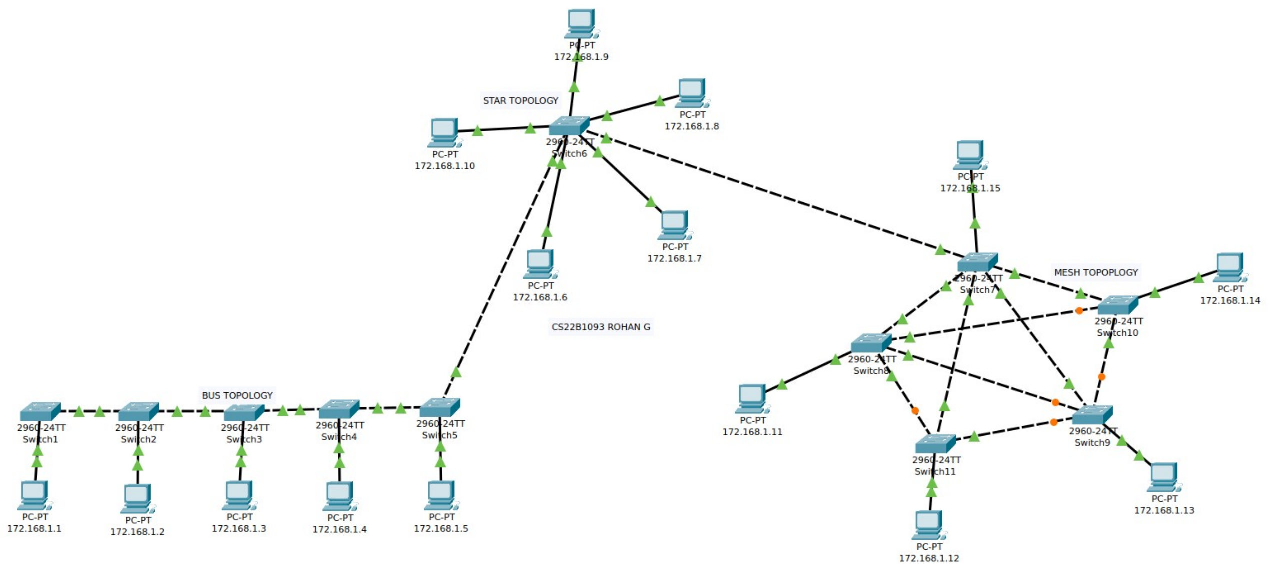
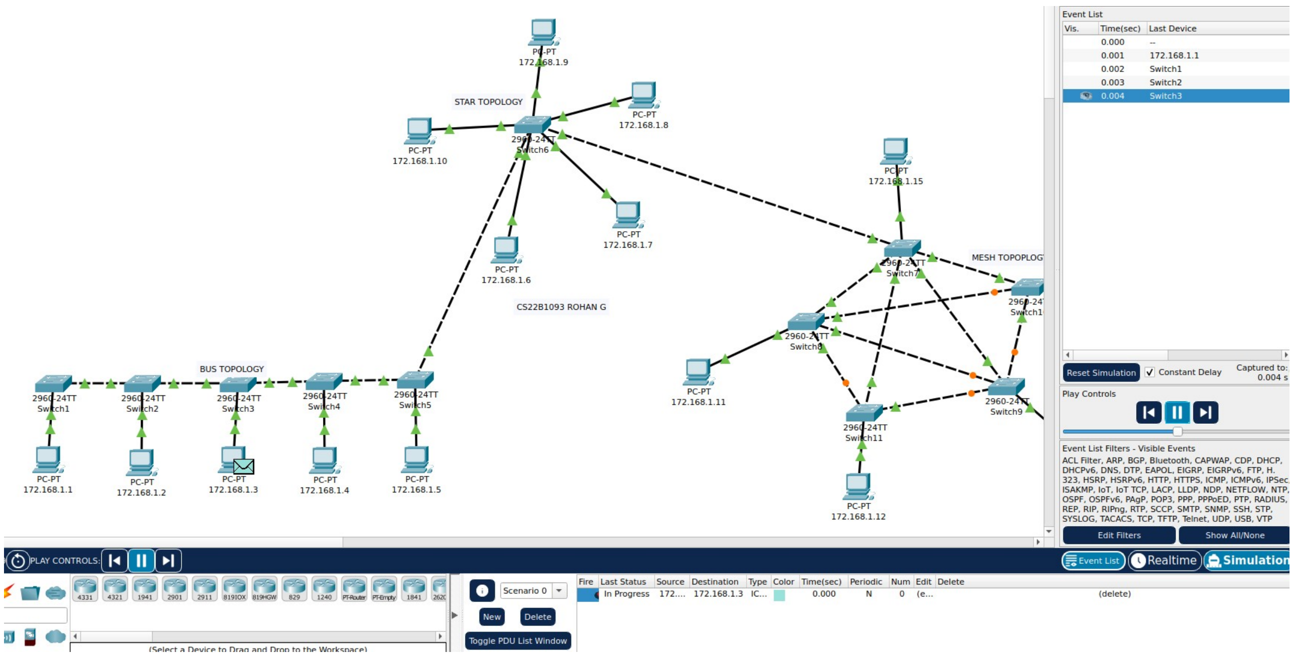
COMPUTER NETWORKS
LAB – 4

ROHAN G
CS22B1093

Q1)





Event List

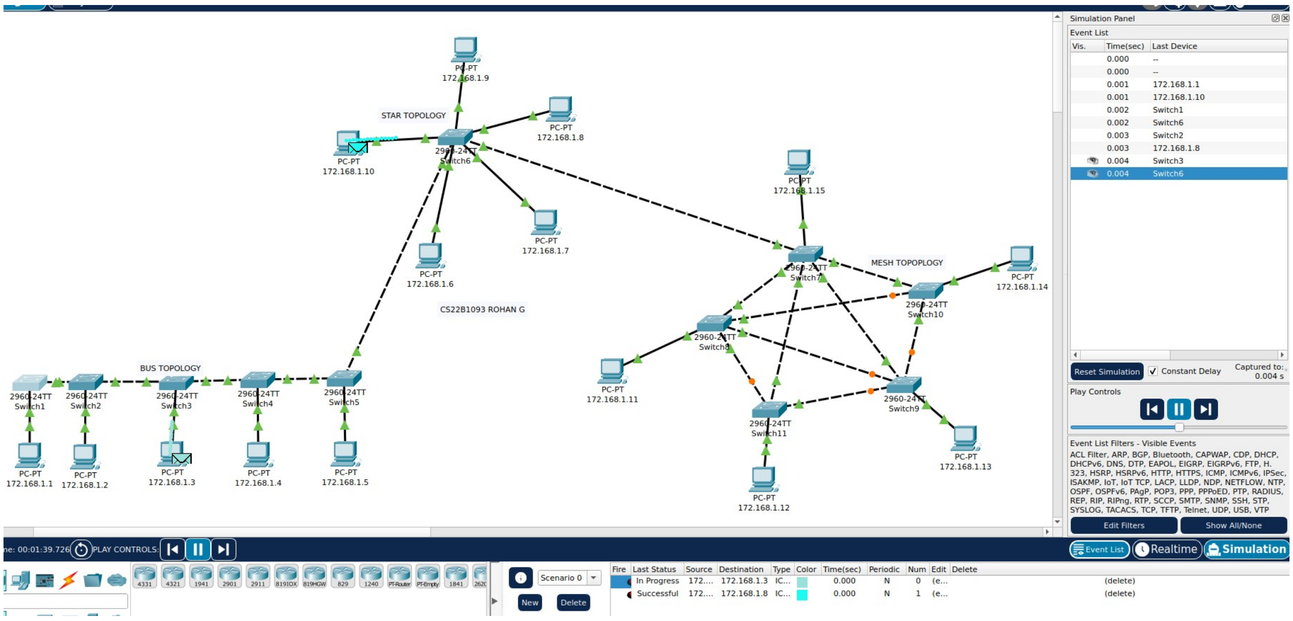
Vis.	Time(sec)	Last Device
	0.000	--
	0.001	172.168.1.1
	0.002	Switch1
	0.003	Switch2
	0.004	Switch3

Event List Filters - Visible Events

ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	In Progress	172....	172.168.1.3	IC...	🟩	0.000	N	0	(e...	(delete)

Intra Bus Topology Packet Transmission(172.168.1.1 – 172.168.1.3)

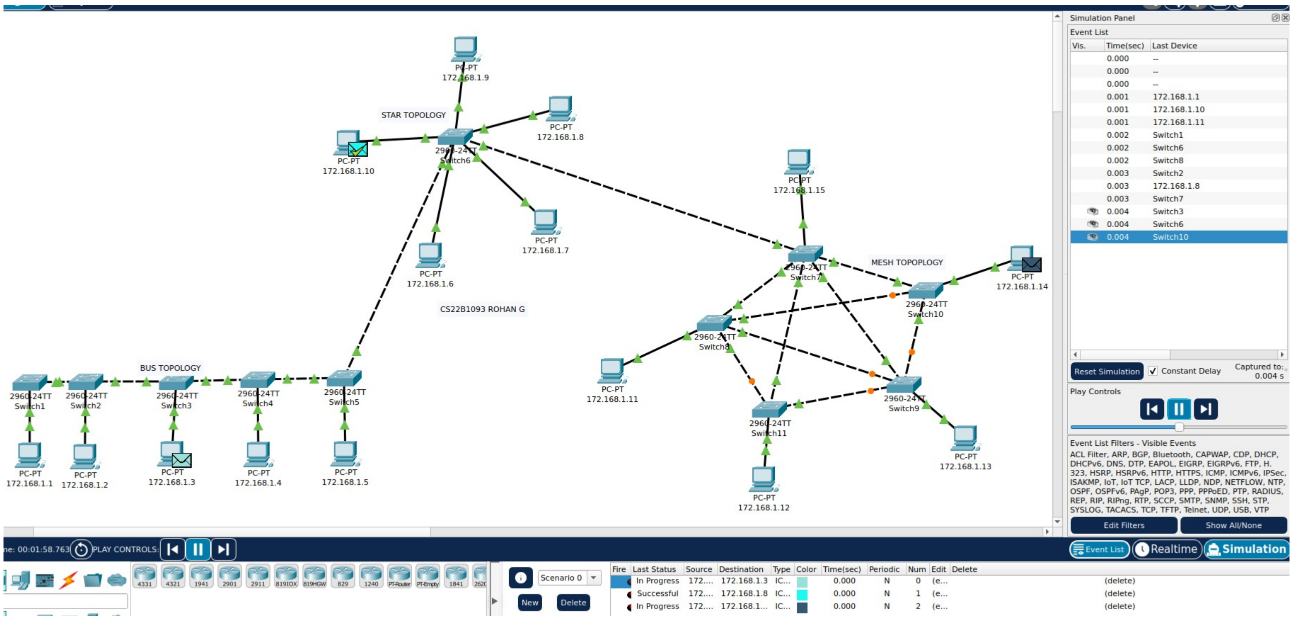


Vis.	Time(sec)	Last Device
	0.000	--
	0.000	--
	0.001	172.168.1.1
	0.001	172.168.1.10
	0.002	Switch1
	0.002	Switch6
	0.003	Switch2
	0.003	172.168.1.8
👁	0.004	Switch3
👁	0.004	Switch6

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H. 323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	In Progress	172....	172.168.1.3	IC...	🟦	0.000	N	0	(e...	(delete)
●	Successful	172....	172.168.1.8	IC...	🟦	0.000	N	1	(e...	(delete)

Intra Star Topology Packet Transmission (172.168.1.10 – 172.168.1.8)

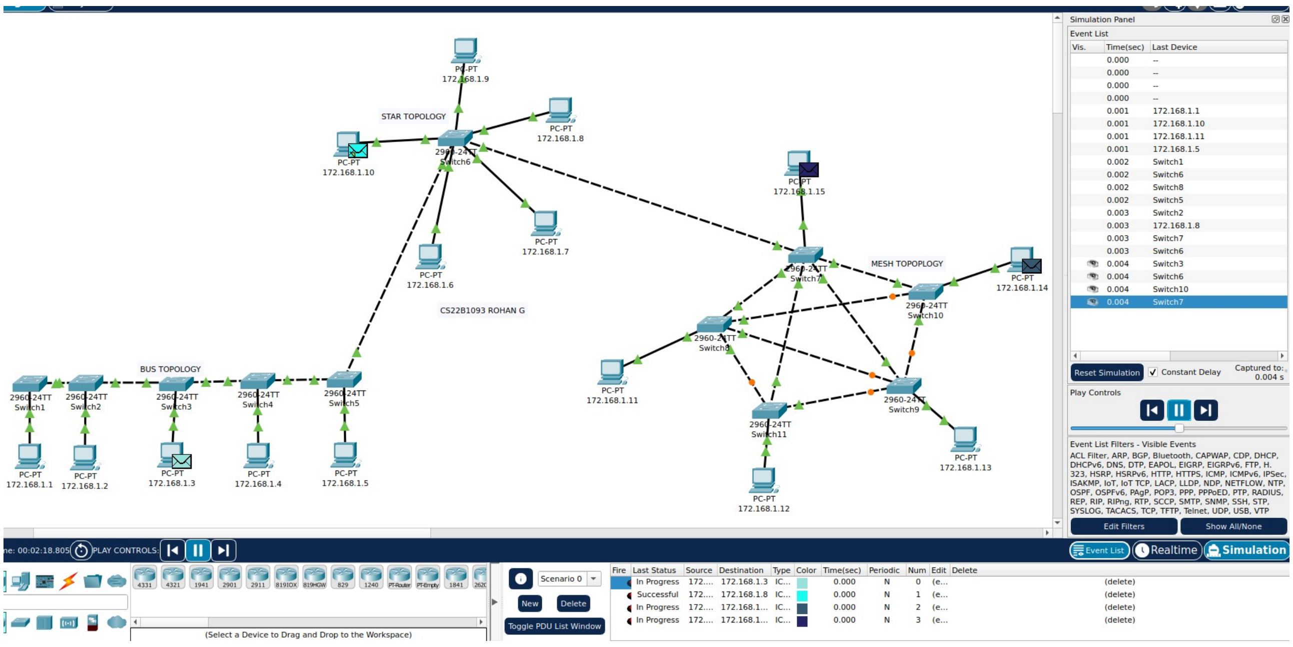


Vis.	Time(sec)	Last Device
	0.000	--
	0.000	--
	0.000	--
	0.001	172.168.1.1
	0.001	172.168.1.10
	0.001	172.168.1.11
	0.002	Switch1
	0.002	Switch6
	0.002	Switch8
	0.003	Switch2
	0.003	172.168.1.8
	0.003	Switch7
👁	0.004	Switch3
👁	0.004	Switch6
👁	0.004	Switch10

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H. 323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	In Progress	172....	172.168.1.3	IC...	🟦	0.000	N	0	(e...	(delete)
●	Successful	172....	172.168.1.8	IC...	🟦	0.000	N	1	(e...	(delete)
●	In Progress	172....	172.168.1...	IC...	🟦	0.000	N	2	(e...	(delete)

Intra Star Topology Packet Transmission(172.168.1.11 – 172.168.1.14)



Event List

Vis.	Time(sec)	Last Device
	0.000	--
	0.000	--
	0.000	--
	0.000	--
	0.001	172.168.1.1
	0.001	172.168.1.10
	0.001	172.168.1.11
	0.001	172.168.1.5
	0.002	Switch1
	0.002	Switch6
	0.002	Switch8
	0.002	Switch5
	0.003	Switch2
	0.003	172.168.1.8
	0.003	Switch7
	0.003	Switch6
👁	0.004	Switch3
👁	0.004	Switch6
👁	0.004	Switch10
👁	0.004	Switch7

✓ Constant Delay   Captured to: 0.004 s

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H. 323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

ne: 00:02:18.805

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	In Progress	172....	172.168.1.3	IC...	🟦	0.000	N	0	(e...	(delete)
●	Successful	172....	172.168.1.8	IC...	🟦	0.000	N	1	(e...	(delete)
●	In Progress	172....	172.168.1...	IC...	🟦	0.000	N	2	(e...	(delete)
●	In Progress	172....	172.168.1...	IC...	🟦	0.000	N	3	(e...	(delete)

Inter Topology Packet Transmission (172.168.1.5 (Bus Topology) – 172.168.1.15(Mesh Topology))

iii)

Bus Topology -
Performance – Performance reduces as number of users and data transmission at same time increases and has limited bandwidth.

Fault Tolerance – Single break or issue in the network causes the whole network collapse and hence the network stops working.

Star Topology -
Performance – Better in performance compared to Bus topology has it has better bandwidth and can scale to more number of users compared to Bus topology.

Fault Tolerance – Has more fault tolerance if there is a breakage or failure between the switch/hub and the connected network device , but the hub/central switch plays a critical role and failure in it causes the whole network to collapse and stop working.

Mesh Topology -
Performance – Highest performance compared to other two with high bandwidth and low latency due to direct connection between the connected devices in the network.

Fault Tolerance – Has the highest fault tolerance compared two the other two , has a lot of reliability as failure in one of the links , the network still functions . But it is expensive to set it up.

Q2)

I) TCP -

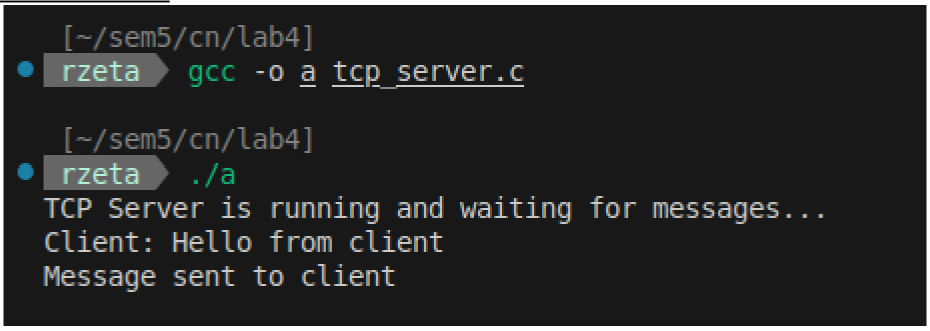
TCP Server -

```c
//CS22B1093 ROHAN G

#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>

#define PORT 8080

int main()
{
    int server_fd, new_socket;
    struct sockaddr_in address;

    char buffer[1024] = {0};
    char *hello = "Hello from server";
    socklen_t addrlen = sizeof(address);

    //create a socket file descriptor
    server_fd = socket(AF_INET, SOCK_STREAM, 0);

    if (server_fd == -1) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    //define the server address
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    //bind the socket
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    //listen for connections
    if (listen(server_fd, 3) < 0) {
        perror("listen");
        close(server_fd);
        exit(EXIT_FAILURE);
    }
    printf("TCP Server is running and waiting for messages...\n");

    //accept connection
    new_socket = accept(server_fd, (struct sockaddr *)&address, &addrlen);

    if(new_socket < 0) {
        perror("Connection acceptance failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    //read message from client
    read(new_socket, buffer, 1024);
    printf("Client: %s\n", buffer);

    //send message to client
    send(new_socket, hello, strlen(hello), 0);
    printf("Message sent to client\n");

    //close the socket
    close(new_socket);
    close(server_fd);

}
```

TCP Client -

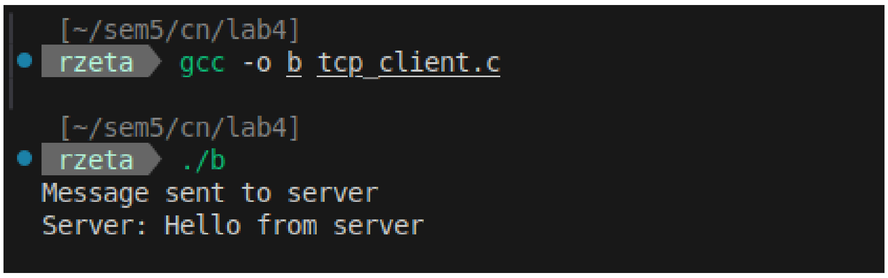
```
//CS22B1093 ROHAN G

#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>

#define PORT 8080

int main()
{
    int sock = 0;
    struct sockaddr_in serv_addr;
    char *message = "Hello from client";
    char buffer[1024] = {0};

    //create a socket file descriptor
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if(sock == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    //define the server address
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    //convert IPv4 and IPv6 addresses from text to binary form
    if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        perror("Invalid address/Address not supported");
        close(sock);
        exit(EXIT_FAILURE);
    }

    //connect to the server
    if(connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("Connection failed");
        close(sock);
        exit(EXIT_FAILURE);
    }

    //send message to server
    send(sock, message, strlen(message), 0);
    printf("Message sent to server\n");

    //read message from server
    read(sock, buffer, 1024);
    printf("Server: %s\n", buffer);

    //close the socket
    close(sock);
    return 0;
}
```

Terminal – 1:



```
 [~/sem5/cn/lab4]
● rzeta  gcc -o a tcp_server.c

 [~/sem5/cn/lab4]
● rzeta  ./a
TCP Server is running and waiting for messages...
Client: Hello from client
Message sent to client
```

Terminal – 2:



```
 [~/sem5/cn/lab4]
● rzeta  gcc -o b tcp_client.c

 [~/sem5/cn/lab4]
● rzeta  ./b
Message sent to server
Server: Hello from server
```

II)UDP

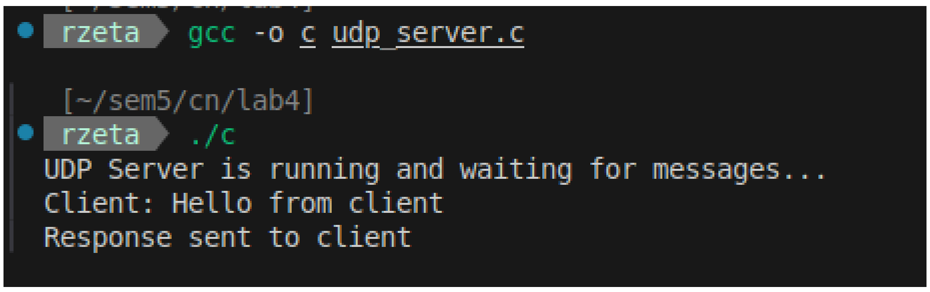
<u>UDP Server -</u>

```c
// CS22B1093 ROHAN G

#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int sockfd;
    char buffer[BUFFER_SIZE];
    struct sockaddr_in serv_addr, client_addr;
    socklen_t addr_len;

    // Create socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Define server address
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;         // IPv4
    serv_addr.sin_addr.s_addr = INADDR_ANY; // Accept any IP
    serv_addr.sin_port = htons(PORT);       // Port

    // Bind the socket to the server address
    if (bind(sockfd, (const struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("Bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    printf("UDP Server is running and waiting for messages...\n");

    addr_len = sizeof(client_addr);

    // Receive data from the client
    int n = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, (struct sockaddr *)&client_addr, &addr_len);
    if (n < 0) {
        perror("Receive failed");
    } else {
        buffer[n] = '\0'; // Null-terminate the received data
        printf("Client: %s\n", buffer);

        // Respond to the client
        char *response = "Hello from server";
        sendto(sockfd, response, strlen(response), 0, (struct sockaddr *)&client_addr, addr_len);
        printf("Response sent to client\n");
    }

    // Close the socket
    close(sockfd);
    return 0;
}
```

<u>UDP Client - </u>

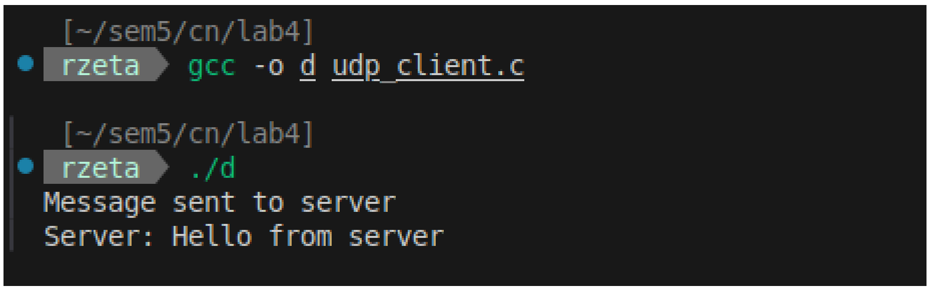
```
// CS22B1093 ROHAN G

#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int sockfd;
    struct sockaddr_in serv_addr;
    char *message = "Hello from client";
    char buffer[BUFFER_SIZE];

    // Create socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Define server address
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    serv_addr.sin_addr.s_addr = INADDR_ANY;

    // Send message to the server
    sendto(sockfd, message, strlen(message), 0, (const struct sockaddr *)&serv_addr,
sizeof(serv_addr));
    printf("Message sent to server\n");
    // Receive response from the server
    socklen_t addr_len = sizeof(serv_addr);
    int n = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, (struct sockaddr *)&serv_addr, &addr_len);
    if (n < 0) {
        perror("Receive failed");
    } else {
        buffer[n] = '\0'; // Null-terminate the received string
        printf("Server: %s\n", buffer);
    }

    // Close the socket
    close(sockfd);
    return 0;
}
```

<u>Terminal -1 :</u>



```
● rzeta  gcc -o c udp_server.c

   [~/sem5/cn/lab4]
● rzeta  ./c
  UDP Server is running and waiting for messages...
  Client: Hello from client
  Response sent to client
```

<u>Terminal – 2 :</u>



```
   [~/sem5/cn/lab4]
● rzeta  gcc -o d udp_client.c

   [~/sem5/cn/lab4]
● rzeta  ./d
  Message sent to server
  Server: Hello from server
```

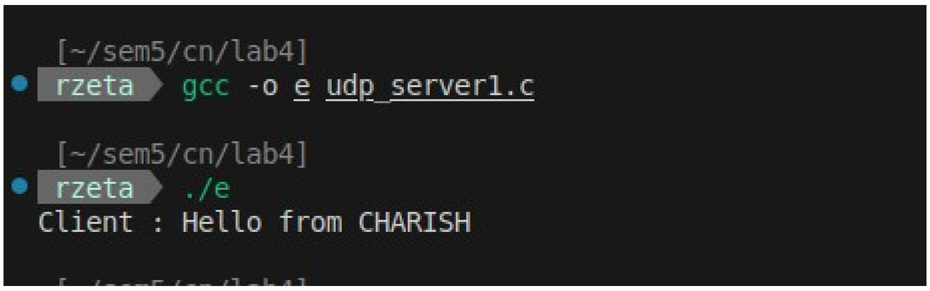
Q3)
UDP Server -

```
//CS22B1093 ROHAN G

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>

#define MAXLINE 1024

int main() {
    int sockfd;
    char buffer[MAXLINE];
    const char *hello = "Hello from Rohan";
    struct sockaddr_in servaddr, cliaddr;
    socklen_t len;

    // Create socket
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(8080);

    if (bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    len = sizeof(cliaddr);

    int n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr
*)&cliaddr, &len);
    buffer[n] = '\0';
    printf("Client : %s\n", buffer);

    sendto(sockfd, (const char *)hello, strlen(hello), 0, (const struct sockaddr *)&cliaddr, len);

    close(sockfd);
    return 0;
}
```

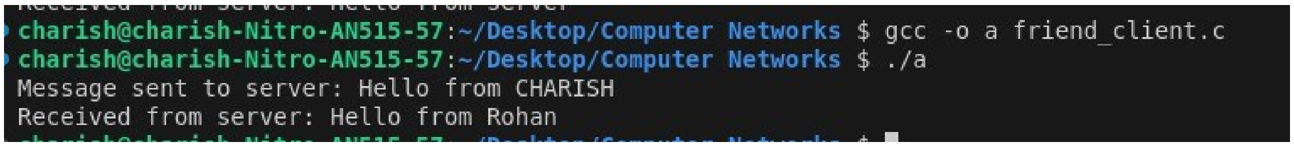
UDP Client -

```
//CS22B1093 ROHAN G

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>

#define MAX 1024

int main() {
    int sockfd;
    char buffer[MAX];
    const char *message = "Hello from ROHAN";
    struct sockaddr_in serv_addr;
    socklen_t len;

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&serv_addr, 0, sizeof(serv_addr));


    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(8080);

    if (inet_pton(AF_INET, "172.16.25.82", &serv_addr.sin_addr) <= 0) {
        perror("Invalid address or Address not supported");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    sendto(sockfd, (const char *)message, strlen(message), 0, (const struct sockaddr *)&serv_addr,
sizeof(serv_addr));

    len = sizeof(serv_addr);
    int n = recvfrom(sockfd, (char *)buffer, MAX, MSG_WAITALL, (struct sockaddr *)&serv_addr,
&len);
    buffer[n] = '\0';
    printf("Server : %s\n", buffer);

    close(sockfd);
    return 0;
}
```

UDP Server (me) , friend is Client -

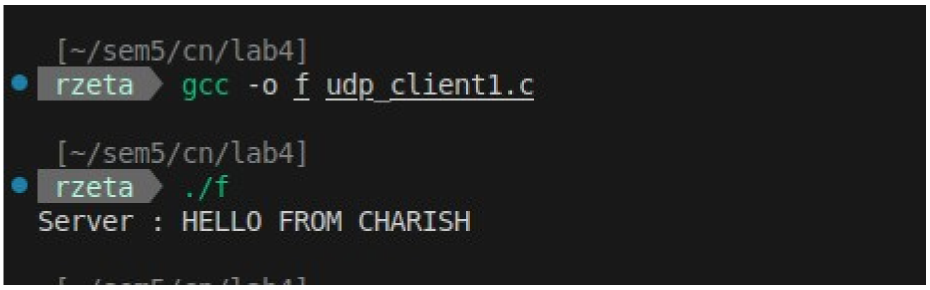


```
[~/sem5/cn/lab4]
● rzeta  gcc -o e udp_server1.c

[~/sem5/cn/lab4]
● rzeta  ./e
Client : Hello from CHARISH
```

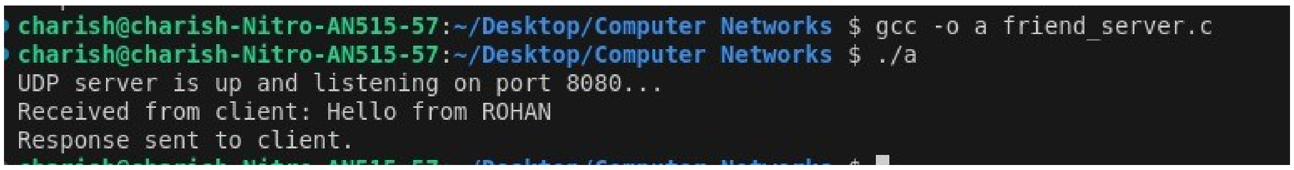


```
charish@charish-Nitro-AN515-57:~/Desktop/Computer Networks $ gcc -o a friend_client.c
charish@charish-Nitro-AN515-57:~/Desktop/Computer Networks $ ./a
Message sent to server: Hello from CHARISH
Received from server: Hello from Rohan
```

UDP Client (me) , friend is Server -



```
[~/sem5/cn/lab4]
● rzeta  gcc -o f udp_client1.c

[~/sem5/cn/lab4]
● rzeta  ./f
Server : HELLO FROM CHARISH
```



```
charish@charish-Nitro-AN515-57:~/Desktop/Computer Networks $ gcc -o a friend_server.c
charish@charish-Nitro-AN515-57:~/Desktop/Computer Networks $ ./a
UDP server is up and listening on port 8080...
Received from client: Hello from ROHAN
Response sent to client.
```