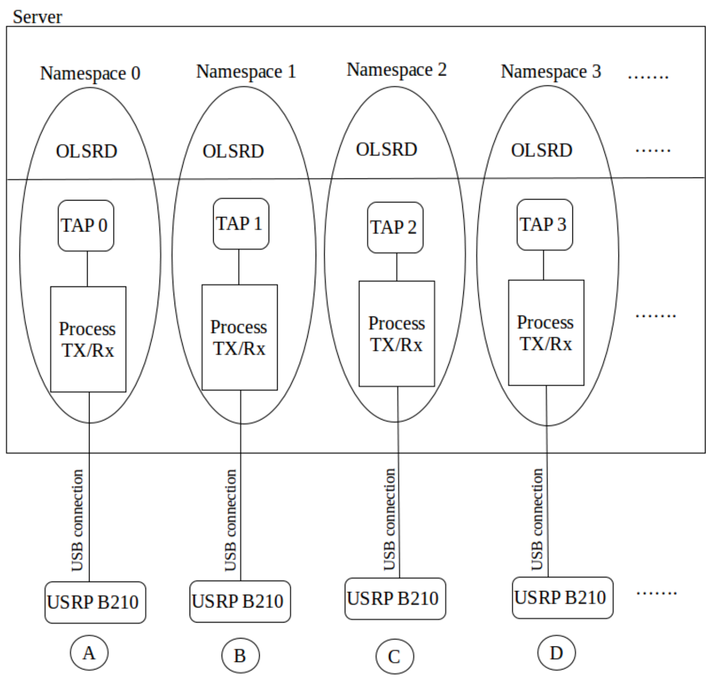


Figure 3.1: The testbed setup using the SDR system with USRP B210.

To get accurate results a script was written to run the commands in the same situation for 100 times and between each runs there is a delay of 30 sec. The timeline for running different commands are as shown in Figure 3.2. It takes 10 sec for a warm-up and configuring the interfaces if there are just two nodes and 120 sec for running the commands. The warm-up and configuring time could be raised to 25 seconds if the number of nodes increased to six nodes. The tunnel works for just two minutes and in while it can transfer approximately 13000 to 14000 Ethernet frames while the packet sizes are 1500 bytes. If it divided by the running time, it can be said that the throughput is around 1.3 to 1.4 Mbps. To open a google home page of 50 KB and for a web page with a picture of 100 KB is needed. Although 2 minutes run is a short period for running the code, it is sufficient to open a couple of web pages. However, it is not enough for video streaming. Since the test for a single value for different parameters are run for 100 times and there are sleep time and warm-up period and there are numerous parameters to be considered, the commands were run just for two minutes in our experiments. However, in a particular situation, the time has been raised to five minutes to see the impact of the running time.

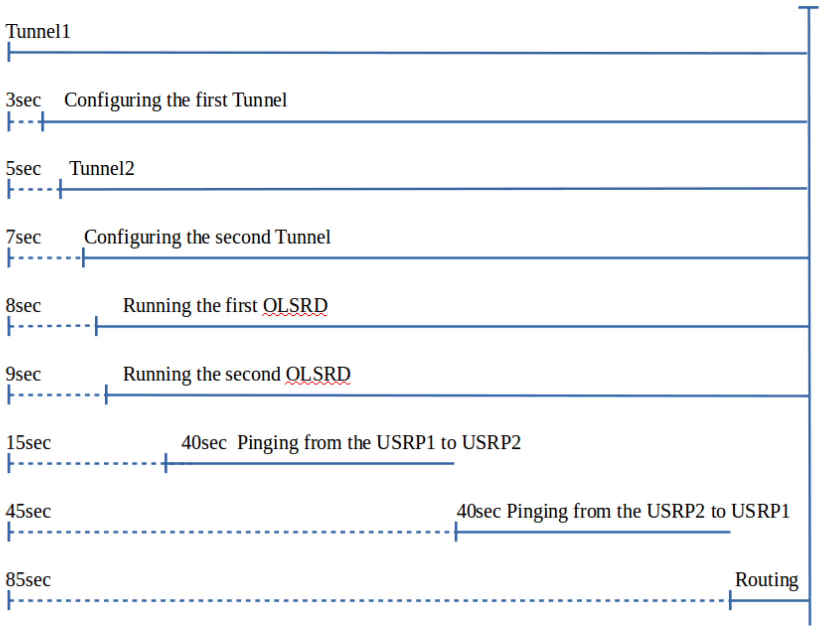

Tunnel1
3sec
Configuring the first Tunnel
5sec
Tunnel2
7sec
Configuring the second Tunnel
8sec
Running the first OLSRD
9sec
Running the second OLSRD
15sec
40sec Pinging from the USRP1 to USRP2
45sec
40sec Pinging from the USRP2 to USRP1
85sec
Routing

Figure 3.2: Commands timeline. 10 sec for a warm-up period and configuring the TAP interfaces and then 120 sec for running the commands.

3.5 Antennas

As it is mentioned in Section 2.2.1, the USRP B210 covers a wide range of frequencies from 70 MHz to 6 GHz and based on the antenna that is connected to the USRP B210 the frequency can be adjusted. In setting up the ad hoc testbed and for signal propagation three different omnidirectional vertical antennas, Vert400, Vert900 and Vert2450 are used in this thesis that each works in different frequency ranges and they all are illustrated in Figure 3.3 and the specifications of them are presented in the following [50].

- Vert400: This antenna covers Tri-band. 144 MHz, 400 MHz and 1200 MHz. The maximum power of this antenna is 10 watts.
- Vert900: This antenna is a quad-band that can be worked in 824 up to 960 MHz and also 1710 up to 1990 MHz.
- Vert2450: This is a dual-band antenna that works in 2.4 to 2.48 GHz and 4.9 up to 5.9 GHz.

The omnidirectional antenna is popular in wireless networks and radio broadcasting. This kind of antennas radiates the radio signal in all directions, not like a directional antenna that radiates the signal directly to the specific destination [51].

In this chapter, the ad hoc testbed setup has been presented. In Chapter 4, the experimental results are evaluated.

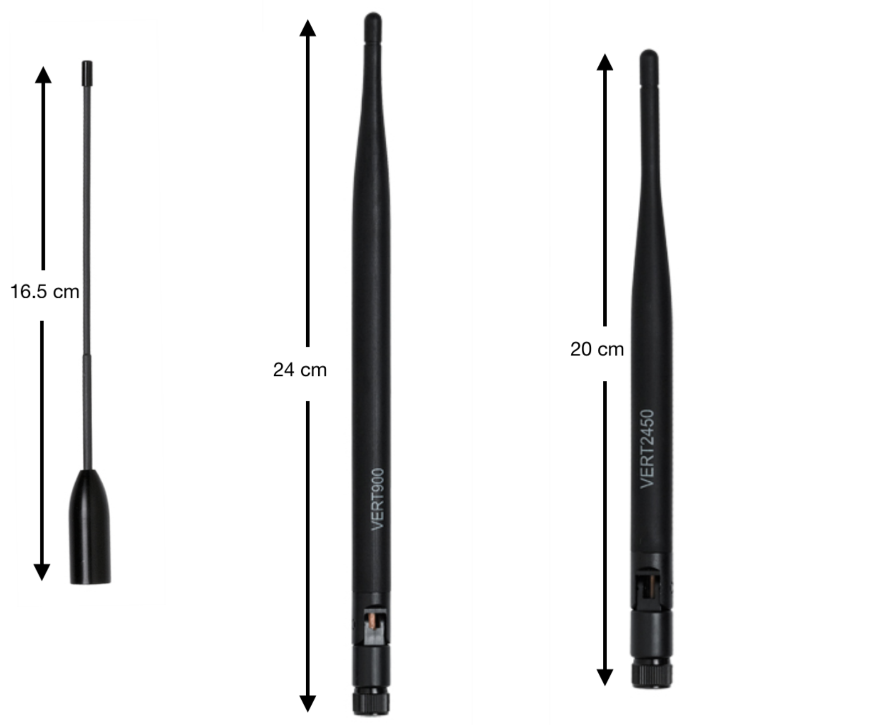


Figure 3.3: Vert400, vert900 and vert2450 antennas respectively from left to right.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, the setup that was explained in Chapter 3 is used for evaluating different parameters and see the impact of each of them. Moreover, different experiments and implementations are made and the results are presented.

4.1 Experiments

In this section, different implementations are prepared and explored and the results are compared. In the first experiments, which is the simplest one just two different USRPs are used, they both use the same vert900 antennas that is explained in Section 3.5, the frequency for sending and receiving are the same and they are on 908.0 MHz and the distance between two USRPs are 1.4 meters.

As it is mentioned in Chapter 3, for making a TCP/IP communication, the $Tunnel$ block of GNU Radio is used and the code of the $Tunnel.py$ is presented in appendix [52]. For running the $Tunnel$, the following command can be used to define some parameters whereas the TAP connection is made.

$$\begin{gathered} ./GNURadio/gr-digital/tunnel.py \quad -f908.0M \quad -r0.5M \\ --rx-gain=40 \quad --tx-gain=40 \quad -c50 \quad -p2 \end{gathered} \tag{4.1}$$

The baseband bit rate is 0.5 Mbit/sec which is represented by r in the command line 4.1 and the sample per symbol (p) is 2 which means final transmitting rate is 1 Msamples/sec. The sample rate shows how many measurements are taken in second.

The gain is 40 dB for both transmitting and receiving antennas that are defined by $tx - gain$ and $rx - gain$. The carrier sense threshold, C is 50 dB.

4.1.1 Evaluation of The Frontends of USRP B210

In this experiment, the throughput is at its highest point as it is illustrated in Figure 4.1. When the same frequency is used and also the antennas are on the same side of the USRPs.

After a packet is received, there is a two-level of controlling:

- First, the header and the packet is controlled by the $CRC32$ algorithms that was explained in Section 3.3 and if it is accepted there is an output line.

- Then the length of the payload is checked and if it is correct as well, then there would be an "$okay$" message in the output line and it can be considered as a correct received payload. Otherwise, the received payload is not considered since there is an error in the received packet or in the length of the payload.

The $MAC\ address$ of the sender is not going to be controlled at this level. Therefore using the same frequency for sending and receiving could make a forgery since we are not sure who is the sender of the packets which is received and there is a chance we receive from the same device that is sending the packets. Consequently some "$packet\ IDs$" could be received more than once.

However, in the routing table, the other nodes are shown as the next hop, the number of packets received by the next hop is higher than the number of sending packets and it shows that there are a duplication or the packets are received from other sources as well. We have no packet loss and the throughput is at its highest point since the packets are received from the same device that transmits the packet and also from the other device.

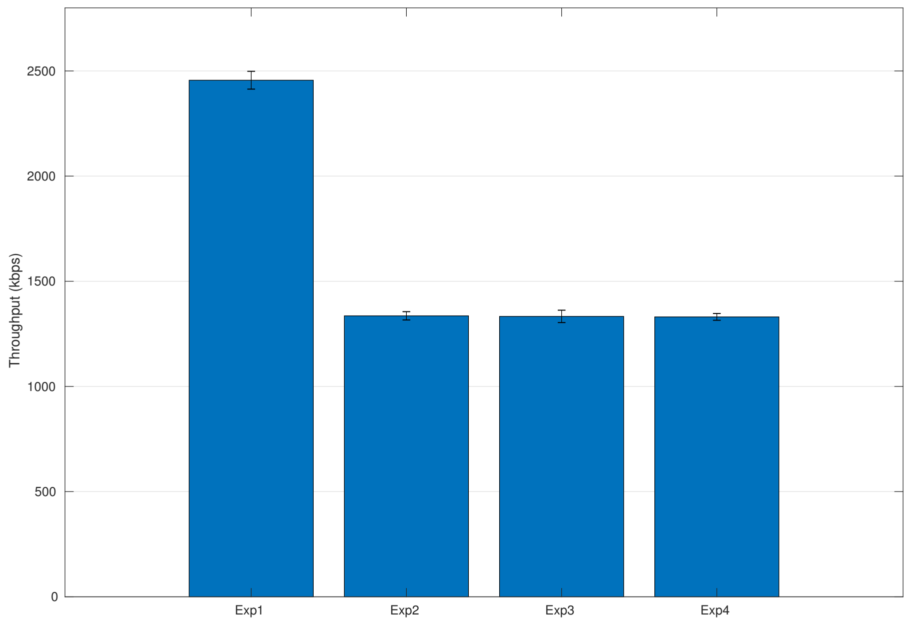
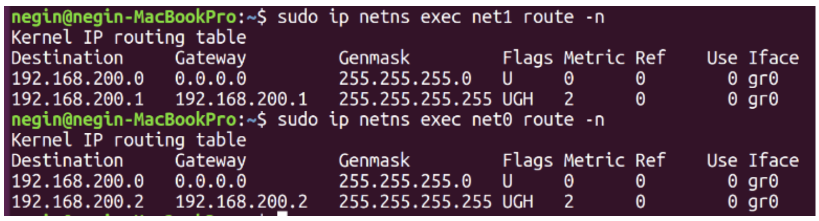


Figure 4.1: 4 Different experiments. Exp1: the same frequency for transceiving, Exp2: different frequency for transceiving, Exp3: using both sides of the USRP, Exp4: using different frequency ranges for transceiving while using different sides. The distance between the two devices is 1.4 m. The running time is 120 sec. The carrier sense threshold is 50 dB and the transmitting rate is 0.5 Mbit/sec. The packet size is 1500 bytes and the transmitting and receiving gains are 40 dB fixed for all the four experiments.

4.1.2 Impact of Frequency Division Duplexity

To overcome this problem in the second experiment, different frequencies are used for sending and receiving while they are in the same range. It can be seen in Figure 4.1, the throughput is much lower in this case while we can be sure about the sender of the packets. For the USRP1 the transmitting frequency is 908.0 MHz and the receiving one is 870.0 MHz and for the USRP2 the sending and receiving frequency is vice-versa. In the routing protocols (Figure 4.2), it can be seen that we consider the next USRP B210 as the next hop.



```
negin@negin-MacBookPro:~$ sudo ip netns exec net1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.200.0   0.0.0.0         255.255.255.0   U     0      0        0 gr0
192.168.200.1   192.168.200.1   255.255.255.255 UGH   2      0        0 gr0
negin@negin-MacBookPro:~$ sudo ip netns exec net0 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.200.0   0.0.0.0         255.255.255.0   U     0      0        0 gr0
192.168.200.2   192.168.200.2   255.255.255.255 UGH   2      0        0 gr0
```

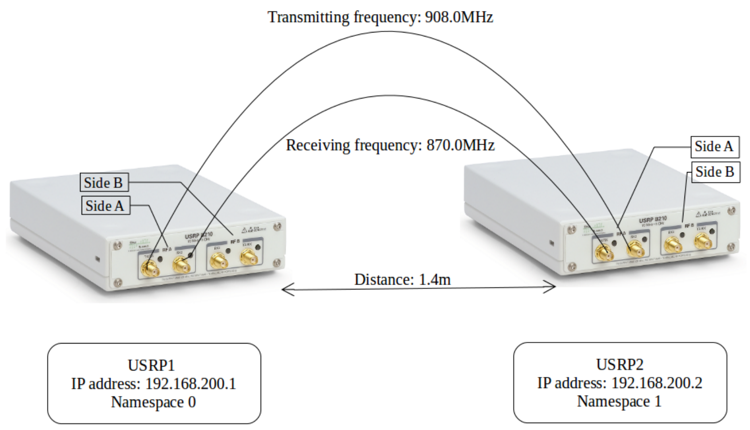

Transmitting frequency: 908.0MHz
Receiving frequency: 870.0MHz
Side B
Side A
Side A
Side B
Distance: 1.4m
USRP1
IP address: 192.168.200.1
Namespace 0
USRP2
IP address: 192.168.200.2
Namespace 1

Figure 4.2: The routing table for the nodes after running the OLSRD protocol. The setup picture of two USRPs and their configurations.

4.1.3 Impact of USRP B210 Transceiver Modules

In the third experiment, I try to use different sides of the USRP, while transmitting by side "A" and receiving the packets by side "B". As it is mentioned in Section 2.2, each USRP B210 has two transmitters and two receivers, one in side "A" and the other on side "B". As it is expected there is no differences in the results since the two sides of the USRP are working as two different devices and completely separated. It is like that we have two devices that one receives while the other one transfers the data.

4.1.4 Impact of Frequency Division While Using Different Transceivers of USRP B210

In the fourth and final experiment, since we do not use the same side of the USRPs we have the ability to use completely different kind of antennas and the frequency can be in different ranges. For USRP1, the data is transmitted with the 908.0 MHz frequency as it was in previous, while the data is received with the frequency of 420.0 MHz and for USRP2, it is conversely. Even in this situation, the results would be the same. As it is illustrated in Figure 4.1, the changes in the throughput based on different experiments can be seen.

4.2 Impact of Carrier Frequency

In this section, based on the antennas that we have and explained in Section 3.5, different frequency ranges are examined.

- The first range is 144 MHz: 146 MHz is used for transmitting and 120 MHz is used for receiving.
- The second range is 400 MHz: 446 MHz for transmitting and 420 MHz on the receiving part.
- In the third range 900 MHz is being used: 908.0 MHz for sending the packets and 870 MHz for receiving.
- The final range is 2.4GHz: In this range, 2.4GHz is being used for transmitting the packets and for receiving 2.48GHz is being used.

As it is demonstrated in Figure 4.3, the interferences in a different range of frequencies vary. As it is expected in the lower frequency, the results are worse since the amount of diffraction would be higher [44] and also in some frequency ranges there are more interferences because of other devices. For instance, in the final range which was 2.4GHz, the electromagnetic interferences are high since it is a Wi-Fi communication range and as it is illustrated in Figure 4.3 the throughput in this range would be in the lowest level. Therefore, higher frequency ranges with the maximum throughput are used for the other implementations.

As the results of these experiments suggest, using one side of the USRPs and also using the 900 MHz range of frequency which means 908.0 MHz for sending and 870.0 MHz for receiving on one USRP, and reversely on the other one would be the most suitable to continue from this part.

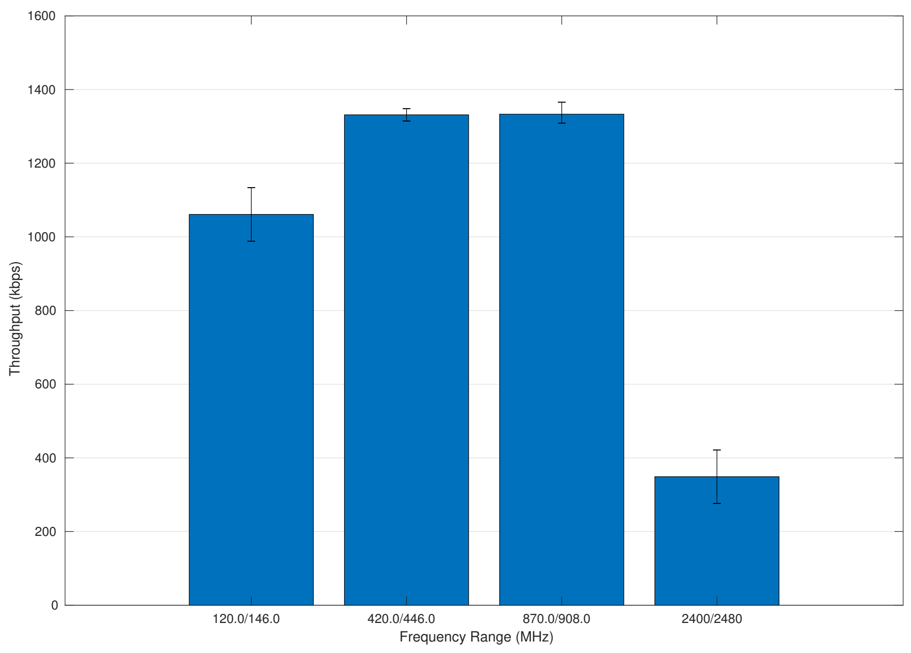


Figure 4.3: The impact of different frequency ranges on throughput while the distance is still 1.4 m and the running time is 120 sec. The threshold is 50 dB. The packet size is 1500 bytes whereas the transmitting rate is 0.5 Mb/sec. The transmitting and receiving gains are 40 dB and the transmitting and receiving antennas are on the different side of USRPs.

4.3 Impact of Transmitting Rate

In the tunnelling and transmitting signals, there are a variety of parameters that could be considered such as frequency, baseband bit rate, carrier threshold, packet size, Transmit (Tx) / Receive (Rx) antennas gain and at the end the number of nodes.

Different frequency ranges have been considered in Section 4.2, then as it is mentioned before the transmitting rate was 0.5 Mbit per second for the previous runs but in here a test is conducted with different bitrate to find the best transmitting rate.

The other setups which were considered are distance, time, carrier threshold, packet size, transmitting and receiving gain are fixed in here and there are respectively, 1.4 m, 130 sec (which is 10 sec for a warm-up and 120 sec for running the Tunnels and

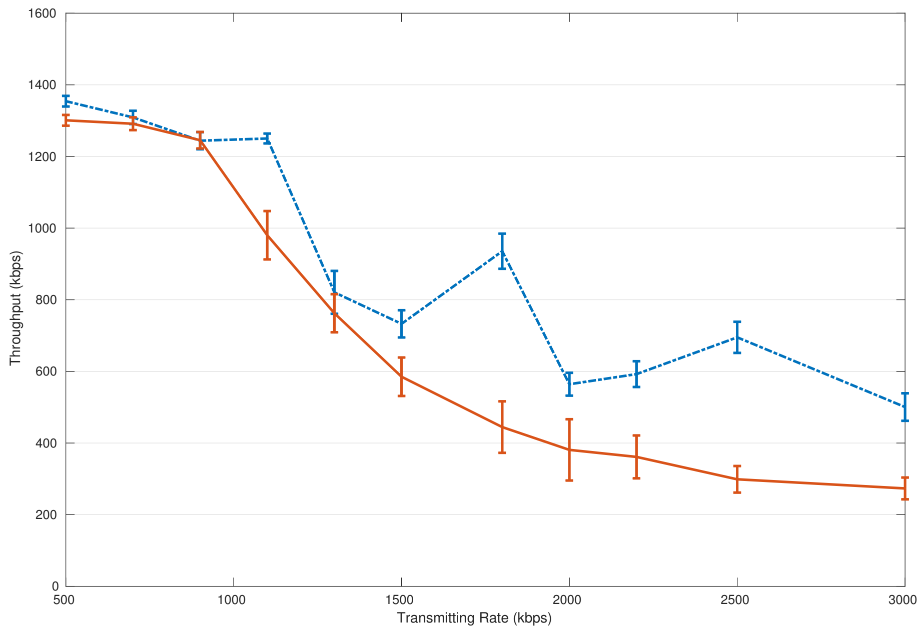


Figure 4.4: The impact of transmitting bitrate on throughput. The distances between the two nodes are 1.4 m and the running time is 120 sec. The carrier sense threshold is 50 dB. The transmitting and receiving frequencies are 908.0 MHz and 870.0 MHz respectively and the gains are 40 dB for both transmitting and receiving and the antennas are on the "A" side of the devices.

other commands), 50 dB, 1500 packets, 40 dB.

As can be seen from Figure 4.4, the best rate is 0.5 Mbit/sec. Based on the transmitting rate (R) and (4.2), the symbol rate (D) can be calculated. The modulation (M) is GMSK as it was mentioned in Section 3.2 and the sample per symbol (S) is 2. Therefore, the bandwidth could be calculated by (4.3) where α is a roll of factor and in general, is equal to 0.35. Base on the $Shannon–Hartley$ theorem, the channel capacity could be measured through (4.4).

$$D = \frac{R}{M \times S} \tag{4.2}$$

$$B = (1 + \alpha) \times D \tag{4.3}$$

$$C = B \log_2 (1 + \frac{S}{N}) \tag{4.4}$$

In (4.4), the capacity of the channel is C, while B represents the bandwidth in Hz, S is the received signal power, N is the power of the noise and in general noise over the bandwidth and S/N is considered as the signal to noise ratio.

By increasing the bitrate the bandwidth would be increased as well based on the equations that are presented and in theory, the noise ratio would be fixed. However, by increasing the bitrate, we compact more data in the transmitted symbol while the minimum SNR is fixed. As a consequence, on the receiving side, the symbols could not be decoded correctly and the packet will be dropped. Accordingly, by increasing the bitrate, the number of dropped packets would go higher and also the noise ratio increases as well and that would cause the throughput to get lower and lower as it is illustrated in Figure 4.4.

For this particular experiment, the tunnels were run for both 2 minutes and 5 minutes. As can be seen by increasing the running time the fluctuations that we faced in 2 min run would be diminished despite it has no fundamental impact on the whole trend and its highest point of throughput.

4.4 Impact of Carrier Sense Threshold

To improve throughput, several mechanisms existed in Carrier Sense Multiple Access (CSMA) wireless networks such as transmit rate (the impact of that are considered in section 4.3), transmit power which will be presented in Section 4.6, control and carrier sensing that is going to be evaluated in this section.

Carrier sense is a mechanism for avoiding packet collision. In more specific detail, if the channel detects energy more than a certain threshold or in another case if a valid signal is detected, the channel would be considered as busy [53].

The carrier threshold has not a major influence on the throughput since the transmit rate is adjusted based on this threshold. Therefore extending the carrier threshold while the bitrate is stable, the throughput would be the same as it is illustrated in

Figure 4.5. Some fluctuations could happen because of the interferences and noise of the physical environment.

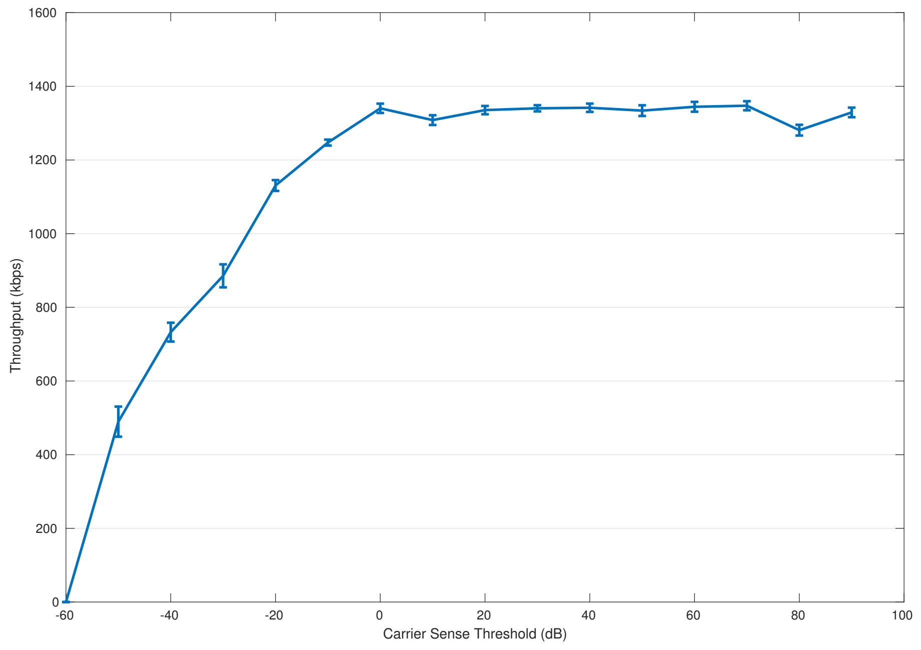


Figure 4.5: The impact of carrier sense threshold on throughput. The distance between the USRPs are 1.4 m and the running time is 120 sec while the transmit rate is 0.5 Mbps. Frequencies for sending and receiving are 908.0 MHz and 870.0 MHz and the gains are 40 dB while the antennas are on the same side while sending and receiving the signals.

4.5 Impact of Packet Size

The next parameter that is considered here is the packet size. As the packet sizes increased the throughput get extensively higher. While transmitting smaller packet sizes, the whole transmitting process takes more time. The process of encoding, decoding, handshaking between different nodes. And it causes, the less number of packets transmitted in the fixed amount of time and the control over the channel is less. However, when the number of packets increased the transmitting process is less and the channel would become more stable that could cause a better throughput as can be seen in Figure 4.6.

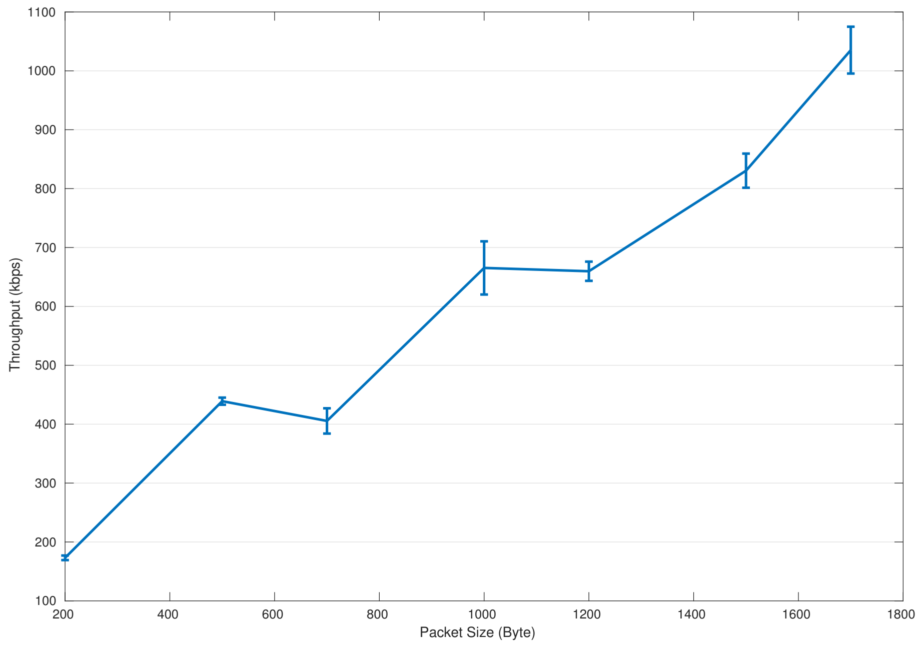


Figure 4.6: Increasing the packet size influence directly the throughput. The other parameters as distance, time, transmitting rate, carrier threshold, Tx/Rx frequency, and Tx/Rx gain are all fixed to 1.4 m, 120 sec, 0.5 Mbps, 70 dB, 908.0/870.0 MHz and 40 dB accordingly.

It should be mentioned that if the transmitted packets are larger, the header overhead would be decreased. However, it may affect the bit error and packet loss too [54]. Although the throughput increased, the packet delivery ratio would be decreased as it is shown in Figure 4.7. While the packet sizes are 200 bytes, in 2 minutes run, almost 196 packets were transmitted. As it is mentioned by increasing the packet sizes to 1700 bytes the number of packets that could be transmitted in the same amount of time would be raised to 200. It means 4 more packets, just in 2 minutes run. In spite of increasing the transmitted packets, the number of packets which are received correctly when the packet sizes are 200 bytes are 99, more than half of the packets that are transmitted while in 1700 bytes just 79 of the packets would be received. It means 20 packets less. However, since each packet that is received correctly carriers more data, it would cause the throughput raise while on the other hand, the delivery

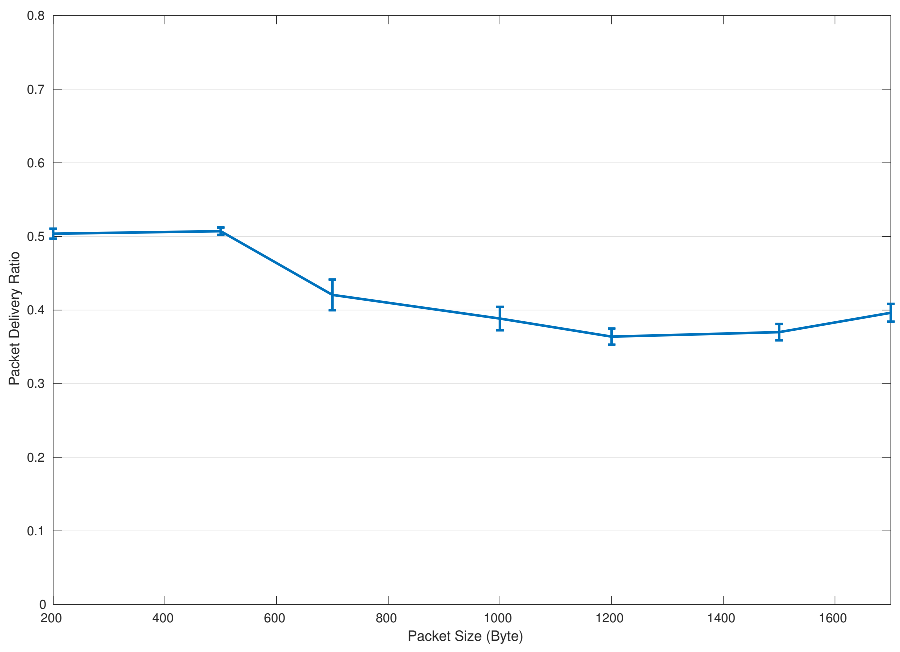


Figure 4.7: The impact of packet size on packet delivery ratio.

ratio reduces.

4.6 Impact of Transmitting and Receiving Power

Transmit power effects on the carrier sense threshold in detecting the busy channel. Therefore to maximize the capacity of the networks all the three transmit rate, transmit power and carrier sense threshold should be considered [53].

In radio propagation, there is a variety of approximations to measure the prediction or in other word calculating the link budget. There could be a signal attenuations due to a variety of reasons, for instance, reflection, refraction, diffraction, the distance between transmitter and receiver and also the power of the transmitter and the gain of the receiver.

As it is expected if the transmitting power and receiving sensitivity are low then signal quality will be lower. There are different formula base on the environment that could measure the link budget. And as it is shown in Figure 4.8 when the Tx/Rx gain is low, attenuations happens and the throughput became zero and also when the TX/Rx gain increased too much, the throughput became zero due to interferences.

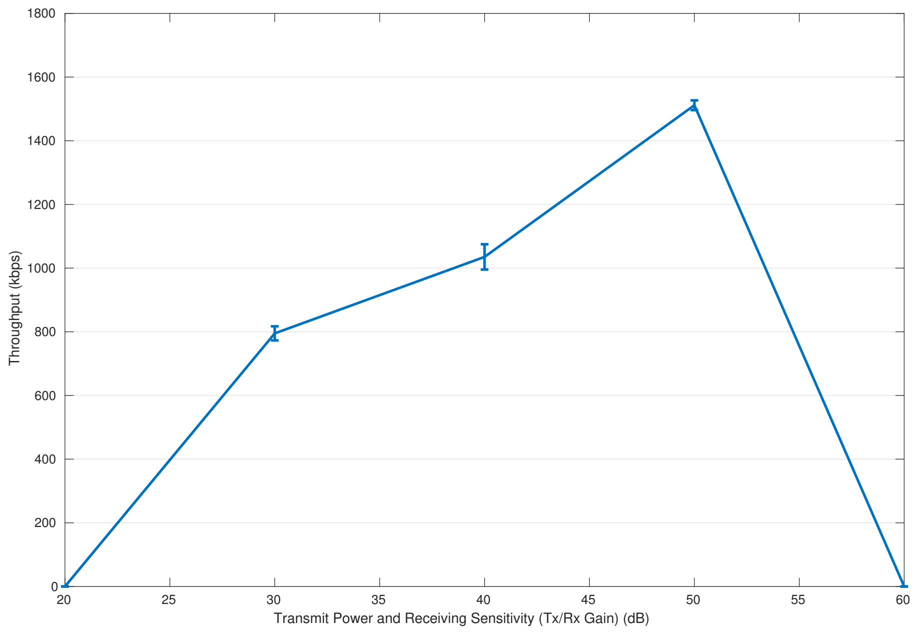


Figure 4.8: Increasing the transmitting power and receive sensitivity (Tx/Rx) affect the throughput. While the other parameters such as distance, time, rate, threshold, packet size, Tx/Rx frequencies are fixed to 1.4 m, 120 sec, 0.5 Mbps, 70 dB, 1700 bytes, 908.0MHz and 870.0 MHz respectively.

Since the transmitting power and receiving gain plays a major role in the number of packets that will be received and in general on the throughput, in this thesis we separate them to see the impact of each of them individually.

4.6.1 Impact of Transmit power

The gain for transmitting and receiving is separated in order to find just the impact of the transmit power in this section. While the receiving sensitivity is fixed to its midpoint which is 38dB. The transmit power can be in the range of zero to 89dB. As it is illustrated in Figure 4.9 by increasing the transmit power, the throughput would increase. However, after increasing the transmit power to its highest value which is 89dB, the reduction in the throughput occurs due to interference.

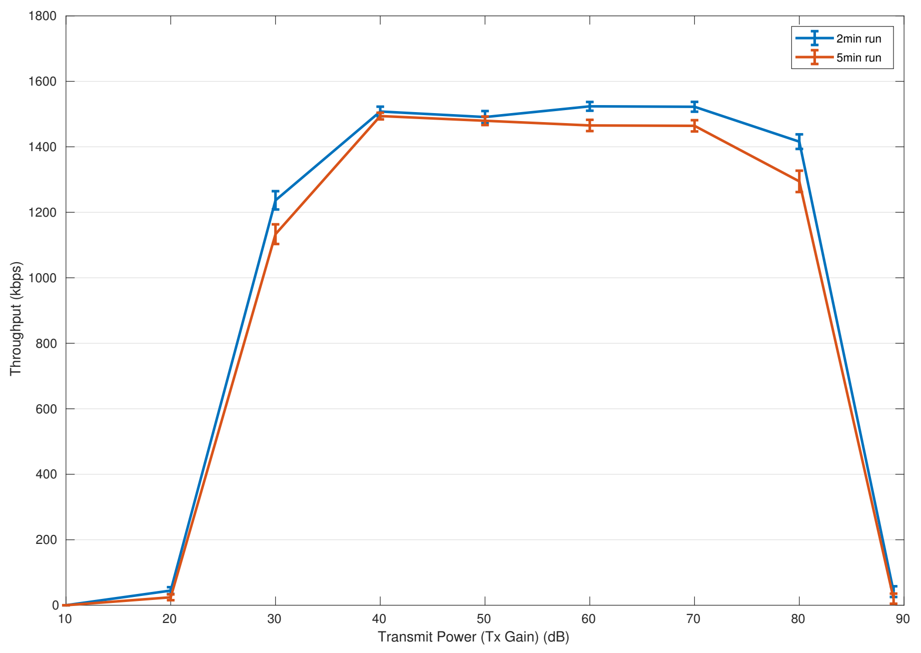


Figure 4.9: Increasing the transmitting power while receive sensitivity is fixed to 38dB affect the throughput. While the other parameters such as distance, time, rate, threshold, packet size, Tx/Rx frequencies are fixed to 1.4 m, 120 sec, 0.5 Mbps, 70 dB, 1700 bytes, 908.0MHz and 870.0 MHz respectively.

Since transmit power is an important parameter and it could affect received signal quality, this experiment is also run for 2 minutes and 5 minutes to see the impact of time on our throughput. As illustrated in Figure 4.9, there is not much contrast between them.

4.6.2 Impact of Receiving Sensitivity

In Section 4.6.1, the influence of the transmitting gain is considered and in this section, the range of receiving gain is evaluated to capture the best throughput. The transmit power is fixed on 60dB which is a safe band. By rising the receiving gain, we actually increase the sensitivity for receiving the packets. Therefore, it is expected to improve throughput. However, increasing the sensitivity excessively could cause a problem in receiving packets and as a consequence, it would make a reduction in the throughput as it is illustrated in Figure 4.10.

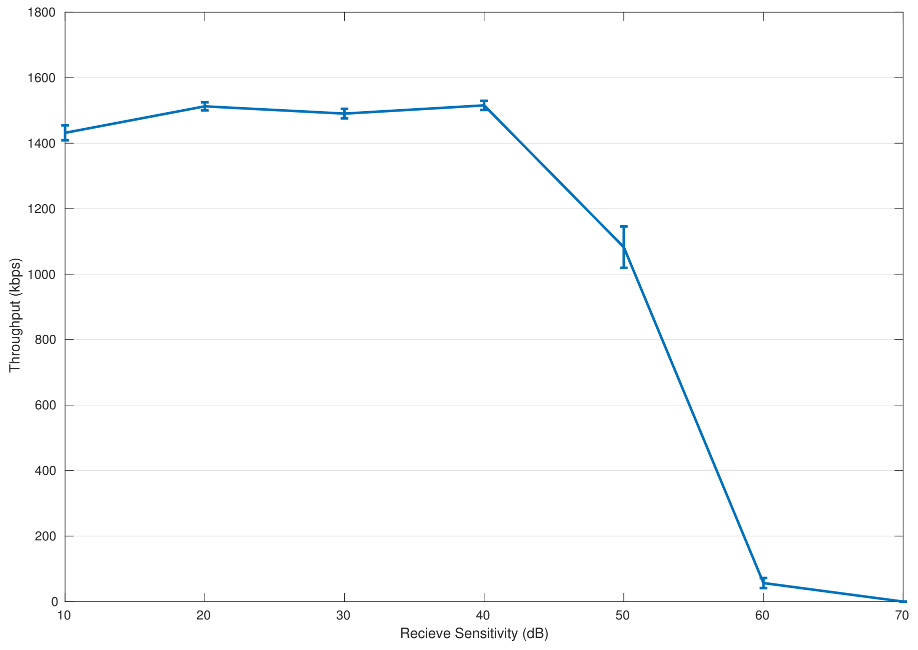


Figure 4.10: Increasing the receiving power while transmitting gain is fixed to 60dB affect the throughput. While the other parameters such as distance, time, rate, threshold, packet size, Tx/Rx frequencies are fixed to 1.4 m, 120 sec, 0.5 Mbps, 70 dB, 1700 bytes, 908.0MHz and 870.0 MHz respectively.

The distance between two USRPs could affect the results, however, the impact of it in a small scale is not that much and preparing a largely isolated environment without any other interferences and intervals is not possible in this level.

4.7 Ad-hoc Testbed Implementation

As it's mentioned before, there are several parameters in the tunnelling and till this level, we tried to optimize the throughput by adjusting these numbers and change them in their range to get the good results.

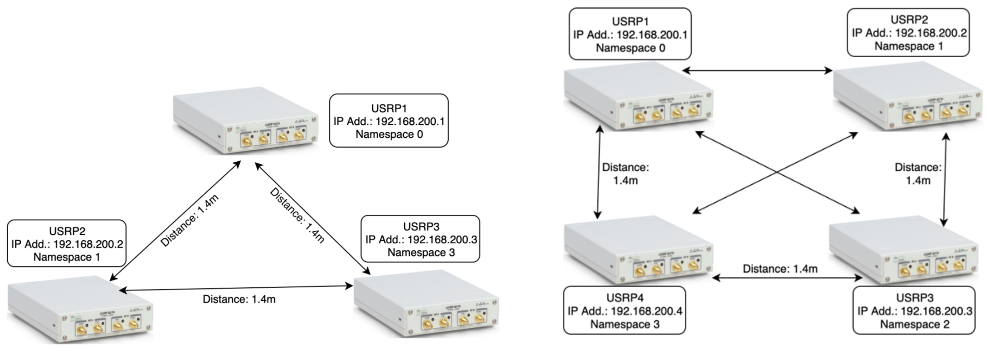
At this point, we try to fulfill our goal which was building an ad-hoc wireless network. To achieve that other nodes are added to the system to see the impact on the system, and see how adding nodes could cause the interferences and if it reduces the availability and reliability of the system.

In the first phase, another node is added to the system which means there are three nodes in the system rather than two. The distance between these nodes like the previous experiments, is 1.4 m while the nodes are located in the corners of the equilateral triangle as it is illustrated in Figure 4.11a. The running time is 300 sec to have more time for routing and produce more stable results while the warm-up period is different as the number of nodes is increased. The transmitting and receiving frequency is 908.0 MHz for all three nodes. The bitrate is 0.5 Mbit/s and the sample per symbol is 2 which means transmitting rate is 1 Mbit sample/sec. The carrier sense threshold is 70 dB which was the highest from the previous experiments. The packet size is still 1700 packets and the transmitting power is 60 dB which is a safe band and receiving gain is 20 dB. All these parameters and their values are shown in Table 4.1. Then node number four, five and six were added in our experiment. By increasing the number of nodes, we were forced to reduce the distance between the nodes. Since the nodes are connected to a single server through the cable and also our experimental environment was limited. The same distance was used for four nodes, whereas for 5 nodes the distance was 1.3 meter between two nodes and for 6 nodes the distance was reduced to 1.2 meter. The setup configuration of these experiments is presented in Figure 4.12.

Table 4.1: The parameters and their values that is used in the ad hoc testbed.

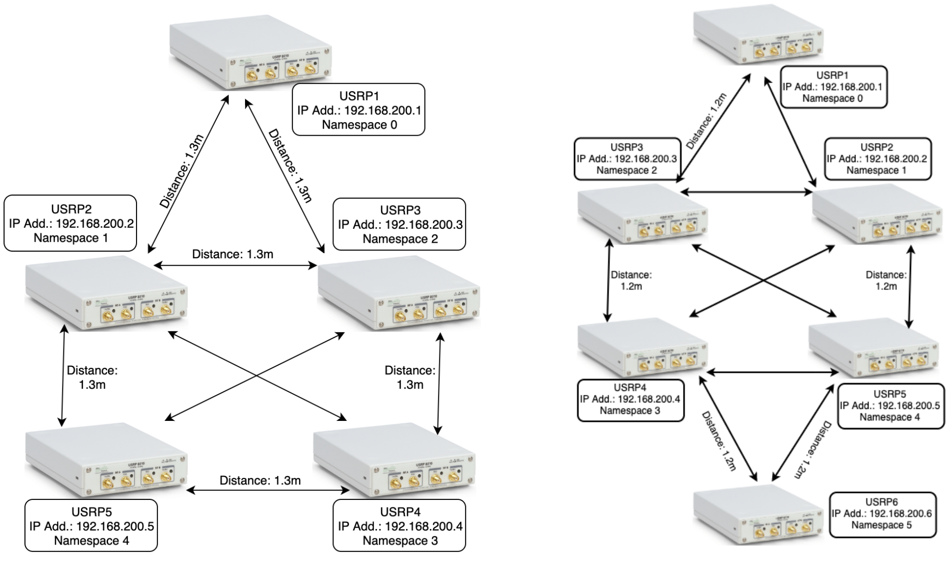
Time	Bitrate	Frequency	Threshold	Packet Size	Tx gain	Rx gain
300 sec	0.5 Mbps	908.0MHHz	70 dB	1700	60 dB	20 dB

The *Iperf* command is used to measure the bandwidth and also see the quality of our network while node number one is considered as a server and the other nodes were clients. The bandwidth is 1.05 Mbits/sec, though by increasing the nodes, we would face some latency variation due to being unsynced with the server node for a brief period, in our case, it was about 13.321 to 13.325 ms.



(a) (b)

Figure 4.11: The ad hoc testbed setup and configuration with 3 nodes (a) and 4 nodes (b).



(a) (b)

Figure 4.12: The ad hoc testbed setup and configuration with 5 nodes (a) and 6 nodes (b).

4.7.1 Impact of Routing Protocol

As can be seen from Figure 4.13 when another node comes to the system the amount of transmitted data between devices is raised and the throughput increase, however, the interferences get higher as well. The number of packets which are received is much higher since there are coming from different resources and nodes that is the reason for the rise in the throughput. As it is illustrated in Figure 4.13, two different routing protocol has been used. While the OLSR is a link-state routing protocol as it was discussed in Section 2.5, the Babel routing protocol is a distance vector routing protocol as it was presented in section 2.6. OLSR, Better Approach To Mobile Ad hoc Network (B.A.T.M.A.N) and Babel are three open-source proactive routing protocols for real-world mesh testbed and there are other researchers on the performance of these routing protocols [43]. While B.A.T.M.A.N, as it is mentioned in its name is for mobile ad hoc networks and since in our experiments mobility is not considered, this routing protocol is not investigated here.

Every four seconds a Babel node broadcasts $hello\ messages$ to all of its neighbors and from the neighbors which had been recently received a $hello\ message$, periodically sends a IHU (I Heard You) message. The throughput of Babel is much lower since the hello message is sent every 4 sec while in the OLSR the message is sent every 2 sec. That is the reason for the difference in the throughput and moreover, due to different metrics, each uses for finding the route. However, in finding the route, Babel finds the route in all the experiments while OLSR routing protocol faces a problem in finding all the routes if the number of nodes is higher than four and since it transmits more packets the overhead gets higher. As a result, Babel outperforms the OLSR routing protocol in finding the route while it transmits fewer packets.

4.7.2 Impact of Topology on OLSR Routing Protocol

There is various kind of topology on wireless mesh networks. In here two different topologies, one full mesh and the other one, star/point-to-multipoint topology are implemented while having 4, 5 and 6 nodes and the results are presented in Figure 4.14 while the other parameters are fixed and as it is displayed in Table 4.1.

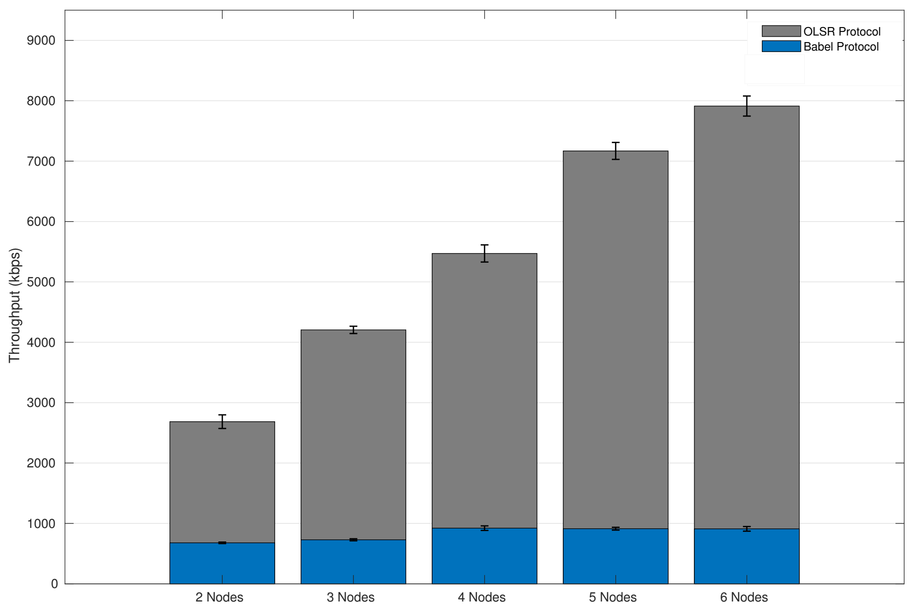


Figure 4.13: Making an ad-hoc with two to six different nodes while running OLSR and Babel routing protocol. The distances between devices are 1.4 m for 2 to 4 nodes and the distance between 5 nodes are 1.3 m and for 6 nodes are 1.2 m. The other parameters as running time, transmitting rate, frequency, carrier sense threshold, packet size, transmit power and receiving sensitivity are as presented in Table 4.1

In the full mesh topology, all nodes are connected while transmitting between themselves. In the star topology, just one node is connected to other nodes as a central node or server and if other nodes wanted to send a message, they have to transmit the data to that defined node at first. The routing table for the central node in a star topology and for one of the clients' node are illustrated in Table 4.2 and Table 4.3 respectively. In Table 4.4 the routing table for one node after running the OLSR in full-mesh topology is presented.

It can be seen that in the full mesh topology each node gets connected to other nodes while the gateway is the other node's IP Address. Whereas in the star topology just for node number one which was a central node the gateway is different since it is connected to all the nodes and for the other nodes the gateway always is node number one. As it is expected in the full mesh the transceiving data are more and also

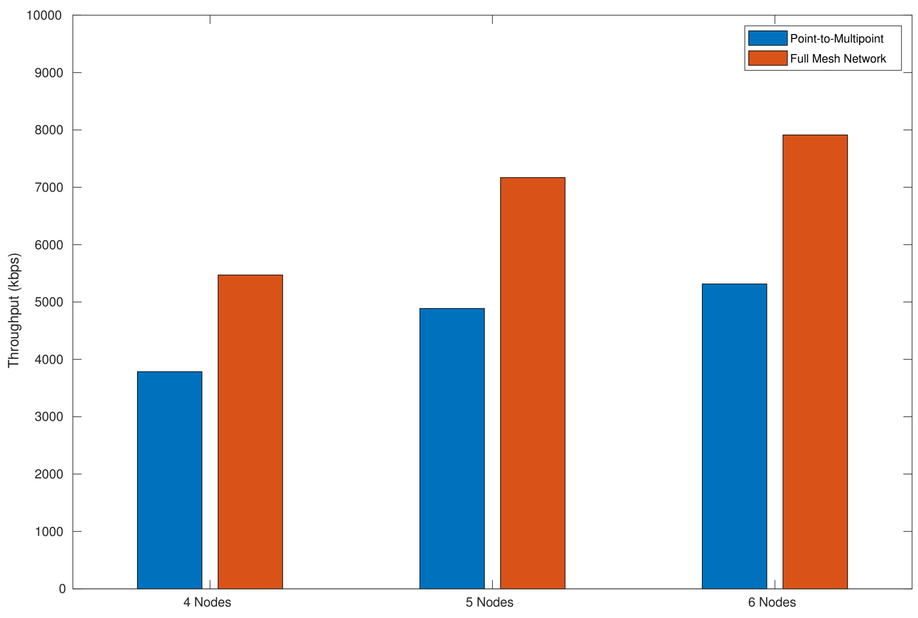


Figure 4.14: The influence of full mesh topology and star topology on OLSR routing protocol. The other parameters and their values are presented in Table 4.1

Table 4.2: Routing table for the central node with IP Add. 192.168.200.1 after running the OLSR in the star topology.

Destination	Gateway
192.168.200.0	0.0.0.0
192.168.200.2	192.168.200.2
192.168.200.3	192.168.200.3
192.168.200.4	192.168.200.4

the throughput is higher in comparison with the star topology. Besides, the routing table and the connections are more stable in full-mesh topology since all nodes are connected and even if one node fails, the traffic can be redirected to other nodes. The full mesh topology has better results in discovering the neighbor nodes as the number of nodes increased to six. However, as it was mentioned in Section 4.7.1 the OLSR routing protocol could not discover all the neighbor nodes and add them to its routing table in all the experiments when the number of nodes is more than four.

Table 4.3: Routing table for the client node with IP Add. 192.168.200.2 after running the OLSR in the star topology.

Destination	Gateway
192.168.200.0	0.0.0.0
192.168.200.1	192.168.200.1
192.168.200.3	192.168.200.1
192.168.200.4	192.168.200.1

Table 4.4: Routing table for a node with IP Add. 192.168.200.1 after running the OLSR in the full mesh topology.

Destination	Gateway
192.168.200.0	0.0.0.0
192.168.200.2	192.168.200.2
192.168.200.3	192.168.200.3
192.168.200.4	192.168.200.4

4.8 Impact of Distance and Running Time

While the nodes are connected to a single server, they are separated and they all are in different namespaces, therefore they can not make a connection internally through the server and they have to just use the wireless connection and signal propagation to be connected to each other. In our experiments, we expand the number of nodes to six however due to limitations of the experimental environment, the cable which devices used to get connected to the server and also the number of USRPs, this research could not be used in larger-scale like a real-life experiment at this level.

By increasing the number of nodes to six, we were forced to decrease the distance to 1.2 m instead of 1.4 m which was used in other experiments. However, we did not increase the nodes more than six since the distance has an impact on the interferences and on the results apart from the influence of adding more nodes. An isolated environment without other interferences could not be implemented, hence our experiment is limited to this level and further research needs more advanced equipment and experimental environment.

The running time was 2 min for evaluating most of the parameters, though as it is mentioned in related sections, for transmitting rate and transmit power the codes were run for both 2 min and 5 min to see the impact of the running time. Consequently, the whole upward and downward trend were the same. Although by extending the running time, the fluctuations have vanished and more stable results were captured. For different routing protocol and different topology, we consider that and the codes were run for 5 minutes rather than 2 minutes to get more robust connections and stable results.

CHAPTER 5

CONCLUSION AND FUTURE WORK

There is a rapid transition to the wireless network and the number of devices that are connected to a single network grows dramatically. Whereas the amount of data that are produced by these users are increasing. Therefore there is an interest in the mesh network while most of the researches in this area are done by simulators. Simulators bring real-world scale environment however there is still a significant gap between real wireless networks and simulations. Since analytical environments are presented by the simulations though by implementing a testbed we tried to create a practical environment.

It is important to prepare a testbed to operate on actual real hardware. Delays due to encoding, decoding, modulation, demodulation, transmission, and reception become more important with real hardware devices and are not usually discovered in the designing phase or in simulations. In addition, signalling issues like timing and synchronization with waveforms do not become apparent without the use of real hardware. The influence of unknown noises and interferences in the environment can not be considered in simulations.

In this thesis, an indoor SDR ad hoc testbed is implemented. The SDR system is used since it is a promising technology for implementing various functions of radio communication in software rather than hardware. SDR brings reconfigurability, adaptability, and flexibility to the system. It can be used in noisy environments like disaster sites or in the military due to the fact that it can be reconfigured easily without the needs for redesigning the hardware and most of the configurations are done by software.

In our implementation, different open-source software and applications are used and the information of them are presented in this thesis. GNU Radio is used for making a TCP/IP connections between nodes whereas the OLSR protocol and also Babel is used for making routing protocols. These two routing protocol is used since they are open-source routing protocols and they use different methods for creating the routing table. By running these two routing protocols we also show that any other open-source routing protocol can be run on this platform for future experiments. Besides different parameters are evaluated to find the best configurations.

Although our ad hoc testbed has just 6 different nodes, there is an ability to expand this work to a larger scale. The USRPs are connected to a single server, therefore there is a chance to add more nodes to the server. Moreover, all the software are multi-platforms and another system can be used for enlarging our testbed environment. In our experiment, we can consider the server and the USRPs which are connected to as a container. For future studies, the number of containers can be expanded while each container could be a server with some USRPs. In this way, the system's kernel is shared between while different applications can be run on the server and like real-world testbed environment can be implemented by increasing number of containers.

In our experiments, all the nodes were fixed and the mobility is not studied at this level due to physical limitations. In the future, mobility can be analyzed as well. In a scenario, a container could be a laptop with a USRP or a robot with USRP, then the mobility issues such as routing issues, the connectivity, the latency can be considered. USRP B210 was designed for low-cost experimentations, however, each of the devices cost around 1.259$ which is expensive for academics research. The throughput of devices is low compared to other wireless devices and it has high latency.

As an outcome, an SDR ad hoc testbed is provided that can be used for more advanced and future experimental researches in ad hoc networks. The implemented testbed is not limited or specified for just this research, therefore the testbed can be reproduced and reused easily for instance in a larger scale in order to provide like a real-world environment. For future works, other applications can be run on top of the implemented testbed. There is a chance to optimize the routing table with better mesh routing protocols. In addition in higher-level, the MAC address of the sender can be recognized for preventing packet receives from different sources. More sophisticated USRPs can be used for better throughput and signal detections.

REFERENCES

[1] N. B. Truong, Y. J. Suh, and C. Yu, “Latency analysis in GNU radio/USRP-based software radio platforms,” *Proceedings - IEEE Military Communications Conference MILCOM*, pp. 305–310, 2013.

[2] GNURadio, “FM reciever.” `https://github.com/csete/gnuradio-grc-examples/blob/master/receiver/fm_rx.grc`. [Online; 2019].

[3] Cisco, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011–2016 [Visual Networking Index (VNI)],” *Cisco*, pp. 2016–2021, 2012.

[4] A. Zahmatkesh and T. Kunz, “Software defined multihop wireless networks: Promises and challenges,” *Journal of Communications and Networks*, vol. 19, no. 6, pp. 546–554, 2017.

[5] P. Sakarindr and N. Ansari, “Security services in group communications over wireless infrastructure, mobile ad hoc, and wireless sensor networks,” *IEEE Wireless Communications*, vol. 14, pp. 8–20, October 2007.

[6] G. Lin, G. Noubir, and R. Rajaraman, “Mobility models for ad hoc network simulation,” in *Proceedings - IEEE INFOCOM*, vol. 10, pp. 454–463, 2004.

[7] Z. Navidi, N. Ronald, and S. Winter, “Comparison between ad-hoc demand responsive and conventional transit: a simulation study,” *Public Transport*, vol. 10, pp. 147–167, May 2018.

[8] I. Dorathy and M. Chandrasekaran, “Simulation tools for mobile ad hoc networks: a survey,” *Journal of applied research and technology*, vol. 16, no. 5, pp. 437–445, 2018.

[9] S. Boehm and H. Koenig, “SEmulate: Seamless Network Protocol Simulation and Radio Channel Emulation for Wireless Sensor Networks,” *Annual Confer-*

ence on Wireless On-demand Network Systems and Services, vol. 7, pp. 111–118, 2019.

[10] M. Dillinger, K. Madani, and N. Alonistioti, *"Software Defined Radio: Architectures, Systems and Functions"*. Willey, 2003.

[11] E. A. Thompson, N. Clem, I. Renninger, and T. Loos, “Software-defined GPS receiver on USRP-platform,” *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1352–1360, 2012.

[12] David A. Clendenen, *"A Software defined Radio Testbed for Research in Dynamic Spectrum Access"*. PhD thesis, 2012.

[13] Wikipedia, “Universal Software Radio Peripheral.” `https://en.wikipedia.org/wiki/Universal_Software_Radio_Peripheral`. [Online; 2019].

[14] M. Ettus, “The Universal Software Radio Peripheral or USRP,” in *Http:/www.Ettus.Com*, 2008.

[15] M. Ettus and M. Braun, *"The Universal Software Radio Peripheral (USRP) Family of Low-Cost SDRs"*, pp. 3–23. 2015.

[16] A. Omar, “Baseband and super-resolution-passband reconstructions in microwave imaging,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 67, pp. 1327–1335, April 2019.

[17] F. Hamza, “The USRP under 1.5 X Magnifying Lens!,” *https://microembedded.googlecode.com/files/USRP_Documentation.pdf*, pp. 1–90, 2008.

[18] Ettus Research B200/B210, “USRP B200/B210 Product Overview,” *http://www.ettus.com/wp-content/uploads/2019/01/b200-b210_spec_sheet.pdf.*

[19] M. Ettus, “USRP Hardware Driver (UHD) - Ettus Research.” `https://github.com/EttusResearch/uhd`. [Online; 2019].

[20] D. Nguyen, *"Implementation of OFDM systems using GNU Radio and USRP"*. PhD thesis, 2013.

[21] E. Blossom, "Exploring GNU Radio," *Ettus research*, pp. 1–11, 2004.

[22] A. Pinomaa, H. Baumgartner, J. Ahola, and A. Kosonen, "Utilization of software-defined radio in power line communication between motor and frequency converter," *IEEE ISPLC 2010 - International Symposium on Power Line Communications and its Applications*, pp. 172–177, 2010.

[23] "GNU Radio." `https://www.gnuradio.org/about/`. [Online; 2019].

[24] Z. M. Laureando and S. K. Albert, *"Development of an Android App To Control and Manage Cognitive Networks"*. PhD thesis, 2014.

[25] OLSR.org, "OLSR.org Network Framework." `http://www.olsr.org/mediawiki/index.php/OLSR.org_Network_Framework`. [Online; May 2016].

[26] F. Haq and T. Kunz, "Simulation vs . Emulation : Evaluating Mobile Ad Hoc Network Routing Protocols," *Network*, 2005.

[27] C. Barz, C. Fuchs, J. Kirchhoff, J. Niewiejska, and H. Rogge, "OLSRv2 for Community Networks: Using Directional Airtime Metric with external radios," *Computer Networks*, vol. 93, pp. 324–341, 2015.

[28] R. Jain and K. Indu, "An QoS Aware Link Defned OLSR (LD-OLSR) Routing Protocol.pdf," *International Journal of Computational Intelligence & IoT*, vol. 1, p. 6, 2018.

[29] S. Naimi, A. Busson, V. Vèque, and R. Bouallegue, "Metric anticipation to manage mobility in mobile mesh and ad hoc wireless networks," *Annals of Telecommunications*, vol. 73, no. 11, pp. 787–802, 2018.

[30] V. Veselý, V. Rek, and O. Ryšavý, "Babel Routing Protocol for OMNeT++ More than just a new simulation module for INET framework," *CoRR*, vol. abs/1609.05215, 2016.

[31] S. COCORAD, "Connecting Omnetpp to virtual Ethernet Interfaces," *ICDCC13 International Conference on Computers, Digital Communications and Computing*, pp. 30–35, 2013.

[32] I. Bekmezci, I. Sen, and E. Erkalkan, “Flying ad hoc networks (fanet) test bed implementation,” in *2015 7th International Conference on Recent Advances in Space Technologies (RAST)*, pp. 665–668, June 2015.

[33] M. Fazeli and H. Vaziri, “Assessment of throughput performance under opnet modeler simulation tools in mobile ad hoc networks (manets),” in *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 328–331, July 2011.

[34] S. Krug, A. Brychcy, and J. Seitz, “A mobile embedded-linux-based testbed for outdoor ad hoc network evaluation,” in *2016 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, pp. 164–166, Sep. 2016.

[35] T. Brown, S. Doshi, S. Jadhav, D. Henkel, and R. Thekkekunnel, “A full scale wireless ad hoc network test bed,” *Access*, pp. 51–60, July 2014.

[36] S. S. Kumar, M. N. Kumar, and S. V.S., “Implementation of hybrid ad-hoc routing protocol,” in *2010 International Conference on Advances in Recent Technologies in Communication and Computing*, pp. 151–154, Oct 2010.

[37] O. Ozdemir, R. Hamila, and N. Al-Dhahir, “A usrp-based experimental testbed for ofdm systems impaired by i/q imbalance,” in *2013 7th IEEE GCC Conference and Exhibition (GCC)*, pp. 98–102, Nov 2013.

[38] W. O. Oduola, N. Okafor, O. Omotere, L. Qian, and D. Kataria, “Experimental study of hierarchical software defined radio controlled wireless sensor network,” in *2015 36th IEEE Sarnoff Symposium*, pp. 18–23, Sep. 2015.

[39] S. Desilva and S. R. Das, “Experimental evaluation of a wireless ad hoc network,” in *Proceedings Ninth International Conference on Computer Communications and Networks (Cat.No.00EX440)*, pp. 528–534, Oct 2002.

[40] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G. M. Voelker, “Sora,” *Communications of the ACM*, vol. 54, no. 1, p. 99, 2011.

[41] L. Sanabria-Russo, “TCP/IP communication between two USRP-E110,” tech. rep., 2012.

[42] M. Segata, N. Facchi, L. Maccari, G. Gemmi, and R. Lo Cigno, “Centrality-based route recovery in wireless mesh networks,” in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2018.

[43] M. Abolhasan, B. Hagelstein, and J. C. P. Wang, “Real-world performance of current proactive multi-hop mesh protocols,” *2009 15th Asia-Pacific Conference on Communications, APCC 2009*, pp. 44–47, 2009.

[44] C. Siva Ram Murthy and B. S. Manoj, *"Ad Hoc Wireless Networks Architectures"*. Pearson Education, Inc., 6th ed., 2004.

[45] K. R. Dandekar, S. Begashaw, M. Jacovic, A. Lackpour, I. Rasheed, X. R. Rey, C. Sahin, S. Shaher, and G. Mainland, “Grid software defined radio network testbed for hybrid measurement and emulation,” in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, June 2019.

[46] A. Wiesler, R. Machauer, and F. Jondral, “Comparison of gmsk and linear approximated gmsk for use in software radio,” in *1998 IEEE 5th International Symposium on Spread Spectrum Techniques and Applications - Proceedings. Spread Technology to Africa (Cat. No.98TH8333)*, vol. 2, pp. 557–560 vol.2, Sep. 1998.

[47] S. H. Goode, “A comparison of gaussian minimum shift keying to frequency shift keying for land mobile radio,” in *34th IEEE Vehicular Technology Conference*, vol. 34, pp. 136–141, May 1984.

[48] S. Gueron, “Speeding up CRC32C computations with Intel CRC32 instruction,” *Information Processing Letters*, vol. 112, no. 5, pp. 179–185, 2012.

[49] T. Schmidt, “CRC Generating and Checking,” *MICROCHIP (AN730) Data Sheet*, p. 24, 2000.

[50] Ettus, “Vert400, Vert900, Vert2450 antennas.” `https://www.ettus.com/product-categories/antennas/`. [Online; 2019].

[51] X. Qing and Z. N. Chen, *"Handbook of Antenna Technologies"*. Springer-Refrence, 2014.

[52] "Tunnel.py." `https://github.com/gnychis/gnuradio-3.5.0-dmr/blob/master/gr-digital/examples/narrowband/tunnel.py`. [Online; 2011].

[53] T. Huehn, R. Merz, and C. Sengul, "Joint transmission rate, power, and carrier sense settings: An initial measurement study," *2010 5th IEEE Workshop on Wireless Mesh Networks, WiMesh 2010*, pp. 49–54, 2010.

[54] J. Korhonen and Ye Wang, "Effect of packet size on loss rate and delay in wireless links," *IEEE Wireless Communications and Networking Conference, WCNC*, vol. 00, pp. 1608–1613, 2005.

APPENDIX A

TUNNEL CODE

This code sets up up a virtual ethernet interface (typically gr0), and relays packets between the interface and the GNU Radio PHY+MAC. What this means in plain language, is that if you've got a couple of USRPs on different machines, and if you run this code on those machines, you can talk between them using normal TCP/IP networking.

```
//////////////////////////////////////////////

from gnuradio import gr, digital, blocks, eng_notation
from gnuradio.eng_option import eng_option
from optparse import OptionParser

from current dir
from receive_path  import receive_path
from transmit_path import transmit_path
from uhd_interface import uhd_transmitter
from uhd_interface import uhd_receiver
```

```
import os, sys
import random, time, struct

////////////////////////////////////////////////
  Use the Universal TUN/TAP device driver to move packets
  to/from kernel
////////////////////////////////////////////////

Linux specific...
TUNSETIFF ifr flags from <linux/tun_if.h>

IFF_TUN = 0x0001  #tunnel IP packets
IFF_TAP = 0x0002  #tunnel Ethernet frames
IFF_NO_PI = 0x1000  #don't pass extra packet info
IFF_ONE_QUEUE = 0x2000 #beats me ;)

global n_rcvd, n_right, n_sent, S_payload, R_payload

def open_tun_interface(tun_device_filename):
    from fcntl import ioctl

    mode = IFF_TAP | IFF_NO_PI
    TUNSETIFF = 0x400454ca

    tun = os.open(tun_device_filename, os.O_RDWR)
    ifs = ioctl(tun, TUNSETIFF, struct.pack
                    ("16sH", "gr%d", mode))
    ifname = ifs[:16].strip("\x00")
    return (tun, ifname)

////////////////////////////////////////////////
    The flow graph
////////////////////////////////////////////////
```

```
class my_top_block(gr.top_block):

    def __init__(self, mod_class, demod_class,
                 rx_callback, options):

        gr.top_block.__init__(self)

        # Get the modulation’s bits_per_symbol
        args=mod_class.extract_kwargs_
                        from_options(options)
        symbol_rate = options.bitrate /
                mod_class(**args).bits_per_symbol()

        self.source = uhd_receiver(
                options.args, symbol_rate,
                options.samples_per_symbol,
                options.rx_freq, options.rx_gain,
                options.spec, options.antenna,
                options.verbose)

        self.sink = uhd_transmitter(
                options.args, symbol_rate,
                options.samples_per_symbol,
                options.tx_freq, options.tx_gain,
                options.spec, options.antenna,
                options.verbose)

        options.samples_per_symbol = self.source._sps

        self.txpath =transmit_path(mod_class, options)
        self.rxpath =receive_path(demod_class,
                            rx_callback, options)
        self.connect(self.txpath, self.sink)
        self.connect(self.source, self.rxpath)

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

    def carrier_sensed(self):
        """
        Return True if the receive path thinks there is
        carrier
        """
        return self.rxpath.carrier_sensed()

    def set_freq(self, target_freq):
        """
        Set the center frequency we're interested in.
        """

        self.sink.set_freq(target_freq)
        self.source.set_freq(target_freq)

////////////////////////////////////////////////
    Carrier Sense MAC
////////////////////////////////////////////////

class cs_mac(object):
    """
    Prototype carrier sense MAC

    Reads packets from the TUN/TAP interface, and sends them
    to the PHY. Receives packets from the PHY via phy_rx_
    callback, and sends them into the TUN/TAP interface.
    Of course, we’re not restricted to getting packets
    via  TUN/TAP, this is just an example.
    """
    global n_sent, S_payload, R_payload
    def __init__(self, tun_fd, verbose=False):
        self.tun_fd = tun_fd
        self.verbose = verbose
        self.tb = None

    def set_top_block(self, tb):
        self.tb = tb

    def phy_rx_callback(self, ok, payload):

global n_rcvd, n_right, R_payload
n_rcvd += 1
        if self.verbose:
            print "Rx: ok = %r  len(payload) =
                        %4d" % (ok, len(payload))
        if ok:
            n_right += 1
    R_payload = R_payload + len(payload)
            print "ok: %r \t n_rcvd: %d \t n_right:
                        %d \t Received_payload: %d"  %
                        (ok, n_rcvd, n_right, R_payload)
            os.write(self.tun_fd, payload)

    def main_loop(self):
        """
        Main loop for MAC.
        Only returns if we get an error reading from TUN.
        """
        min_delay = 0.001               # seconds
```

```
n_sent = 0
S_payload = 0
        while 1:
            payload = os.read(self.tun_fd, 1700)
            if not payload:
                self.tb.send_pkt(eof=True)
                break

            if self.verbose:
n_sent += 1
S_payload = S_payload + len(payload)
                print "Tx: len(payload) = %4d \t n_sent: %d \t
                            Send_payload: %d"  % (len(payload),
                            n_sent, S_payload)

            delay = min_delay
            while self.tb.carrier_sensed():
print "USRP is Busy"
                sys.stderr.write('Busy')
                time.sleep(delay)
                if delay < 0.050:
                    delay = delay * 2  #exponential back-off

            self.tb.send_pkt(payload)

//////////////////////////////////////////////
        main
//////////////////////////////////////////////

def main():

    global n_rcvd, n_right, n_sent, S_payload, R_payload
    n_rcvd = 0
```

```
n_right = 0
n_sent = 0
S_payload = 0
R_payload = 0

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

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=1700, help="set packet
                size [default=%default]")
parser.add_option("-v","--verbose", action=
                "store_true", default=False)
expert_grp.add_option("-c", "--carrier-threshold",
                type="eng_float", default=70,
                help="set carrier detect threshold
                (dB) [default=%default]")
expert_grp.add_option("","--tun-device-filename",
               default="/dev/net/tun", help="path to
               tun device file [default=%default]")

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

```
uhd_transmitter.add_options(parser)


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

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

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

# open the TUN/TAP interface
(tun_fd, tun_ifname) = open_tun_interface(
                    options.tun_device_filename)

# Attempt to enable realtime scheduling
r = gr.enable_realtime_scheduling()
if r == gr.RT_OK:
    realtime = True
else:
    realtime = False
    print "Note: failed to enable realtime scheduling"

# instantiate the MAC
mac = cs_mac(tun_fd, verbose=True)

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

mac.set_top_block(tb)
# give the MAC a handle for the PHY

if tb.txpath.bitrate() != tb.rxpath.bitrate():
    print "WARNING: Transmit bitrate = %sb/sec,
       Receive bitrate = %sb/sec" % (
       eng_notation.num_to_str(tb.txpath.bitrate()),
       eng_notation.num_to_str(tb.rxpath.bitrate()))

print "modulation: %s"   % (options.modulation,)
print "freq:  %s"      % (eng_notation.num_to_str(
                        options.tx_freq))
print "bitrate:        %sb/sec" % (eng_notation.num_
                       to_str(tb.txpath.bitrate()),)
print "samples/symbol: %3d" % (tb.txpath.samples_
                        per_symbol(),)

tb.rxpath.set_carrier_threshold(
                    options.carrier_threshold)
print "Carrier sense threshold:",
                    options.carrier_threshold, "dB"

print "Allocated virtual ethernet interface:
                        %s" % (tun_ifname,)
print "You must now use ifconfig to set its
                        IP address. E.g.,"

print "  $ sudo ifconfig %s 192.168.200.1"
                        % (tun_ifname,)

print "Be sure to use a different address in the
```

```
                same subnet for each machine."



    tb.start()     # Start executing the flow graph
                    (runs in separate threads)

    mac.main_loop()    # don’t expect this to return...

    tb.stop()      # but if it does, tell flow graph to stop.
    tb.wait()      # wait for it to finish


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