NAME

MOHAMMAD SHAAD

REGISTRATION NUMBER

21BCE1542

CLASS

COMPUTER NETWORKS

FACULTY

PUNITHA K MA’AM

LAB

EXERCISE 6

AIM

TO IMPLEMENT THE SOCKET PROGRAMMING USING **TCP**

PROCEDURE

1. Open two terminals (one for server and another one for client)
2. Execute the server socket program
3. Execute the client program

CODES

server.c

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

#define BUF_SIZE 1024

void error_handler(char *msg) {
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[]) {
```

```
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <port>\n", argv[0]);
        exit(1);
    }

    int server_sock, client_sock;
    struct sockaddr_in server_addr, client_addr;

    // Create socket
    server_sock = socket(AF_INET, SOCK_STREAM, 0);
    if (server_sock == -1) {
        error_handler("Failed to create socket");
    }

    // Bind socket to address and port
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(atoi(argv[1]));

    if (bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) ==
-1) {
        error_handler("Failed to bind socket");
    }

    // Listen for connections
    if (listen(server_sock, 5) == -1) {
```

```
        error_handler("Failed to listen for connections");
    }

    printf("Server started. Waiting for connections...\n");

    socklen_t client_addr_size = sizeof(client_addr);

    // Accept incoming connections
    client_sock = accept(server_sock, (struct sockaddr *)&client_addr,
&client_addr_size);
    if (client_sock == -1) {
        error_handler("Failed to accept connection");
    }

    printf("Client connected: %s:%d\n", inet_ntoa(client_addr.sin_addr),
ntohs(client_addr.sin_port));

    char buf[BUF_SIZE];
    int read_size;

    while ((read_size = recv(client_sock, buf, BUF_SIZE, 0)) > 0) {

        // Get current time
        time_t now = time(0);
        struct tm *local_time = localtime(&now);
        char timestamp[20];
        strftime(timestamp, sizeof(timestamp), "%Y-%m-%d %H:%M:%S",
local_time);
```

```
        // Add timestamp to message
        char message[BUF_SIZE + 38]; // Maximum possible string length is 19 +
BUF_SIZE - 1 + 19
        sprintf(message, "[%s] %s", timestamp, buf);

        printf("%s", message);

        // Send message back to client
        if (send(client_sock, message, strlen(message), 0) == -1) {
            error_handler("Failed to send message");
        }

        memset(buf, 0, sizeof(buf));
    }

    if (read_size == 0) {
        puts("Client disconnected");
    } else {
        error_handler("Failed to receive data");
    }

    close(client_sock);
    close(server_sock);

    return 0;
}
```

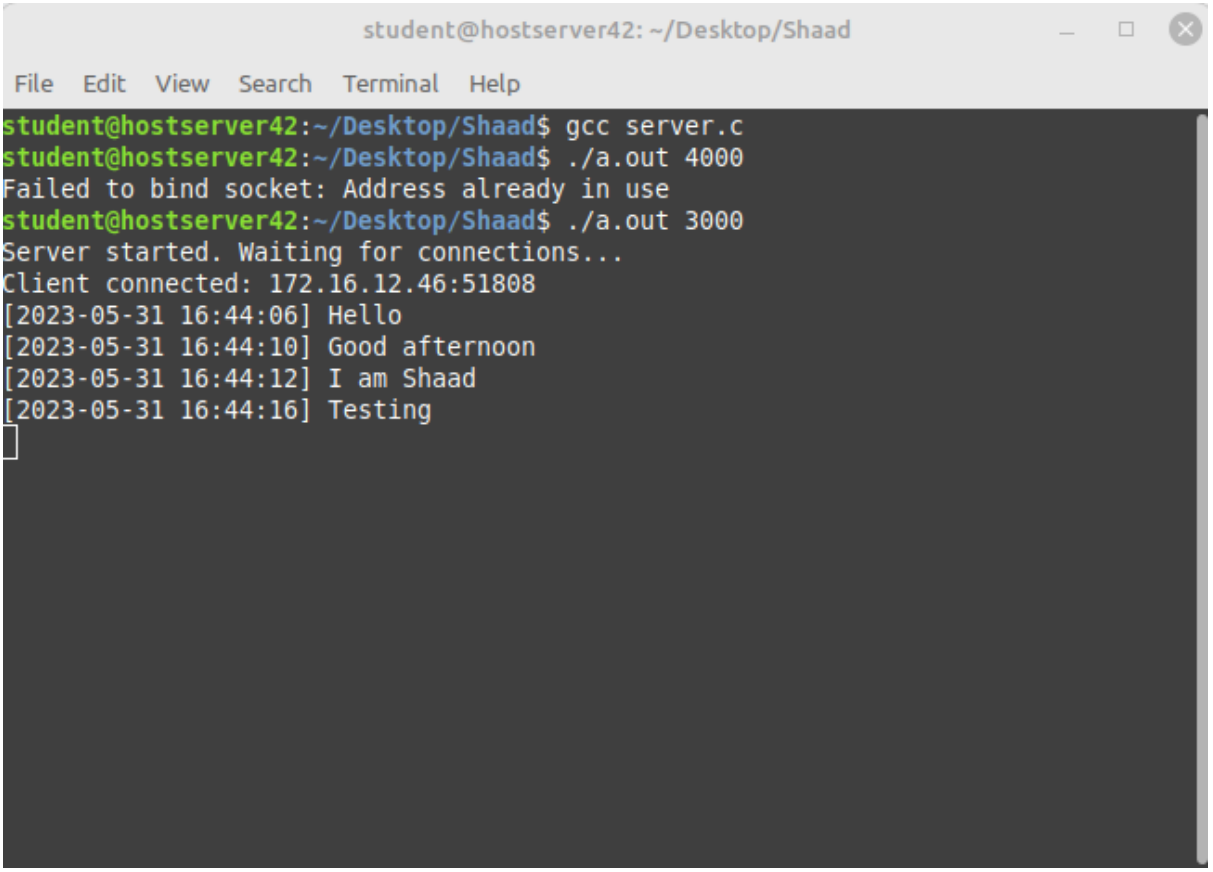
client.c

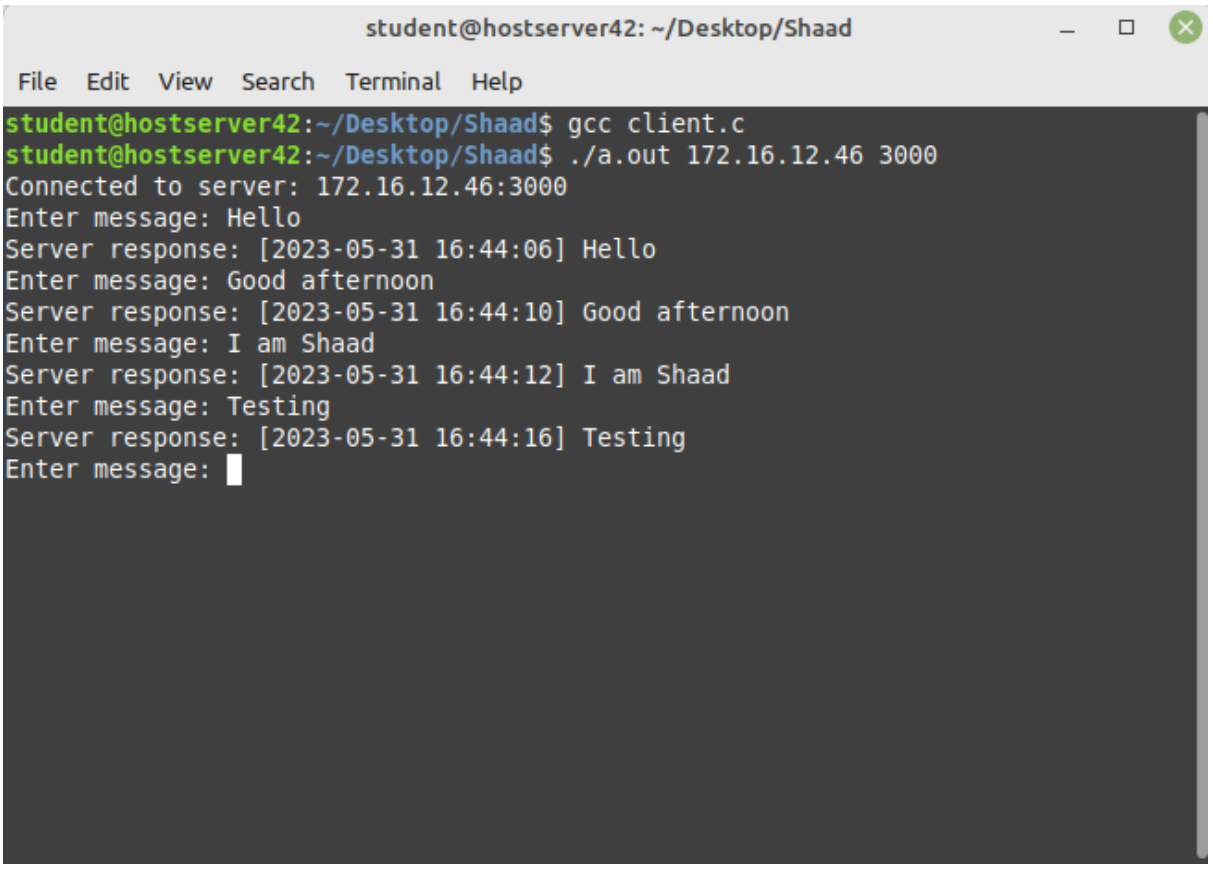
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>

#define BUF_SIZE 1024

void error_handler(char *msg) {
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[]) {

    if (argc != 3) {
        fprintf(stderr, "Usage: %s <server_ip> <port>\n", argv[0]);
        exit(1);
    }

    int sock;
    struct sockaddr_in server_addr;

    // Create socket
    sock = socket(AF_INET, SOCK_STREAM, 0);
```

```
if (sock == -1) {
    error_handler("Failed to create socket");
}

// Configure server address
memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr(argv[1]);
server_addr.sin_port = htons(atoi(argv[2]));

// Connect to server
if (connect(sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) == -1) {
    error_handler("Failed to connect to server");
}

printf("Connected to server: %s:%d\n", argv[1], atoi(argv[2]));

char message[BUF_SIZE];
int read_size;

while (1) {
    printf("Enter message: ");
    fgets(message, BUF_SIZE, stdin);

    // Send message to server
    if (send(sock, message, strlen(message), 0) == -1) {
        error_handler("Failed to send message");
    }
```

```
        // Receive response from server
        memset(message, 0, sizeof(message));
        read_size = recv(sock, message, BUF_SIZE, 0);
        if (read_size == -1) {
            error_handler("Failed to receive response");
        } else if (read_size == 0) {
            puts("Server disconnected");
            break;
        }

        printf("Server response: %s", message);
    }

    close(sock);

    return 0;
}
```

OUTPUT


student@hostserver42: ~/Desktop/Shaad
File Edit View Search Terminal Help
student@hostserver42:~/Desktop/Shaad$ gcc server.c
student@hostserver42:~/Desktop/Shaad$ ./a.out 4000
Failed to bind socket: Address already in use
student@hostserver42:~/Desktop/Shaad$ ./a.out 3000
Server started. Waiting for connections...
Client connected: 172.16.12.46:51808
[2023-05-31 16:44:06] Hello
[2023-05-31 16:44:10] Good afternoon
[2023-05-31 16:44:12] I am Shaad
[2023-05-31 16:44:16] Testing


student@hostserver42: ~/Desktop/Shaad
File Edit View Search Terminal Help
student@hostserver42:~/Desktop/Shaad$ gcc client.c
student@hostserver42:~/Desktop/Shaad$ ./a.out 172.16.12.46 3000
Connected to server: 172.16.12.46:3000
Enter message: Hello
Server response: [2023-05-31 16:44:06] Hello
Enter message: Good afternoon
Server response: [2023-05-31 16:44:10] Good afternoon
Enter message: I am Shaad
Server response: [2023-05-31 16:44:12] I am Shaad
Enter message: Testing
Server response: [2023-05-31 16:44:16] Testing
Enter message:

NAME

MOHAMMAD SHAAD

REGISTRATION NUMBER

21BCE1542

CLASS

COMPUTER NETWORKS

FACULTY

PUNITHA K MA’AM

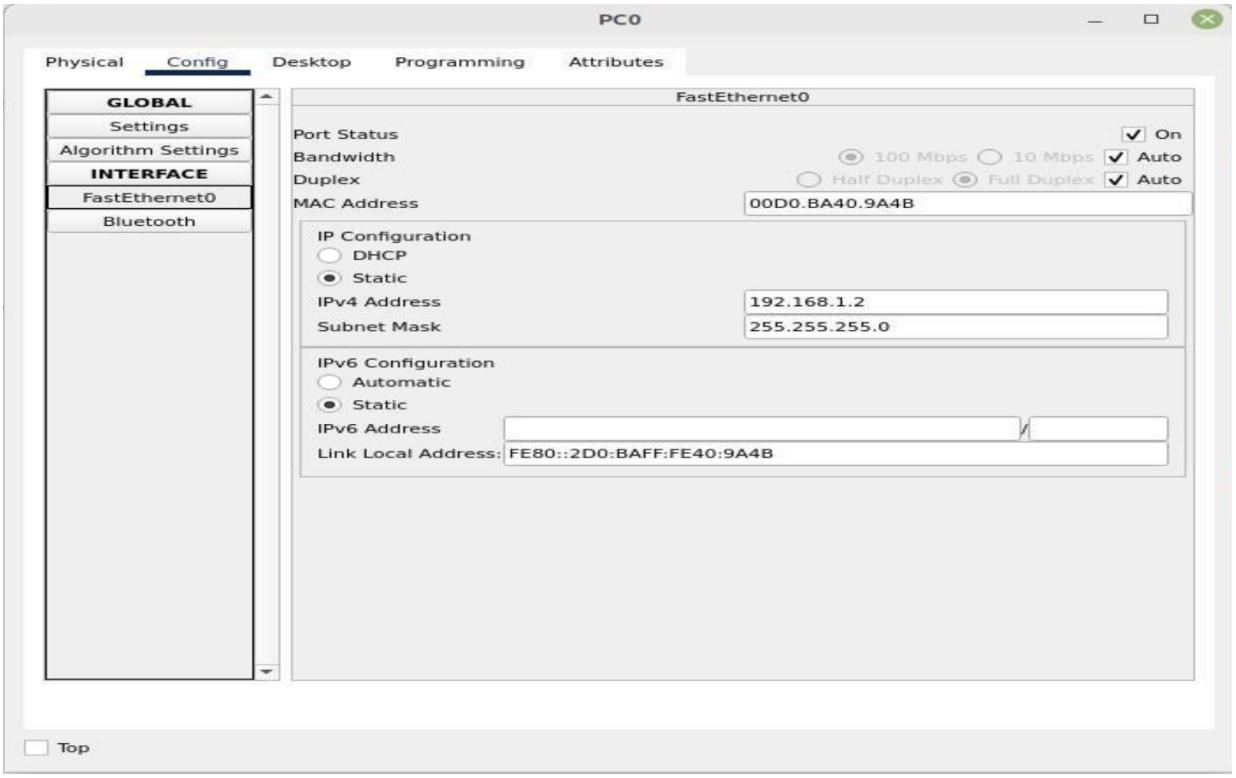
LAB

EXERCISE 8

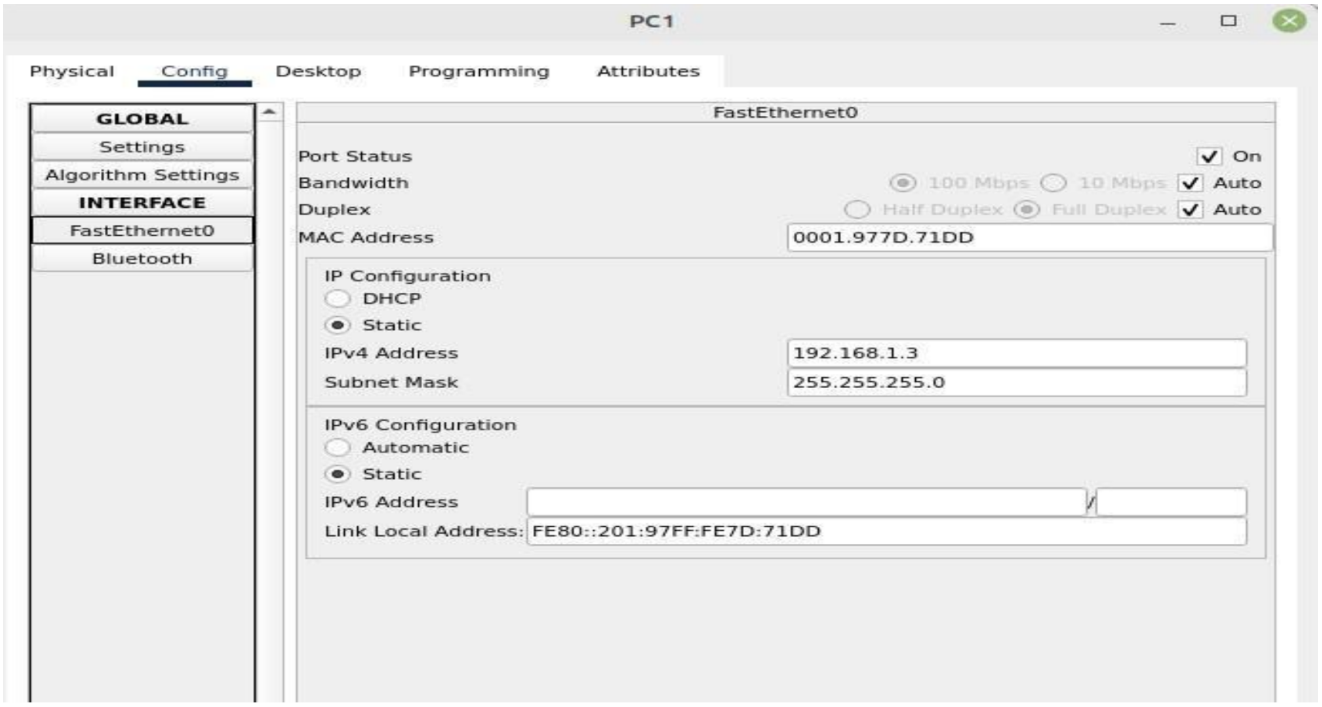
CISCO PACKET TRACER

1. Switch1

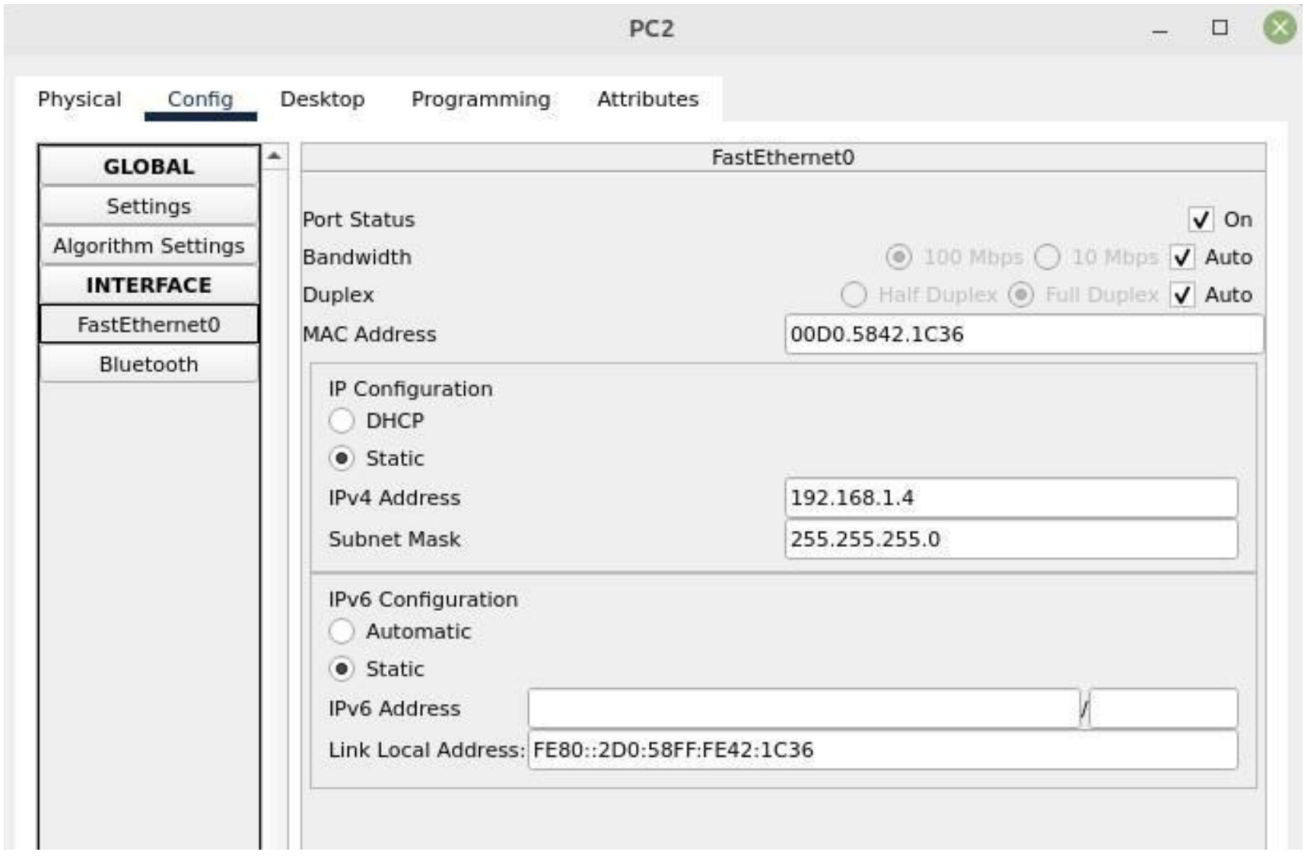
💻 PC0



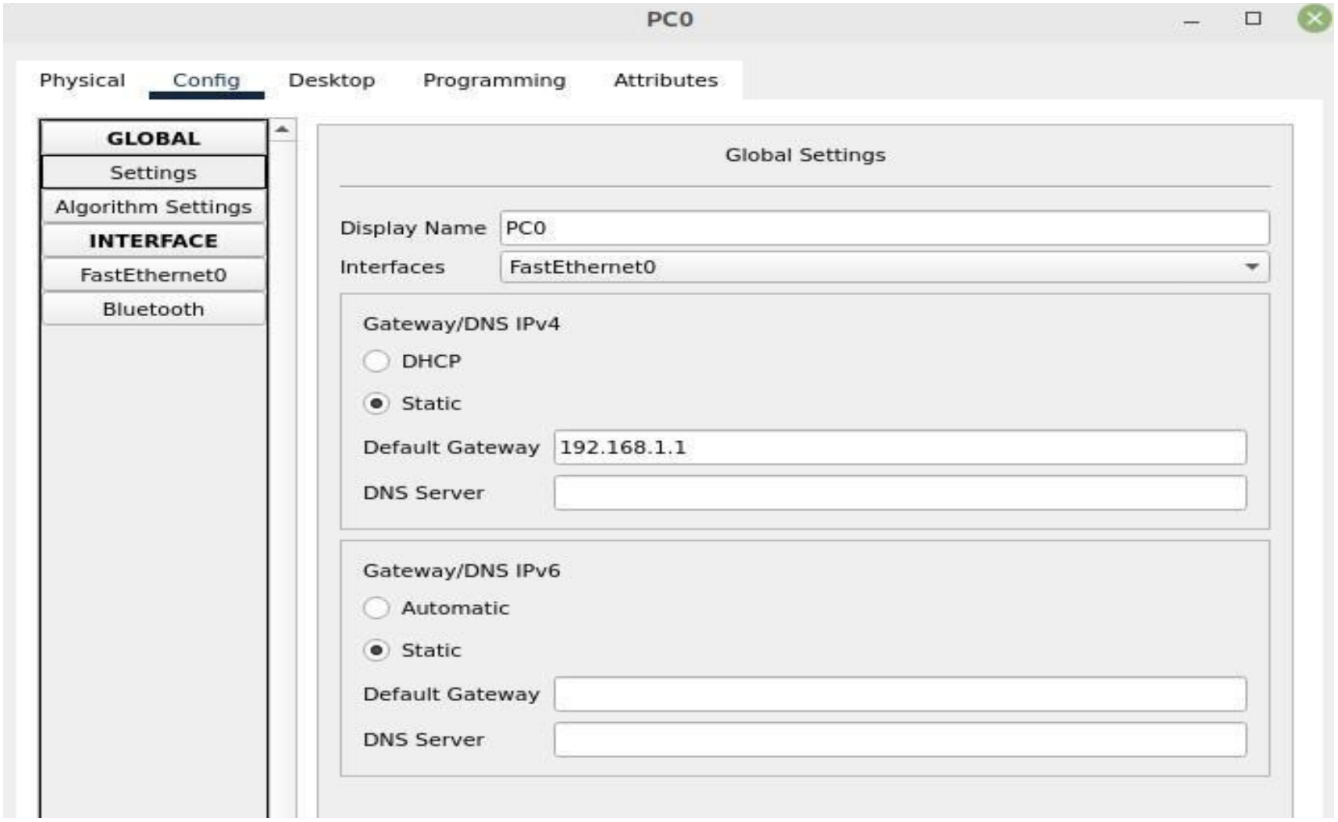
💻 PC1


PC1
Physical
Config
Desktop
Programming
Attributes
GLOBAL
Settings
Algorithm Settings
INTERFACE
FastEthernet0
Bluetooth
FastEthernet0
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
0001.977D.71DD
IP Configuration
DHCP
Static
IPv4 Address
192.168.1.3
Subnet Mask
255.255.255.0
IPv6 Configuration
Automatic
Static
IPv6 Address
Link Local Address:
FE80::201:97FF:FE7D:71DD

💻 PC2

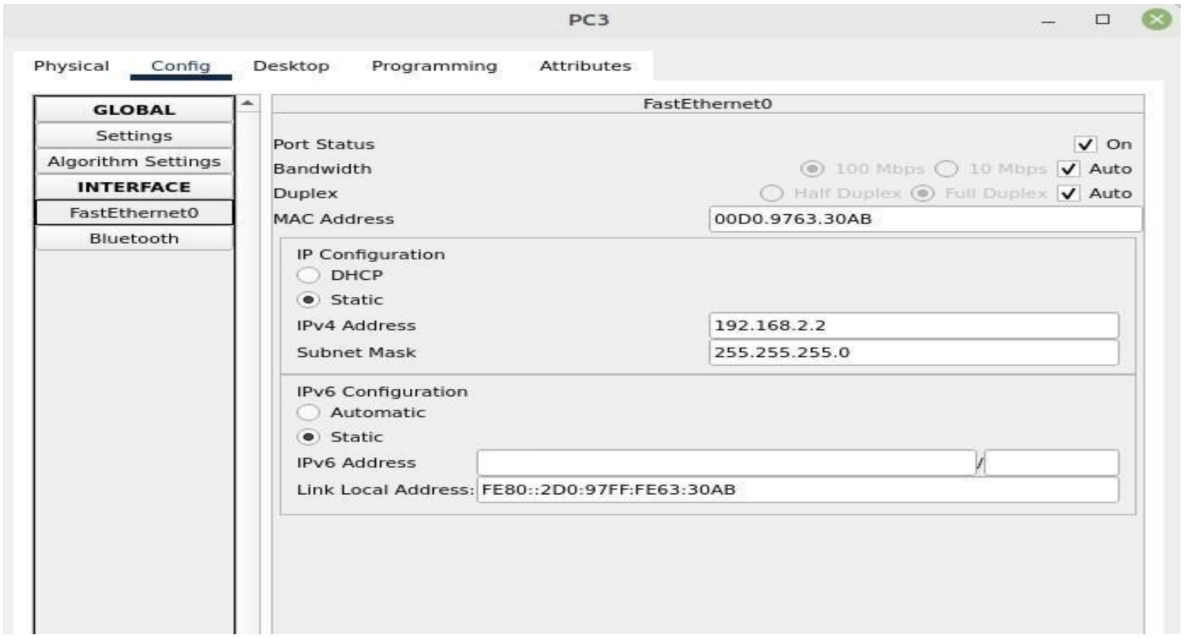

PC2
Physical
Config
Desktop
Programming
Attributes
GLOBAL
Settings
Algorithm Settings
INTERFACE
FastEthernet0
Bluetooth
FastEthernet0
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
00D0.5842.1C36
IP Configuration
DHCP
Static
IPv4 Address
192.168.1.4
Subnet Mask
255.255.255.0
IPv6 Configuration
Automatic
Static
IPv6 Address
Link Local Address:
FE80::2D0:58FF:FE42:1C36

🔢Default gateway for all the PC's in Switch1 is 192.168.1.1.

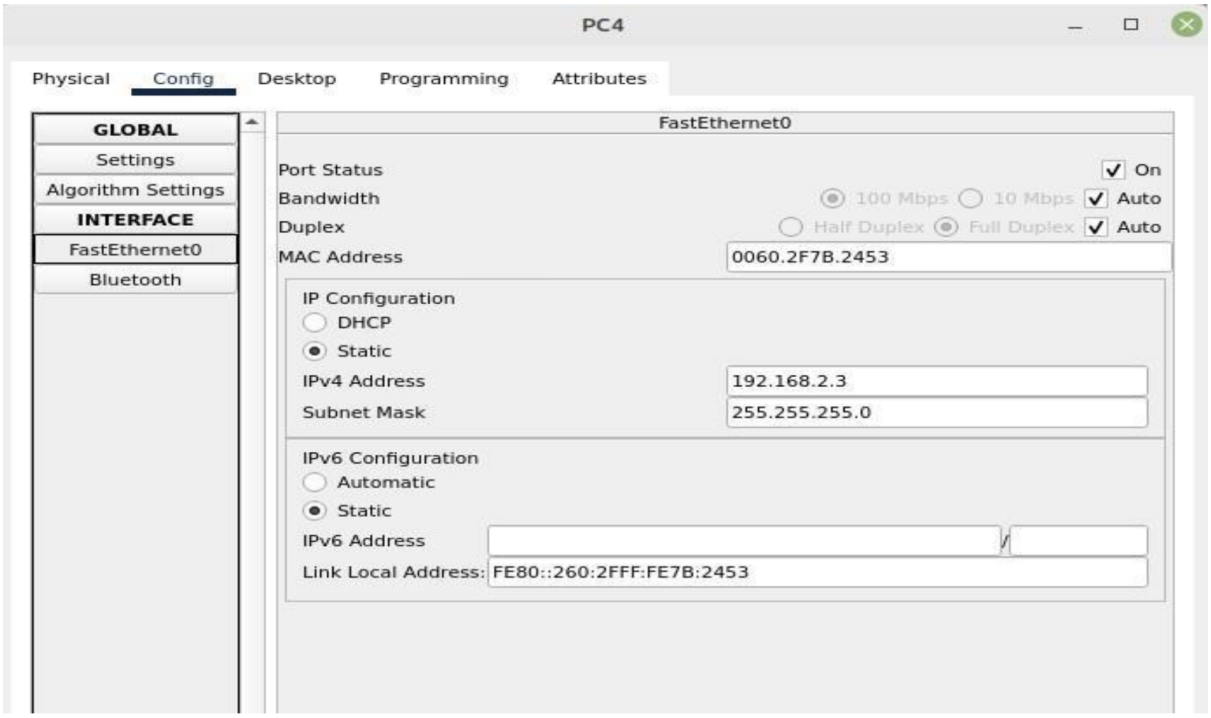


2. Switch2

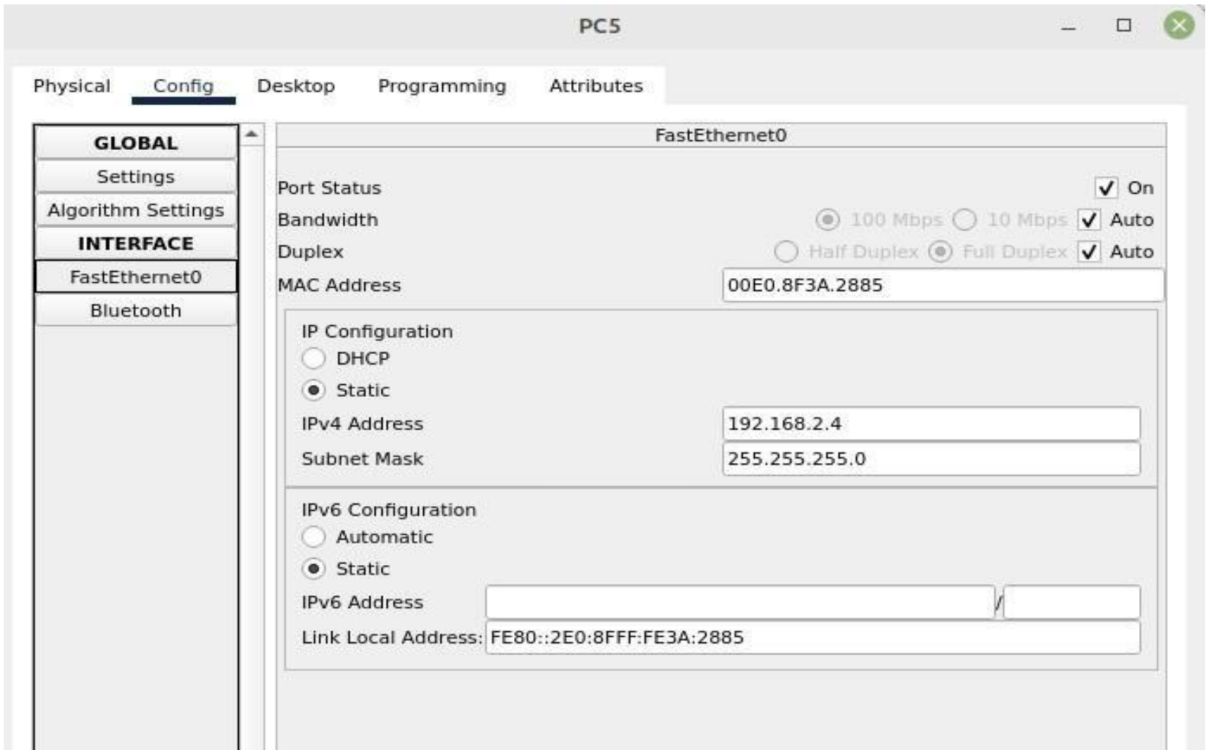
💻 PC3



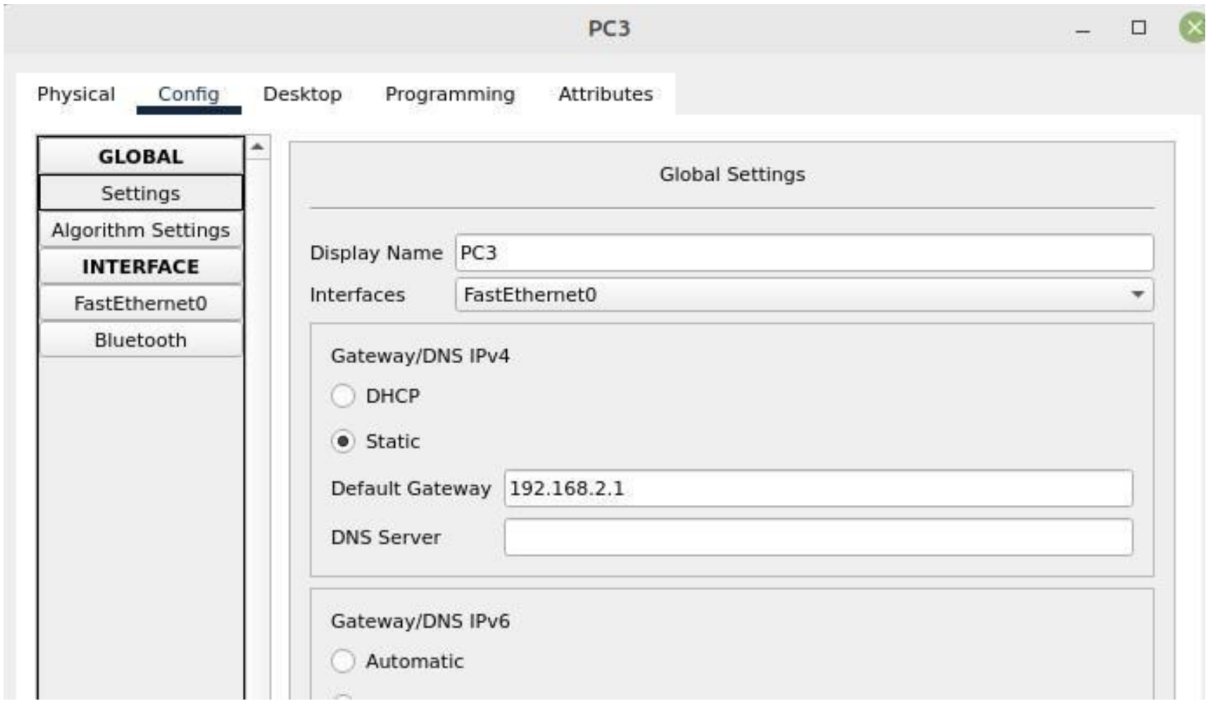
💻 PC4



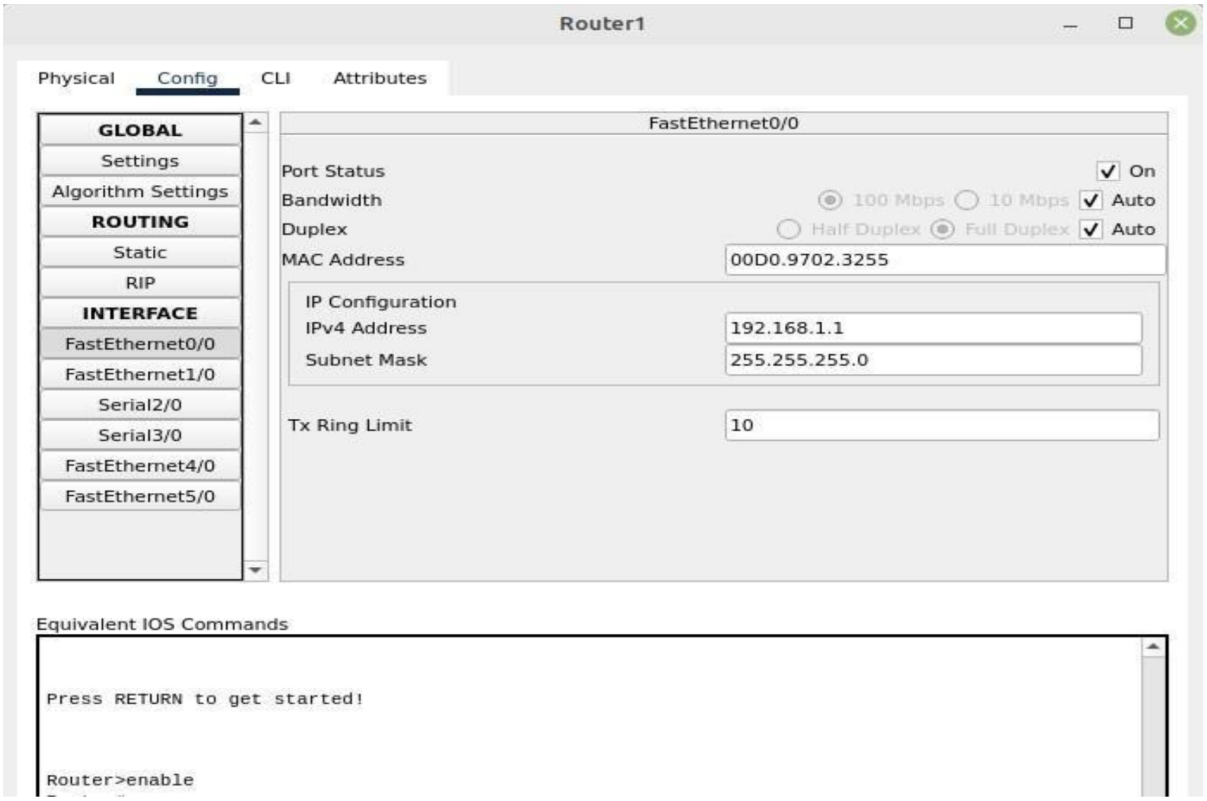
💻 PC5

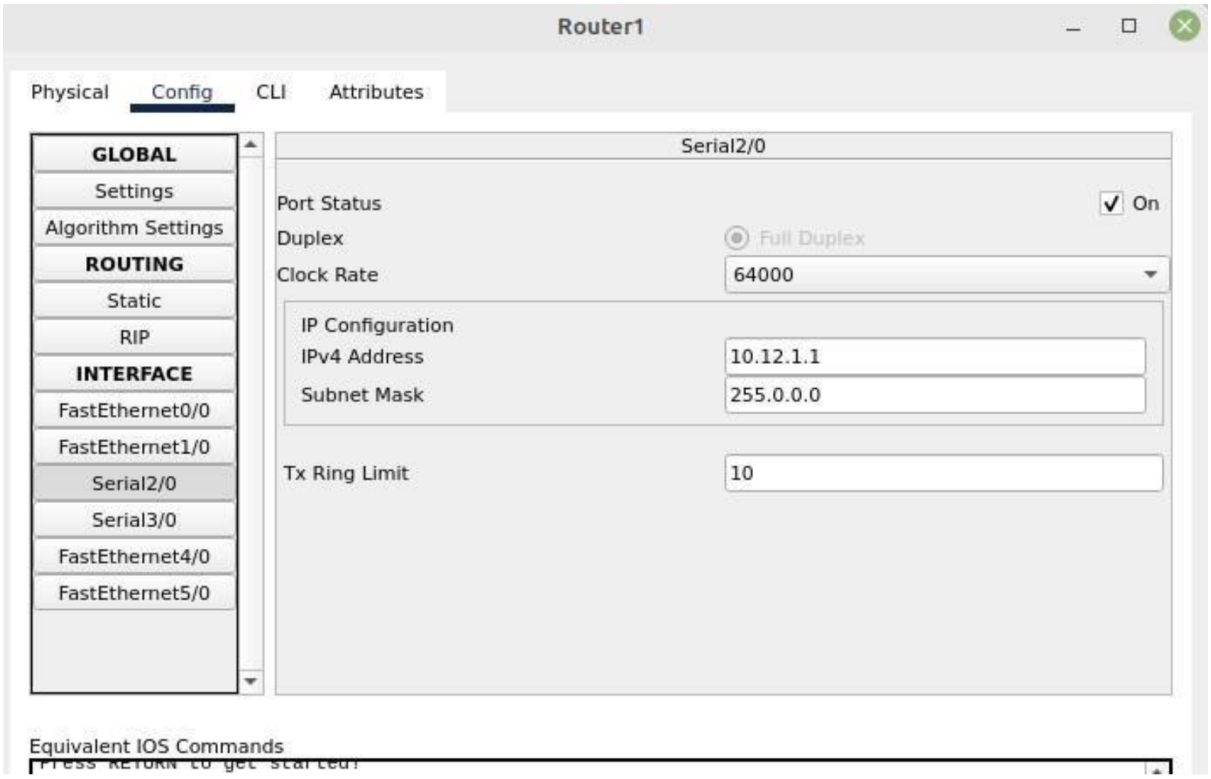


🔢 Default gateway for all the PC's in Switch2 is 192.168.2.1.

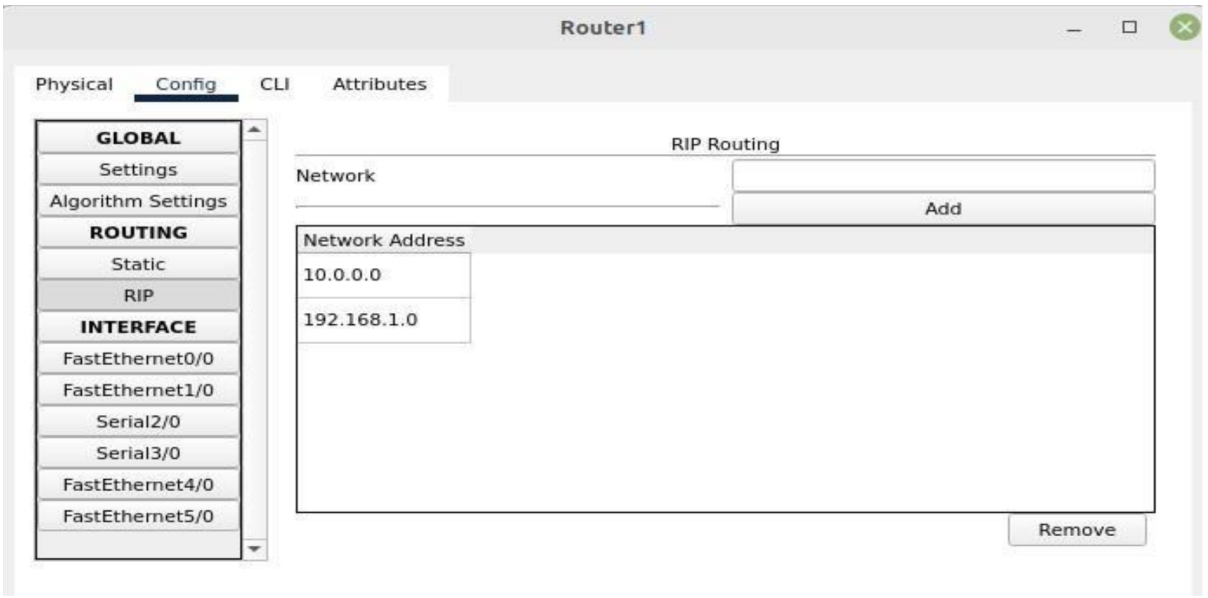

PC3
Physical
Config
Desktop
Programming
Attributes
GLOBAL
Settings
Algorithm Settings
INTERFACE
FastEthernet0
Bluetooth
Global Settings
Display Name
PC3
Interfaces
FastEthernet0
Gateway/DNS IPv4
DHCP
Static
Default Gateway
192.168.2.1
DNS Server
Gateway/DNS IPv6
Automatic

🖨️ Router1

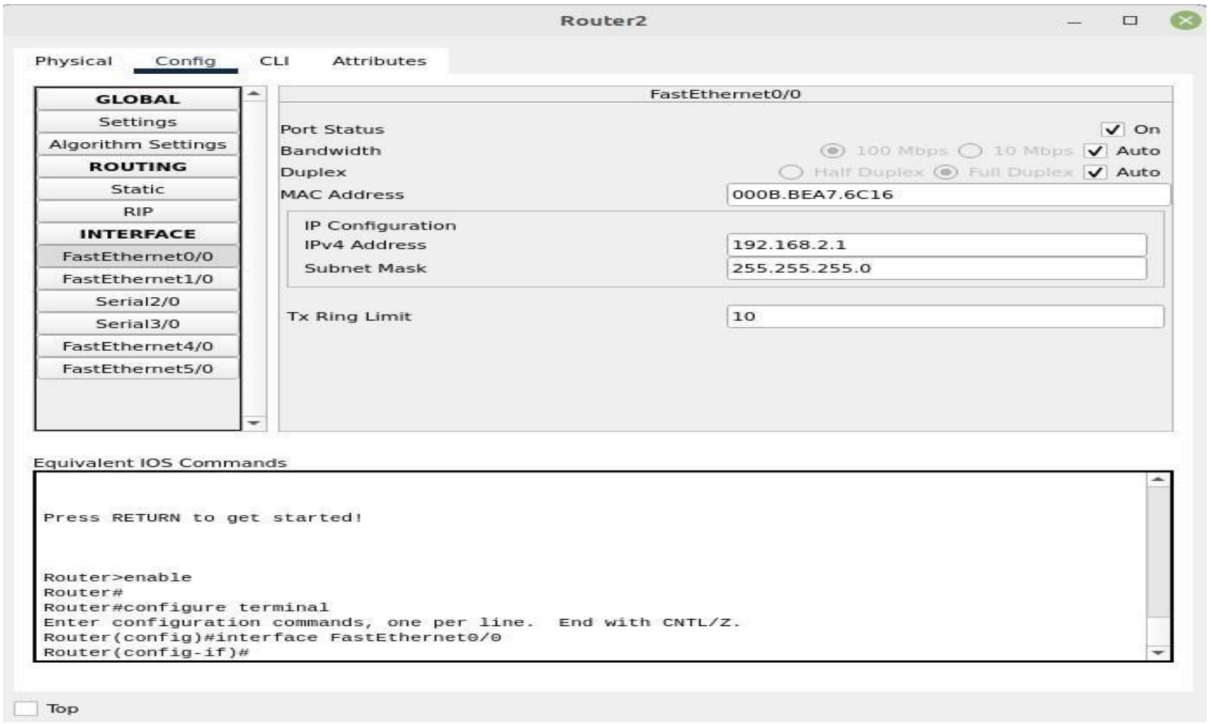

Router1
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
FastEthernet0/0
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
00D0.9702.3255
IP Configuration
IPv4 Address
192.168.1.1
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Press RETURN to get started!
Router>enable


Router1
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
Serial2/0
Port Status
On
Duplex
Full Duplex
Clock Rate
64000
IP Configuration
IPv4 Address
10.12.1.1
Subnet Mask
255.0.0.0
Tx Ring Limit
10
Equivalent IOS Commands

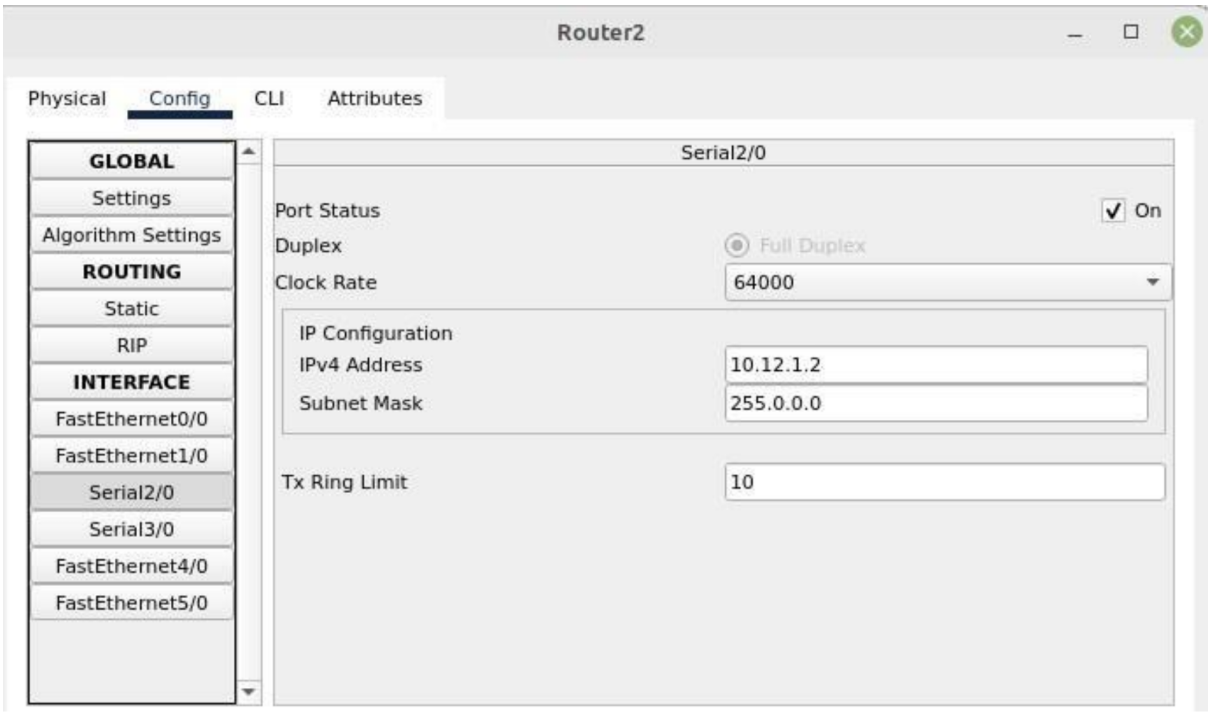
RIP


Router1
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
RIP Routing
Network
Add
Network Address
10.0.0.0
192.168.1.0
Remove

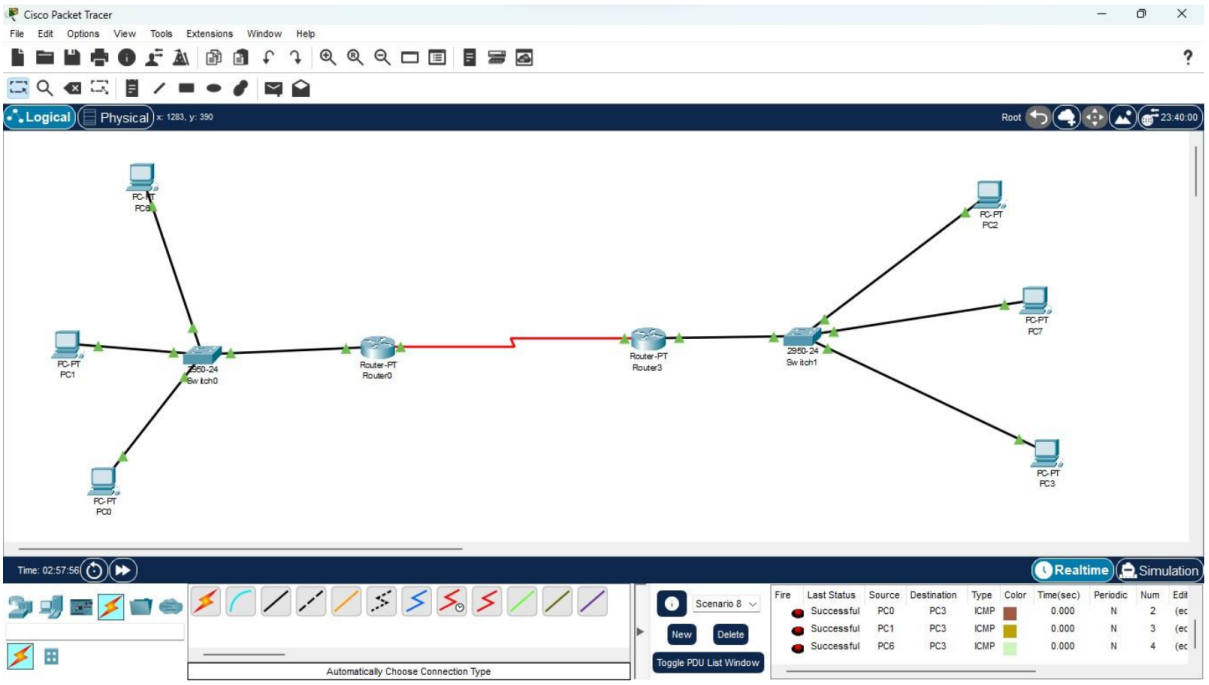
🖨️ Router2

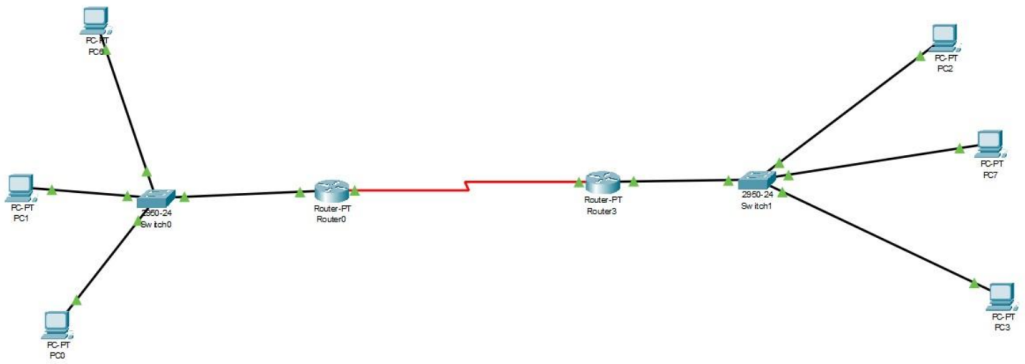

Router2
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
FastEthernet0/0
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
000B.BEA7.6C16
IP Configuration
IPv4 Address
192.168.2.1
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Press RETURN to get started!
Router>enable
Router#
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#
Top

RIP


Router2
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
Serial2/0
Port Status
On
Duplex
Full Duplex
Clock Rate
64000
IP Configuration
IPv4 Address
10.12.1.2
Subnet Mask
255.0.0.0
Tx Ring Limit
10

OUTPUT





Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Successful	PC6	PC3	ICMP		0.000	N	4	(ec
	Successful	PC6	PC7	ICMP		0.000	N	5	(ec
	Successful	PC0	PC2	ICMP		0.000	N	6	(ec

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Successful	PC1	PC3	ICMP		0.000	N	3	(ec
	Successful	PC6	PC3	ICMP		0.000	N	4	(ec
	Successful	PC6	PC7	ICMP		0.000	N	5	(ec

NAME

MOHAMMAD SHAAD

REGISTRATION NUMBER

21BCE1542

CLASS

COMPUTER NETWORKS

FACULTY

PUNITHA K MA’AM

LAB

EXERCISE 9

CISCO PACKET TRACER PRACTICE

AIM

To check make connection between two topology (Ring and Mesh) using Cisco Packet Tracer

Procedure

1. **Select appropriate Cisco devices**: Choose Cisco networking devices that support both Ring and Bus topologies. For example, you can consider switches or routers that offer the necessary features for creating and managing these topologies.

2. **Design the network layout**: Plan the physical and logical layout of your network. Identify the locations where devices will be placed, the connections between them, and any additional components required (such as hubs or repeaters).

3. **Implement the Ring topology**: Connect the devices in a circular manner, forming a ring. Use appropriate Ethernet cables to establish the connections between devices. Ensure that each device has two connections to form a redundant ring for fault tolerance.

4. **Configure Ring-specific protocols**: Depending on the Cisco devices you're using, configure protocols like Rapid Spanning Tree Protocol (RSTP) or Multiple Spanning Tree Protocol (MSTP) to manage the ring's redundancy and prevent loops.

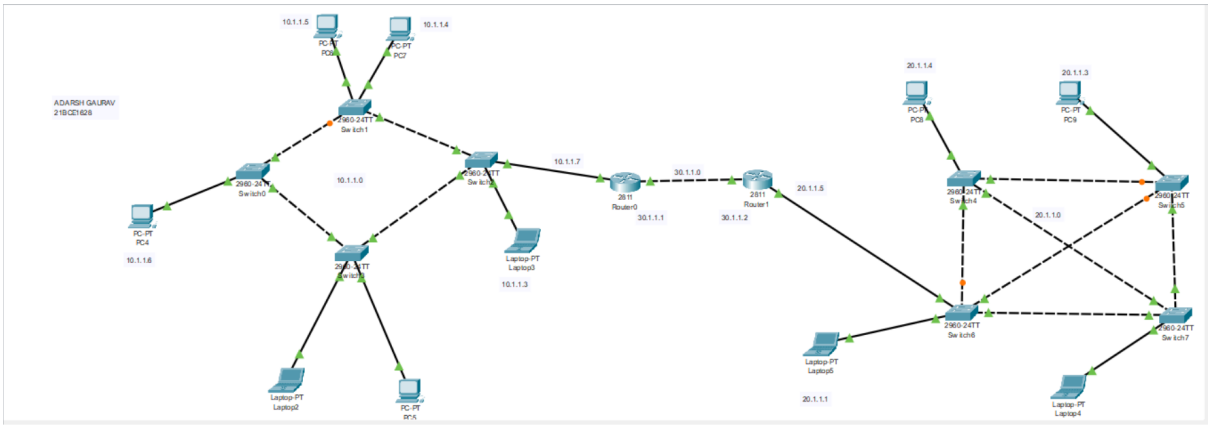
5. **Implement the MESH topology**: Extend the network by connecting additional devices to the ring using appropriate connectors, such as taps or drop cables. These connected devices will serve as bus segments branching out from the ring backbone.

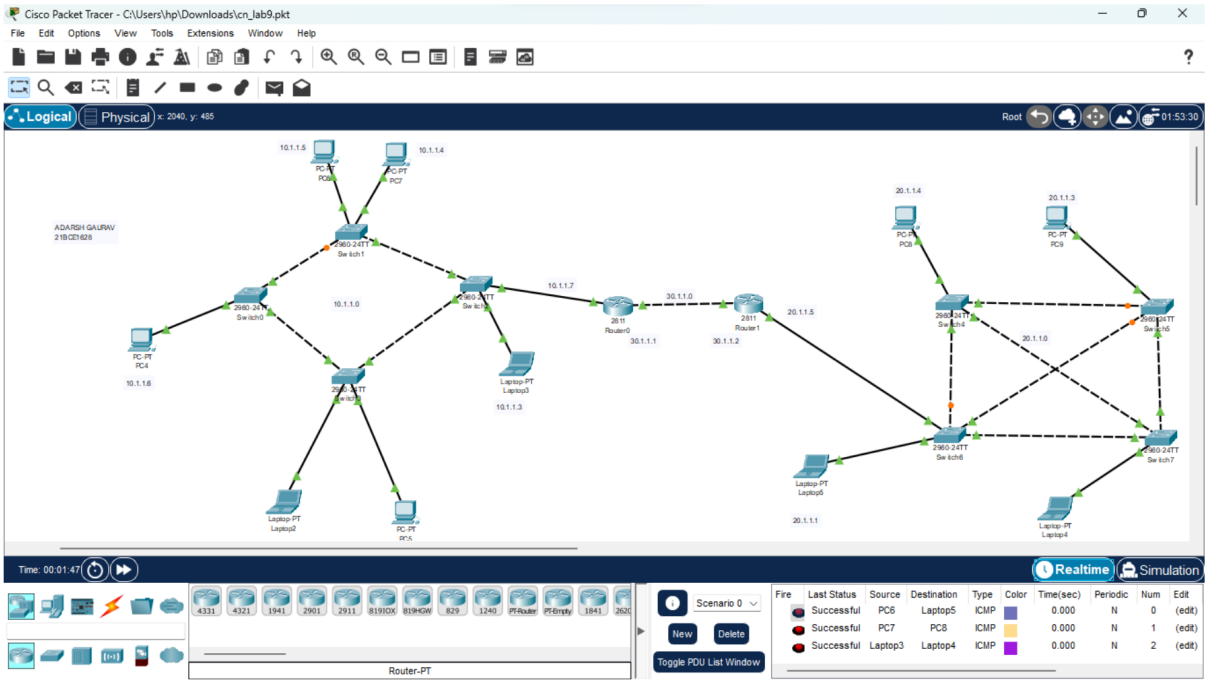
6. **Configure Mesh-specific protocols**: Configure protocols like Ethernet or VLANs to facilitate communication within the mesh segments. You may need to configure VLAN tagging or trunking on the Cisco devices to segregate traffic and ensure efficient communication.

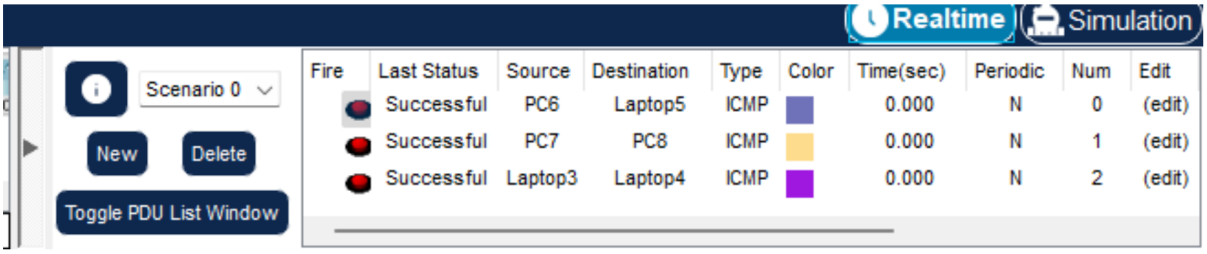
7. **Test and verify the connectivity**: Once the physical connections and configurations are in place, test the network's connectivity. Verify that devices can communicate with each other and that the desired data flow is achieved.

8. **Monitor and maintain the network**: Regularly monitor the network to identify any performance issues or faults. Implement proper maintenance procedures and keep track of any changes or updates made to the network.

OUTPUTS






Realtime
Simulation
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Edit
Successful
PC6
Laptop5
ICMP
0.000
N
0
(edit)
Successful
PC7
PC8
ICMP
0.000
N
1
(edit)
Successful
Laptop3
Laptop4
ICMP
0.000
N
2
(edit)

NAME

MOHAMMAD SHAAD

REGISTRATION NUMBER

21BCE1542

CLASS

COMPUTER NETWORKS

FACULTY

PUNITHA K MA’AM

LAB

EXERCISE 10 [PAT]

Question

Implement TCP client server program to find the shortest path algorithm using Dijkstra's Algorithm

Code

Server Side

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

#define MAX_BUFFER_SIZE 1024
#define SERVER_PORT 8080

// Function to find the shortest path using Dijkstra's algorithm
// Implementation of the algorithm goes here

int main() {
    int serverSocket, clientSocket;
    struct sockaddr_in serverAddress, clientAddress;
    char buffer[MAX_BUFFER_SIZE];

    // Create server socket
    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket < 0) {
        perror("Error creating socket");
        exit(1);
    }

    // Prepare server address structure
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(SERVER_PORT);
    serverAddress.sin_addr.s_addr = INADDR_ANY;

    // Bind the socket to the server address
```

```
    if (bind(serverSocket, (struct sockaddr *)&serverAddress,
sizeof(serverAddress)) < 0) {
        perror("Error binding");
        exit(1);
    }

    // Listen for client connections
    if (listen(serverSocket, 5) < 0) {
        perror("Error listening");
        exit(1);
    }

    printf("Server listening on port %d\n", SERVER_PORT);

    // Accept client connections
    socklen_t clientAddressLength = sizeof(clientAddress);
    clientSocket = accept(serverSocket, (struct sockaddr *)&clientAddress,
&clientAddressLength);
    if (clientSocket < 0) {
        perror("Error accepting connection");
        exit(1);
    }

    printf("Client connected\n");

    // Receive data from the client
    memset(buffer, 0, MAX_BUFFER_SIZE);
    if (recv(clientSocket, buffer, MAX_BUFFER_SIZE, 0) < 0) {
        perror("Error receiving data");
        exit(1);
    }

    // Process the received data and find the shortest path using Dijkstra's
algorithm
    // ...

    // Send the result back to the client
```

```
    char result[MAX_BUFFER_SIZE] = "Shortest path: ...";  // Replace "..." with the
actual result
    if (send(clientSocket, result, strlen(result), 0) < 0) {
        perror("Error sending data");
        exit(1);
    }

    // Close the sockets
    close(clientSocket);
    close(serverSocket);

    return 0;
}
```

Code

Client Side

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

#define MAX_BUFFER_SIZE 1024
#define SERVER_PORT 8080
#define SERVER_ADDRESS "127.0.0.1"

int main() {
    int clientSocket;
    struct sockaddr_in serverAddress;
    char buffer[MAX_BUFFER_SIZE];

    // Create client socket
    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (clientSocket < 0) {
        perror("Error creating socket");
```

```
        exit(1);
    }

    // Prepare server address structure
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(SERVER_PORT);
    if (inet_pton(AF_INET, SERVER_ADDRESS, &(serverAddress.sin_addr)) <= 0) {
        perror("Error converting server address");
        exit(1);
    }

    // Connect to the server
    if (connect(clientSocket, (struct sockaddr *)&serverAddress,
sizeof(serverAddress)) < 0) {
        perror("Error connecting to server");
        exit(1);
    }

    printf("Connected to server\n");

    // Send data to the server
    char data[] = "Graph data...";  // Replace "Graph data..." with the actual graph
data
    if (send(clientSocket, data, strlen(data), 0) < 0) {
        perror("Error sending data");
        exit(1);
    }

    // Receive the result from the server
    memset(buffer, 0, MAX_BUFFER_SIZE);
    if (recv(clientSocket, buffer, MAX_BUFFER_SIZE, 0) < 0) {
        perror("Error receiving data");
        exit(1);
    }

    printf("Result: %s\n", buffer);
```

```
    // Close the socket
    close(clientSocket);

    return 0;
}
```

OUTPUT

```
    // Graph representation
    int graph[V][V] = {
        {0, 4, 0, 0, 0, 0, 0, 8},
        {4, 0, 8, 0, 0, 0, 0, 11},
        {0, 8, 0, 7, 0, 4, 0, 0},
        {0, 0, 7, 0, 9, 14, 0, 0},
        {0, 0, 0, 9, 0, 10, 0, 0},
        {0, 0, 4, 14, 10, 0, 2, 0},
        {0, 0, 0, 0, 0, 2, 0, 1},
        {8, 11, 0, 0, 0, 0, 1, 0}
    };
```