Studying the Impact of Protocol Conversions on Multi-Hop Routing

Mallinatha, Niketan Shahapur
M Tech 1st year
IIT MADRAS
Chennai,India
cs18m003@smail.iitm.ac.in,
cs18m036@smail.iitm.ac.in

Abstract — **The aim of this project is to study the impact of protocol conversion in a multi-technology IoT system on multi-hop routing during best effort transmission. This project also aims to study the packet loss due to multiple conversions, fragmentation and defragmentation. This is achieved by implementing multiple ways of protocol conversion using two laptops and two practical and commonly used multi-technology IoT platform- Raspberry Pi.**

I. INTRODUCTION

Protocol conversion offers several challenges like frame format differences, conversion complexities, fragmentation issues etc. The overhead associated with the same can be quantified in terms of cost, packet loss, delay, energy or link utilization.
Here protocol conversion at each of the hop is implemented using the approach that the packet payload is retrieved first and the new technology header is appended only to the payload, this is the most efficient conversion scheme.
The expected outcome of task is to show the impact and performance of protocol conversion of the Wi-Fi based packet to other protocols relayed through RPi platform. This is done by measuring the packet loss with respect to both the protocols.

A. Phases of the project

The project has been implemented in the following three phases:

- In the first phase we implemented and studied packet loss during best effort transmission of 100 packets of each technology between two laptops
- In second phase we implemented and studied the packet loss and impact of 4 different combinations of protocol conversion in a multi hop transmission with 2 R-Pi in between.
- In third phase we implemented and studied the packet loss and impact of fragmentation and defragmentation in a multi hop transmission with 2 R-Pi in between.

II. INSTALLATIONS AND SETUPS

A. Raspberry Pi

Raspbian OS was installed on Raspberry pi3 using NOOBS installer and configured as per the project requirement. In our implementation we require two Raspberry Pi devices in a multi hop transmission.

B. Bluetooth and WiFi

In-built Bluetooth and WiFi support in Raspberry Pi and laptop is used for Bluetooth and WiFi communication.

C. ZigBee

For ZigBee communication, Digi xbee S2S module has been used. It is connected to both Raspberry Pi and client-2 through Uni4 ZigBee/xbee USB module. The Digi xbee S2S module is configured through X-CTU. Configuration is summarized in Table 1.

Parameter	**Configuration**
SC scan channel	8
ID PAN ID	2015
DH Destination Address High	Higher bits of serial no of other device
DH Destination Address Lower	Lower bits of serial no of other device
CE (Coordinator Enabled)	enabled for relay and disabled for client 2
AP API Enabled	Transparent Mode

Table 1: ZigBee Configuration

D. Packet Formats

The general packet format for Bluetooth is shown below:

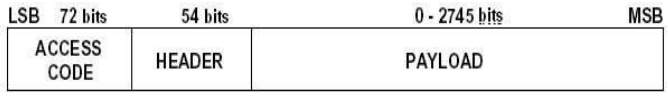

Fig 1: Bluetooth packet format

A packet could contain a shorthanded access code (72 bits) part of the packet only, or access code plus header, or all the three parts. Access code is used for synchronization, DC offset compensation and identification.

The packet format developed for ZigBee is as following:

Start Byte (0x7e)	**Data (1KB)**	**Stop Byte (0x7e)**

Fig 2: ZigBee packet format

III. Procedure and steps followed

3.1 Phase 1

The First Phase involved understanding of Best Effort Transmission using WiFi, Bluetooth and ZigBee technology. Here the packet loss % is calculated for each of the technologies by varying the distance between two laptops.

4.1.1 Work Done

A. Best Effort Transmission using WiFi

The first setup of this phase involved direct communication between two laptops connected through Wi-Fi. One laptop is configured to act as a server. Other laptop acts as client and sends 100 packets to the server. The setup for direct communication between two laptops connected through Wi-Fi is depicted in Fig 3.

Fig 3: Two Laptops directly connected through WiFi

4.1.2 Observations

100 packets were sent from client to server, % number of packets dropped at the server side are noted for 10 varying distances . We can observe from the findings that WiFi transmission is stable for initial 10-15 meters wherein data loss was 0%. However % of packets dropped increase with increasing distance implying energy loss due to distance.

Distance between laptops (in m)	Packets Dropped (in %)
1	0
2	0
4	0
6	0
8	0
10	0
12	0
14	0
16	12
18	18
20	26

Table 2: Packet drop percentage for WiFi

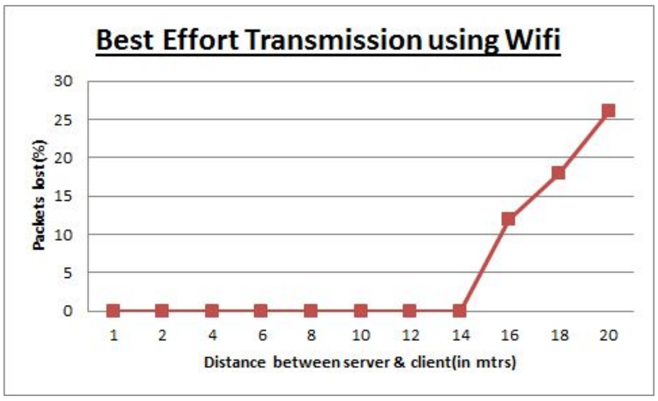

Fig 4: Packets lost percentage graph for WiFi

4.1.3 Work Done

B. Best Effort Transmission using Bluetooth

The setup involved direct communication between two laptops connected through Bluetooth. One laptop is configured to act as a server. Other laptop acts as client and sends 100 packets to the server. The setup for direct communication between two laptops connected through Bluetooth is depicted in Fig 5.

Fig 5: Two Laptops directly connected through Bluetooth

4.1.4 Observations

100 packets were sent from client to server, % number of packets dropped at the server side are noted for 5 different Flush timeout values. Here L2CAP protocol is used to establish connection between two laptops through Bluetooth. L2CAP, by default, provides a connection-oriented protocol that sends individual datagrams of fixed maximum length. In order to make the connection unreliable, a Flush timeout value is set in the program which ensures that, if the packet is not delivered for the duration of flush timeout value, the buffer is flushed thus making it best effort transmission. The % of packet loss was initially studied and found that packet loss by varying the distance was almost zero due to very minimal interference in the environment where the readings were taken. Hence Flush time out value was taken as varying parameter for analyzing % packet drops. We can see from the graph at Fig 6 that, the number of packet loss % gradually decreases with increase in the timeout value. Here timeout value of 1 by default implies 0.625ms, hence with increase in the timeout value, the program is able to retransmit more number of packets before flushing the buffer thus reducing the % of packet loss.

Flush_timeout value	Packets Dropped (in %)
1	65.75
2	63.5
3	62.75
5	49.8

Table 3: Packet drop percentage for Bluetooth

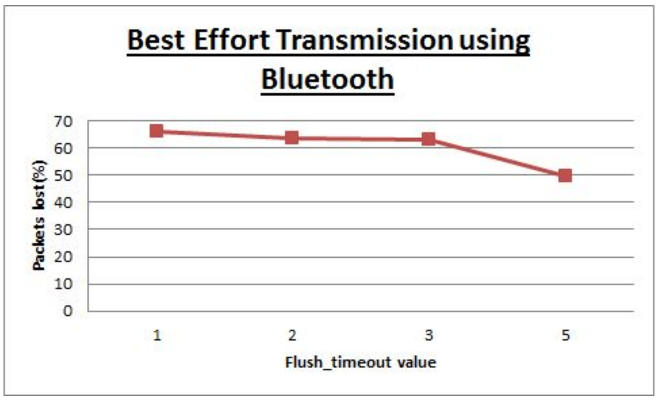

Fig 6: Packets lost percentage graph for Bluetooth

4.1.5 *Work Done*

C. Best Effort Transmission using ZigBee

The setup involved direct communication between two laptops connected through ZigBee. One laptop is configured to act as a server. Other laptop acts as client and sends 100 packets to the server. The setup for direct communication between two laptops connected through ZigBee is depicted in Fig 7

Fig 7: Two Laptops directly connected through ZigBee

4.1.6 *Observations*

100 packets were sent from client to server, % bytes dropped at the server side are noted for 10 varying distances. However % of bytes dropped increase with increasing distance implying energy loss due to distance. The ZigBee protocol uses serial communication to transmit the packets byte by byte. Since it is best effort transmission, any bytes undelivered would be dropped and hence we can see in the graph Fig 8 that % of byet loss keeps increasing with distance.

Distance between laptops (in m)	Packets Dropped (in %)
1	0.4
2	1.86
4	2.03
6	3.22
8	3.1
10	5.32
12	6.76
14	8.49
16	16.83
18	21.27
20	24.29

Table 4: Packet drop percentage for ZigBee

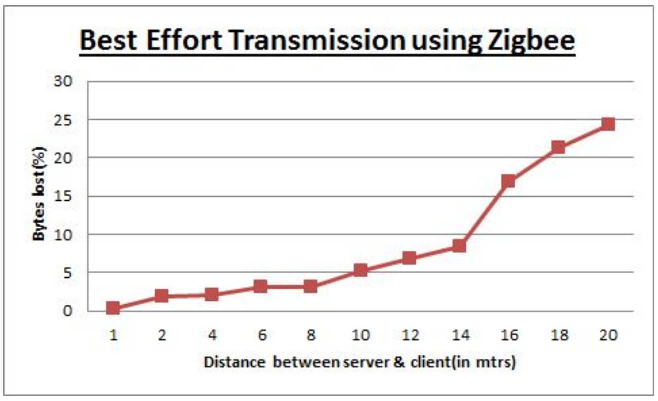

Fig 8: Bytes lost percentage graph for ZigBee

3.2 *Phase 2*

The task of Second Phase is to show the impact of protocol conversion in a multi hop transmission. Four Different Combinations were observed and analysed for packet loss by varying the size of the packets being transmitted from 1KB to 10KB.

3.2.1 *Work Done*

A. Zero protocol conversion

The setup for zero protocol conversion consists of two laptops and two relay agents (RPi). The relay agents support Wi-Fi communication. The network setup is depicted in Fig 9.

Fig 9: Two Laptops connected through WiFi with zero protocol conversions

3.2.2 Observations

100 packets were sent from laptop1 to laptop2 with 2 hops in between. The packets were transmitted and received using WiFi protocol at all the four systems. % packets dropped at the destination laptop are noted for 10 varying packet sizes. We can observe that that the packet loss % increases with increase in the packet size transmitted over 2 hops. The increase in the loss % is mainly attributable to the fragmentation happening at RPi1 as packets of size greater than 1KB sent from laptop are getting fragmented at RPi1 and then transmitted to RPi2. The setup was changed wherein Laptop1 to Laptop2 Wifi Communication was observed and no considerable loss was observed. This reiterates the limitation of RPi in handling large size packets. The same observation can be seen clearly in Fig 10.

Packet Size (in KB)	Packets Dropped (in %)
1	0
2	26
3	40
4	44
5	53
6	54
7	55
8	64
9	65
10	65

Table 5: Packet drop percentage for zero conversion using WiFi-WiFi-WiFi-WiFi

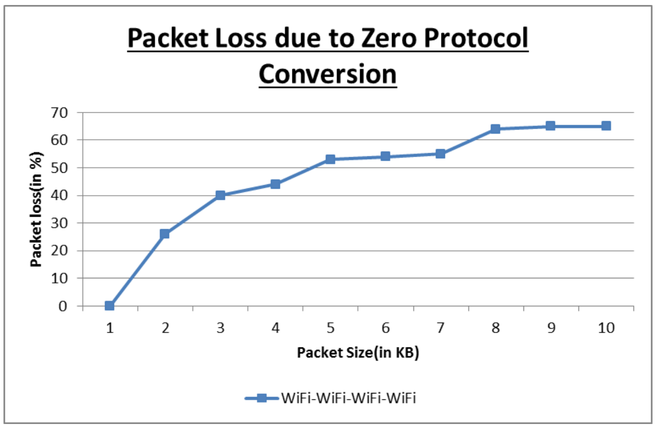

Fig 10: Packet Loss Percentage graph due to Zero Protocol Conversion

2.3 Work Done

B. One Protocol Conversion(ZigBee)

The setup for ZigBee communication consists of two laptops and two relay agents. The relay agent supports both Wi-Fi and ZigBee interfaces. Laptop1 is connected to relay agent(RPi 1) through Wi-Fi and acts as sender. RPi 1 is connected to RPi2 through Wi-Fi, wherein RPi 1 receives packets from laptop1 and transmits the same to RPi2. In the second hop RPi2 is connected to laptop2 through ZigBee, wherein the RPi2 receives packets from RPi1 through WiFi and transmits the same to laptop2 through ZigBee. Laptop2 acts as the final receiver. The network setup is depicted in Fig 11.

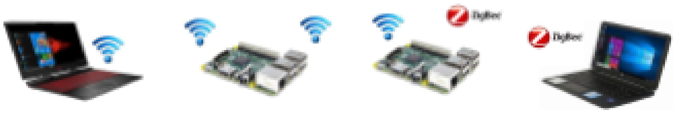

Fig 11: Two Laptops connected with one protocol conversion(WiFi-ZigBee)

3.2.4 Observations

100 packets were sent from laptop1 to laptop2 with 2 hops in between. The packets were transmitted and received using WiFi protocol at first three systems and the last transmission from RPi2 to laptop2 was through ZigBee. % packets dropped at the destination laptop are noted for 10 varying packet sizes. We can observe that that the packet loss % increases with increase in the packet size transmitted over 2 hops. The increase in the loss % is mainly attributable to transmission delay at the intermediate two nodes where the nodes take longer time to process the large size packets compared to smaller size packets. However the loss increases due to mismatch in the way the two protocols WiFi and ZigBee communicate. At RPi 2 the packets are received at much higher rate than compared to the packets transmitted from RPi2 to laptop2 due to nature of ZigBee which uses serial communication where each packet is sent Byte by Byte. The same observation can be seen clearly in Fig 12.

Packet Size (in KB)	Packets Dropped (in %)
1	27
2	49
3	66
4	71
5	77
6	81
7	83
8	85
9	86
10	88

Table 6: Packet drop percentage for one protocol conversion using WiFi-WiFi-ZB-ZB

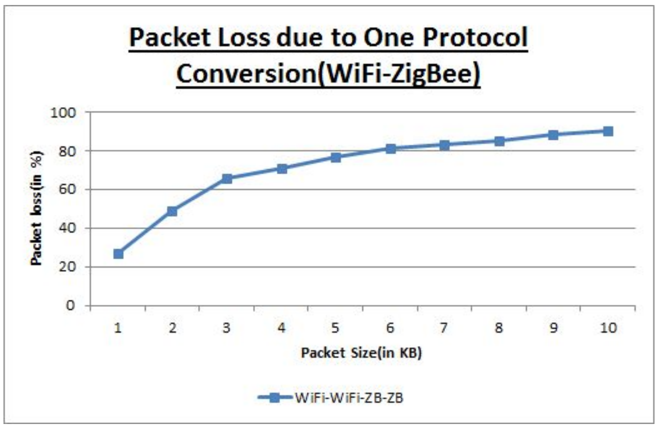

Fig 12: Packet Loss Percentage graph due to One Protocol Conversion(WiFi-ZigBee)

3.2.5 Work Done

C. One Protocol Conversion(Bluetooth)

The setup for Bluetooth communication consists of two laptops and two relay agents. The relay agent supports both Wi-Fi and Bluetooth interfaces. Laptop1 is connected to relay agent(RPi 1) through Wi-Fi and acts as sender. RPi 1 is connected to RPi2 through Wi-Fi, wherein RPi 1 receives packets from laptop1 and transmits the same to RPi2. In the second hop RPi2 is connected to laptop2 through Bluetooth, wherein the RPi2 receives packets from RPi1 through WiFi and transmits the same to laptop2 through Bluetooth. Laptop2 acts as the final receiver. The network setup is depicted in Fig 13.

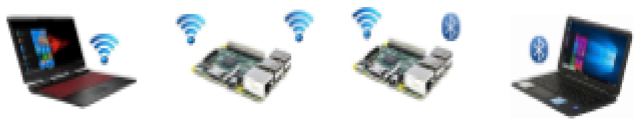

Fig 13: Two Laptops connected with one protocol conversion(WiFi-Bluetooth)

3.2.6 Observations

100 packets were sent from laptop1 to laptop2 with 2 hops in between. The packets were transmitted and received using WiFi protocol at first three systems and the last transmission from RPi2 to laptop2 was through Bluettoth. % packets dropped at the destination laptop are noted for 10 varying packet sizes. We can observe that that the packet loss % increases with increase in the packet size transmitted over 2 hops. The increase in the loss % is mainly attributable to transmission delay at the intermediate two nodes where the nodes take longer time to process the large size packets compared to smaller size packets. However the loss increases due to mismatch in the way the two protocols WiFi and Bluetooth communicate. At RPi 2 the packets are received at much higher rate than compared to the packets transmitted from RPi2 to laptop2. Irrespective of the size of packet transmitted through WiFi, Bluetooth uses constant MTU size of 256 bytes, hence the processing delay occurs at RPi2 where packets received at varying size are transmitted with MTU size of 256 Bytes after fragmentation. The same observation can be seen clearly in Fig 14.

Packet Size (in KB)	Packets Dropped (in %)
1	20
2	36
3	58
4	65
5	72
6	76
7	78
8	82
9	85
10	86

Table 7: Packet drop percentage for one protocol conversion using WiFi-WiFi-BT-BT

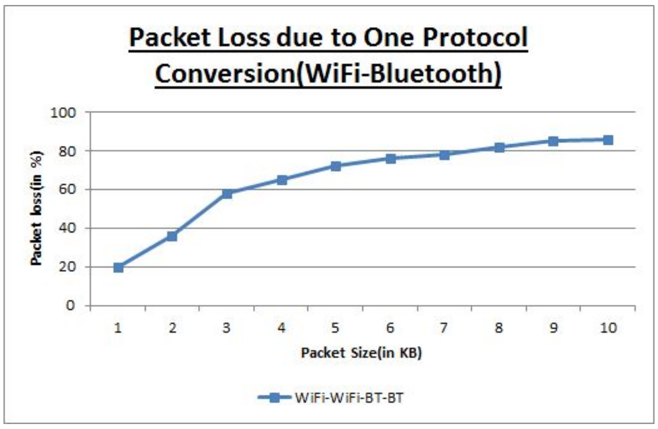

Fig 14: Packet Loss Percentage graph due to One Protocol Conversion(WiFi-BT)

3.2.7 Work Done

D. Two Protocol Conversions(Bluetooth-ZigBee)

The setup for two protocol conversion communication consists of two laptops and two relay agenst. The relay agent supports both Wi-Fi, Bluetooth and ZigBee interfaces. Laptop1 is connected to relay agent(RPi 1) through Wi-Fi and acts as sender. RPi 1 is connected to Laptop2 through Bluetooth, wherein RPi 1 receives packets from laptop1 on WiFi and transmits the same to Laptop2 on Bluetooth. In the second hop Laptop2 is connected to RPi2 through ZigBee, wherein the Laptop2 receives packets from RPi1 through Bluetooth and transmits the same to RPi2 through ZigBee. RPi2 acts as the final receiver. The network setup is depicted in Fig 15.

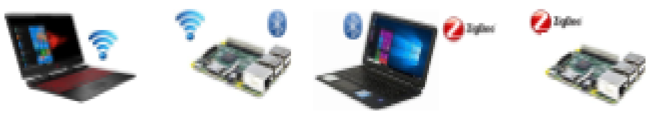

Fig 15: Two Systems connected with two protocol conversions(WiFi-Bluetooth-ZigBee)

3.2.8 *Observations*

100 packets were sent from laptop1 to RPi2 with 2 hops in between. The packets were transmitted and received using WiFi protocol at first the system and then Bluetooth at second system and then using ZigBee at third system. % of packets dropped at the destination laptop are noted for 10 varying packet sizes. We can observe that the packet loss % is 57% even for 1KB packet size keeps increasing with increase in the packet size transmitted over 2 hops. The increase in the loss % is mainly attributable to transmission delay at the intermediate two nodes where initially the packet received on WiFi needs to be transmitted over Bluetooth protocol. Here the delay caused in processing 1KB packets and sending them over Bluetooth at 256 Bytes results in packet loss. Again at third node, packets received on Bluetooth are transmitted over ZigBee to the last node resulting in further losses due to mismatch in the way the two protocols ZigBee and Bluetooth communicate. At laptop 2 the packets are received at much higher rate than compared to the packets transmitted from laptop2 to RPi2 due to nature of ZigBee which uses serial communication where each packet is sent Byte by Byte. The same observation can be seen clearly in Fig 16.

Packet Size (in KB)	Packets Dropped (in %)
1	84
2	85
3	91
4	93
5	95
6	96
7	96
8	97
9	97
10	98

Table 8: Packet drop percentage for two protocol conversion using WiFi-WiFi-BT-ZB

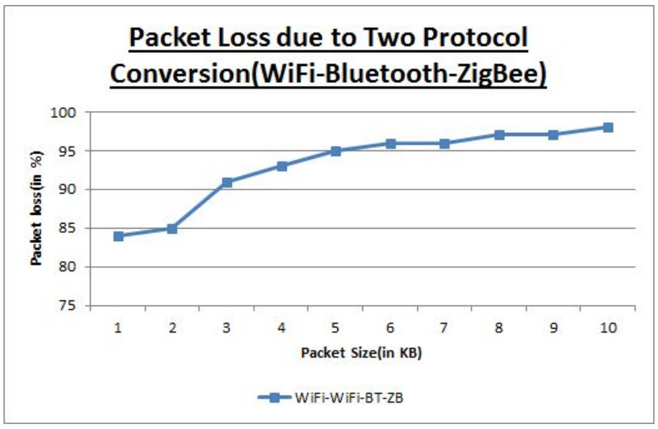

Fig 16: Packet Loss Percentage graph due to Two Protocol Conversions(WiFi-BT-ZB)

3.2.9 *Observations on comparison of multiple protocol conversion combinations*

The comparison of the packet loss percentage due to multiple conversions clearly shows that packet loss increases as the number of conversions at each hop increases. The packet loss due to zero protocol conversion WiFi-WiFi-WiFi-WiFi is the least as there is no transmission delay in protocol conversion. Packet loss due to single protocol conversion WiFi-WiFi-ZB-ZB is more than WiFi-WiFi-BT-BT as large size packets undergo more number of fragmentations in ZigBee transmission compared to Bluetooth as ZigBee being Byte Oriented and Bluetooth being burst transmission where packets are of size 256 Bytes. Loss due to two protocol conversion WiFi-WiFi-BT-ZB is the highest due to multiple fragmentation happening during protocol conversions. Since all the communication is best effort, the delay in processing the packets due to fragmentations results in packet loss.

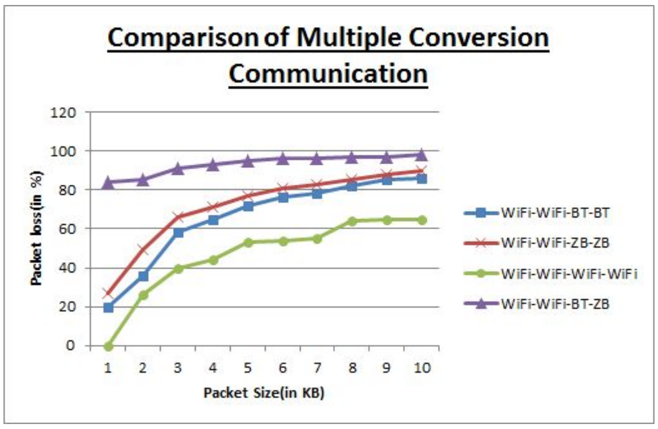

Fig 16: Packet Loss Percentage graph due to 4 different Protocol Conversion combinations

3.3 *Phase 3*

In the third phase we implemented and studied the impact of fragmentation and defragmentation during multi hop transmissions.

3.3.1 *Work Done*

A. Communication involving one Fragmentation and Defragmentation

The setup for this communication consists of two laptops and two relay agents. The relay agent supports both Wi-Fi, Bluetooth interfaces. RPi1 is connected to Laptop1 through Bluetooth and acts as sender. Laptop1 is connected to RPi2 through WiFi, wherein Laptop1 receives packets from RPi1 on Bluetooth and transmits the same to RPi2 on Wifi. In the second hop RPi2 is connected to Laptop2 through WiFi, wherein the RPi2 receives packets from Laptop1 through WiFi and transmits the same to Laptop2 through Wi-Fi. Laptop2 acts as the final receiver. The network setup is depicted in Fig 17.

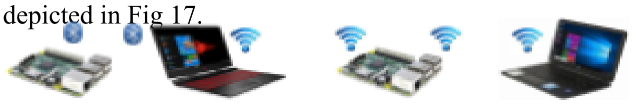

Fig 17: Network Setup for Parallel Transmission

3.3.2 Observations

100 packets are sent from RPi-1 to Laptop-2. RPi1 will send 100 Bluetooth packets to laptop1. Whatever is the size of the packets, Bluetooth fragments them into 256 bytes packets and then transmits it to Laptop1. At Laptop1 the packets are again defragmented to its original size and transmitted to RPi2 and further to Laptop2 on WiFi. We can observe considerable packet loss due to delay involved in defragmenting the Bluetooth fragments at Laptop1 (node2). Since bluetooth uses 256 bytes fragments, number of fragments multiple with increasing size of packets thus increasing the processing delay. The same can be seen in the graph at Fig 19.

3.3.3 Work Done

B. Communication involving Two Fragmentation and Defragmentation

The setup for this communication consists of two laptops and two relay agents. The relay agent supports both Wi-Fi, Bluetooth and ZigBee interfaces. RPi1 is connected to Laptop1 through Bluetooth and acts as sender. Laptop1 is connected to RPi2 through WiFi, wherein Laptop1 receives packets from RPi1 on Bluetooth and transmits the same to RPi2 on Wifi. In the second hop RPi2 is connected to Laptop2 through ZigBee, wherein the RPi2 receives packets from Laptop1 through WiFi and transmits the same to Laptop2 through ZigBee. Laptop2 acts as the final receiver. The network setup is depicted in Fig 18.

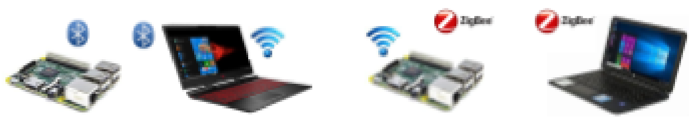

Fig 18: Network Setup for Parallel Transmission

3.3.4 Observations

100 packets are sent from RPi-1 to Laptop-2. RPi1 will send 100 Bluetooth packets to laptop1. Whatever is the size of packets, Bluetooth fragments them into 256 bytes packets and then transmits it to Laptop1. At Laptop1 the packets are again defragmented to its original size and transmitted to RPi2 on WiFi. At RPi2 the packets received from Laptop1 are again fragmented into size of 1 bytes by default as the ZigBee protocol uses serial communication to transmit the packets. At the receiver Laptop2 defragments these bytes in order to packets of size 32 bytes. We can observe considerable packet loss due to delay involved in defragmenting the Bluetooth fragments at Laptop1 (node2). Since bluetooth uses 256 bytes fragments, number of fragments multiple with increasing size of packets thus increasing the processing delay. However we cannot observe a significant rise in the losses due to additional fragmentation and defragmentation occuring at RPi2 due to ZigBee. This observation can be explained by the fact that ZigBee communication is much reliable at shorter distances and the losses are minimal as shown in Fig8. The comparison between these multiple fragmentation and defragmentation can be seen in the graph at Fig 19.

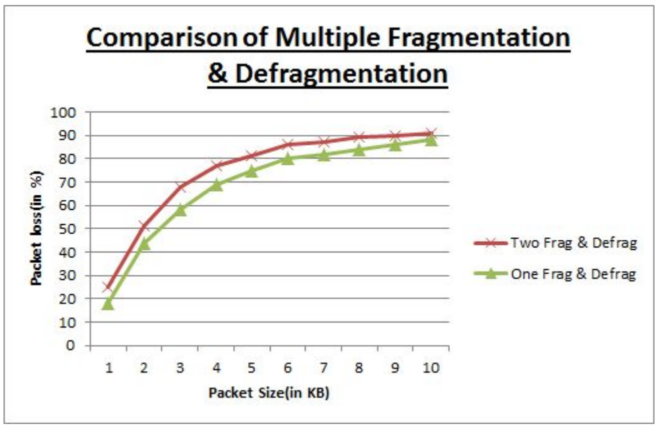

Fig 19: Packet loss % graph showing comparison of multiple fragmentation and defragmentation

IV. CONCLUSION.

- In this project three wireless technologies have been studied i.e. Wi-Fi, ZigBee and Bluetooth over multi hop relaying agent. WiFi supports long distance communication at high speed. Bluetooth technology is best suited for close-distance communication at high speed. However ZigBee provides comparatively longer distance communication with low power consumption , but at reduced speed.
- Hence depending on scenario, Bluetooth or ZigBee could be used for IOT devices. Wi-Fi technology which works on IP, can be used on relay agents supporting protocol conversion to connect to Bluetooth and ZigBee IOT devices.
- We also studied the fragmentation and defragmentation impact on all the three wireless technology. If there are more number of fragmentation/defragmentation at each protocol conversion on the relay agent then its transmission delay also increases which lead to more % loss of data.
- Repeater enhances the data transfer distance between the clients.
- Use of Protocol conversion supports interoperability of heterogeneous network.

V. ACKNOWLEDGMENT.

We wish to acknowledge the contributions of Professor Mr. C.Siva Ram Murthy, CSE, IIT Madras for his constant guidance and support. We would like to acknowledge Ms. Revathy Narayanan, PhD Scholar, CSE, IIT Madras for her continuous support, motivation and valuable discussions and suggestions. We would also like to acknowledge Mr. Ravichandran S, Technical Officer, CSE, IIT Madras for his continuous support, suggestions, motivation and providing lab infrastructure.

VI. REFERENCES

https://howtoraspberrypi.com/create-a-wi-fi-hotspot-in-less-than-10-minutes-with-pi-raspberry/

https://www.imore.com/how-get-started-using-raspberry-pi

https://people.csail.mit.edu/albert/bluez-intro/

http://people.csail.mit.edu/rudolph/Teaching/Articles/BTBook.pdf

http://www.libelium.com/development/waspmote/documentation/x-ctu-tutorial/

https://www.digi.com/resources/documentation/digidocs/pdfs/90001458-13.pdf

https://www.tecmint.com/synchronize-time-with-ntp-in-linux/

APPENDICES

A. *Phase 1 Procedure*

A.1. Best Effort Transmission using WiFi

I. Client-side:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the server information including IP address, port number.
- Fill server information in the above created structure.
- Create a buffer of some fixed size which contains message to be sent to the server.
- Use a for loop to send 100 packets i.e. above created buffer and use sendto() method to send buffer message to the server, where as sendto() method includes all the information related to the socket file descriptor, server and buffer.
- Close the socket.

II. Server-side:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the client and server information including IP address, port number.
- Fill server information in the above created structure.
- Bind the above created socket with the sockaddr_in structure type of server.
- Create a buffer of some fixed size which is used to receive the message from the client.
- Use a for loop to receive 100 packets and store in the above created buffer, here recvfrom() method is used to receive buffer message from the client, where as recvfrom() method fetches all the information related to the client and stores in the sockaddr_in structure type of client and also related to the socket and buffer.
- Close the socket.

A.2. Best Effort Transmission using Bluetooth

I. Client-side:

- Create socket file descriptor of type Bluetooth.
- Create a sockaddr_l2 structure type to store the client and server information including Bluetooth address (BD_ADDR), port number.
- Fill server information in the above created structure.
- Bind the above created socket with the sockaddr_l2 structure type of server.
- Set the Bluetooth MTU from default value to the required value using struct l2cap_opt, getsockopt() and setsockopt() library.
- Connect to the server using connect() method by passing above created socket file descriptor and server Bluetooth address.
- Send hello message to the server using write() method.
- Use a for loop to send 100 multiplied by number of fragments of messages, using write() method to send the fragmented messages to the server using established Bluetooth connection and L2CAP protocol.
- Close the socket.

II. Server-side:

- Create socket file descriptor of type Bluetooth.
- Create a sockaddr_l2 structure type to store the client and server information including Bluetooth address (BD_ADDR), port number.
- Fill server information in the above created structure.
- Bind the above created sockets with the sockaddr_l2 structure type of server.
- Set the Bluetooth MTU from default value to the required value using struct l2cap_opt, getsockopt() and setsockopt() library.
- Use listen() for any Bluetooth request to connect to the server and accept the Bluetooth connection using accept() and store the client's Bluetooth address in the sockaddr_l2 structure type variable.
- Create a buffer of some fixed size which is used to receive the hello message from the client using read() method.
- Use a for loop to receive 100 packets from client and store in the above created buffer, here read() method is used to receive Bluetooth buffer message fragments from the client whereas read() method fetches all the information related to the client and stores in the sockaddr_l2 structure type of client and also related to the socket and buffer.
- Close both the sockets.

A.3. Best Effort Transmission using ZigBee

I. Laptop-1:

- Create a termios structure, for communication on serial port. Configure port for 8 bit data, 1 stop bit, baud rate of 9600, no parity and no hardware or software flow control.
- Create a packet of size 1KB which includes alpha numeric data, store it in a buffer.
- Receive a hello message from server using read() method.
- Use a for loop to send 100 packets from client to server, add start and finish byte to buffer and forward the message to server using serial communication through write() method, where

as write() method uses descriptor, buffer and length of buffer to the server. Continue writing until all buffer bytes are forwarded.

II. Server-side:

- Create a termios structure, for communication on serial port. Configure port for 8 bit data, 1 stop bit, no parity, no hardware and software flow control and baud rate of 9600.
- Create a buffer of some fixed size which contains message to be received from the client.
- Send a hello message to the client using write() method.
- Use a do while loop to receive packets and store in the above created buffer,
- read() method is used to receive buffer message from the client. When start byte is received, keep saving bytes until finish byte is received..
- Calculate count of received bytes and display the summary.

B. Phase 2 Procedure

B.1. Zero protocol conversion

I. Laptop1-side:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the laptop1 and RPi1 information including IP address, port number.
- Fill laptop1 information in the above created structure.
- Create a buffer which contains message of 1KB size to be sent to the RPi1 over UDP protocol.
- Use a for loop to send 100 packets i.e. above created buffer and use sendto() method to send buffer message to the RPi1, whereas sendto() method includes all the information related to the socket file descriptor, RPi1 and buffer.
- Close the socket.

II. RPi-1/2:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the laptop1/RPi1, RPi1/RPi2 and RPi2/laptop2 information including IP address, port number.
- Fill RPi1/RPi2 information in the above created structure.
- Bind the above created socket with the sockaddr_in structure type of RPi1/RPi2.
- Create a buffer of some fixed size which is used to receive and forward the message from laptop1 to RPi2/laptop2.
- Receive a hello message from RPi2/laptop2 using recvfrom() method, where recvfrom() method fetches all the information related to the RPi2/laptop2 and stores it in the structure type of RPi2/laptop2.
- Use a for loop to forward 100 packets from laptop1/RPi1 to RPi2/laptop2 by temporarily storing in the above created buffer, here recvfrom() method is used to receive buffer message from the laptop1/RPi1, whereas recvfrom() method fetches all the information related to the laptop1/RPi1 and stores in the sockaddr_in structure type of laptop1/RPi1, buffer and forwards the message to RPi2/laptop2 using sendto() method, whereas sendto() method uses socket, buffer and sockaddr_in structure type of RPi2/laptop2 stored above to send the buffer message to the RPi2/laptop2.
- Close the socket.

III. Laptop-2 side:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the RPi2 information including IP address, port number.
- Fill RPi2 information in the above created structure.
- Create a buffer of some fixed size which contains message to be received from the RPi2.
- Send a hello message to the RPi2 using sendto() method, which uses socket, buffer and sockaddr_in structure type of RPi2.
- Use a for loop to receive 100 packets and store in the above created buffer, here recvfrom() method is used to receive buffer message from the RPi2, whereas recvfrom() method fetches all the information related to the RPi2 and stores in the sockaddr_in structure type of RPi2 and also related to the socket and buffer.
- Close the socket.

B.2. One Protocol Conversion(ZigBee)

I. Laptop1-side:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the laptop1 and RPi1 information including IP address, port number.
- Fill laptop1 information in the above created structure.
- Create a buffer which contains message of 1KB size to be sent to the RPi1 over UDP protocol.
- Use a for loop to send 100 packets i.e. above created buffer and use sendto() method to send buffer message to the RPi1, whereas sendto() method includes all the information related to the socket file descriptor, RPi1 and buffer.
- Close the socket.

II. RPi-1:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the laptop1, RPi1 and RPi2 information including IP address, port number.
- Fill RPi1 information in the above created structure.
- Bind the above created socket with the sockaddr_in structure type of RPi1.

- Create a buffer of some fixed size which is used to receive and forward the message from laptop1 to RPi2.
- Receive a hello message from RPi2 using recvfrom() method, where recvfrom() method fetches all the information related to the RPi2 and stores it in the structure type of RPi2.
- Use a for loop to forward 100 packets from laptop1 to RPi2 by temporarily storing in the above created buffer, here recvfrom() method is used to receive buffer message from the laptop1, whereas recvfrom() method fetches all the information related to the laptop1 and stores in the sockaddr_in structure type of laptop1, buffer and forwards the message to RPi2 using sendto() method, whereas sendto() method uses socket, buffer and sockaddr_in structure type of RPi2 stored above to send the buffer message to the RPi2.
- Close the socket.

III. RPi-2:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the RPi1 and RPi2 information including IP address, port number.
- Fill RPi2 information in the above created structure.
- Bind the above created socket with the sockaddr_in structure type of RPi2.
- Create a buffer of some fixed size which is used to receive and forward the message from RPi1 to laptop2.
- Create a termios structure, for communication on serial port. Configure port for 8 bit data, 1 stop bit, no parity, no hardware and software flow control and baud rate of 9600.
- Receive a hello message from laptop2 using read() method.
- Use a do while loop to receive 100 packets from Rpi1 using recvfrom() method over Wi-Fi protocol and store in the above created buffer, add start and finish byte to buffer and forward the message to laptop2 using serial communication through write() method, where as write() method uses descriptor, buffer and length of buffer to thelaptop2. Continue writing until all buffer bytes are forwarded.
- Calculate count of received bytes and display summary.

IV. laptop2-side:

- Create a termios structure, for communication on serial port. Configure port for 8 bit data, 1 stop bit, no parity, no hardware and software flow control and baud rate of 9600.
- Create a buffer of some fixed size which contains message to be received from the RPi2.
- Send a hello message to the RPi2 using write() method.
- Use a do while loop to receive packets and store in the above created buffer,
- read() method is used to receive buffer message from the RPi2. When start byte is received, keep saving bytes until finish byte is received.
- Calculate count of received bytes and display summary.

B.3. One Protocol Conversion(Bluetooth)

I. Laptop1-side:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the laptop1 and RPi1 information including IP address, port number.
- Fill laptop1 information in the above created structure.
- Create a buffer which contains message of 1KB size to be sent to the RPi1 over UDP protocol.
- Use a for loop to send 100 packets i.e. above created buffer and use sendto() method to send buffer message to the RPi1, whereas sendto() method includes all the information related to the socket file descriptor, RPi1 and buffer.
- Close the socket.

II. RPi-1:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the laptop1, RPi1 and RPi2 information including IP address, port number.
- Fill RPi1 information in the above created structure.
- Bind the above created socket with the sockaddr_in structure type of RPi1.
- Create a buffer of some fixed size which is used to receive and forward the message from laptop1 to RPi2.
- Receive a hello message from RPi2 using recvfrom() method, where recvfrom() method fetches all the information related to the RPi2 and stores it in the structure type of RPi2.
- Use a for loop to forward 100 packets from laptop1 to RPi2 by temporarily storing in the above created buffer, here recvfrom() method is used to receive buffer message from the laptop1, whereas recvfrom() method fetches all the information related to the laptop1 and stores in the sockaddr_in structure type of laptop1, buffer and forwards the message to RPi2 using sendto() method, whereas sendto() method uses socket, buffer and sockaddr_in structure type of RPi2 stored above to send the buffer message to the RPi2.
- Close the socket.

III. RPi-2:

- Create socket file descriptor of type datagram and Bluetooth.
- Create a sockaddr_in structure type to store the RPi1 and RPi2 information including IP address, port number.
- Fill RPi2 information in the above created structure.

- Bind the above created socket with the sockaddr_in structure type of RPi2.
- Create a sockaddr_l2 structure type to store the RPi2 and laptop2 information including Bluetooth address (BD_ADDR), port number.
- Fill laptop2 information in the above created structure.
- Bind the above created socket with the sockaddr_l2 structure type of laptop2.
- Set the Bluetooth MTU from default value to the required value using struct l2cap_opt, getsockopt() and setsockopt() library.
- Connect to the laptop2 using connect() method by passing above created socket file descriptor and laptop2 Bluetooth address.
- Send hello message to the RPi1 using sendto() method.
- Send hello message to the laptop2 using write() method.
- Create a buffer of some fixed size which is used to receive and forward the message from RPi1 to laptop2.
- Use a do while loop to receive 100 packets from Rpi1 using recvfrom() method over Wi-Fi protocol and store in the above created buffer, forward the message to laptop2 using l2cap over bluetooth protocol with write() method, where as write() method uses descriptor, buffer and length of buffer to the laptop2.
- Close the socket.

IV. Laptop2-side:
- Create socket file descriptor of type Bluetooth.
- Create a sockaddr_l2 structure type to store the RPi2 and laptop2 information including Bluetooth address (BD_ADDR), port number.
- Fill laptop2 information in the above created structure.
- Bind the above created sockets with the sockaddr_l2 structure type of RPi1.
- Set the Bluetooth MTU from default value to the required value using struct l2cap_opt, getsockopt() and setsockopt() library.
- Use listen() for any Bluetooth request to connect to the laptop2 and accept the Bluetooth connection using accept() and store the RPi2's Bluetooth address in the sockaddr_l2 structure type variable.
- Create a buffer of some fixed size which is used to receive the hello message from the RPi2 using read() method.
- Use a for loop to receive 100 packets from RPi2 and store in the above created buffer, here read() method is used to receive Bluetooth buffer message fragments from the RPi2 where as read() method fetches all the information related to the RPi2 and stores in the sockaddr_l2 structure type of RPi2 and also related to the socket and buffer.
- Close both the sockets.

B.4. Two Protocol Conversions(Bluetooth-ZigBee)

I. Laptop1-side:
- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the laptop1 and RPi1 information including IP address, port number.
- Fill RPi1 information in the above created structure.
- Create a buffer which contains message of 1KB size to be sent to the RPi1 over UDP protocol.
- Use a for loop to send 100 packets i.e. above created buffer and use sendto() method to send buffer message to the RPi1, whereas sendto() method includes all the information related to the socket file descriptor, RPi1 and buffer.
- Close the socket.

II. RPi-1:
- Create socket file descriptor of type datagram and Bluetooth.
- Create a sockaddr_in structure type to store the laptop1 and RPi1 information including IP address, port number.
- Fill RPi1 information in the above created structure.
- Bind the above created socket with the sockaddr_in structure type of RPi1.
- Create a sockaddr_l2 structure type to store the RPi1 and RPi2 information including Bluetooth address (BD_ADDR), port number.
- Fill RPi2 information in the above created structure.
- Bind the above created socket with the sockaddr_l2 structure type of RPi2.
- Set the Bluetooth MTU from default value to the required value using struct l2cap_opt, getsockopt() and setsockopt() library.
- Connect to the RPi2 using connect() method by passing above created socket file descriptor and RPi2 Bluetooth address.
- Send hello message to the laptop1 using sendto() method.
- Send hello message to the RPi2 using write() method.
- Create a buffer of some fixed size which is used to receive and forward the message from laptop1 to RPi2.
- Use a do while loop to receive 100 packets from laptop1 using recvfrom() method over Wi-Fi protocol and store in the above created buffer, forward the message to RPi2 using l2cap over bluetooth protocol with write() method, where as write() method uses descriptor, buffer and length of buffer to the RPi2.
- Close the socket.

III. RPi-2:
- Create socket file descriptor of type Bluetooth.
- Create a sockaddr_l2 structure type to store the RPi2 and laptop2 information including Bluetooth address (BD_ADDR), port number.

- Fill laptop2 information in the above created structure.
- Bind the above created socket with the sockaddr_l2 structure type of laptop2.
- Set the Bluetooth MTU from default value to the required value using struct l2cap_opt, getsockopt() and setsockopt() library.
- Create a termios structure, for communication on serial port. Configure port for 8 bit data, 1 stop bit, no parity, no hardware and software flow control and baud rate of 9600.
- Use listen() for any Bluetooth request to connect to the laptop2 and accept the Bluetooth connection using accept() and store the RPi2's Bluetooth address in the sockaddr_l2 structure type variable.
- Send hello message to the laptop2 using write() method.
- Create a buffer of some fixed size which is used to receive the hello message from the RPi2 using read() method.
- Use a do while loop to receive 100 packets multiplied by number of fragments from Rpi2 using read() method over l2cap using bluetooth protocol where as read() method fetches all the information related to the RPi1 and store in the above created buffer, add start and finish byte to buffer and forward the message to laptop2 using serial communication through write() method, where as write() method uses descriptor, buffer and length of buffer to the laptop2. Continue writing until all buffer bytes are forwarded.
- Close the socket.

IV. laptop2-side:

- Create a termios structure, for communication on serial port. Configure port for 8 bit data, 1 stop bit, no parity, no hardware and software flow control and baud rate of 9600.
- Create a buffer of some fixed size which contains message to be received from the RPi2.
- Send a hello message to the RPi2 using write() method.
- Use a do while loop to receive packets and store in the above created buffer, read() method is used to receive buffer message from the RPi2. When start byte is received, keep saving bytes until finish byte is received.
- Calculate count of received bytes and display summary.

C. Phase 3 Procedure

C.1. Communication involving one Fragmentation and Defragmentation

I. Laptop1-side:

- Create socket file descriptor of type Bluetooth.
- Create a sockaddr_l2 structure type to store the laptop1 and RPi1 information including Bluetooth address (BD_ADDR), port number.
- Fill RPi1 information in the above created structure.
- Bind the above created socket with the sockaddr_l2 structure type of RPi1.
- Set the Bluetooth MTU from default value to the required value using struct l2cap_opt, getsockopt() and setsockopt() library by doing this we fragment the actual packet into MTU size chunks.
- Create a buffer which contains message of 1KB size which consist of alphanumeric characters.
- Connect to the RPi1 using connect() method by passing above created socket file descriptor and RPi1 Bluetooth address.
- Send hello message to the RPi1 using write() method.
- Use a for loop to send 100 multiplied by number of fragments of messages, using write() method to send the fragmented messages to the RPi1 using established Bluetooth connection and L2CAP protocol.
- Close the socket.

II. RPi-1:

- Create socket file descriptor of type datagram and Bluetooth.
- Create a sockaddr_in structure type to store the RPi1 and RPi2 information including IP address, port number.
- Fill RPi2 information in the above created structure.
- Bind the above created socket with the sockaddr_in structure type of RPi1.
- Create a sockaddr_l2 structure type to store the laptop1 and RPi1 information including Bluetooth address (BD_ADDR), port number.
- Fill RPi1 information in the above created structure.
- Bind the above created socket with the sockaddr_l2 structure type of RPi1.
- Set the Bluetooth MTU from default value to the required value using struct l2cap_opt, getsockopt() and setsockopt() library.
- Use listen() for any Bluetooth request to connect to the RPi1 and accept the Bluetooth connection using accept() and store the RPi1's Bluetooth address in the sockaddr_l2 structure type variable.
- Receive a hello message from RPi2 using recvfrom() method, where recvfrom() method fetches all the information related to the RPi2 and stores it in the structure type of RPi2.
- Use a do while loop to receive 100 packets multiplied by number of fragments from laptop1 using read() method using l2cap over bluetooth protocol and store in the above created buffer, here defragmentation is done according to the desired packet size and forward the message to RPi2 over Wifi protocol with sendto() method, whereas

sendto() method uses descriptor, buffer and length of buffer to the RPi2.

III. RPi-2:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the RPi1, RPi2 and laptop2 information including IP address, port number.
- Fill RPi2 information in the above created structure.
- Bind the above created socket with the sockaddr_in structure type of RPi2.
- Create a buffer of some fixed size which is used to receive and forward the message from RPi1 to laptop2.
- Receive a hello message from laptop2 using recvfrom() method, where recvfrom() method fetches all the information related to the laptop2 and stores it in the structure type of laptop2.
- Use a do while loop to forward 100 packets from RPi1 to laptop2 by temporarily storing in the above created buffer, here recvfrom() method is used to receive buffer message from the RPi1, whereas recvfrom() method fetches all the information related to the RPi1 and stores in the sockaddr_in structure type of RPi1, buffer and forwards the message to laptop2 using sendto() method, whereas sendto() method uses socket, buffer and sockaddr_in structure type of laptop2 stored above to send the buffer message to the laptop2.
- Close the socket.

IV. Laptop-2 side:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the RPi2 and laptop2 information including IP address, port number.
- Fill RPi2 information in the above created structure.
- Create a buffer of some fixed size which contains message to be received from the RPi2.
- Send a hello message to the RPi2 using sendto() method, which uses socket, buffer and sockaddr_in structure type of RPi2.
- Use a do while loop to receive 100 packets and store in the above created buffer, here recvfrom() method is used to receive buffer message from the RPi2, whereas recvfrom() method fetches all the information related to the RPi2 and stores in the sockaddr_in structure type of RPi2 and also related to the socket and buffer.
- Close the socket.

C.2. Communication involving Two Fragmentation and Defragmentation

I. Laptop1-side:

- Create socket file descriptor of type Bluetooth.
- Create a sockaddr_l2 structure type to store the client and RPi1 information including Bluetooth address (BD_ADDR), port number.
- Fill RPi1 information in the above created structure.
- Bind the above created socket with the sockaddr_l2 structure type of RPi1.
- Set the Bluetooth MTU from default value to the required value using struct l2cap_opt, getsockopt() and setsockopt() library by doing this we fragment the actual packet into MTU size chunks.
- Create a buffer which contains message of 1KB size which consist of alphanumeric characters.
- Connect to the RPi1 using connect() method by passing above created socket file descriptor and RPi1 Bluetooth address.
- Send hello message to the RPi1 using write() method.
- Use a for loop to send 100 multiplied by number of fragments of messages, using write() method to send the fragmented messages to the RPi1 using established Bluetooth connection and L2CAP protocol.
- Close the socket.

II. RPi-1:

- Create socket file descriptor of type datagram and Bluetooth.
- Create a sockaddr_in structure type to store the RPi1 and RPi2 information including IP address, port number.
- Fill RPi2 information in the above created structure.
- Bind the above created socket with the sockaddr_in structure type of RPi1.
- Create a sockaddr_l2 structure type to store the laptop1 and RPi1 information including Bluetooth address (BD_ADDR), port number.
- Fill RPi1 information in the above created structure.
- Bind the above created socket with the sockaddr_l2 structure type of RPi1.
- Set the Bluetooth MTU from default value to the required value using struct l2cap_opt, getsockopt() and setsockopt() library.
- Use listen() for any Bluetooth request to connect to the RPi1 and accept the Bluetooth connection using accept() and store the RPi1's Bluetooth address in the sockaddr_l2 structure type variable.
- Receive a hello message from RPi2 using recvfrom() method, where recvfrom() method fetches all the information related to the RPi2 and stores it in the structure type of RPi2.
- Use a do while loop to receive 100 packets multiplied by number of fragments from laptop1 using read() method using l2cap over bluetooth protocol and store in the above created buffer, here defragmentation is done

according to the desired packet size and forward the message to RPi2 over Wifi protocol with sendto() method, whereas sendto() method uses descriptor, buffer and length of buffer to the RPi2.

III. RPi-2:

- Create socket file descriptor of datagram type.
- Create a sockaddr_in structure type to store the RPi1 and RPi2 information including IP address, port number.
- Fill RPi2 information in the above created structure.
- Bind the above created socket with the sockaddr_in structure type of RPi2.
- Create a buffer of some fixed size which is used to receive and forward the message from RPi1 to laptop2.
- Create a termios structure, for communication on serial port. Configure port for 8 bit data, 1 stop bit, no parity, no hardware and software flow control and baud rate of 9600.
- Receive a hello message from laptop2 using read() method.
- Use a do while loop to receive 100 packets from Rpi1 using recvfrom() method over Wi-Fi protocol and store in the above created buffer, add start and finish byte to buffer and forward the message to laptop2 using serial communication through write() method, where as write() method uses descriptor, buffer and length of buffer to the laptop2. Continue writing until all buffer bytes are forwarded.
- Calculate count of received bytes and display summary.

IV. laptop2-side:

- Create a termios structure, for communication on serial port. Configure port for 8 bit data, 1 stop bit, no parity, no hardware and software flow control and baud rate of 9600.
- Create a buffer of some fixed size which contains message to be received from the RPi2.
- Send a hello message to the RPi2 using write() method.
- Use a do while loop to receive packets and store in the above created buffer, read() method is used to receive buffer message from the RPi2. When start byte is received, keep saving bytes until finish byte is received.
- Calculate count of received bytes and display summary.