STUDYING THE IMPACT OF PROTOCOL CONVERSIONS

Sahil Mahajan
M Tech 1st year
IIT MADRAS
Chennai,India
cs18m001@smail.iitm.ac.in

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

Ankit Goyal
M Tech 1st year
IIT MADRAS
Chennai,India
cs18m010@smail.iitm.ac.in

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

Sunku Bala Bharat
M Tech 1st year
IIT MADRAS
Chennai,India
cs18m054@smail.iitm.ac.in

***Abstract* — The aim of this project is to study the impact of protocol conversion in a multi-technology IoT system. This is achieved by implementing multiple ways of protocol conversion on a practical and commonly used multi-technology IoT platform- a 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 a relay gateway 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 relaying 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 delay with respect to both the protocols.

A. Phases of the project

The project has been implemented in the following three phases:

- In first phase we implemented a Wi-Fi packet based relay at RPi to communicate between two laptops.
- In second phase we implemented and studied separately the impact of protocol conversion (Wi-Fi to Bluetooth and Wi-Fi to Zigbee) through a Raspberry Pi acting as the relay gateway.
- In the third phase we implemented and studied the impact of protocol conversion (Wi-Fi to Bluetooth and Wi-Fi to Zigbee) through a Raspberry Pi acting as the relay gateway on parallel transmissions.

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 Raspberry Pi to act as a Wi-Fi repeater. Accordingly, Raspberry Pi was configured as Wi-Fi hotspot. Once Wi-Fi hotspot is configured Raspberry Pi is ready to act as repeater. Now client-1 will send data to client-2 through Raspberry Pi as a repeater.

B. Bluetooth

In-built Bluetooth support in Raspberry Pi and laptop is used for Bluetooth 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:

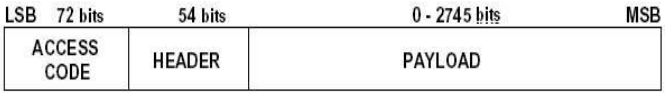

Fig 1: Bluetooth packet format

A packet could contain a shorthanded access code (68 bytes) 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

E. NTP time sync

Since there is requirement of measuring the delay caused due to protocol conversion, there is need of synchronizing client1,client2 and the server(Raspberry Pi) to a central time

server. This was achieved by installing ntpdate and synchronizing the system time to the server from pool.ntp.br.
The commands used to sync are :

```
# dpkg-reconfigure tzdata        // set up of correct time zone
# apt-get install ntpdate        // install ntpdate
# ntpdate  -v pool.ntp.br
```

III. PROCEDURE AND STEPS FOLLOWED

3.1 Phase 1

The First Phase involved understanding of WiFi technology and study of using WiFi based repeater for interaction between two clients.

3.1.1 Network setup

The setup for direct communication between IOT devices consists of two laptops connected through Wi-Fi as depicted in Fig 3. The network setup for RPi based relaying consists of two laptops acting as IOT devices and RPi acting as Wi-Fi based relaying gateway interconnecting two IOT devices as depicted in Fig 4.

Fig 3: Two Laptops directly connected

Fig 4: Two Laptops through RPi Gateway

3.1.2 Work Done

A. Transmission of Wi-Fi packets without repeater

The first setup of this phase involved direct communication between IOT devices consists of two laptops connected through Wi-Fi. One laptop is configured as Wi-Fi hotspot to act as a server. Other laptop acts as client and sends 100 packets to the server without Wi-Fi repeater. The above transmission process is implemented using socket programming and the steps followed are detailed in Appendix A.1

B. Transmission of Wi-Fi packets through repeater

The second setup of this phase is for RPi based relaying consists of two laptops acting as IOT devices and RPi acting as Wi-Fi based relaying gateway interconnecting two IOT devices. One laptop acting as client-1 will send 100 packets to another laptop (client-2) through Raspberry Pi (Wi-Fi repeater). First client-2 will send a “hello” message to server (Raspberry Pi), so that server knows the address of client2. Now packet transfer from client1 to client2 can be initialized. The above transmission process is implemented using socket programming and the steps are described in Appendix A.2

3.1.3 Observations

100 packets were sent from client to server without using Raspberry Pi repeater, number of packets dropped and received Wi-Fi signal strength at the client side are noted for 10 varying distances . Similarly, packet drop and received Wi-Fi signal strength is recorded after introducing Raspberry Pi as a Wi-Fi repeater in between client1 and client2 as shown in below two tables.

Distance between laptops (in m)	Signal Strength Without Repeater (in dBm)	Signal Strength With Repeater (in dBm)
1	-49	-44
2	-47	-44
4	-74	-49.5
6	-84	-61.5
8	-80	-69
10	-80	-72.5
12	-92	-72.5
14	-94	-72
16	-94	-77
18	-86	-77.5
20	-96	-78

Table 2: Signal strength comparison

Distance between laptops (in m)	Number of Packets Dropped Without Repeater	Number of Packets Dropped With Repeater
1	0	0
2	0	0
4	0	0
6	1	0
8	2	0
10	5	2
12	7	4

14	29	4
16	29	5
18	30	5
20	32	6

Table 3: Packet drop comparison

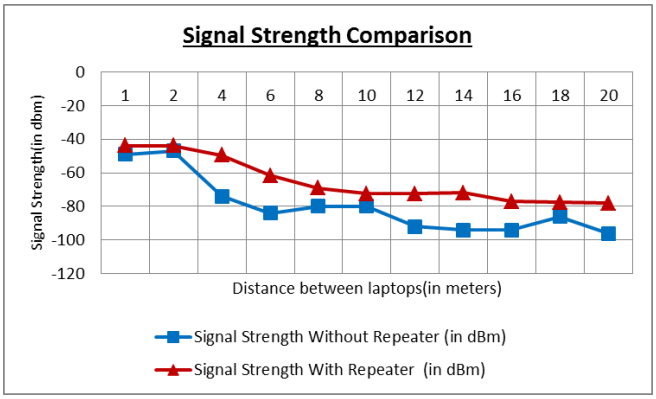

Fig 5: Signal Strength comparison graph

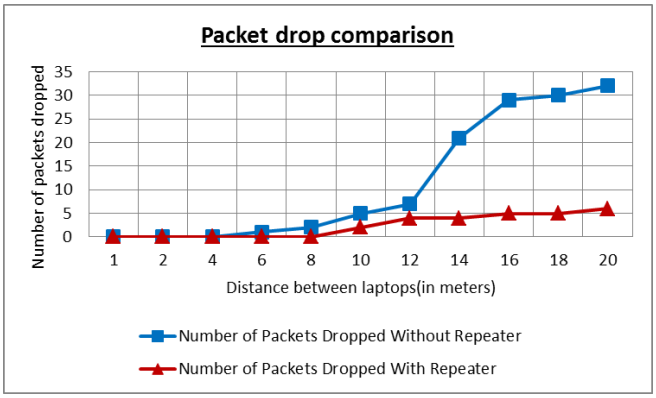

Fig 6: Packet drop comparison graph

3.2 Phase 2

The task of Second Phase 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 data loss in case of Wi-Fi to Zigbee conversion and delay with increasing distance in case of Wi-Fi to Bluetooth conversion.

3.2.1 Network setup

Bluetooth communication: The setup for Bluetooth communication consists of two laptops and relay agent (RPi). The relay agent supports both Wi-Fi and Bluetooth interfaces. One of the client is connected to relay agent through Wi-Fi and acts as sender. The other client is connected to relay agent through Bluetooth and acts as receiver. The network setup is depicted in Fig 7.

Zigbee Communication: The setup for Zigbee communication consists of two laptops and relay agent. The relay agent supports both Wi-Fi and Zigbee interfaces. One of the client is connected to relay agent through Wi-Fi and acts as sender. The other client is connected to relay agent through Zigbee and acts as receiver. The network setup is depicted in Fig 8.

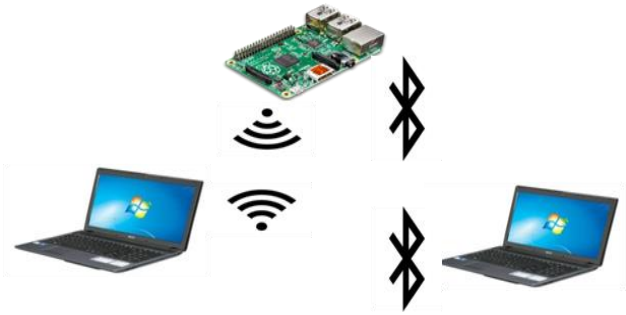

Fig 7: Bluetooth Communication setup

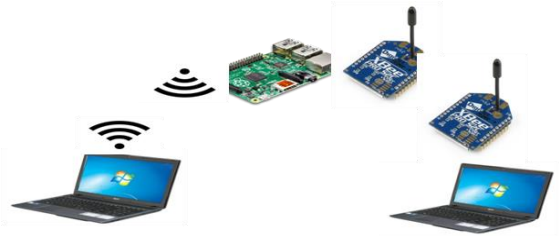

Fig 8: Zigbee Communication Setup

3.2.2 Work Done

A. Transmission of Wi-Fi packets after relaying and getting converted at the server using Bluetooth protocol

One laptop acting as client-1 will send 100 Wi-Fi packets using UDP protocol to the Raspberry Pi (server) which acts as protocol converter using Bluetooth inbuilt interfaces and forwards the converted fragments to the other laptop which acts as client-2.

First client-2 will send a "hello" message to the server (Raspberry Pi) using Bluetooth protocol, so that server knows the address of client2. Now packet transfer from client1 to client2 can be initialized. This transmission process is implemented using socket programming and steps followed are detailed in Appendix B.1

B. Transmission of Wi-Fi packets after relaying and getting converted at the server using Zigbee protocol

One laptop acting as client-1 will send 100 packets to another laptop (client-2) through Raspberry Pi (repeater).

The above transmission process is implemented using socket programming and serial communication and the steps followed are detailed in Appendix B.2

3.2.3 Observations

Bluetooth communication: 100 packets are sent from client-1 to client-2. Client-1 will send 100 Wi-Fi packets to another laptop (client-2) through Raspberry Pi (Wi-Fi-Bluetooth repeater) where packets have been fragmented based on MTU of bluetooth. Table 4 and Fig 9, depicts the delay of protocol conversion from Wi-Fi to bluetooth and propagation on Bluetooth with respect to MTU of Bluetooth.

MTU of Bluetooth (in bytes)	Total Delay (in msec)
32	19257.71
64	16750.25
128	8178.18
256	5185.91
512	3444.76
1024	3378.62

Table 4: Packet Delay Server to Client2 (Bluetooth Conversion)

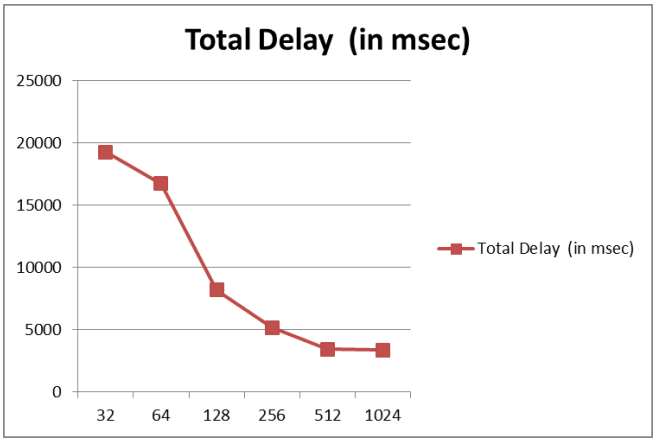

Fig 9: Conversion delay graph Client1 to Client2 (Bluetooth Conversion)

Zigbee Communication: 100 packets are sent from client-1 to client-2. Client-1 will send 100 packets to another laptop (client-2) through Raspberry Pi (Wi-Fi-Zigbee repeater). Table 5 and Fig 10, depicts the delay at various distances.

Distance between Server and Client2(in metre)	Delay (in ms)
1	109229
2	113229
3	115225
4	120459
5	124068
6	127543
7	128161
8	130998
9	131223
10	132434
12	135665
14	138440
16	139776
18	140243
20	142654

Table 5: Delay Server to Client2 (Zigbee Conversion)

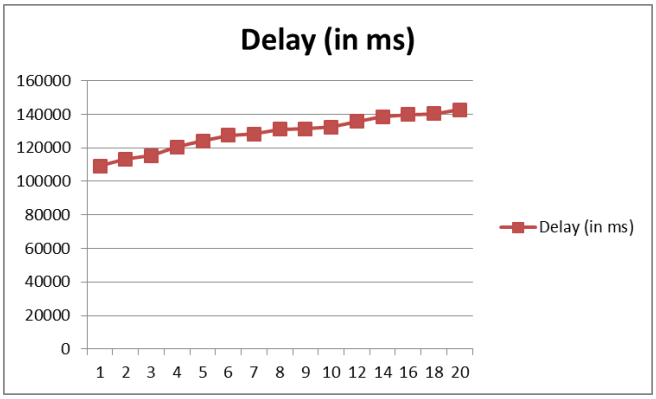

Fig 10: Delay server to Client2 (Zigbee Conversion)

3.3 Phase 3

In the third phase we implemented and studied the impact of protocol conversion (Wi-Fi to Bluetooth and Wi-Fi to Zigbee) through a Raspberry Pi acting as the relay gateway on parallel transmissions.

3.3.1 Network setup

The setup consists of two laptops and relay agent (RPi). The relay agent supports Wi-Fi, Bluetooth and Zigbee interfaces. One of the client is connected to relay agent through Wi-Fi and acts as sender. The other client is connected to relay agent through both Bluetooth and Zigbee and acts as receiver. At the receiver two processes are running one to receive Bluetooth packets and other to receive Zigbee packets.

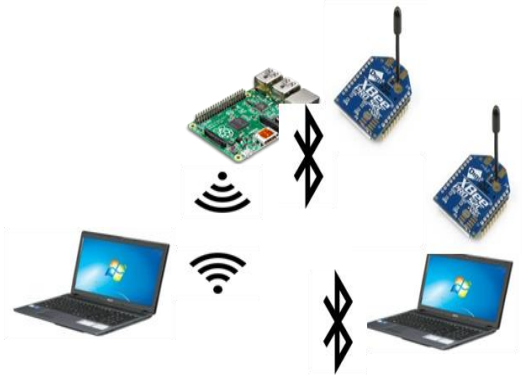

Fig 11: Network Setup for Parallel Transmission

3.3.2 Work Done

One laptop acting as client-1 will send 100 packets to another laptop (client-2) through Raspberry Pi (repeater). The Rpi Repeater forwards packets to client-2 over both Bluetooth and Zigbee tested under various ratio. Client-2 has two processes, one listening on Bluetooth and other on Zigbee. The above transmission process is implemented using socket programming and serial communication and the steps followed are detailed in Appendix C.

3.3.3 *Observations*

100 packets are sent from client-1 to client-2. Client-1 will send 100 Wi-Fi packets to another laptop (client-2) through Raspberry Pi under various ratios for Bluetooth and Zigbee Packets. For Bluetooth, packets have been fragmented based on MTU of bluetooth. For Zigbee header and footer has been added to data. Table 6 and Fig 12, depicts the delay of protocol conversion and propagation from Wi-Fi to Bluetooth and Zigbee under various ratios of Bluetooth-Zigbee Packets.

Bluetooth-Zigbee Ratio	Delay(msec)
100-0	2909
75-25	28964
50-50	51099
25-75	72812
0-100	104814

Table 6: Delay for Bluetooth-Zigbee Ratio

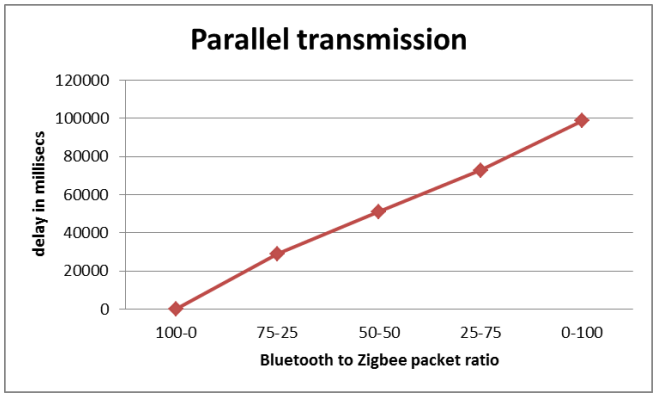

Fig 12: Delay for Bluetooth-Zigbee Ratio

IV. CONCLUSION.

- In this project three wireless technologies have been studied i.e. Wi-Fi, Zigbee and Bluetooth. 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.
- Repeater enhances the data transfer distance between the clients.
- Use of Protocol conversion supports inter operability 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 also like to acknowledge Ms. Revathy Narayanan, PhD Scholar, CSE, IIT Madras for her continuous support, motivation and valuable discussions and suggestions.

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. Without repeater.

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 infromation 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. With Repeater

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, whereas 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 client1, server and client2 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 and forward the message from client1 to client2.
- Receive a hello message from client2 using recvfrom() method, where recvfrom() method fetches all the information related to the client2 and stores it in the structure type of client2 .
- Use a for loop to forward 100 packets from client1 to client2 by temporarily storing in the above created buffer, here recvfrom() method is used to receive buffer message from the client1, whereas recvfrom() method fetches all the information related to the client1 and stores in the sockaddr_in structure type of client1, buffer and forwards the message to client2 using sendto() method, whereas sendto() method uses socket, buffer and sockaddr_in structure type of client2 stored above to send the buffer message to the client2.
- Close the socket.

III. Client2-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 received from the server.
- Send a hello message to the server using sendto() method, which uses socket, buffer and sockaddr_in structure type of server.
- 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 server, whereas recvfrom() method fetches all the information related to the server and stores in the sockaddr_in structure type of server and also related to the socket and buffer.
- Close the socket.

B. Phase 2 Procedure

B.1. Bluetooth

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 which contains message of 1KB size to be sent to the server 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 server, whereas sendto() method includes all the information related to the socket file descriptor, server and buffer.
- Close the socket.

II. Server-side:

- Create two socket file descriptor of type datagram and Bluetooth.
- Create a sockaddr_in structure type to store the client and server information including IP address, port number.
- 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_in structure type and 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-2'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-2 using read() method.
- Use a for loop to receive 100 packets from client-1 and store in the above created buffer, here recvfrom() method is used to receive Wi-Fi buffer message from the client1, whereas recvfrom() method fetches all the information related to the client1 and stores in the sockaddr_in structure type of client1 and also related to the socket and buffer.
- Create message fragments according to the Wi-Fi packet size and specified Bluetooth MTU size.
- Use write() method to send the fragmented messages to the client-2 using established Bluetooth connection and L2CAP protocol.
- Close both the sockets.

III. Client-2 side:

- Create socket file descriptor of Bluetooth type.
- Create a sockaddr_l2 structure type to store the server information including Bluetooth address (BD_ADDR), port number.
- Fill server information in the above created structure.
- 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 receive 100 multiplied by number of fragments of messages using read() method from server through established Bluetooth connection and L2CAP protocol.
- Close the socket.

B.2. Zigbee

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 which contains message of 1KB size to be sent to the server 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 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 client1, server and client2 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 and forward the message from client1 to client2.
- 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.
- Use a for loop to forward 100 packets from client1 to client2 by temporarily storing in the above created buffer, here recvfrom() method is used to receive buffer message from the client1, where as recvfrom() method fetches all the information related to the client1 and stores in the sockaddr_in structure type of client1, buffer.
- Add start and finish byte to buffer and forward the message to client2 using serial communication through write() method, where as write() method uses descriptor, buffer and length of buffer to the client2. Continue writing until all received bytes are forwarded.
- Close the socket.

III. Client2-side:

- Create socket file descriptor of datagram type.
- 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 server.
- Send a hello message to the server 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 server .When start byte is received, keep saving bytes until finish byte received when.
- Calculate count of received bytes and display summery

C. Phase 3 Procedure

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 which contains message of 1KB size to be sent to the server 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 server, where as sendto() method includes all the information related to the socket file descriptor, server and buffer.
- Close the socket.
- Create message fragments according to the Wi-Fi packet size and specified Bluetooth MTU size.
- Use write() method to send the fragmented messages to the client-2 using established Bluetooth connection and L2CAP protocol.
- Close both the sockets.

II. Server-side:

- Create two socket file descriptor of type datagram and Bluetooth.
- Create a sockaddr_in structure type to store the client and server information including IP address, port number.
- 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_in structure type and 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-2's Bluetooth address in the sockaddr_l2 structure type variable.
- 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 buffer of some fixed size which is used to receive the hello message from the client-2 using read() method.
- Use a for loop to receive 100 packets from client-1 and store in the above created buffer, here recvfrom() method is used to receive Wi-Fi buffer message from the client1, whereas recvfrom() method fetches all the information related to the client1 and stores in the sockaddr_in structure type of client1 and also related to the socket and buffer.
- Forward packets on bluetooth and zigbee as per desired ratios.
- For Bluetooth
  - Create message fragments according to the Wi-Fi packet size and specified Bluetooth MTU size.
  - Use write() method to send the fragmented messages to the client-2 using established Bluetooth connection and L2CAP protocol.
- For zigbee
  - Add start and finish byte to buffer and forward the message to client2 using serial communication through write() method, where as write() method uses descriptor, buffer and length of buffer to the client2. Continue writing until all received bytes are forwarded.
- Close both the sockets.

III. Client-2 side:

i. Bluetooth

- Create socket file descriptor of Bluetooth type.
- Create a sockaddr_l2 structure type to store the server information including Bluetooth address (BD_ADDR), port number.
- Fill server information in the above created structure.
- 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 receive 100 multiplied by number of fragments of messages using read() method from server through established Bluetooth connection and L2CAP protocol.
- Close the socket.

ii. Zigbee

- Create socket file descriptor of datagram type.
- 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 server.
- Send a hello message to the server 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 server .When start byte is received, keep saving bytes until finish byte received when.
- Calculate count of received bytes and display summery