Exp.NO:-1 Design a local Area Network for
Data:-07.08.24 Organization of 5 laboratories with inter departmental connectivity and show that simulated output

AIM:-

To design a local area Network for Organization of 5 laboratories with inter departmental connectivity and show that the simulated output.

Algorithm.

Step:-1 Start the process.

step:-2 Install the dependecies for Cisco Packet racer.

step:-3 Use the 10 Pc's connected them with five switches (5 Laboratories) by the Fast Ethernet

Step:-4 Connect the Switch with the Routers by the Ethernet connection

step:-5 configure the ip address for the router with specified enable and exit

Step:-6 Fix the ip address for Pc's (Separately)

Step:-7 Pass the messages from one Pc to neighbouring Pc

Step:8 · When the messages passes successfully through realtime and simulation

Step:9 Succes message is displayed.

Step:10 End the Procers.

Program

Router-config

Router> enable

Router# configure terminal

Enter configuration commands, one per Line.

Router(config)# interface go/0

Router(config-if)# ip address 192.168.141.255.255.255.0

Router(config-if)# no shutdown.

Router(config-if)# exit

Router(config)# interface go/0

Router(config-if)# ip address 192.168.1.1 255.255.255.0

Router(config-if)# no shutdown.

Router(config-if)# exit

Router.

Switch-config

Switch > enable

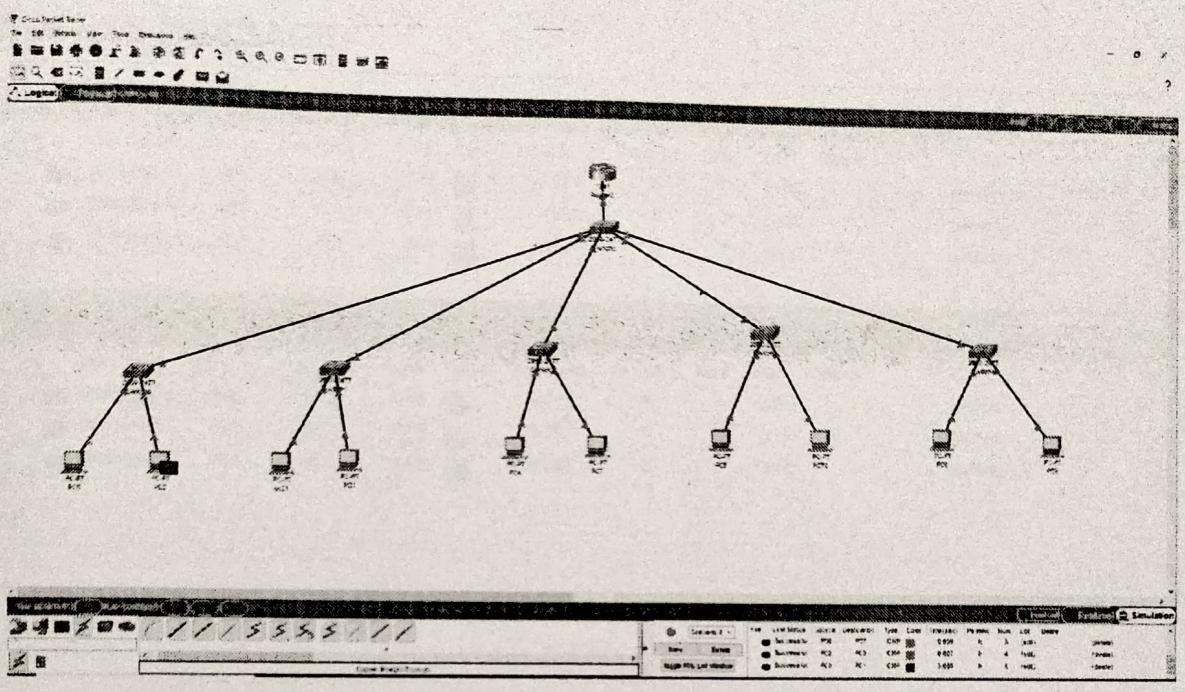
Switch# configure terminal

Switch (config)# interface go/1
Switch (config-if) #ip address 192.168.1.1 255.255.255.0
Switch.(config-if)# no shutdown.
Switch (config-if)# exit
Switch (config)# interface go/1
Switch. (config-if)# no shutdown.
Switch (config-if)# exit
Switch

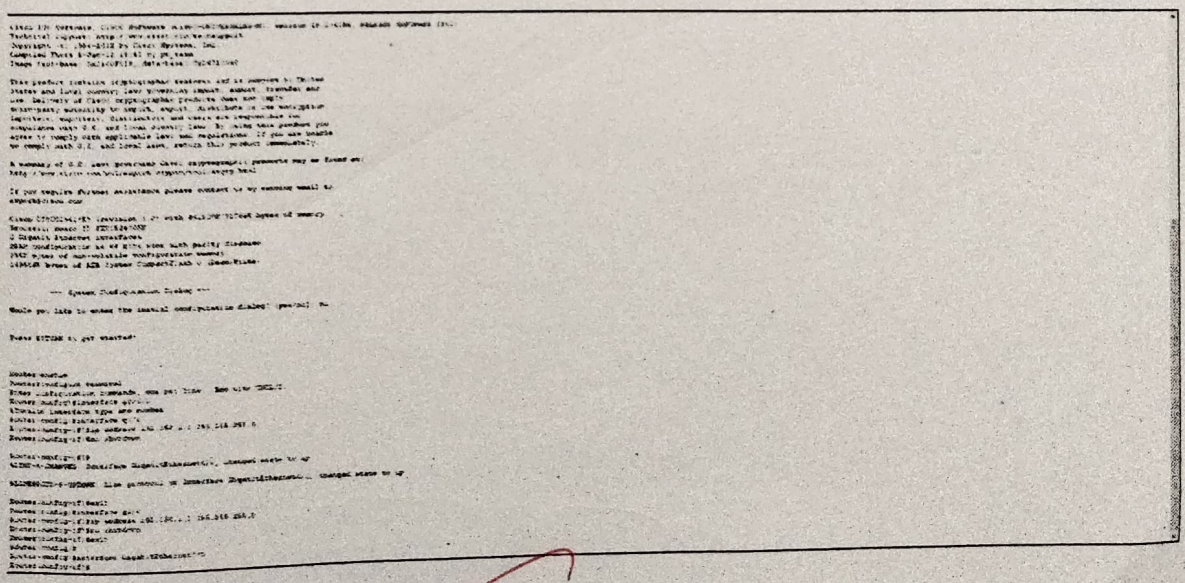
PC- configuration
Interface : FastEthernet

IPV4 Address: 192.168.3.11
Subnet Mask: 255.255.255.0
Default Gateway: 0.0.0.0
DNS Server : 0.0.0.0

OUTPUT:



CONFIGURATION:



SUCESS MESSAGE:

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC6	ICMP		0.000	N	5	(edit)	(delete)
	Successful	PC0	PC7	ICMP		0.000	N	6	(edit)	(delete)
	Successful	PC1	PC8	ICMP		0.000	N	7	(edit)	(delete)

Event List | Realtime | Simulation

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC6	PC7	ICMP		0.606	N	3	(edit)	(delete)
	Successful	PC2	PC3	ICMP		0.607	N	4	(edit)	(delete)
	Successful	PC0	PC1	ICMP		3.685	N	5	(edit)	(delete)

	Marks	
AIM	10	10
ALGORITHM	20	20
PROGRAM	40	35
OUTPUT	10	10
VIVA VOICE	10	5
RESULT	10	90
		90

RESULT :-

Local Area Network for Organization of 5 hub interdetermental connectivity has been designed successfully and message passed successfully.

7/8/24.

Exp. No:-2
20.09.2024

Implement Cyclic Redundancy check and Checksum algorithms to detect and correct errors while Transferring files (jpeg, txt, csv) over unreliable networks.

AIM:

To implement cyclic redundancy check and checksum algorithms to detect and correct errors while Transferring files.

Algorithm

step:-1 open the file in binary mode using open() function

step:-2 Read the entire content of the file using the .read() method.

step:-3 Calculate the CRC of the file content using binascii.crc32() function

step:-4 Apply the bitmask to ensure the CRC is within a 32 bit range.

step:-5 Return the CRC value

Send the file with CRC

Step:-1 First call the calc_crc() function and get the returned CRC value

Step:-2 Again Open the file in binary mode and read the content using .read() function

Step:-3 Return both the file data and CRC value

Reciever

Step:-1 Reciever the file and CRC value.

Step:-2 Recompute the CRC for the received file.

Step:-3 Then compare the recomputed CRC with the returned CRC value

check the CRC value match

Step:-4 If yes

Step:-5 Print the file has been received successfully. If not print the file is corrupted

CHECKSUM :-

Step:-1 - Get the input as an binary digit dividend and divisor

Step:-2 Add the n bits and if carry is there add that carry also at the end.

Step:-3 Now 1's complement the answer and that is called checksum

Step:-4 The sender sends the n bits and checksum to the receiver

Step:-5 In receiver side, it adds all the n bits and checksum

Step:-6 If the sum contains all '1's means no error otherwise error is spotted.

CRC

```
def xor(a,b):
    result = []
    for i in range(1, len(b)):
        result.append('0' if a[i] == b[i] else '1')
    return ''.join(result)
def mododiv(div, divisor):
    pick = len(divisor)
    tmp = dividend[:pick]

    while pick < len(dividend):
        if temp[0] == '1':
            tomp = xor(divisor, tmp) + div[pick]
        else:
            temp = xor('0' * pick, tmp) + div[pick]
        pick += 1
    if tmp[0] == '1':
        tmp = xor(divisor, tmp)

    else:
        tmp = xor('0' * pick, tmp)

    return tmp

def encode_crc(data, divisor):
    l_key = len(divisor)
    append_data = data + '0' * (l_key - 1)
    remainder = mod2div(append_data, divisor)
    return data + remainder
```

```
def crc_check(received_data, divisor):
    remainder = mod2div(received_data, divisor)
    return '1' not in remainder

def main():
    dividend = input("Enter a dividend:")
    divisor = input("Enter a divisor:")
    encoded_data = encode_src(dividend, divisor)
    print("Encoded data with CRC:", encoded_data)

    crc_status = crc_check(encoded_data, divisor)
    print("CRC status:", "No Error" if crc_status
                        else "Error Detected")

if __name__ == "__main__":
    main()
```

Output:

```
Enter the dividend: 1001
Enter the divisor : 1101
Encoded Data with CRC : 1001011
CRC status : No Error.
```

checksum

```
def xor_division(dividend, divisor):
    remainder = dividend[:len(divisor)]
    for i in range(len(dividend) - len(divisor) + 1):
        if remainder[0] == '1':
            remainder = ''.join(str[int(remainder[i]) ^
                        int(divisor[i]))
                for j in range(len(divisor))[1]) + dividend
                        [len(divisor) + i : len(divisor) + i + 1]
        else:
            remainder = remainder[1:] + dividend
                [len(divisor) + i : len(divisor) + i + 1]
    return remainder
def send_dividend_with_division_and_checksum(dividend,
                                             divisor):
    padded_dividend = dividend + '0' * (len(divisor) - 1):
    remainder = xor_division(padded_dividend, divisor)
    checksum_value = sum([int(bit) for bit in remainder])
    print(f"Orginal dividend : {dividend}")
    print(f"Remainder after division : {remainder}")
    print(f"checksum : {checksum_value}")
    transmitted_data = dividend + remainder
    return transmitted_data, checksum_value
```

```
def receiver (received_data, divisor):
    remainder = xor_division ( received_data , divisor)
    received_checksum = sum ( [int(bit) for bit in
                                remainder ])
    print (f" Remainder after division at receiver:
                                {remainder }")

    return 1 if received_checksum == 0 else 0

divisor = input (" Enter a divisor:")
dividend = input (" Entora dividend:")
sent_data , check_sum = send_dividend_with_division_and_
                        checksum (dividend, divisor)

result = receiver (sent_data , divisor)

if result == 1:
    print ("Data received correctly (checksum
                                match).")

else:
    print ("Data corrupted (checksum mismatch).")
```

Output

```
Enter a divisor: 000
Enter a dividend: 001
Checksum: 0
Remainder after division at receiver: 000
Data Received Correctly
```

	Marks	
AIM	10	10
ALGORITHMS	20	20
PROGRAM	40	40
OUTPUT	10	10
VIVA VOICE	10	10
RESULT	10	10
		100

RESULT:-

Cyclic Redundancy check and checksum algorithms were used to detected and correct the errors while pass the binary value as input were successfully created and executed.

21/9/24

Ex-No:-3
Date:-09-24.

Pg.

Configure routers and switches to manage and optimize network traffic, ensuring reliable internet connectivity and efficient data flow for home or office networks and show the simulated output

AIM:-

To configure routers and switches to manage and optimize network traffic, ensuring reliable internet connectivity and efficient data flow for home (or) office networks and show the simulated output.

ALGORITHM:-

Step:-1 Determine the number of computer (or) other...

Step:-2 Identify the type of connections (wired (or) wireless

Step:-3 Choose the Network Topology and Place the two routers.

Step:-4 Connect the computer or other devices to the Switches.

Step:-5 Then configure the router

Step:-6 Set up the WAN interface (eg: fastEthernet)

Step:-7 Configure the switches connected to the devices.

Step:-8 From the command Prompt of a PC, ping the routers IP address.

ping 192.168.1.1

Step:-9 click on the "Simulation" mode in Packet Tracer. and Observe the data packets between devices.

Step:-10 check the data flows are expected.

Step:-11 End the Process

Router:-

Router > enable

Router # configure terminal.

Router (config) # Interface gigabitEthernet 0/0

Router (config-if) # ip address 192.168.10.2. 255.255.255.0

Router (config-if) # no shutdown.

Router (config-if) # exit

Router (config) # default-router 192.168.10.2.

Switch:-

Switch > enable.

Switch # configure terminal

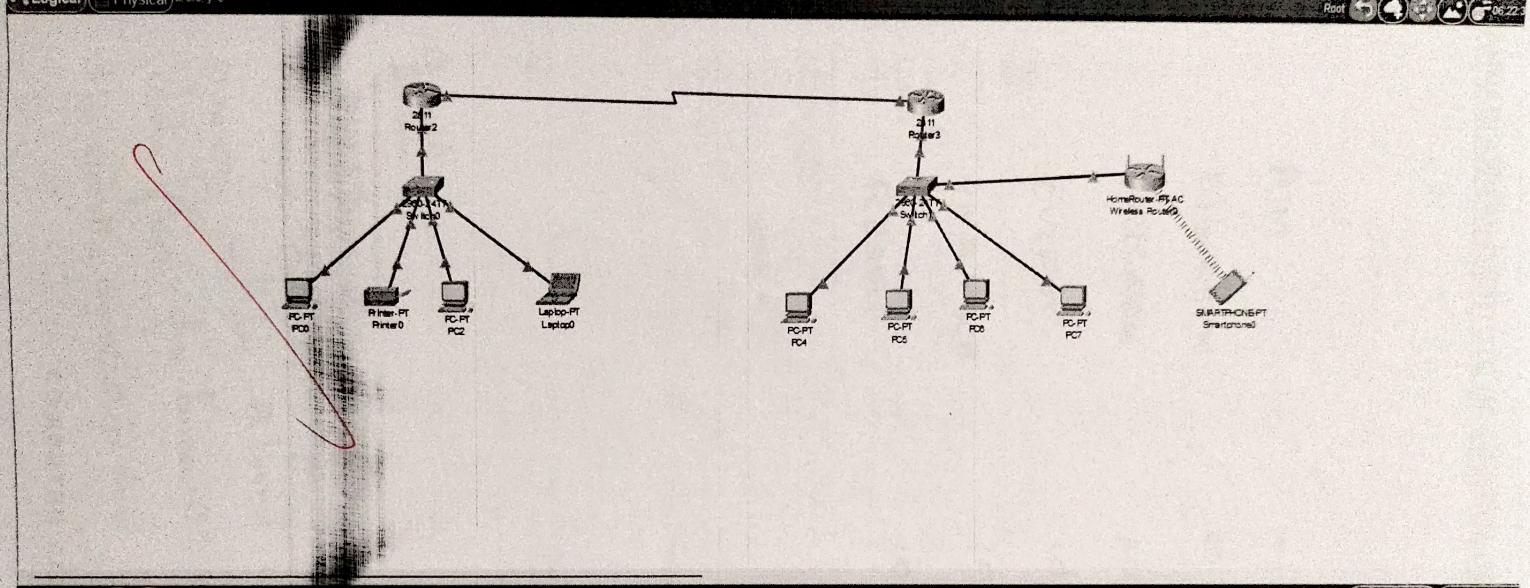
Switch (config) # Vlan 10

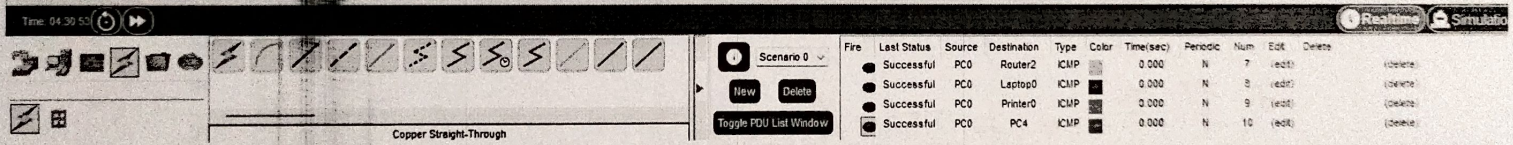
Switch (config-vlan) # name [redacted]. mk

Switch (config-vlan) # exit

Switch (config) # vlan 20

Switch (config-vlan) # name AK.


Router2
Router3
Switch0
Switch1
HomeRouter-PT-AC
Wireless Router0
PC-PT
PC0
Printer-PT
Printer0
PC-PT
PC2
Laptop-PT
Laptop0
PC-PT
PC4
PC-PT
PC5
PC-PT
PC6
PC-PT
PC7
SMARTPHONE-PT
Smartphone0


Time: 04:39:53
Realtime
Simulatio
Copper Straight-Through
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Edit
Delete
Successful PC0 Router2 ICMP 0.000 N 7 (edit) (delete)
Successful PC0 Laptop0 ICMP 0.000 N 8 (edit) (delete)
Successful PC0 Printer0 ICMP 0.000 N 9 (edit) (delete)
Successful PC0 PC4 ICMP 0.000 N 10 (edit) (delete)

35

AIM	10	10
ALGORITHM	20	20
PROGRAM	40	40
OUTPUT	10	10
VIVA VOICE	10	10
RESULT	10	10
		100

Result:

Thus the configuration routers and switch to manage and optimize network traffic through ensuring reliable internet connectivity and effective data flow for home or office networks and shown the the result were expected 21/9/24.

Exp No:- 4
Date:-

Configure the network address using Address Resolution Protocol (ARP) to map IP addresses to MAC address in a college network and Reverse ~~in a college~~ ARP ~~Netwo~~ (RARP) to obtain their IP addresses from a available server and show the simulated output

AIM

To configure the network address using Address Resolution Protocol (ARP) to map IP addresses to MAC address in a college network and Reverse ARP addresses from a available server and show the simulated Output.

ALGORITHM

Step:-1 Start the Process.

Step:-2 Drag and Drop the Pc's, laptop's and Server in the cisco Packet Racer.

Step:-3 Connect the Pc's with copper cable

Step:-4 In Pc's → Desktop → ipconfig set the different IP address for the Pc's

Step:-5 : In Pc's Open the command prompt ping the other Pc's or Server with IP addresses

Step:-6 The messages will be passed to matched MAC address

Step:-7 In command prompt enter the arp -a the ARP table will displayed with MAC address

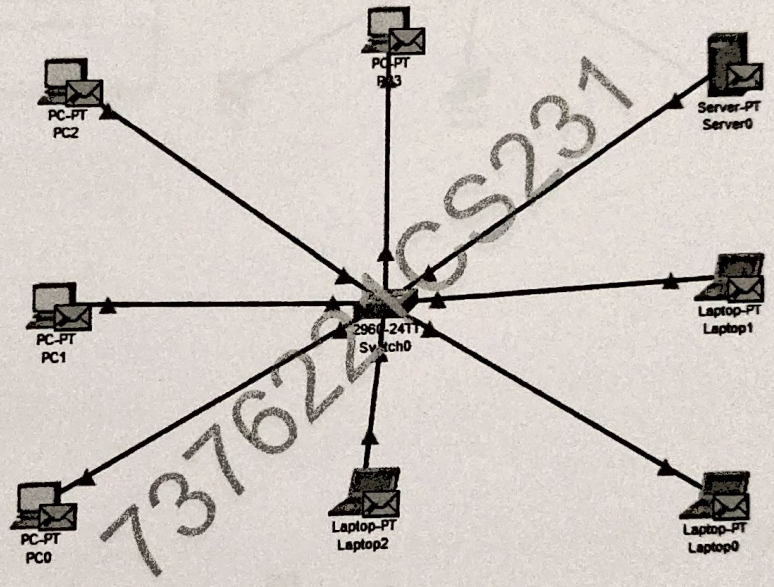
Step:-8 Then on the stimulation, the stimulated messages will be passed.

Step:-9 In RARP Reverse Address Resolution Protocal MAC address is mapped to matched IP address

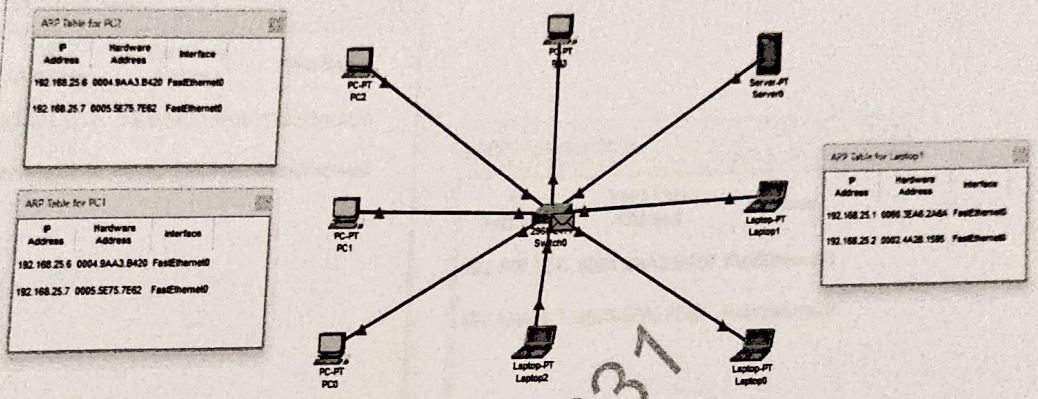
Step:-10 In Command prompt enter ipconfig then the matched IP address will be displayed.

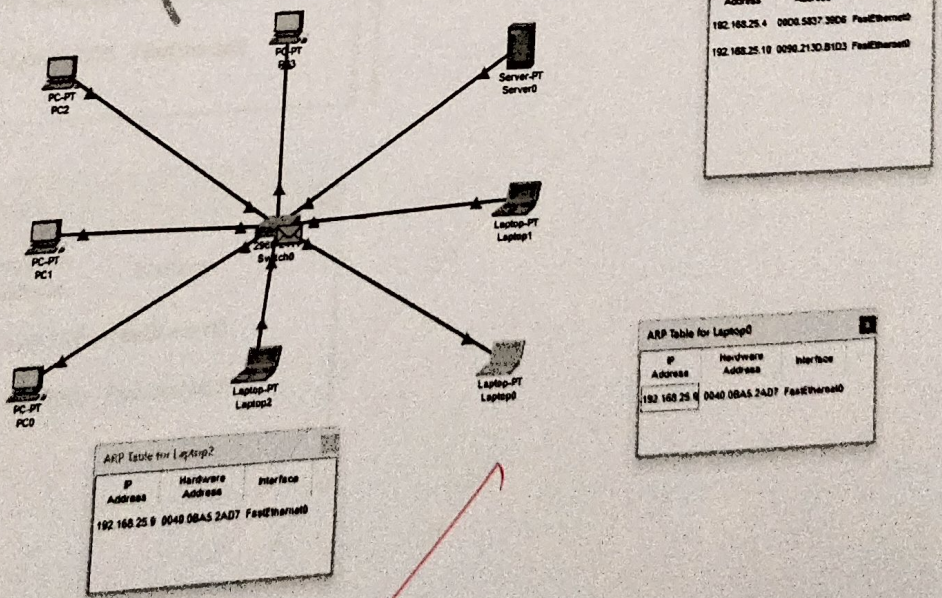
Step:-11 End the Process.

OUTPUT:



z





ARP TABLE:

ARP Table for PC3

IP Address	Hardware Address	Interface
192.168.25.4	00D0.5837.39D6	FastEthernet0
192.168.25.10	0090.213D.B1D3	FastEthernet0

ARP Table for PC2

IP Address	Hardware Address	Interface
192.168.25.6	0004.9AA3.B420	FastEthernet0
192.168.25.7	0005.5E75.7E62	FastEthernet0

ARP Table for Laptop1

IP Address	Hardware Address	Interface
192.168.25.1	0060.3EA6.2A6A	FastEthernet0
192.168.25.2	0002.4A26.1595	FastEthernet0

ARP Table for Laptop2

IP Address	Hardware Address	Interface
192.168.25.9	0040.0BA5.2AD7	FastEthernet0

ARP Table for Server0

IP Address	Hardware Address	Interface
192.168.25.1	0060.3EA6.2A6A	FastEthernet0
192.168.25.2	0002.4A26.1595	FastEthernet0

AIM	10	10
ALGORITHM	20	20
PROGRAM	40	36
OUTPUT	10	10
VIVA VOICE	10	8
OUTPUT	10	94

RESULT:

Thus the configure the network address using. Address Resolution Protocol (ARP) to map IP address to MAC addresses in a college network and Reverse ARP (RARP) to obtain their IP addresses from a available server were successfully designed and stimulated output message were passed.

18/10/24

Exp.No:5

Date: 22.10.22

Implement Distance vector and link state Routing algorithm to determine the most effective path for data transmission across large corporate networks

AIM

To implement Distance Vector and link state Routing Algorithm to determine the most effective path for data transmission across large corporate networks

ALGORITHM (Distance vector Routing)

Step:-1 Input the number of vectors and their neighbours with distances. Initialize each router's distance vector.

step:-2 Each router shares its distance vector with its neighbours. calculating distances to each destination through its neighbors

Step:-3 After each iteration, check if any router's distance vector has been updated. if any update occurs, continue, otherwise stop

Step:-4 Repeat the step-3 until no more updates happen, until the shortest distance found

Step:5 : Output the final distance vector for each router showing the shortest path to every other router.

Link state Routing

Step:-1 Input the number of routers, the cost matrix and the source router, Initialize the distance visited array and previous node array for all routers.

Step:-2 In each iteration, find the unvisited node with smallest distance from the source code.

Step:-3 Once the next node with the smallest distance is found, mark it is visited to avoid reprocessing it

Step:-4 For each unvisited neighbour of the current node. update the distance if a shorter path is found through the current node

Step:-5 After processing all nodes, print the shortest path from the source to each router, along the total distance for that path.

1. DISTANCE VECTOR ALGORITHM:

```
#include <stdio.h>
#include <limits.h>
#define MAX 10
struct Router {
    int distance[MAX];
    int via[MAX];
};
void distanceVectorRouting(int costMatrix[MAX][MAX], int n) {
    struct Router routers[MAX];
    int i, j, k;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            routers[i].distance[j] = costMatrix[i][j];
            routers[i].via[j] = j;
        }
    }
    int updated;
    do {
        updated = 0;
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                for (k = 0; k < n; k++) {
                    if (routers[i].distance[j] > costMatrix[i][k] + routers[k].distance[j]) {
                        routers[i].distance[j] = costMatrix[i][k] + routers[k].distance[j];
                        routers[i].via[j] = k;
                        updated = 1;
                    }
                }
            }
        }
    } while (updated);
    for (i = 0; i < n; i++) {
        printf("Router %d:\n", i + 1);
        for (j = 0; j < n; j++) {
```

```
            printf("  To %d: Distance = %d, Via = %d\n", j + 1, routers[i].distance[j],
routers[i].via[j] + 1);
        }
        printf("\n");
    }
}
int main() {
    int costMatrix[MAX][MAX], n;
    printf("Enter number of routers: ");
    scanf("%d", &n);
    printf("Enter cost matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &costMatrix[i][j]);
            if (i != j && costMatrix[i][j] == 0) {
                costMatrix[i][j] = INT_MAX; // Infinity
            }
        }
    }
    distanceVectorRouting(costMatrix, n);
    return 0;
}
```

OUTPUT:

```
/tmp/gAAWOOr7nP.o
Enter number of routers: 3

Enter cost matrix:
0 2 7
2 0 1
7 1 0
Router 1:
   To 1: Distance = 0, Via = 1
   To 2: Distance = 2, Via = 2
   To 3: Distance = 3, Via = 2

Router 2:
   To 1: Distance = 2, Via = 1
   To 2: Distance = 0, Via = 2
   To 3: Distance = 1, Via = 3

Router 3:
   To 1: Distance = 3, Via = 2
   To 2: Distance = 1, Via = 2
   To 3: Distance = 0, Via = 3
```

2.LINK STATE ROUTING ALGORITHM

```
#include <stdio.h>
#include <limits.h>
#define MAX 10
void dijkstra(int costMatrix[MAX][MAX], int n, int src) {
   int dist[MAX], visited[MAX], prev[MAX];
   int i, j, count, minDist, nextNode;
   for (i = 0; i < n; i++) {
      dist[i] = costMatrix[src][i];
      prev[i] = src;
      visited[i] = 0;
```

```
    }
    dist[src] = 0;
    visited[src] = 1;
    count = 1;
    while (count < n - 1) {
        minDist = INT_MAX;
        for (i = 0; i < n; i++) {
            if (dist[i] < minDist && !visited[i]) {
                minDist = dist[i];
                nextNode = i;
            }
        }
        visited[nextNode] = 1;
        for (i = 0; i < n; i++) {
            if (!visited[i] && minDist + costMatrix[nextNode][i] < dist[i]) {
                dist[i] = minDist + costMatrix[nextNode][i];
                prev[i] = nextNode;
            }
        }
        count++;
    }
    for (i = 0; i < n; i++) {
        if (i != src) {
            printf("\nShortest path from Router %d to Router %d is: ", src + 1, i + 1);
            printf("%d", i + 1);
            j = i;
            do {
                j = prev[j];
                printf(" <- %d", j + 1);
            } while (j != src);
            printf("\nDistance: %d\n", dist[i]);
        }
    }
}
int main() {
    int costMatrix[MAX][MAX], n, src;
    printf("Enter number of routers: ");
```

```
    scanf("%d", &n);
    printf("Enter cost matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &costMatrix[i][j]);
            if (i != j && costMatrix[i][j] == 0) {
                costMatrix[i][j] = INT_MAX;  // Infinity
            }
        }
    }
    printf("Enter source router: ");
    scanf("%d", &src);
    dijkstra(costMatrix, n, src - 1);
    return 0;
}
```

Output

```
/tmp/L3OrAsnNkG.o
Enter number of routers: 3
Enter cost matrix:
0 2 7
2 0 1
7 1 0
Enter source router: 1

Shortest path from Router 1 to Router 2 is: 2 <- 1
Distance: 2

Shortest path from Router 1 to Router 3 is: 3 <- 2 <- 1
Distance: 3


=== Code Execution Successful ===
```

AIM	10	10
ALGORITHM	20	20
PROGRAM	40	38
OUTPUT	10	10
VIVA VOICE	10	10
RESULT	10	10
		98

RESULT

Thus the implementation of Distance Vector routing and Link state algorithm to determine the most effective path for data transmission across large corporate wore successfully executed.

23/10

Exp.No:. 06
Date :- 28.10.24.

Develop a real time chat application that user TCP for reliable ordered communication and User UDP for faster connectionless messaging

AIM

To Develop a real time chat application that user TCP for reliable ordered communication and User UDP for faster connectionless messaging.

ALGORITHM

Step:-1 Start the Process

Step:-2 Intialize the TCP socket and connect to the TCP Server and UDP socket for sending typing indicators to the UDP Server

Step:-3 Create a Tkinter window like TCP/UDP chat Application. Bind a key event on the entry box to send typing indicators.

Step:-4 Retrieve the message from the entry box. if the message is not empty, encode it and Send via the TCP socket

Step :-5 Continuously editing the input from the client side.

Step :-6 Uses the multiple client side for more more message passing

Step :-7 End the Process.

SERVER.JS

```
import socket
import threading

# TCP and UDP server setup
tcp_server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp_server.bind(('localhost', 9000))
tcp_server.listen()

udp_server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
udp_server.bind(('localhost', 9001))

clients = []
typing_clients = set()  # Set to keep track of clients who are currently typing

# Handle TCP messages from clients
def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")
    clients.append(conn)
    while True:
        try:
            message = conn.recv(1024).decode('utf-8')
            if message:
                print(f"[{addr}] {message}")
                broadcast_tcp(message, conn)
            else:
                break
        except:
            break
    conn.close()
    clients.remove(conn)

# Broadcast TCP message to all connected clients with a TCP prefix for clarity
def broadcast_tcp(message, sender):
    for client in clients:
        if client != sender:
            try:
                client.send(f"TCP:{message}".encode('utf-8'))
            except Exception as e:
```

```
            print(f"[ERROR] Could not send TCP message to {client}: {e}")

# Handle UDP typing indicators with a flag to avoid repeated "typing" messages
def handle_udp():
    while True:
        message, addr = udp_server.recvfrom(1024)
        if addr not in typing_clients:
            typing_clients.add(addr)
            print(f"[UDP MESSAGE] {addr}: {message.decode('utf-8')}")
            broadcast_udp(message.decode('utf-8'), addr)

# Broadcast UDP message with a UDP prefix for typing indication
def broadcast_udp(message, sender):
    for client in clients:
        try:
            client.send(f"UDP:{message}".encode('utf-8'))
        except Exception as e:
            print(f"[ERROR] Could not send UDP message to {client}: {e}")

# Start server threads
def start():
    print("[STARTING] Server is starting...")
    threading.Thread(target=handle_udp, daemon=True).start()
    while True:
        conn, addr = tcp_server.accept()
        threading.Thread(target=handle_client, args=(conn, addr), daemon=True).start()

start()
```

CLIENT.JS

```
import socket
import threading
import tkinter as tk
from tkinter import scrolledtext
import time

# TCP and UDP setup
```

```
tcp_client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp_client.connect(('localhost', 9000))

udp_client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
udp_server_addr = ('localhost', 9001)

last_udp_sent = 0  # Timestamp of the last sent UDP message

# Send a TCP message
def send_tcp_message():
    message = message_entry.get()
    if message:
        tcp_client.send(message.encode('utf-8'))
        display_message(f"You: {message}")  # Display your own message in the GUI
        message_entry.delete(0, tk.END)

# Send a UDP typing indicator with rate limiting
def send_udp_typing():
    global last_udp_sent
    current_time = time.time()
    if current_time - last_udp_sent > 1:
        udp_client.sendto("User is typing...".encode('utf-8'), udp_server_addr)
        last_udp_sent = current_time

# Display messages in the GUI
def display_message(msg):
    chat_display.config(state=tk.NORMAL)
    chat_display.insert(tk.END, msg + "\n")
    chat_display.config(state=tk.DISABLED)
    chat_display.see(tk.END)  # Auto-scroll to the latest message

# Listen for incoming messages
def listen_for_messages():
    while True:
        try:
            message = tcp_client.recv(1024).decode('utf-8')
            if message.startswith("UDP:"):
                display_message(f"[Indicator] {message[4:]}")
            else:
                display_message(f"[New Message] {message}")
```

```
        except:
            print("Error receiving message")
            break

# GUI setup
window = tk.Tk()
window.title("TCP/UDP Chat Application")

# Chat display (read-only)
chat_display = scrolledtext.ScrolledText(window, state=tk.DISABLED, width=50,
height=20, wrap=tk.WORD)
chat_display.grid(row=0, column=0, columnspan=2, padx=10, pady=10)

# Message entry box
message_entry = tk.Entry(window, width=40)
message_entry.grid(row=1, column=0, padx=10, pady=10)

# Send button
send_button = tk.Button(window, text="Send", command=send_tcp_message)
send_button.grid(row=1, column=1, padx=10, pady=10)

# Typing indicator (UDP) sent on key press with rate limiting
message_entry.bind("<Key>", lambda event: send_udp_typing())

# Start listening thread
threading.Thread(target=listen_for_messages, daemon=True).start()

# Run the GUI
window.mainloop()
```

OUTPUT:

TCP/UDP Chat Application

```
[Indicator] User is typing...
You: hello
You: hru
You: fine
```

Send

```
Terminal  Local x  Local (2) x  Local (3) x  Local (4) x  Local (5) x  Local (6) x  +  v
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(venv) PS C:\Users\moune\PycharmProjects\pythonProject3> python server.py
[STARTING] Server is starting...
[NEW CONNECTION] ('127.0.0.1', 63361) connected.
[UDP MESSAGE] ('127.0.0.1', 57190): User is typing...
[('127.0.0.1', 63361)] hello
[('127.0.0.1', 63361)] hru
[('127.0.0.1', 63361)] fine
```

AIM	10	10
ALGORITHM	20	20
PROGRAM	40	40
OUTPUT	10	10
VIVA VOICE	10	9
RESULT	10	10
TOTAL	100	99

RESULT:-

Thus the real-time chat application that user TCP for reliable order communication and User UDP for faster connectionless messaging were successfully executed.

28/10/24

COMPUTER CENTRE