Middle East Technical University
Department of Computer Engineering
Wireless Systems, Networks and Cybersecurity (WINS) Laboratory

CENG513 Wireless Communications and Networking
2018-2019 Spring
Design of the Stack

Prepared by
Ilker GURCAN - 1777929
Umay Ezgi KADAN - 2404457
Fehime Betul CAVDARLI 2339372

e177792@metu.edu.tr
e240445@metu.edu.tr
betul.cavdarli@metu.edu.tr
Computer Engineering
30 May 2019

Abstract

The purpose of this project is to simulate multi-user multi-input multi-output (MIMO) environment by communicating two RF devices called bladeRF x115 software defined radio (SDR), while implementing the Internet Protocol Architecture model. The Round-Robin scheduling algorithm is realized to utilize orthogonal frequency division multiple access (OFDMA) time-frequency resources among users. Socket programming, TUN/TAP interface, liquidSDR and bladeRF libraries are used to build the protocol architecture. After implementation, the effect of different parameters such as OFDMA numerology, forward error correction (FEC) techniques, modulation scheme on some performance criteria like system throughput and bit-error-ratio (BER) are observed by conducting various experiments. Moreover, to validate every step of our results, the Osmocom sofware including osmocom_siggen and osmocom_fft tools is run with bladeRF devices. Thanks to this study, we experience how the user packet is transmitted on the air via RF devices and observing the impact of real environment conditions such as attenuation and multipath effects.

Table of Contents

Abstract ii
List of Figures iv
List of Tables v
1 Introduction 1
2 Background and Related Work 2
2.1 Background 2
2.2 Related Work 2
2.2.1 Channel Independent Scheduling Algorithms 2
2.2.2 Channel Dependent Scheduling Algorithms 4
3 Implementation Details 6
3.1 System Architecture 6
3.1.1 Transmitter Side 6
3.1.2 Receiver Side 6
3.2 System Implementation 8
3.2.1 Transmitter Side 8
3.2.2 Receiver Side 10
4 Results and Discussion 11
4.1 Methodology 11
4.2 Results 14
4.3 Discussion 17
5 Conclusion 18

List of Figures

Figure 1 Flow chart for Round Robin algorithm 5
Figure 2 System Architecture 7
Figure 3 Illustration of Routing Configuration 9
Figure 4 Default subcarrier allocation defined by *liquidSDR* [1] 10
Figure 5 Half and full duplex communication 15
Figure 6 Illustration of Packet Delivery Ratio versus Distance 16
Figure 7 Illustration of BER versus SNR 17
Figure 8 Illustration of Throughput versus Gain 18
Figure 9 Illustration of IP Packet Loss versus Distance 19

List of Tables

Table 1 BladeRF calibration parameters 15

1 Introduction

The problem we are trying to solve in this work is to make OFDMA scheduling in multi user MIMO system on SDR at the beginning. However, due to the insufficiency of the subcarrier allocation in the liquidSDR library which is used for physical layer implementation, the problem is altered to implement single user MIMO. In addition, since we have used single antenna at both transmitter and receiver sides, the problem can also be considered as single user single-input-single-output (SISO) problem.

The Internet protocol architecture model used for realizing of this project consists of five layers. The socket programming is implemented by UDP packets in order to communication application and transport layers. Next, we decided to use network tunnel (TUN) which is a virtual network layer. At the transmitter side, TUN interface receives the UDP packets from transport layer and sends IP packets to link layer. At the receiver side, the reverse process is occurred. In the following link layer, the Round-Robin algorithm would be applied for OFDMA scheduling task, yet due to the project alteration, this algorithm is not implemented. We just mention it in this report and presentation. The details of the algorithm is provided in Section 2.2.1. In the last layer called physical layer, we perform the operations in this layer by using the liquidSDR library. Here, the payload and the OFDM header is assembled and inverse FFT (IFFT) operations are achieved. After this process, the packet is returned into complex buffer. In order to transmit data to air interface, this complex buffer is converted into the format by which the RF device manages to understand. the bladeRF device uses the libbladeRF library to operate the process.

The importance of solving this problem is to test the field performance of communication systems in real life. The simulation environment can be used to test the performance of the practical cellular communication system including a base station at the transmitter side and user equipment at the receiver side. By changing some parameters such as modulation order, forward error correction technique or SNR value, one can observe their effects on the system very easily. Testing the performance of the system before it is deployed to the real test site is very important to see possible breakthroughs about it.

This project in general is not a very common concept. First of all, the detailed expression on how the TUN / TAP interface communicates is not conclusive except for the detailed description of how to use the TUN, or the code-sharing, except for a few examples in its own library. In addition, it is said that the TUN interface accepts the IP packet, but there is not enough information about how to overcome the problem due to the loop back event on the local table of the Linux operating system. It is not mentioned that UDP packet that is created by UDP Socket programming should be written to the TUN interface in order to create IP packets. Users should carry out long research to grasp that it will communicate with the TUN.

The problem with the libbladeRF library is that which version will be supported by bladeRF device. The compatibility between the FPGA, libbladeRF and the firmware versions is not understood until the device starts and fails. In addition, how to calibrate is still not clearly specified. We can only use the console interface (bladeRF-cli) for this purpose, we can not figure out to make this with coding implementation. Finally, there is not so specific examples about various input parameters for some methods of liquidSDR library.

We noticed that the bladeRF device must be calibrated very well in order to obtain meaningful results. In other words, it is affected very easily from changing environmental conditions; therefore, calibration process plays an important role to ensure good performance.

2 Background and Related Work

2.1 Background

In wireless communication systems, the number of users and the demands for high data rates are increasing tremendously. However, each system has a limited bandwidth, hence using spectrum efficiently is very crucial. In order to serve a large number of users in one communication cell, different techniques are utilized to increase spectral efficiency. In 4G LTE and also 5G NR technologies, both MIMO and OFDMA techniques can be thought as efficient approaches for this purpose. In MIMO communication systems, by employing multiple antennas at both transmitter and receiver side, multiple data streams can be transmitted to users by allocating the same resource. Moreover, the diversity and multiplexing gain are important advantages of these systems regarding spectral efficiency. On the other hand, due to its immunity to frequency selective fading and close-spaced overlapping subcarriers, high data rate and reliable communications are possible [2].

The data is transmitted in time-frequency resource blocks. In time domain, transmission is achieved in every transmission time interval (TTI) while in frequency domain the divided sub-channels are used. For example, according to the 3GPP standards, in 5G communication, 1 ms TTI and 180 KHz sub-channel bandwidth numerology can be used as one alternative option. The scheduling process helps to dynamically adjust the time and frequency resources in order to optimize different cost functions for example maximizing the overall throughput in a fair or unfair fashion for user equipments [3].

In most cases, the downlink scheduling algorithms can be split into two broad categories:

- Channel-independent scheduling
- Channel-dependent scheduling

Before describing the details of all algorithms, the scheduling metric should be defined. $m_{i,k}$ represents the scheduling metric of the i^{th} user on the k^{th} RB. The user who has the maximum $m_{i,k}$ metric allocates the k^{th} RB.

2.2 Related Work

2.2.1 Channel Independent Scheduling Algorithms

It would make sense to utilize these type of strategies if there was no channel impairments such as fading, path loss or shadowing. However, in wireless communication environment, these are unrealistic approaches when the system conditions are considered. Channel independent algorithms are easy to implement, yet the system performance can be easily affected from instantaneously changing channel conditions.

The channel-independent algorithms are firstly used in wired communication purposes which has a stable transmission media in time. Therefore, only using these methods for wireless communication technologies is not very feasible. They can be utilized together with channel-dependent algorithms in order to improve system performance.

In this review, three types of channel independent scheduling algorithms are examined. Detailed explanation of them are provided in the following.

1) **First In First Out (FIFO) Scheduling:** FIFO strategy is serving according to the time of arrival of data packets. For this method, the scheduling metric of i^{th} user for the k^{th} RB can be represented as

$$m_{i,k}{}^{FIFO} = t - T_i \tag{1}$$

where t denotes the current time and T_i denotes the requesting time of the i^{th} user [4]. In FIFO method, if the queue uses full of its capacity, the incoming data packets are dropped out. This method is the easiest one to implement; however, the its performance is not well as expected due to dropping of packets and being unaware of channel conditions.

2) **Round-Robin Scheduling:** Round-Robin (RR) scheduling is one of the preemptive algorithms which means the resources are allocated to a process for a limited time. It is very similar to FIFO strategy; however, in order to switch the system between process, preemption property is added. The scheduling metric of the k^{th} RB for i^{th} user can be defined as

$$m_{i,k}{}^{RR} = t - T_i \tag{2}$$

where t denotes the current time and T_i denotes the last serving time of the i^{th} user [5].

In this method, equal time slots called quantum are used for each process. The first part of the process in the queue is assigned to resources until the first quantum time is finished. This occurrence continues in the circular order without any priority until all the process is completed. RR algorithm can be easily implemented and all the process can allocate some resources in time. Therefore, it is also a starvation-free method.

3) **Blind Equal Throughput:** The most prominent aim of the Blind Equal Throughput (BET) method is provide fair throughput values between users. The scheduling metric of BET method for i^{th} user over k^{th} RB can be defined as [6]

$$m_{i,k}^{BET} = \frac{1}{\overline{\overline{R_i^{(t-1)}}}} \tag{3}$$

where $R_i^{(t-1)}$ is the average throughput value of i^{th} user in the past. Here, T represents the window size of average throughput. Then, the instantaneous throughput of the i^{th} user at time t can be calculated as

$$\overline{R_i^t} = \left(1 - \frac{1}{T}\right)\overline{R_i^{(t-1)}} + \frac{1}{T}R_i^t \tag{4}$$

As noticed from Equation 3, the users who have the average worse throughput values in the past has a larger priority in the present time. Therefore, having bad channel conditions means higher resource allocation. However, it causes having degraded spectral efficiency.

2.2.2 Channel Dependent Scheduling Algorithms

In channel-dependent scheduling, the resource allocation is tried to be achieved according to the dynamically changed channel state information (CSI). The channel is estimated by users and channel quality indicator (CQI) reports are transmitted to the eNodeB in LTE in order to specify the scheduling strategy. It is expected that the performance of the channel-dependent algorithms is better than channel-independent ones, since they attempt to provide fairness between users who have different channel qualities.

In this report, two types of channel dependent scheduling algorithms are analyzed.

1) **Maximum Throughput Scheduling:** In Maximum Throughput scheduling method, the user who has the best channel conditions meaning that having the highest throughput value allocates the resource. The scheduler analyzes the CQI reports in order to calculate throughput values of all users. The scheduling metric at time t for i^{th} user over k^{th} RB can be defined as [6]

$$m_{i,k}^{Max-T} = log\left(1 + SINR_k^i\left(t\right)\right) \tag{5}$$

where $SINR_k^i\left(t\right)$ represents the signal to noise and interference ratio of the i^{th} user at time t.

2) **Proportional Fair Scheduling:** This method can be considered as the combination of Maximum Throughput and BET algorithms. The resources are allocated according to the ratio of instantaneous throughput of the i^{th} user. Therefore, the priority metric of Proportional Fair (PF) strategy at time t for the i^{th} user is expressed as

$$m_{i,k}^{PF} = m_{i,k}^{Max-T} m_{i,k}^{BET} = \frac{log\left(1 + SINR_k^i\left(t\right)\right)}{\overline{R_i^{(t-1)}}} \tag{6}$$

The user who has the highest $m_{i,k}^{PF}$ value has priority to have resources. PF algorithm tries to achieve fairness between users over the combined effect of instantaneous and average throughput values. Calculation of $\overline{R_i^{(t)}}$ can be done in two different ways according to the different cases [7],[8].

If at time instant t, the i^{th} user is selected, $\overline{R_i^{(t)}}$ becomes

$$\overline{R_i^{(t)}} = \left(1 - \frac{1}{T_{PF}}\right)\overline{R_i^{(t-1)}} + \frac{1}{T_{PF}} R^i\left(t\right) \tag{7}$$

If at time instant t, the i^{th} user is not selected, $\overline{R_i^{(t)}}$ becomes

$$\overline{R_i^{(t)}} = \left(1 - \frac{1}{T_{PF}}\right)\overline{R_i^{(t-1)}} \tag{8}$$

In this study, we have decided to implement round robin algorithm for scheduling. Although it does not incorporate channel information into account; it is easy to implement. Moreover, every user is treated in a fair manner which means that it is a starvation free method. In Figure-1 flow chart for round robin algorithm is illustrated.

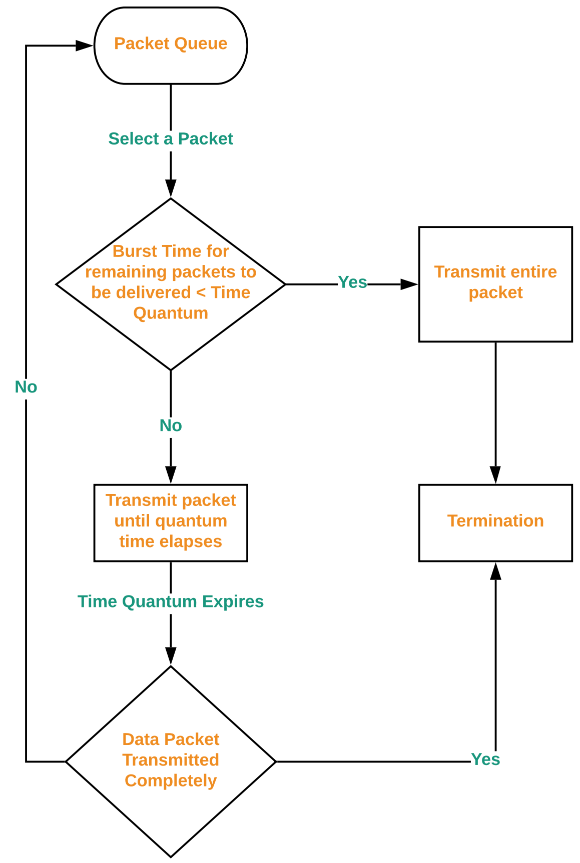
Packet Queue
Select a Packet
Burst Time for remaining packets to be delivered < Time Quantum
Yes
Transmit entire packet
No
No
Transmit packet until quantum time elapses
Termination
Time Quantum Expires
Data Packet Transmitted Completely
Yes

Figure 1 . Flow chart for Round Robin algorithm

3 Implementation Details

The detailed description of system architecture is provided in this section.

3.1 System Architecture

3.1.1 Transmitter Side

In order to simulate desired multi-user environment of this project, we have decided to create n different sockets where n denotes the number of users, but we have only one TUN interface. While creating sockets, we set different port addresses. Moreover, each port is connected to single TUN interface in order to send UDP packets from each port to destination.

The existing users desire to access to medium instantaneously. In round-robin scheduling part, each thread corresponding to the each client, namely the messages of each client are pushed into the associated queues. In this way, each user is fairly served. The scheduling approach is meaningful when multiuser communication is required. However, in our case, due to the alterations in the project, we are only desired to implement single user communication.

By binding each UDP packet to TUN interface, it is routed to its destination throughout these interfaces. However, there is a problem about routing of these packets. In other words, under normal conditions, the packet traffic originated from local host and routed to another local destination IP is sent through a loopback interface regardless of the IP address of destination interface. Therefore, we can not get any message from TUN interface when the interface is listened using *tcpdump* command. In order to deal with this problem, we create some routing configuration.

IP packets received from TUN interface is read and using *liquidSDR* library, the operation doing by physical layer such as modulation and coding schemes are achieved over payload.

libBladeRF library receives the packet which is prepared from the liquidSDR library. Before that, this packet is converted to the appropriate format for bladeRF and it is transmitted to the transmission antenna of bladeRF device.

3.1.2 Receiver Side

The signal received by the receiver antenna of bladeRF device using libBladeRF library. Then it is converted to the proper format for liquidSDR. After that, demodulation and decoding tasks are performed by liquidSDR library. The OFDM header and payload are splitted by this library and the payload (IP header + user packet) is written to TUN interface. At this stage, the TUN interface separates the IP header. The remained payload (UDP header + user packet) is received from UDP socket. Finally, raw user packet is obtained and written to user-space program.

The detailed design stack of our system is illustrated in the Figure 2 .

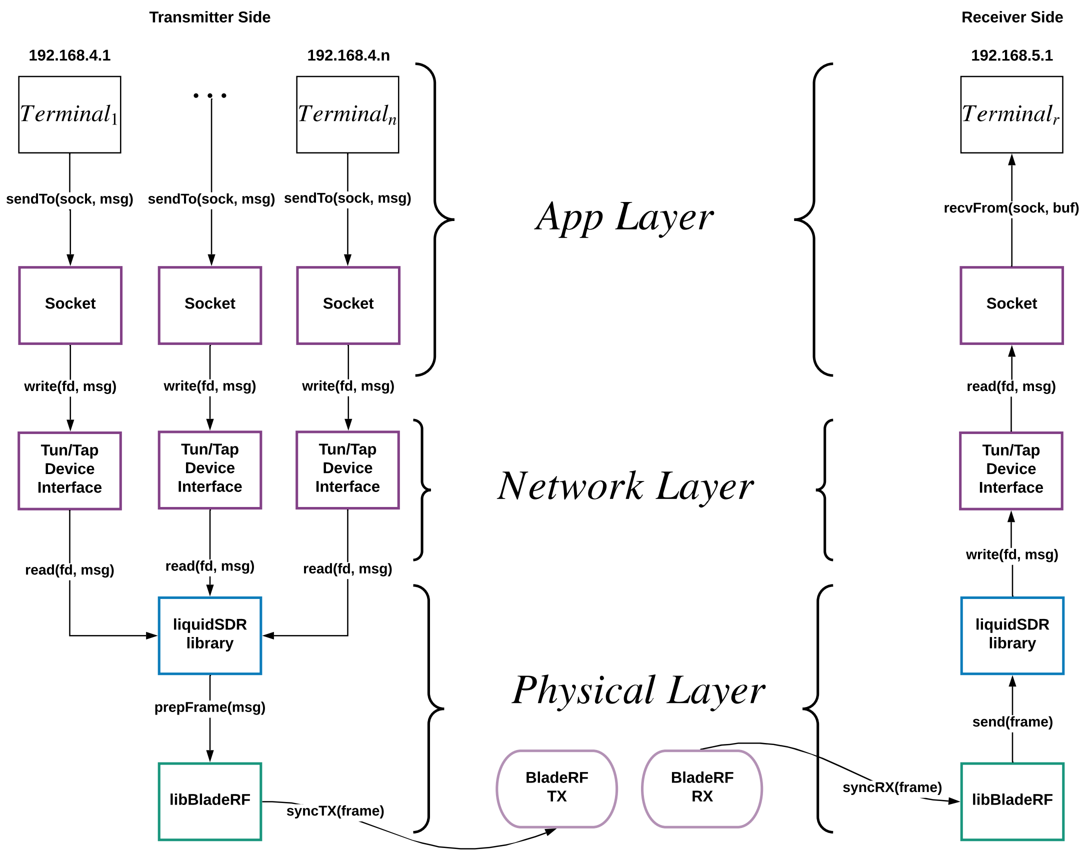

Figure 2 . System Architecture

3.2 System Implementation

The detailed expressions of implementation process of transmitter and receiver side are provided in the following subsections.

3.2.1 Transmitter Side

In this part, the layer based design description of the transmitter (TX) side is given.

- **Application Layer Design**
  In order to use TUN interface, we need to create IP packets. Therefore, we use Linux socket programming interface, namely UDP sockets. As specified in [9], due to the concept of TCP over TCP which is known as a disadvantage leading to reduce the packet delivery performance of the system, we decided to use UDP packets instead of TCP ones.

- **Network Layer Design**
  As explained in the Section 3.1.1., because of the problem of not to be able to observe the desired traffic flow over TUN interfaces, we configured some routing paths. For instance, it is observed that when we ping the TUN interface, Kernel response to this ping itself. Therefore, no packet is sent to the wire and we can not observe any packet in the *tcpdump*. In order to overcome this problem, we used the routing configuration described in [10]. Since TUN interfaces are for tunneling purposes, they need a peer-to-peer address to be set. However, when a network interface is created by *ip tuntap* command; because of its side effect, a default routing configuration is set up by the Kernel. These default configuration settings enforce packets to be routed to lo (loopback) interface and they never reach to the desired location namely the TUN interface. To avoid this incidence, entries in local routing table should be removed and new routing configurations should be created.

  In the network layer, TUN interface is created to transform the UDP packets to IP packets. Here, it should be emphasized that the subnet mask of TUN interfaces at the transmitter and receiver sides must be different in order to help Kernel to create a routing decision. The routing configuration described above is illustrated in Figure- 3 is realized through this TUN interface.

- **Link Layer Design**
  At the beginning of the project, we were planned to design an OFDM scheduling algorithm. However, we have noticed that liquidSDR does not have an ability about frequency allocation for specific users. As a result, it is decided that OFDM scheduling task will be excluded. Therefore, there is no need to make multi-user scheduling, and link layer design is not realized. Only the theoretical description of some scheduling algorithms are given in Section 2.2.1 and 2.2.2.

- **Physical Layer Design**
  IP packets can be read from the TUN interface so it is ready to handle by physical layer operation. The packet read is the one with payload and ip header parts. We employ default OFDM subcarrier allocation scheme provided by *liquidSDR* as shown in Figure-4 . In this representation, guard bands indicates no transmission (i.e. wasted

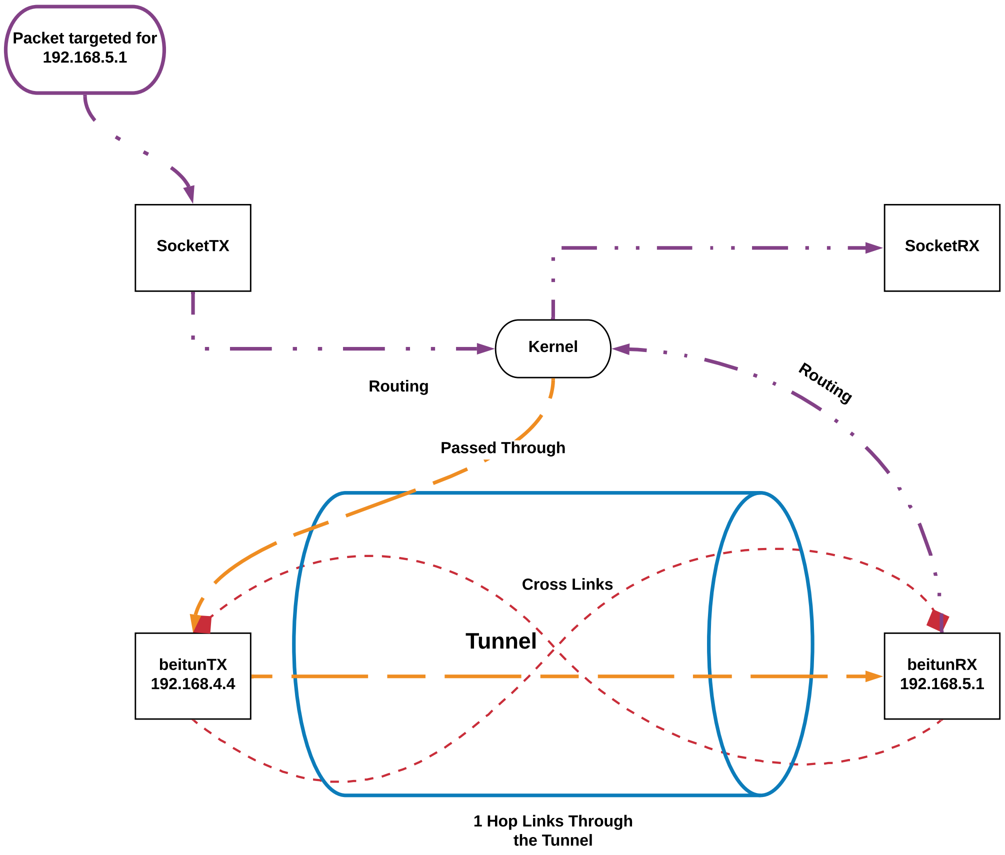

Figure 3 . Illustration of Routing Configuration subcarriers), spectral null symbols are used to avoid side-lobes between consecutive symbol transmissions, and pilot symbols are included to recognize channel impairments for recovery at the receiver side. Before complex data prepared by *liquidSDR* is sent to the *bladeRF* driver, **SC16Q11** encoding scheme is applied to this complex data.

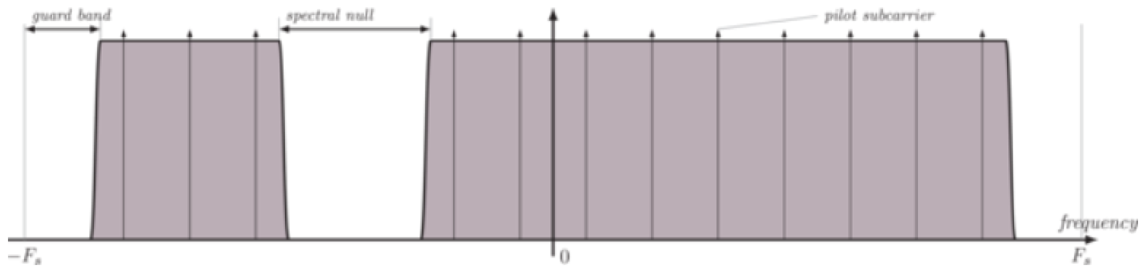

Figure 4 . Default subcarrier allocation defined by *liquidSDR* [1]

3.2.2 Receiver Side

- **Physical Layer Design**
  Physical layer packets received from bladeRF device at the receiver side are given to *liquidSDR*'s *execute* command as parameter in order to parse the OFDM header and to extract payload in proper format. Before this parsing takes place, inverse transformation of **SC16Q11** is applied to decode complex payload. After that, the OFDM callback method is called to provide packet validation information and payload data.

- **Network Layer Design**
  The IP packet obtained from the OFDM callback method is written to the TUN interface at the receiver side. By doing so, the kernel parses IP packet and forward payload to receiving applications which listen to this UDP channel. TUN interface routing configuration set up at the transmitter side is also valid for receiver.

- **Application Layer Design**
  recvFrom method is summoned for listening to incoming packets coming through TUN interface associated with the receiver application. When a packet is delivered to the application layer its content is parsed to extract corresponding packet count for statistical purposes. We keep track of each client's received healthy packets as well as the count for the first packet owned by that particular client. The reason why we store the count for the first packet is due to delayed start of the receiver application relative to the transmitter one.

4 Results and Discussion

4.1 Methodology

- **OFDM Technology**

  OFDM (Orthogonal Frequency Division Multiplexing) is a technology in which a wide frequency band is splitted into many small frequencies called subcarriers. Each of these subcarriers is assigned to carry a complex number (modulated symbol). Therefore, one OFDM symbol can be represented as the sum of N complex exponentials given in Equation 9

  $$S_s(t) = \frac{1}{N}\sum_{n=0}^{N-1} A_n(t)\, e^{j(W_n(t)+\phi_n(t))} \tag{9}$$

  where $N, S_s(t), A_n(t), W_n(t)$ and $\phi_n(t)$ represent the number of subcarriers, the total OFDM symbol, the amplitude, center frequency and the phase of the n^{th} symbol respectively [11].

- **Advantages and Disadvantages of OFDM**

  In the traditional multicarrier systems, in order to prevent the interference between carriers, subcarriers are designed as non-overlapping with each other. However, this causes to have a larger bandwidth; therefore, decrease the spectral efficiency. To deal with this problem, OFDM offers a solution by using an overlapping subcarriers in frequency domain which are orthogonal to each other. In this way, the spectral efficiency is increased which is the most important advantages of the OFDM.

  On the other hand, OFDM is vulnerable to the multipath phenomenon. In order to overcome this environmental effect, a cyclic prefix can be utilized. The cyclic prefix behaves as a buffer or guard interval in order to prevent the OFDM signal from intersymbol interference (ISI). This technique repeats the end of the symbol; therefore, the linear convolution of the channel can be modeled as circular convolution, which in turn may transform to the frequency domain via a discrete Fourier transform. Adding cyclic prefix provide robustness for data to multipath conditions. Yet, using it also causes to reduce the throughput capacity since a part of the spectrum is utilized for cyclic prefix which is the same as the part of the actual data that is already transmitted. Another major drawback of the OFDM technology is related with peak-to-average-power-ratio (PAPR) problem. Due to the central limit theorem, addition of many complex exponentials given in Equation 9, goes to a Gaussian distribution which means the OFDM signal may take extremely large values instantaneously [12]. This fluctuations cause distortion in the operations regions of the linear amplifiers at the receiver side; therefore, the signal can not be correctly decoded.

- **OFDMA Technique and *liquidSDR* Library**

  OFDMA (Orthogonal Frequency Division Multiple Access) is an medium access technique used in the link layer of the communication system. Here, each user trying to access the channel is assigned to a certain frequency-time block that includes one or many subcarriers at the frequency domain. The planning about time-frequency assignment is achieved by scheduling algorithms. At the beginning of this project, we are expected to design an OFDMA scheduling algorithm by using *liquidSDR* library to implement physical layer. However, due to the fact that assigning frequency subcarriers using *liquidSDR* is not very easy task, scheduling only in the time domain is decided to be enough. Therefore, at the transmitter side we used *liquidSDR* library only for OFDM signal generation while at the receiver side it is utilized for OFDM decoding.

In *liquidSDR* library, *ofdmflexframe* structure is used for this purpose. It is a family of methods achieving OFDM generation and synchronization. With this structure, the user can specify the number of subcarriers, the length of CP length, forward error correction (FEC) type or modulation and coding scheme (MCS) etc.

Besides the multipath effects, the multicarrier systems are also very sensitive to the channel frequency offset or Doppler shift phenomena. In order to achieve synchronization, the preamble part is attached by the transmitter at the beginning of each OFDM frame. This part is already known both by transmitter and receiver part and it can be utilized to correct the channel frequency response and eliminate the channel effects on the actual data by channel estimation algorithms.

The *ofdmflexframe* structure construct an OFDM signal consisting of three main parts called as preamble, header and payload. In the preamble part, carrier frequency offset and timing offset are tried to be eliminated with the help of two sequences called Section s and Section l sequences. In this way, the receiver timing is aligned to that of the transmitter. The header part is utilized to provide the forward error correction (FEC) or modulation information to the receiver. Finally, in the payload part, the actual data desired to transmit exists. The length of the payload depends on the number of subcarriers.

When the transmitted packet is detected at the receiver side, firstly, the header is tried to be decoded. If the header decoding is successful, then the payload is decoded. When the data is decoded at the receiver side, the subcarrier allocation is an important issue in order to achieve this task correctly. Details of the subcarrier allocation process are given in last part of the methodology section.

To sum up, in order to implement physical layer, two objects called *ofdmflexframegen* and *ofdmflexframesync* are used from *liquidSDR* library. The *ofdmflexframegen* is used at the transmitter side and the raw data is assembled into OFDM symbols in order to create OFDM signal. It is responsible for creating the frame generator, specifying the number of subcarriers, the length of the cyclic prefix and applying modulation and coding scheme (MCS) or FEC techniques. After creating the OFDM frames, they are written into the buffer in order to transmit. On the other hand, at the receiver side, *ofdmflexframesync* method is utilized to detect frames and then to achieve decoding the header and payload parts [1].

- **OFDM Parameter Selection**

  In this project, we set the number of subcarriers to 64, the length of the cyclic prefix to 16. In *liquidSDR*, the window tapering technique which is utilized for reducing the effect of spectral sidelobes by providing smooth transition between adjacent OFDM symbols. In this work, we set the window taper length to 4. In addition, FEC type 1 is set and Hamming 128 coding technique is used. There can be many parameter selection combination all of which have some advantages and drawbacks. For instance, as we increase the number of subcarriers we can obtain better equalization results with the cost of a hardly achievable carrier frequency requirements. Moreover, using longer cyclic prefix leads to have a better robustness to multipath effects with the cost of reduced throughput and spectral efficiency. Therefore, parameter selection should be analyzed according to the operating conditions.

- **Subcarrier Allocation**

  In OFDMA, subcarriers can be allocated by three types of carriers called as null, pilot and data. In the null part, there is no symbol assignment for the subcarriers. This region is also called guard band or spectral notch in order to prevent interference from neighbour frequency bands. Furthermore, pilot carriers is necessary in order to estimate impairments due to channel condition such as carrier frequency offset. By comparing the pilot carriers at the transmitter and the receiver side, one can achieve the channel estimation. Finally, data subcarriers is used for actual payload transmission. As the number of subcarriers increases, the spectral efficiency also increases, as expected.

  In this project, subcarrier allocation in OFDMA is a crucial part in order to correctly decode the data at the receiver side. In other words, the allocation order of null/pilot and data subcarriers should be known both the transmitter and receiver sides, namely *ofdmflexframegen* and *ofdmflexframesync* methods should use the same *p* vector as an input.

- **Tunneling**

  As discussed in subsection 3.2.1, the purpose of tunneling is to force kernel to do routing. First off, point-to-point addresses for tun interfaces are cross-linked via the shell script given in Code-1. The objective for this step is to create an automatic routing between these two interfaces through Linux kernel.

```
    # ifconfig beitunT0 192.168.4.4 pointopoint 192.168.5.1
    # ifconfig beitunS 192.168.5.1 pointopoint 192.168.4.4
```

Listing 1: P-t-p addressing

  This is not sufficient to have network packets routed through tun interfaces; because Linux kernel automatically generates configuration entries in routing table which causes packets not to be delivered to destined tun interfaces; but to be routed through loopback interface. To avoid this wrong packet delivery; we should remove these default entries from these tables by executing shell commands shown in Code-2.

```
    # ip route del 192.168.5.1 table local
    # ip route del 192.168.4.4 table local
```

Listing 2: Avoiding kernel loopback interface

In order to have these two tun interfaces (namely *beitunS* and *beitunT0*) communicate with each other, we create separate routing table for each interface (i.e. table-**4** and table-**5**) and add rules specified in Code-3. By doing so, we could take advantage of having these two interfaces been located under different subnets.

```
# ip route add local 192.168.5.1 dev beitunS table 5
# ip route add local 192.168.4.4 dev beitunT0 table 4
# ip rule add iif beitunS lookup 5
# ip rule add iif beitunT0 lookup 4
```

Listing 3: Custom routing entries

- **Calibrating BladeRF**

  BladeRF is a half-duplex SDR (sofware-defined radio) device (see Figure-5). It splits bandwidth into half and utilizes one part of it for TX; while the other for RX. Moreover, it only works at critical sampling rate. Thus, one should set sampling rate equal to bandwidth. We set central frequency to 455.55Mhz which is mainly used for security purposes such as police and security guards; because around this frequency it seems less noisy by looking at **osmocom_fft** application. In accordance with the range between TX and RX in which they operate, it is sufficient to set gain for TX to 60dbm; while to 15dbm for RX. However, we recalibrate devices to run on different gain values for few different number of scenarios they are tested against. We use **bladeRF-cli** application to calibrate devices and do not include any code in our implementation regarding calibration process. Commands executed in **bladeRF-cli** application for both TX and RX devices are listed in Code-4 and Code-5.

```
$ set frequency tx 455.55M
$ set bandwidth tx 5M
$ set samplerate tx 5M
$ set gain tx 60
$ cal lms
$ cal dc tx
```

Listing 4: Setting TX calibration params via bladeRF-cli

```
$ set frequency rx 455.55M
$ set bandwidth rx 5M
$ set samplerate rx 5M
$ set gain rx 15
$ cal lms
$ cal dc rx
```

Listing 5: Setting RX calibration params via bladeRF-cli

4.2 Results

In this section, we show the output of evaluating our application against various parameters and environmental conditions. We measure packet delivery ratio, BER (bit error rate), throughput, and IP packet loss.

Packet delivery ratio is evaluated against varying distance between TX and RX devices. They are captured at 6, 36, and 235cm. Packet size is of length 50 characters long and it

Table 1. BladeRF calibration parameters

	TX params	**RX params**
Frequency	455.55M	455.55M
Bandwidth	5M	5M
Sample Rate	5M	5M
Gain	60dbm	15dbm

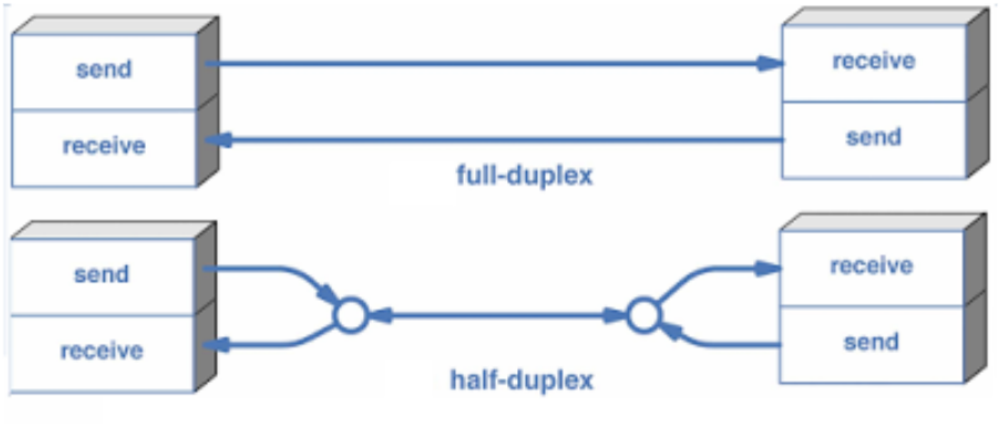

Figure 5 . Half and full duplex communication

remains constant throughout this experiment. It is calculated using Equation-10. It exhibits the best packet delivery ratio when distance between TX and RX is 6 cm and the worst is obtained when it is equal to 36 cm. The mid-rate is attained at 235 cm. This is illustrated in Figure-6 .

$$PDR = \frac{\#\ recv\ packets\ by\ app\ layer}{\#\ sent\ packets\ by\ TX} \tag{10}$$

BER (bit error rate) is computed at application layer (where final payload is extracted). We measure it at this layer; because the broken packets drop at lower layers. There exists a method called **count_bit_errors_array** provided by *liquidSDR* library to compare a known payload to the received one. BER is measured against SNR (Signal-to-Noise). An artificial impairment is added to distort the message immediately after it is received and converted to complex representation. We conduct our experiment with two different modulation schemes namely *QAM16* and *QPSK* shown in blue and red in Figure-7 respectively. Devices are 72 cm apart from each other during this experiment. SNR values set for channel impairment purposes for QPSK are 38, 38.5, 39, 39.5, and 40 db; while 50, 51, 52, 53, and 54 for QAM16. BER is displayed in logarithmic scale and averaged over all packages received till the end of this experiment.

Throughput is sensitive to gain set on TX device, hence its change against gain is recorded in this third experiment. We start a timer when the first packet is delivered and stop it right after approximately 6000 packets (which sum up to the whole experiment) received. Then we divide number of healthy packets captured at application layer to elapsed time in

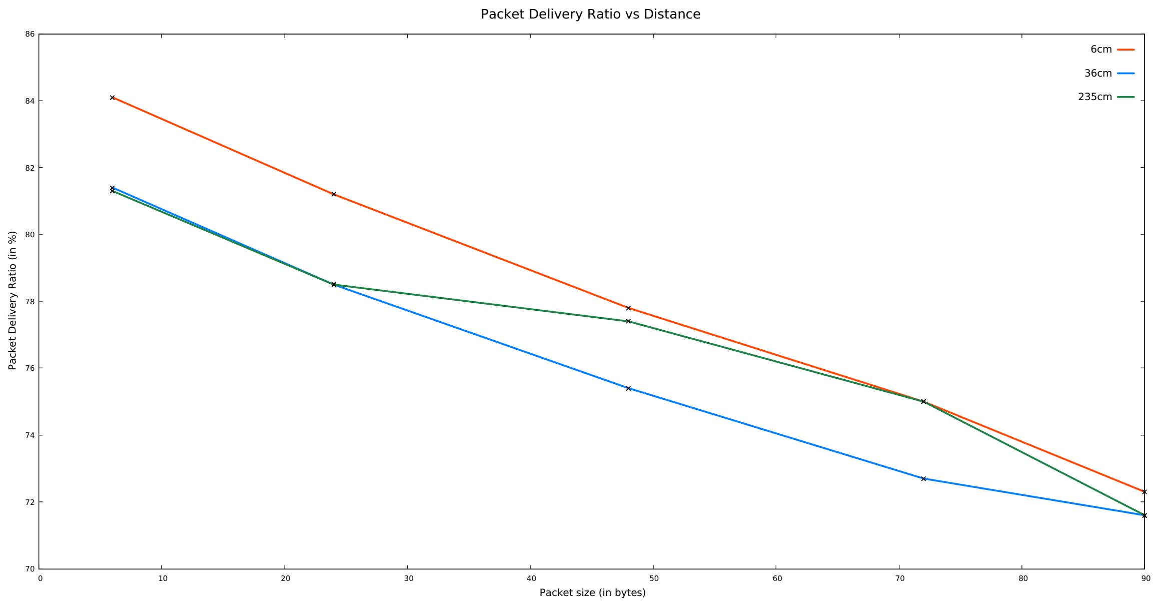

Figure 6 . Illustration of Packet Delivery Ratio versus Distance

```
        float noiseFloor = -80.0f; // noise floor [dB]
        float SNRdB      = 56.0f;// signal-to-noise ratio [dB]
        float dphi       = 0.02f; // carrier frequency offset
        //
        // Create channel impairement
        //
        channel_cccf channel = channel_cccf_create();
        channel_cccf_add_awgn(channel, noiseFloor, SNRdB);
        channel_cccf_add_carrier_offset(channel, dphi, 0.0f);
        channel_cccf_execute_block(channel, y, sampleLen, y);
```

Listing 6: Adding channel impairment

seconds. Gain is set to 45, 48, 50, 52, and 70 db. Each time gain is updated, a calibration is performed. Packet size is of 50 characters long and distance is kept at 72 cm far. Its output is given in Figure-8 .

IP packet loss is the last metric we collect. It refers to packets lost at network layer. That is to say, Linux kernel drops IP packets with corrupted IP headers. This metric shows how many such headers are received by RX's tun interface. IP packet loss is observed with changing distance between TX and RX devices. Distances at which experiments conducted are 6, 36, 72, 144, and 235 cm in order. Equation-11 is employed to compute the loss. Result is displayed in Figure-9 .

$$IPLoss = \frac{\#\ valid\ IP\ packets}{\#\ recv\ liquidSDR\ frames} \tag{11}$$

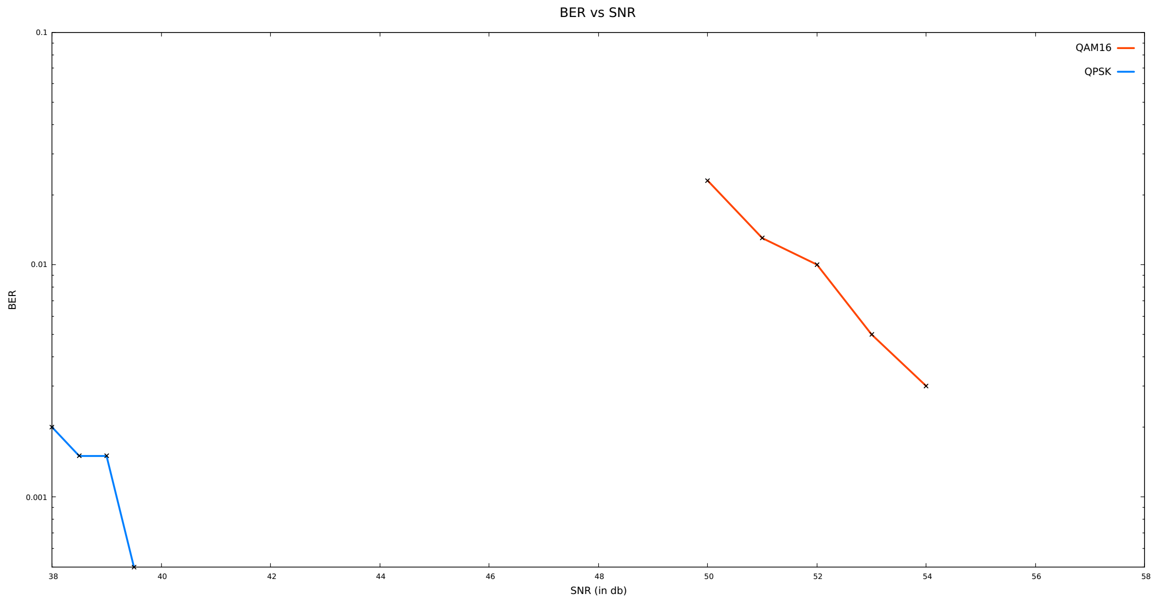

Figure 7 . Illustration of BER versus SNR

4.3 Discussion

As shown in Figure-6 , results for packet delivery ratio is consistent with what is expected in experiments with 6 and 36 cm; but as for 235 cm it fails to demonstrate this consistency. In experiment with 235 cm, due to *near field* issue at our operating central frequency 455.55M and lack of equipment for extending two devices to a range of $10 * \lambda$, (where λ is wavelength of carrier frequency) it does not exhibit this linear decrease in packet delivery. To actually observe a constant decrease in packet delivery ratio as distance grows up, we should have performed this experiment at far field.

Since QPSK and QAM16 represent OFDM symbols with 2 and 4 bits respectively, the latter one requires more power at TX side. We conclude that more bits to be transmitted demand more power and this phenomenon is illustrated in Figure-7 . These findings comply with the theoretical background which explains BER.

In IP packet loss experiment, we observe higher number of packet loss as we increase distance between TX and RX devices. This is an expected behaviour and given in Figure-9 . However, until 140 cm we do not encounter a significant increase in IP packet loss; while right after we exceed this threshold, a large amount of loss occurs due to multipath effect such as reflection and refraction.

Finally, throughput is measured against TX gain as illustrated in Figure-8 . Until 52 dbm, a significant growth in throughput is realized as we increase gain at TX side. We also conduct this experiment in both in-door and out-door environments; though it proves not to be worthy in affecting the throughput. That is to say, we do not observe any multi path effect at all.

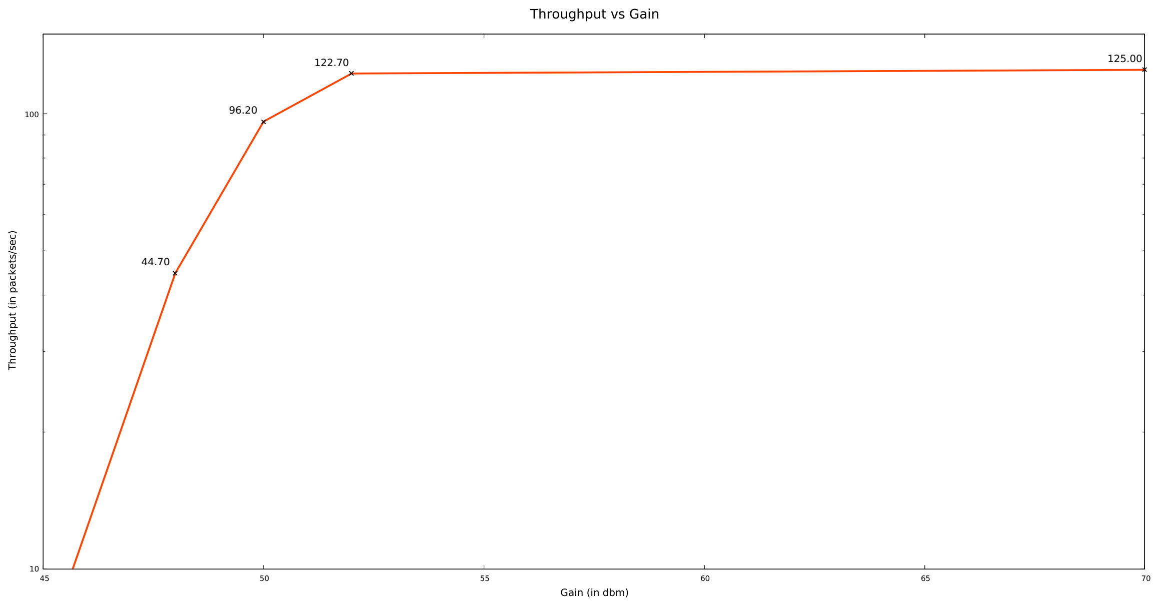

Figure 8 . Illustration of Throughput versus Gain

5 Conclusion

In this project, we strive to implement four layers of Internet Protocol Architecture (application, network, link, and physical layers) by communicating two bladeRF devices, one TX and one RX. Although we do not employ scheduling algorithm for ordering packet transmission; we have the chance to inspect various scheduling algorithms and their corresponding implementations. In the context of networking, we learn how to code with different software packages such as bladeRF, liquidSDR, and TUN/TAP interface. To realize the effect of different parameter settings, we conduct several experiments. We learn how to handle several difficult issues as well. For instance, we tackle calibration problem in which we strive to grasp the idea behind half/full duplex notion for setting correct sample rate and bandwidth parameters. While doing so, we learn how to use osmocom to probe the hardware and to generate a DC (direct conversion) table. Another problem we face during the implementation is due to not flushing USB buffers after we close the bladeRF device in our application code. We notice this trouble when we do not compile our application after the very first execution. This causes not to obtain meaningful results in accordance with our parameter settings. In order to reload dynamic usb libraries and to flush usb buffers forcefully, we recompile our executable after each parameter update.

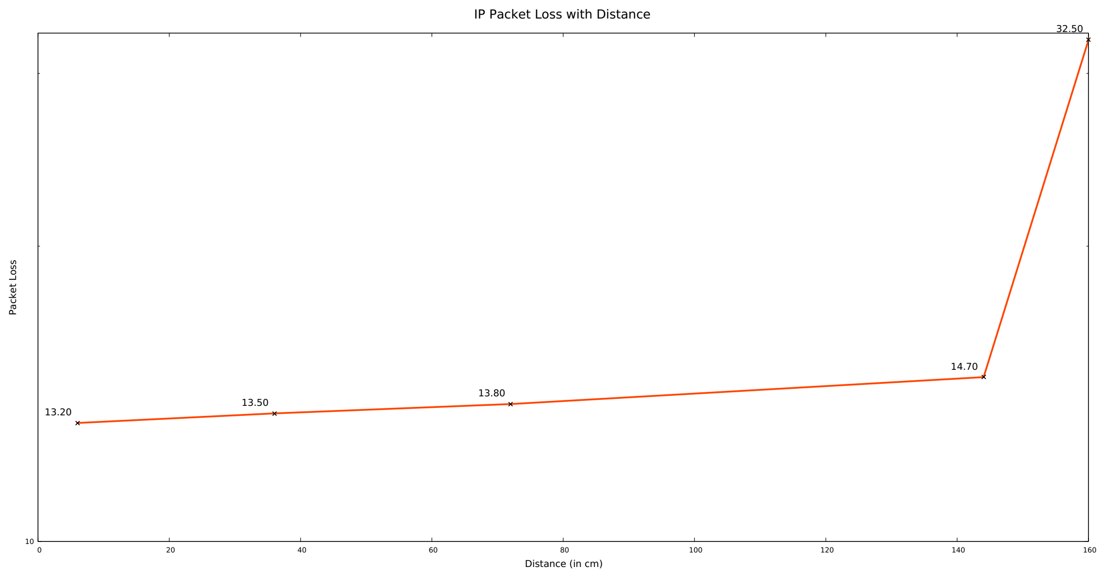

Figure 9 . Illustration of IP Packet Loss versus Distance

References

[1] "OFDM Flexible Framing Structure (ofdmflexframe)," https://liquidsdr.org/doc/ofdmflexframe/.

[2] "3GPP,Technical Specifications Group Radio Access Network -Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN)," http://www.3gpp.org, 3GPP TS 25.913.

[3] "3GPP, Technical Specifications Group Radio Access Network-Physical channel and Modulation (release 8)," http://www.3gpp.org, 3GPP TS 36.211.

[4] A. S. Tanenbaum, *Modern operating system.* Pearson Education, Inc, 2009.

[5] S. Hussain, "Dynamic radio resource management in 3gpp lte," 2009.

[6] P. Kela, J. Puttonen, N. Kolehmainen, T. Ristaniemi, T. Henttonen, and M. Moisio, "Dynamic packet scheduling performance in utra long term evolution downlink," in *2008 3rd International Symposium on Wireless Pervasive Computing.* IEEE, 2008, pp. 308--313.

[7] K. W. Choi, W. S. Jeon, and D. G. Jeong, "Resource allocation in ofdma wireless communications systems supporting multimedia services," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 3, pp. 926--935, 2009.

[8] R. K. Almatarneh, M. H. Ahmed, and O. A. Dobre, "Performance analysis of proportional fair scheduling in ofdma wireless systems," in *2010 IEEE 72nd Vehicular Technology Conference-Fall.* IEEE, 2010, pp. 1--5.

[9] "Why TCP Over TCP Is A Bad Idea," http://sites.inka.de/~W1011/devel/tcp-tcp.html.

[10] "TUN/TAP Tunnels," http://rrendec.mindbit.ro/post/tun-tap-tunnels/.

[11] "Orthogonal Frequency Division Modulation (OFDM) ," https://www.csie.ntu.edu.tw/~hsinmu/courses/_media/wn_11fall/ofdm_new.pdf.

[12] M. C. P. Paredes and M. García, "The problem of peak-to-average power ratio in ofdm systems," *arXiv preprint arXiv:1503.08271*, 2015.