COMPUTER NETWORK LAB

NAME: JATIN BANSAL

REG. NO: 21BPS1342

IMPLEMENTATION OF LINK STATE ROUTING ALGORITHM

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <limits.h>

#define MAX_NODES 100

// Struct to represent a node in the network
typedef struct {
    int cost;
    bool visited;
    int prev_node;
} Node;

// Function to find the node with the minimum cost
int findMinCostNode(Node nodes[], int num_nodes) {
    int min_cost = INT_MAX;
```

```c
    int min_node = -1;

    for (int i = 0; i < num_nodes; i++) {
        if (!nodes[i].visited && nodes[i].cost < min_cost) {
            min_cost = nodes[i].cost;
            min_node = i;
        }
    }

    return min_node;
}

// Function to print the shortest path from source to destination
void printShortestPath(Node nodes[], int dest) {
    if (nodes[dest].prev_node != -1) {
        printShortestPath(nodes, nodes[dest].prev_node);
    }

    printf("%d ", dest);
}

// Dijkstra's algorithm for link state routing
void dijkstra(int graph[MAX_NODES][MAX_NODES], int
num_nodes, int source) {
    Node nodes[MAX_NODES];

    for (int i = 0; i < num_nodes; i++) {
        nodes[i].cost = INT_MAX;
        nodes[i].visited = false;
        nodes[i].prev_node = -1;
    }

    nodes[source].cost = 0;

    for (int count = 0; count < num_nodes - 1; count++) {
        int current_node = findMinCostNode(nodes, num_nodes);
```

```c
      if (current_node == -1) {
         break;
      }

      nodes[current_node].visited = true;

      for (int i = 0; i < num_nodes; i++) {
         if (!nodes[i].visited && graph[current_node][i] &&
nodes[current_node].cost != INT_MAX &&
            nodes[current_node].cost + graph[current_node][i] <
nodes[i].cost) {
            nodes[i].cost = nodes[current_node].cost +
graph[current_node][i];
            nodes[i].prev_node = current_node;
         }
      }
   }

   // Print shortest paths to all nodes
   printf("Shortest paths from node %d:\n", source);
   for (int i = 0; i < num_nodes; i++) {
      if (i != source) {
         printf("Node %d: Cost = %d, Path: ", i, nodes[i].cost);
         printShortestPath(nodes, i);
         printf("\n");
      }
   }
}

int main() {
   int num_nodes, source;
   int graph[MAX_NODES][MAX_NODES];

   printf("Enter the number of nodes: ");
   scanf("%d", &num_nodes);

   printf("Enter the adjacency matrix:\n");
```

```
    for (int i = 0; i < num_nodes; i++) {
        for (int j = 0; j < num_nodes; j++) {
            scanf("%d", &graph[i][j]);
        }
    }

    printf("Enter the source node: ");
    scanf("%d", &source);

    dijkstra(graph, num_nodes, source);

    return 0;
}
```

OUTPUT:

```
student@hostserver42:~$ ./a.out
Enter the number of nodes: 9
Enter the adjacency matrix:
0 6 0 0 0 0 0 8 0
6 0 8 0 0 0 0 13 0
 0 8 0 7 0 6 0 0 2
0 0 7 0 9 14 0 0 0
0 0 0 9 0 10 0 0 0
0 0 6 14 10 0 2 0 0
0 0 0 0 0 2 0 1 6
8 13 0 0 0 0 1 0 7
0 0 2 0 0 0 6 7 0
Enter the source node: 0
Shortest paths from node 0:
Node 1: Cost = 6, Path: 0 1
Node 2: Cost = 14, Path: 0 1 2
Node 3: Cost = 21, Path: 0 1 2 3
Node 4: Cost = 21, Path: 0 7 6 5 4
Node 5: Cost = 11, Path: 0 7 6 5
Node 6: Cost = 9, Path: 0 7 6
Node 7: Cost = 8, Path: 0 7
Node 8: Cost = 15, Path: 0 7 8
student@hostserver42:~$
```

```
student@hostserver42:~/Desktop$ gcc server.c
student@hostserver42:~/Desktop$ ./a.out
Server listening on port 8080
Client connected from 127.0.0.1
Received IP address from client: 240.255.0.1
Sent response to client
```

COMPUTER NETWORKS LAB

LAB ASSIGNMENT

NAME: JATIN BANSAL

REG. NO: 21BPS1342

FILE TRANSFER PROTOCOL IMPLEMENTATION

Client:

```python
import socket
import os

class Client:
    def __init__(self):
        self.s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        self.connect_to_server()

    def connect_to_server(self):
        self.target_ip = input('Enter ip --> ')
```

```python
        self.target_port = input('Enter port --> ')

        self.s.connect((self.target_ip,int(self.target_port)))

        self.main()


    def reconnect(self):
        self.s =
socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        self.s.connect((self.target_ip,int(self.target_port)))


    def main(self):
        while 1:
            file_name = input('Enter file name on server --> ')
            self.s.send(file_name.encode())

            confirmation = self.s.recv(1024)
            if confirmation.decode() == "file-doesn't-exist":
                print("File doesn't exist on server.")

                self.s.shutdown(socket.SHUT_RDWR)
                self.s.close()
                self.reconnect()

            else:
                write_name = 'from_server '+file_name
```

```python
                if os.path.exists(write_name):
os.remove(write_name)

                with open(write_name,'wb') as file:
                    while 1:
                        data = self.s.recv(1024)

                        if not data:
                            break

                        file.write(data)

                print(file_name,'successfully downloaded.')

                self.s.shutdown(socket.SHUT_RDWR)
                self.s.close()
                self.reconnect()
client = Client()
```

Server:

```python
import socket
import threading
import os

class Server:
```

```python
    def __init__(self):
        self.s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        self.accept_connections()


    def accept_connections(self):
        ip = socket.gethostbyname(socket.gethostname())
        port = int(input('Enter desired port --> '))

        self.s.bind((ip,port))
        self.s.listen(100)

        print('Running on IP: '+ip)
        print('Running on port: '+str(port))

        while 1:
            c, addr = self.s.accept()
            print(c)

            threading.Thread(target=self.handle_client,args=(c,addr,)).start()

    def handle_client(self,c,addr):
        data = c.recv(1024).decode()

        if not os.path.exists(data):
```

```python
                c.send("file-doesn't-exist".encode())

        else:
            c.send("file-exists".encode())
            print('Sending',data)
            if data != '':
                file = open(data,'rb')
                data = file.read(1024)
                while data:
                    c.send(data)
                    data = file.read(1024)

            c.shutdown(socket.SHUT_RDWR)
            c.close()

server = Server()
```

Output:

```
student@hostserver42:~/Desktop/21bai1293s$ python3 serverftp.py
Enter desired port --> 1539
Running on IP: 127.0.1.1
Running on port: 1539
<socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
proto=0, laddr=('127.0.1.1', 1539), raddr=('127.0.0.1', 57452)>
Sending jatin.txt
<socket.socket fd=5, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
proto=0, laddr=('127.0.1.1', 1539), raddr=('127.0.0.1', 57456)>
```

student@hostserver42: ~/Desktop/21bai1293c

File   Edit   View   Search   Terminal   Help

```
student@hostserver42:~/Desktop/21bai1293c$ python3 clientftp.py
Enter ip --> 127.0.1.1
Enter port --> 1539
Enter file name on server --> jatin.txt
jatin.txt successfully downloaded.
Enter file name on server -->
```

NAME : Jatin Bansal
Reg No. : 21BRS1342
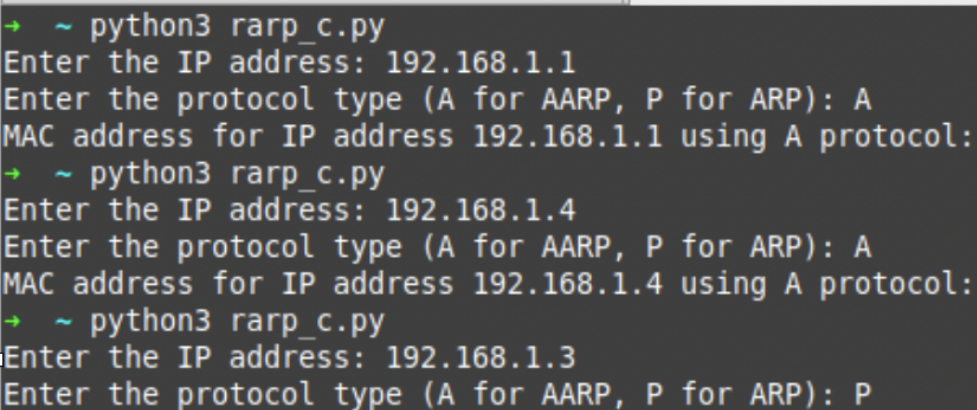# ARP & RARP Protocol for IP & MAC Address

ARP

Client

```python
import socket
import struct

SERVER_IP = '127.0.0.1'
SERVER_PORT = 9999

ip_address_str = input("Enter the IP address: ")

ip_address_binary = socket.inet_aton(ip_address_str)

client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

client_socket.sendto(ip_address_binary, (SERVER_IP, SERVER_PORT))

mac_address_binary, server_address = client_socket.recvfrom(1024)

mac_address = ':'.join('{:02x}'.format(b) for b in mac_address_binary)

print('MAC address for IP address {}: {}'.format(ip_address_str, mac_address))
```

Server

```python
import socket
import struct
SERVER_IP = '127.0.0.1'
SERVER_PORT = 9999
ip_mac_map = {
    '192.168.1.1': '00:11:22:33:44:55',
    '192.168.1.2': '11:22:33:44:55:66',
    '192.168.1.3': '22:33:44:55:66:77',
    '192.168.1.4': '33:44:55:66:77:88'
}
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind((SERVER_IP, SERVER_PORT))
while True:

    data, client_address = server_socket.recvfrom(1024)
    ip_address = socket.inet_ntoa(data)
```

```
    mac_address = ip_mac_map.get(ip_address, 'Not found')
    mac_address_binary = bytes.fromhex(mac_address.replace(':', ''))
    server_socket.sendto(mac_address_binary, client_address)
```

OUTPUT


```
→   ~ gedit arp_c.py
→   ~ python3 arp_c.py
Enter the IP address: 192.168.1.1
MAC address for IP address 192.168.1.1: 00:11:22:33:44:55
→   ~ python3 arp_c.py
Enter the IP address: 192.168.1.4
MAC address for IP address 192.168.1.4: 33:44:55:66:77:88
→   ~ python3 arp_c.py
Enter the IP address: 192.168.1.77
```

RARP CLIENT

```
import socket
import struct
SERVER_IP = '127.0.0.1'
SERVER_PORT = 9999
ip_address_str = input("Enter the IP address: ")
ip_address_binary = socket.inet_aton(ip_address_str)
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
protocol_type_binary = protocol_type_str.encode('utf-8')
data = struct.pack('!Ic', int.from_bytes(ip_address_binary, 'big'), protocol_type_binary)
client_socket.sendto(data, (SERVER_IP, SERVER_PORT))
mac_address_binary, server_address = client_socket.recvfrom(1024)
mac_address = ':'.join('{:02x}'.format(b) for b in mac_address_binary)
print('MAC address for IP address {} using {} protocol: {}'.format(ip_address_str, protocol_type_str, mac_address))
```

SERVER

```
import socket
import struct
SERVER_IP = '127.0.0.1'
SERVER_PORT = 9999
ip_mac_map = {
    '192.168.1.1': '00:11:22:33:44:55',
    '192.168.1.2': '11:22:33:44:55:66',
    '192.168.1.3': '22:33:44:55:66:77',
    '192.168.1.4': '33:44:55:66:77:88'
}
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
server_socket.bind((SERVER_IP, SERVER_PORT))
while True:
    data, client_address = server_socket.recvfrom(1024)
    ip_address, protocol_type = struct.unpack('!Ic', data)
    mac_address = None
    if protocol_type == b'A':
        mac_address = ip_mac_map.get(str(ip_address), None)
    elif protocol_type == b'P':
        mac_address = ip_mac_map.get(socket.inet_ntoa(ip_address), None)
    mac_address_binary = bytes.fromhex(mac_address.replace(':', '')) if mac_address else b''
    server_socket.sendto(mac_address_binary, client_address)
```

OUTPUT

Lab-7
Computer Networks

Jatin Bansal

21BPS1342

E2 slot

Server Side

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#define MAX_BUFFER_SIZE 1024
// Jatin Bansal 21BPS1342
unsigned char calculateChecksum(char *data, int length) {
unsigned char checksum = 0;
for (int i = 0; i < length; i++) {
checksum ^= data[i];
}
return checksum;
}
int main() {
int sockfd, newSockfd;
struct sockaddr_in serverAddr, clientAddr;
socklen_t addrSize;
char buffer[MAX_BUFFER_SIZE];
unsigned char receivedData[4];
unsigned char receivedChecksum, calculatedChecksum;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0) {
perror("Error in socket");
exit(1); }
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(12345);
serverAddr.sin_addr.s_addr = INADDR_ANY;
```

```c
if (bind(sockfd, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) < 0) {
perror("Error in bind");
exit(1);
}
if (listen(sockfd, 10) < 0) {
perror("Error in listen");
exit(1);
}
printf("Server listening on port 12345...\n");
addrSize = sizeof(clientAddr);

newSockfd = accept(sockfd, (struct sockaddr *)&clientAddr, &addrSize);
if (newSockfd < 0) {
perror("Error in accept");
exit(1); }
read(newSockfd, receivedData, sizeof(receivedData));
printf("Received data from client.\n");
printf("Data received from client: %d %d %d %d\n", receivedData[0],
receivedData[1], receivedData[2], receivedData[3]);
read(newSockfd, &receivedChecksum, sizeof(receivedChecksum));
calculatedChecksum = calculateChecksum((char *)receivedData, sizeof(receivedData));
if (receivedChecksum == calculatedChecksum) {
printf("Checksum matched. No error detected.\n");
} else {
printf("Checksum mismatch. Error detected in data.\n");
}
close(newSockfd);
close(sockfd);
return 0;
}
```

Client side

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#define MAX_BUFFER_SIZE 1024
// Jatin Banasal 21BPS1342
unsigned char calculateChecksum(char *data, int length) {
```

```c
unsigned char checksum = 0;
for (int i = 0; i < length; i++) {
checksum ^= data[i];
}
return checksum;
}
int main() {
int sockfd;
struct sockaddr_in serverAddr;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0) {
perror("Error in socket");
exit(1); }
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(12345);
serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
if (connect(sockfd, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) < 0) {
perror("Error in connect");
exit(1);
}
unsigned char data[4] = { 0b1111, 0b1101, 0b1011, 0b0001 };
printf("Data to be sent to the server: 1111 1101 1011 0001\n");
unsigned char checksum = calculateChecksum((char *)data, sizeof(data));
printf("Checksum: %d\n", checksum);
write(sockfd, data, sizeof(data));
write(sockfd, &checksum, sizeof(checksum));
printf("Data sent to the server.\n");
close(sockfd);
return 0; }
```

Output

```
┌──(jatin㉿jatin2209)-[~/Documents]
└─$ gcc Lab7_server.c

┌──(jatin㉿jatin2209)-[~/Documents]
└─$ ./a.out
Server listening on port 12345...
Received data from client.
Data received from client: 15 13 11 1
Checksum matched. No error detected.

┌──(jatin㉿jatin2209)-[~/Documents]
└─$
```

```
┌──(jatin㉿jatin2209)-[~/Documents]
└─$ gcc Lab7_client.c

┌──(jatin㉿jatin2209)-[~/Documents]
└─$ ./a.out
Data to be sent to the server: 1111 1101 1011 0001
Checksum: 8
Data sent to the server.

┌──(jatin㉿jatin2209)-[~/Documents]
└─$
```

Computer Networks
CRC Error Detection

Jatin Bansal

21BPS1342

L27+L28

Source code:
#include<stdio.h>
#include<string.h>
//Jatin Bansal 21BPS1342
#define N strlen(gen_poly)
char data[28];
char check_value[28];
char gen_poly[10];
int data_length,i,j;
void XOR(){
*   for(j = 1;j < N; j++)*
*   check_value[j] = ((check_value[j] == gen_poly[j])?'0':'1');*

}
void receiver(){
*   printf("Enter the received data: ");*
*   scanf("%s", data);*
*   printf("\n-----------------------------\n");*
*   printf("Data received: %s", data);*
*   crc();*
*   for(i=0;(i<N-1) && (check_value[i]!='1');i++);*
*   if(i<N-1)*
*   printf("\nError detected\n\n");*
*   else*
*   printf("\nNo error detected\n\n");*
}

void crc(){
*   for(i=0;i<N;i++)*
*   check_value[i]=data[i];*
*   do{*
*   if(check_value[0]=='1')*

```c
            XOR();
        for(j=0;j<N-1;j++)
            check_value[j]=check_value[j+1];
        check_value[j]=data[i++];
    }while(i<=data_length+N-1);
}

int main()
{
    printf("\nJatin Bansal\n21BPS1342 ");
    printf("\nEnter data to be transmitted: ");
    scanf("%s",data);
    printf("\n Enter the Generating polynomial: ");
    scanf("%s",gen_poly);
    data_length=strlen(data);
    for(i=data_length;i<data_length+N-1;i++)
        data[i]='0';
    printf("\n----------------------------------------");
    printf("\n Data padded with n-1 zeros : %s",data);
    printf("\n----------------------------------------");
    crc();
    printf("\nCRC or Check value is : %s",check_value);
    for(i=data_length;i<data_length+N-1;i++)
        data[i]=check_value[i-data_length];
    printf("\n----------------------------------------");
    printf("\n Final data to be sent : %s",data);
    printf("\n----------------------------------------\n");
    receiver();
        return 0;
}
```

Output

```
/tmp/w2jtA9mwT0.o
Jatin Bansal
21BPS1342
Enter data to be transmitted:110101
Enter the Generating polynomial: 1101
------------------------------------------
 Data padded with n-1 zeros : 10101000
------------------------------------------
CRC or Check value is : 111
------------------------------------------
 Final data to be sent : 10101111
------------------------------------------
Enter the received data: 110011
-----------------------------
Data received: 110011
Error detected
```

Output

```
/tmp/w2jtA9mwT0.o
Jatin Bansal
21BPS1342
Enter data to be transmitted: 110011
Enter the Generating polynomial: 101
------------------------------------------
 Data padded with n-1 zeros : 11001100
------------------------------------------
CRC or Check value is : 00
------------------------------------------
 Final data to be sent : 11001100
------------------------------------------
Enter the received data: 110011
-----------------------------
Data received: 110011
No error detected
```

21BPS1342

JATIN BANSAL

CN LAB ASSIGNMENT

SUBMITTED TO: DR. RAJESH R

1. **BUS TOPOLOGY**

   Bus topology is a type of network topology where all the nodes or devices are connected to a single cable or transmission line. In a bus network, each device is connected to the cable through a connector called a "tap" or "drop". In a bus topology, data travels along the cable in both directions from one device to another. When a device transmits data, the data is broadcasted to all the devices on the network. Each device on the network then checks to see if the transmitted data is addressed to it. If the data is not intended for a particular device, it is simply ignored

   Steps: 1. Identify the devices that you want to connect in a Bus Topology. This may include computers, servers, printers, and other networking devices.

   2. Choose a Cisco switch with enough ports to connect all of your devices. If needed, you can

connect multiple switches to create a larger Bus Topology.

 3. Connect one end of a twisted pair Ethernet cable to the Ethernet port of the device that you want to connect to the Bus Topology.

4. Connect the other end of the cable to an unused port on the Cisco switch.

5. Repeat steps 3-4 for all devices that you want to connect to the Bus Topology.

6. Assign the ip address to each computer and send the packet to test the connectivity
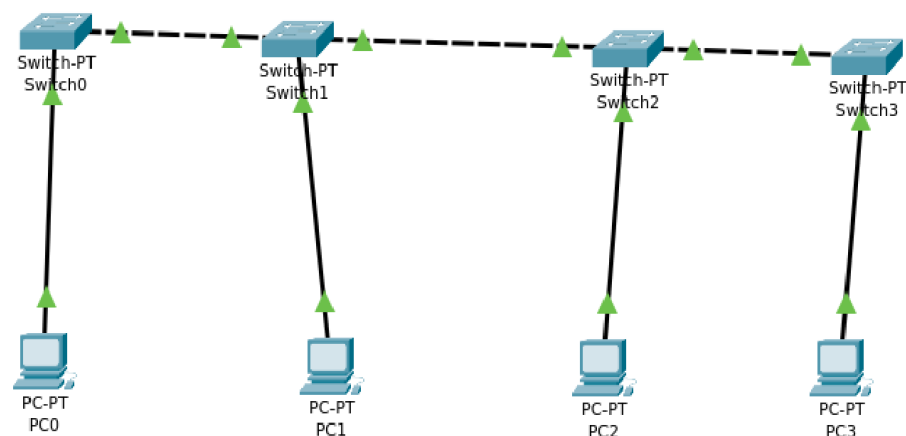
2. **RING TOPLOGY**

Ring topology is a type of network topology where the devices or nodes are connected to each other in a circular arrangement, forming a closed loop. In a ring network, data travels in one direction around the loop, passing from one device to the

next until it reaches its destination. In a ring topology, each device or node is connected to its nearest neighbors through point-to-point connections. When a device on the network sends data, it travels around the ring until it reaches its destination. Each device on the network reads the data as it passes by and forwards it to the next device, until it reaches its intended destination.
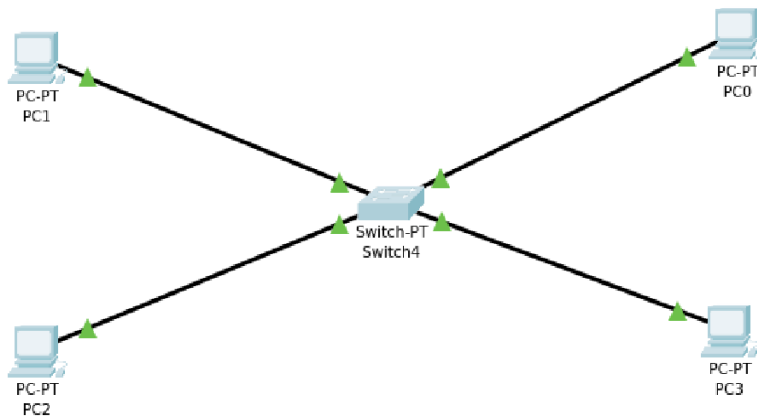
Steps: 1. Identify the devices that you want to connect in a Ring Topology. This may include computers, servers, printers, and other networking devices.

2. Choose a Cisco switch with enough ports to connect all of your devices in a ring. If needed, you can connect multiple switches to create a larger Ring Topology.

3. Connect each device to the switch using a twisted pair Ethernet cable. Each device should have a dedicated connection to the switch, creating a ring topology.

4. Configure each device with a unique IP address within the same subnet.

5. Test the connectivity between the devices by sending a message command from one device to another
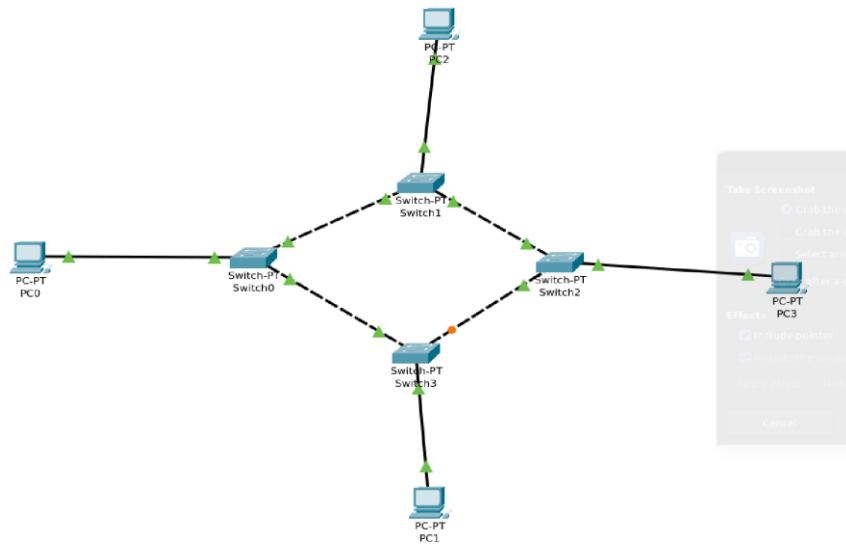
3. STAR TOPOLOGY

Star topology is a type of network topology where each device or node is connected to a central hub or switch, forming a star-shaped network. In a star network, all communication between devices on the network is routed through the central hub or switch. In a star topology, each device on the network has its own dedicated point-to-point connection to the central hub or switch. When a device on the network sends data, it is transmitted to the hub or switch, which then forwards the data to its intended destination. This allows for fast data transmission and easy scalability, as new devices can be added to the network by simply connecting them to the central hub or switch.

Steps: 1. Identify the devices that you want to connect in a Star Topology. This may include computers, servers, printers, and other networking devices.

2. Choose a Cisco switch with enough ports to connect all of your devices. If needed, you can connect multiple switches to create a larger Star Topology.

3. Connect each device to the switch using a twisted pair Ethernet cable. Each device should have a dedicated connection to the switch, creating a star topology.

4. Configure each device with a unique IP address within the same subnet.

5. Test the connectivity between the devices by sending a message command from one device to another

4. MESH TOPOLOGY

Mesh topology is a type of network topology where every device or node is connected to every other device or node on the network. This creates multiple paths for data to travel from one device to another, providing redundancy and ensuring that if one path fails, there are alternative paths for the data to take. In a mesh topology, each device or node acts as a repeater, forwarding data to other devices on the network until it reaches its destination. This allows for high-speed data transmission and reduces the chance of data collisions.

Steps: 1. Identify the devices that you want to connect in a Mesh Topology. This may include computers, servers, printers, and other networking devices.

2. Choose a Cisco switch with enough ports to connect all of your devices. If needed, you can connect multiple switches to create a larger Mesh Topology.

3. Connect each device to the switch using a twisted pair Ethernet cable. Each device should have a dedicated connection to the switch, creating a fully connected Mesh Topology.

4. Configure each device with a unique IP address within the same subnet.

5. Test the connectivity between the devices by sending a message command from one device to another

Output
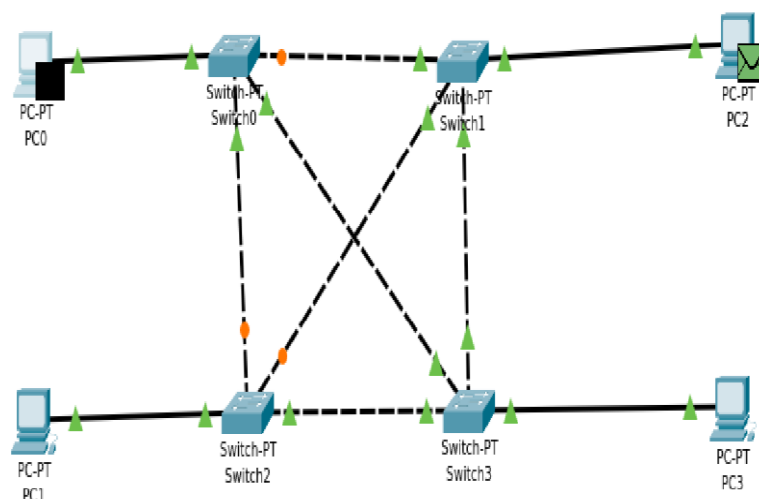
21BPS1342
JATIN BANSAL
CN LAB ASSIGNMENT
LAB SLOT : L27+28
SUBMITTED TO : DR. RAJESH R

SOCKET PROGRAMMING USING UDP

Q1.
SERVER SIDE:

```c
#include <netinet/in.h>
#define PORT    8080
#define MAXLINE 1024


int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello JATIN BANSAL from server";
    struct sockaddr_in servaddr, cliaddr;

    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; // IPv4
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);


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

    int len, n;

    len = sizeof(cliaddr); //len is value/result

    n = recvfrom(sockfd, (char *)buffer, MAXLINE,
                MSG_WAITALL, ( struct sockaddr *) &cliaddr,
                &len);
```
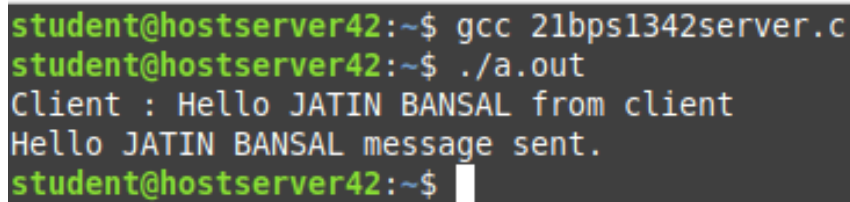
```
        buffer[n] = '\0';
        printf("Client : %s\n", buffer);
        sendto(sockfd, (const char *)hello, strlen(hello),
                MSG_CONFIRM, (const struct sockaddr *) &cliaddr,
                    len);
        printf("Hello JATIN BANSAL message sent.\n");

        return 0;
}
```


CLIENT SIDE:
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT     8080
#define MAXLINE 1024


int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello JATIN BANSAL from client";
    struct sockaddr_in      servaddr;

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

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

```c
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

    int n, len;

    sendto(sockfd, (const char *)hello, strlen(hello),
        MSG_CONFIRM, (const struct sockaddr *) &servaddr,
            sizeof(servaddr));
    printf("Hello JATIN BANSAL message sent.\n");

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

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

```
student@hostserver42:~$ gcc 21bps1342client.c
student@hostserver42:~$ ./a.out
Hello JATIN BANSAL message sent.
Server : Hello JATIN BANSAL from server
student@hostserver42:~$ []
```

Lab 3
Computer Networks

Name: Jatin Bansal
Reg No: 21BPS1342
Slot: L27+L28
Submitted to: Dr Rajesh R

Date & Time Server

Server:

```
#include"unistd.h"
#include"netinet/in.h"
#include"stdlib.h"
#include"sys/socket.h"
#include"stdio.h"
#include"string.h"
#include"time.h"
//Jatin Bansal 21BPS1342
void main( ) {
struct sockaddr_in sa;
struct sockaddr_in cli;
int sockfd,conntfd;
int len,ch;
char str[100];
time_t tick;
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0) {
printf("error in socket\n");
exit(0);
}
else
printf("Socket opened");
bzero(&sa,sizeof(sa));
sa.sin_port=htons(5600); sa.sin_addr.s_addr=htonl(0);
if(bind(sockfd,(struct sockaddr*)&sa,sizeof(sa))<0) {
printf("Error in binding\n");
}
else
printf("Binded Successfully");
listen(sockfd,50);
for(;;) { len=sizeof(ch); conntfd=accept(sockfd,(struct sockaddr*)&cli,&len); printf("Accepted");
tick=time(NULL); snprintf(str,sizeof(str),"%s",ctime(&tick)); printf("%s",str);
write(conntfd,str,100);
}
}
```

```c
#include"unistd.h"
#include"netinet/in.h"
#include"stdlib.h"
#include"sys/socket.h"
#include"stdio.h"
#include"string.h"
#include"time.h"
//Jatin Bansal 21BPS1342
void main( ) {
struct sockaddr_in sa;
struct sockaddr_in cli;
int sockfd,conntfd;
int len,ch;
char str[100];
time_t tick;
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0) {
printf("error in socket\n");
exit(0);
}
else
printf("Socket opened");
bzero(&sa,sizeof(sa));
sa.sin_port=htons(5600); sa.sin_addr.s_addr=htonl(0);
if(bind(sockfd,(struct sockaddr*)&sa,sizeof(sa))<0) {
printf("Error in binding\n");
}
else
printf("Binded Successfully");
listen(sockfd,50);
for(;;) { len=sizeof(ch); conntfd=accept(sockfd,(struct sockaddr*)&cli,&len); printf("Accepted");
tick=time(NULL); snprintf(str,sizeof(str),"%s",ctime(&tick)); printf("%s",str);
write(conntfd,str,100);
}
}
```

Client:

```c
#include"unistd.h"
#include"netinet/in.h"
#include"stdlib.h"
#include"sys/socket.h"
#include"stdio.h"
#include"string.h"
#include"time.h"
//Jatin Bansal 21BPS1342
void main() {
struct sockaddr_in sa,cli;
int n,sockfd;
int len;
char buff[100];
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0) {
printf("\nError in Socket");
exit(0);
}
else
printf("\nSocket is Opened");
bzero(&sa,sizeof(sa));
sa.sin_family=AF_INET;
sa.sin_port=htons(5600);
if(connect(sockfd,(struct sockaddr*)&sa,sizeof(sa))<0) {
printf("\nError in connection failed");
exit(0);
}
else
printf("\nconnected successfully");
if(n=read(sockfd,buff,sizeof(buff))<0) {
printf("\nError in Reading");
exit(0);
}
else
{
printf("\nMessage Read %s",buff); }
```

}

```c
#include"unistd.h"
#include"netinet/in.h"
#include"stdlib.h"
#include"sys/socket.h"
#include"stdio.h"
#include"string.h"
#include"time.h"
//Jatin Bansal 21BPS1342
void main() {
struct sockaddr_in sa,cli;
int n,sockfd;
int len;
char buff[100];
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0) {
printf("\nError in Socket");
exit(0);
}
else
printf("\nSocket is Opened");
bzero(&sa,sizeof(sa));
sa.sin_family=AF_INET;
sa.sin_port=htons(5600);
if(connect(sockfd,(struct sockaddr*)&sa,sizeof(sa))<0) {
printf("\nError in connection failed");
exit(0);
}
else
printf("\nconnected successfully");
if(n=read(sockfd,buff,sizeof(buff))<0) {
printf("\nError in Reading");
exit(0);
}
else
{
printf("\nMessage Read %s",buff); }
}
```

Output:
Server:

```
student@hostserver42:~$ gcc server_time.c
student@hostserver42:~$ ./a.out
Socket openedBinded SuccessfullyAcceptedFri May 12 10:43:17 2023
```

Client:

```
student@hostserver42:~$ gcc client_timer.c
student@hostserver42:~$ ./a.out

Socket is Opened
connected successfully
Message Read Fri May 12 10:43:17 2023
student@hostserver42:~$
```