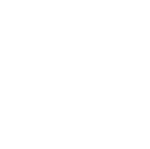
**B.M.S. COLLEGE OF ENGINEERING BENGALURU**

Autonomous Institute, Affiliated to VTU



Lab Record

**Computer Networks – 23CS5PCCON**

*Submitted in partial fulfillment for the 5th Semester Laboratory*

Bachelor of Engineering

in

Computer Science and Engineering

*Submitted by:*

**PRADEEP G**

(1BM24CS414)

Department of Computer Science and Engineering

B.M.S. College of Engineering

Bull Temple Road, Basavanagudi, Bangalore 560 019

August 2025-December 2025

**B.M.S. College of EngineerinG**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

***CERTIFICATE***

This is to certify that the Computer Networks (23CS5PCCON) laboratory has been carried out by Pradeep G (1BM24CS414) during the 5th Semester August 2025-December 2025

Signature of the Faculty Incharge:

**Sarala D V**

**Assistant Professor**

Department of Computer Science and Engineering

B.M.S. College of Engineering, Bangalore

**Table of Contents**

|  |  |
| --- | --- |
| **PART - A** | |
| **Serial No.** | **Name of Expirement** |
| 1. | Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message. |
| 2. | Configure DHCP within a LAN and outside LAN. |
| 3. | Configure Web Server, DNS within a LAN. |
| 4. | Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply. |
| 5. | Configure default route, static route to the Router. |
| 6. | Configure RIP routing Protocol in Routers. |
| 7. | Configure OSPF routing protocol. |
| 8. | To construct a VLAN and make the PC’s communicate among a VLAN. |
| 9. | To construct a WLAN and make the nodes communicate wirelessly. |
| 10. | Demonstrate the TTL/ Life of a Packet. |
| 11. | To understand the operation of TELNET by accessing the router in server room from a PC in IT office. |
| 12. | To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP). |

|  |  |
| --- | --- |
| **PART – B** | |
| **Serial No.** | **Name of Expirement** |
| 1. | Write a program for congestion control using Leaky bucket algorithm. |
| 2. | Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present. |
| 3. | Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present. |
| 4. | Write a program for error detecting code using CRC-CCITT (16-bits). |

**PART - A**

Program 1: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Network diagram:

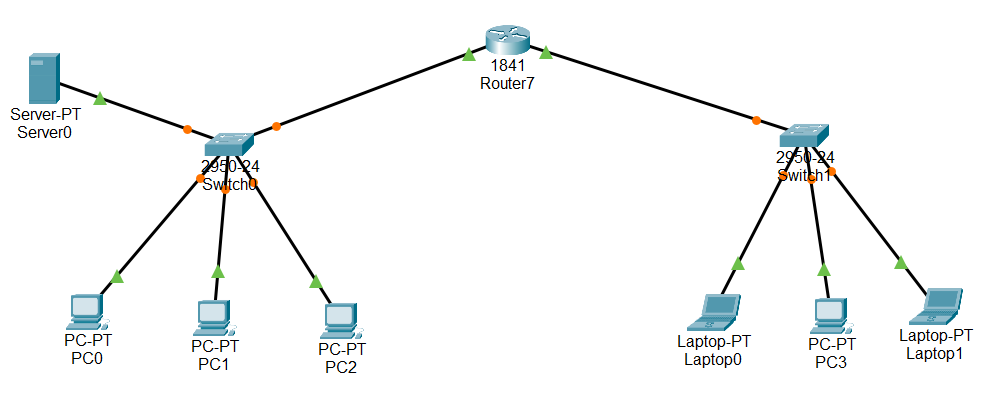
|  |  |
| --- | --- |
| 1. STAR Topology with Switch: | 2. MESH Topology with Switch: |
| 3. HUB-Based Network Topology: |

Configuration:

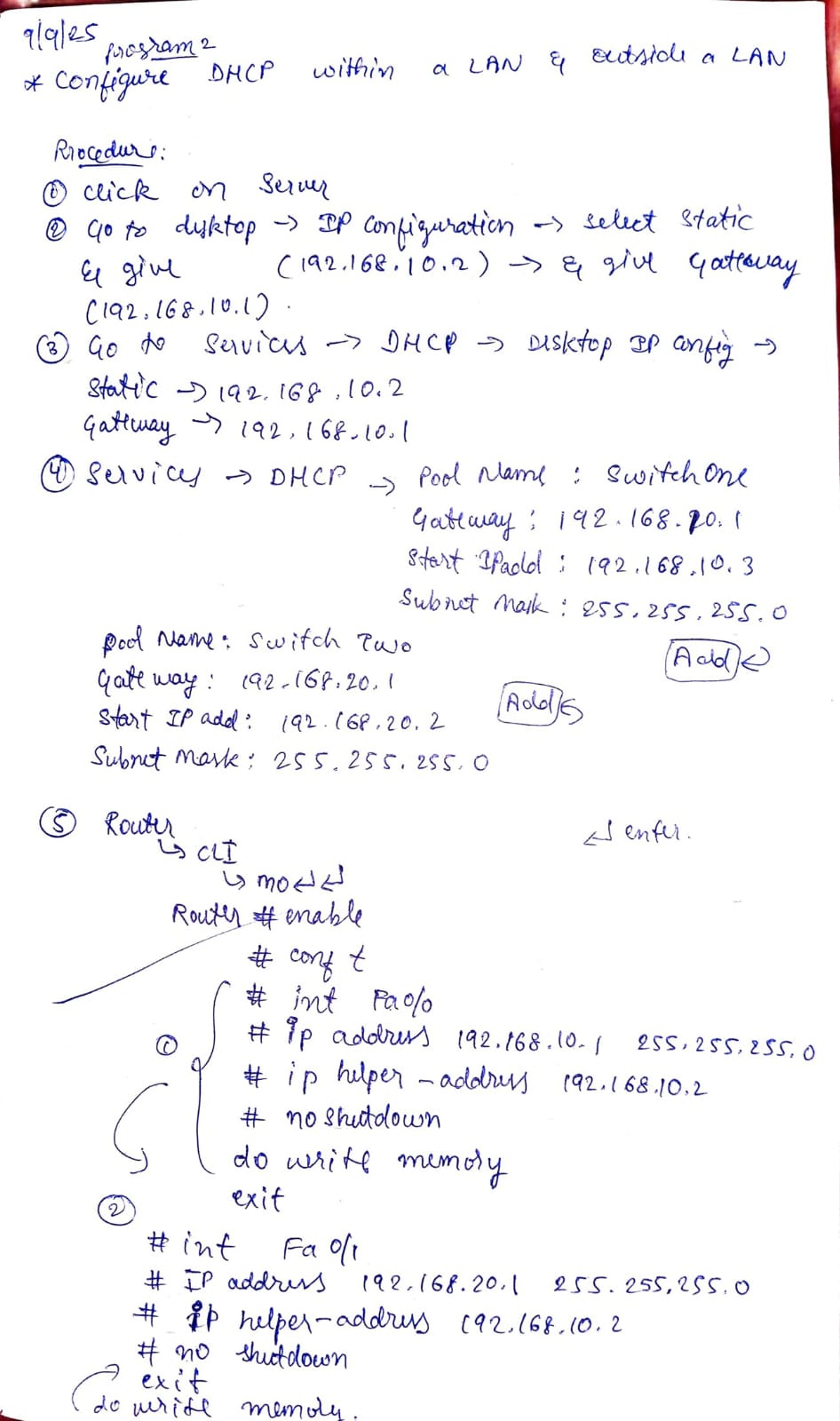
|  |  |
| --- | --- |
|  |  |
|  |  |

Program 2: Configure DHCP within a LAN and outside LAN.

Network diagram:

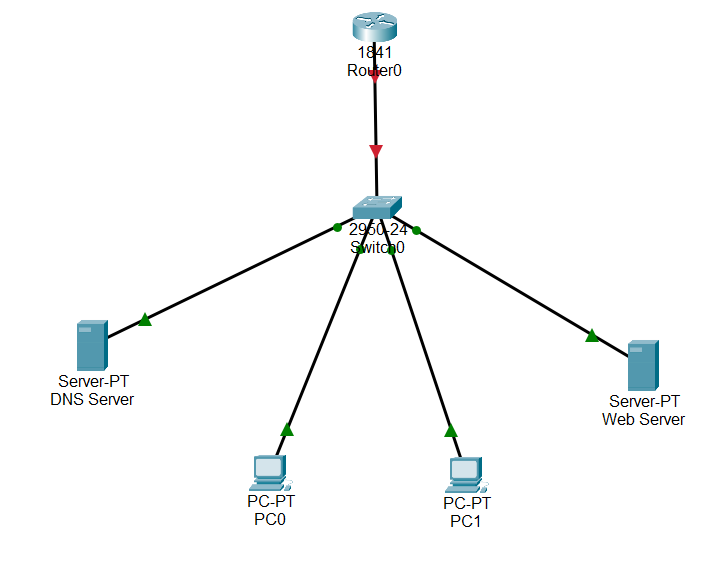


Configuration:

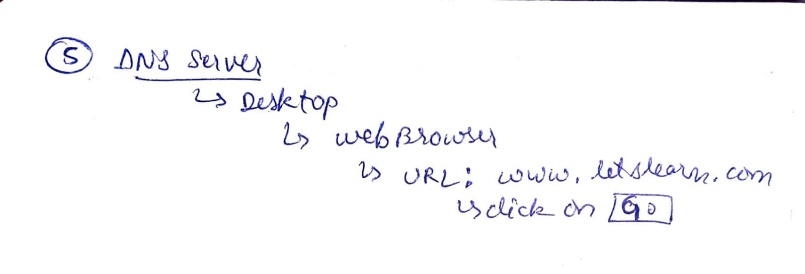
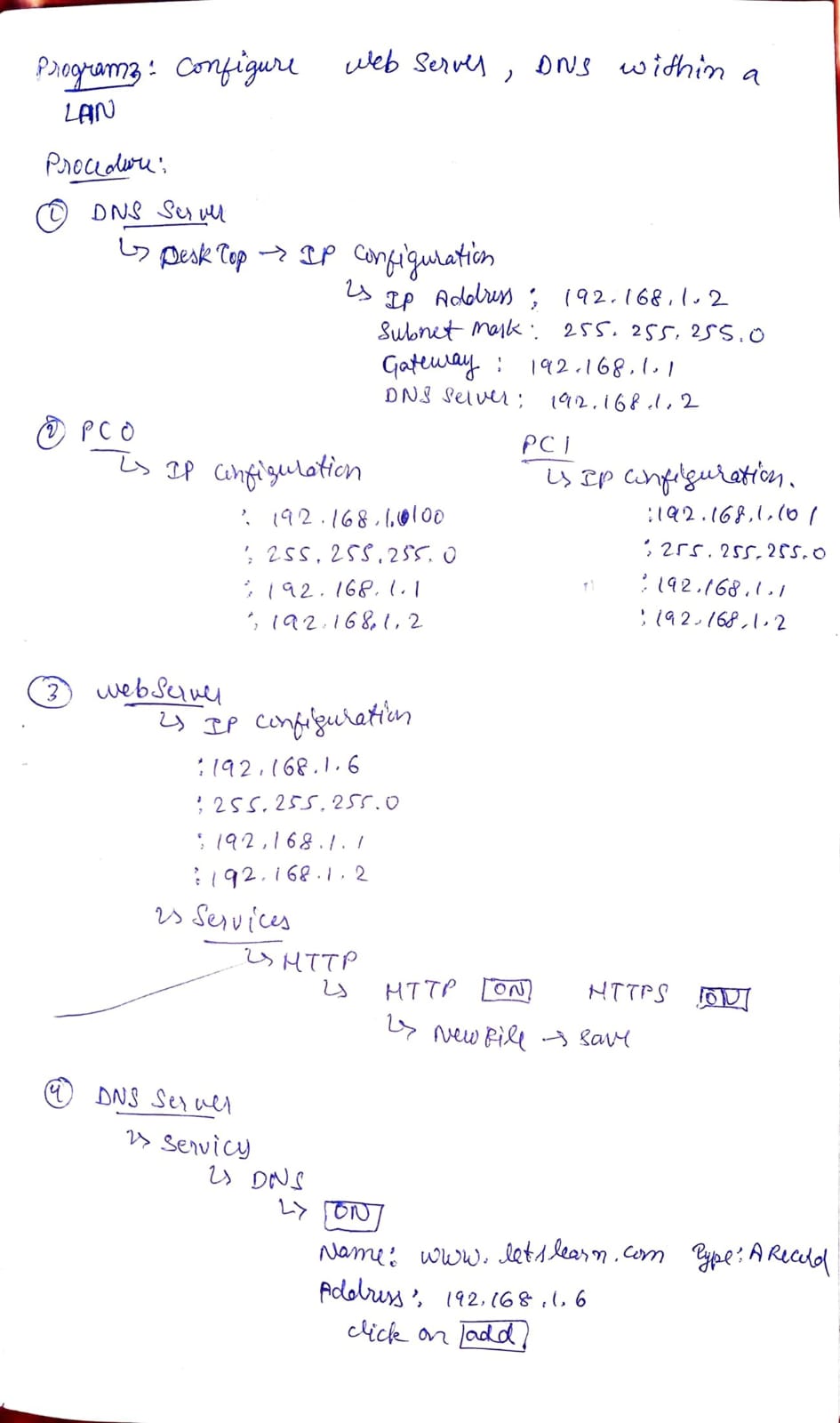


Program 3: Configure Web Server, DNS within a LAN.

Network diagram:

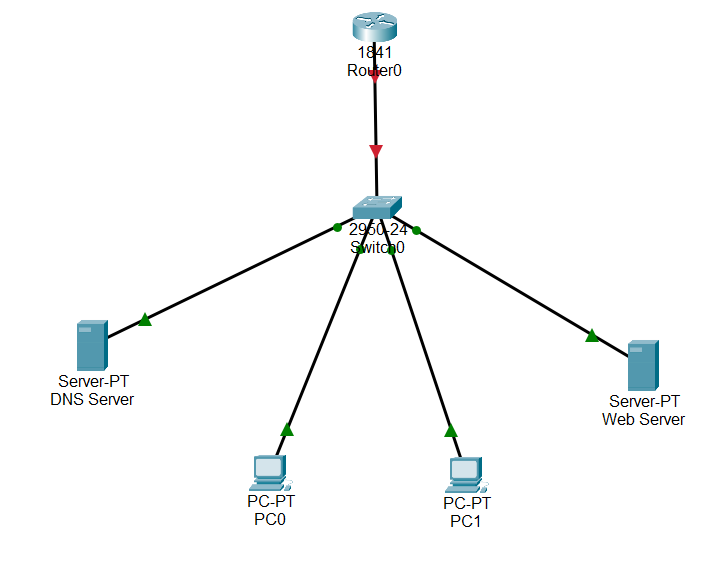


Configuration:



Program 4: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

Network diagram:

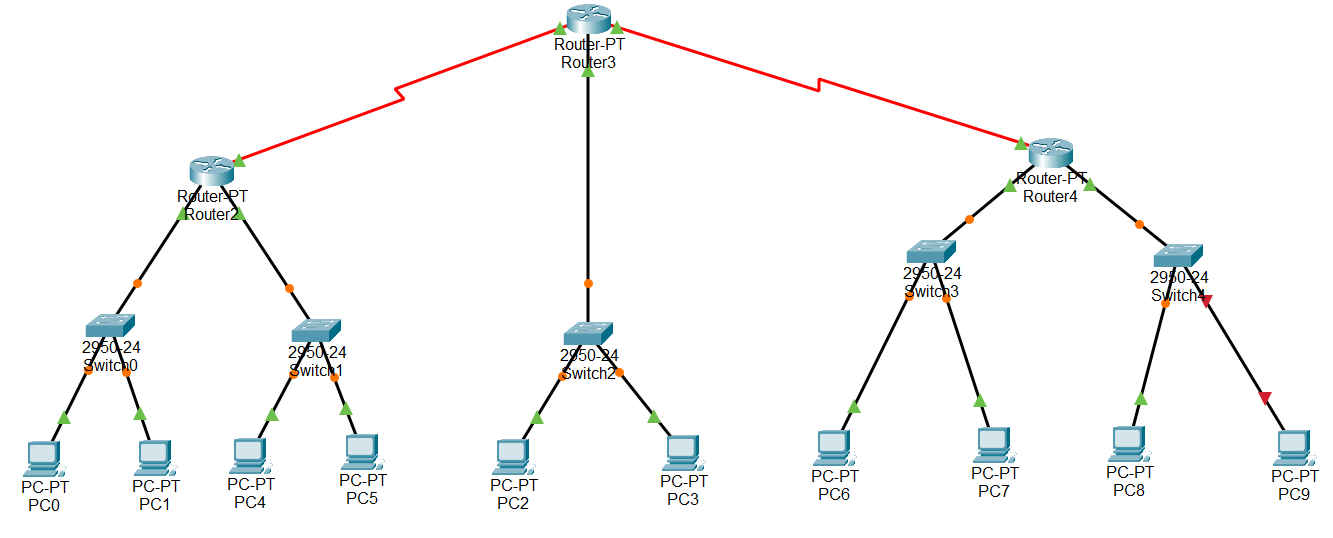


Configuration:

|  |  |
| --- | --- |
|  |  |

Program 5: Configure default route, static route to the Router.

Network diagram:

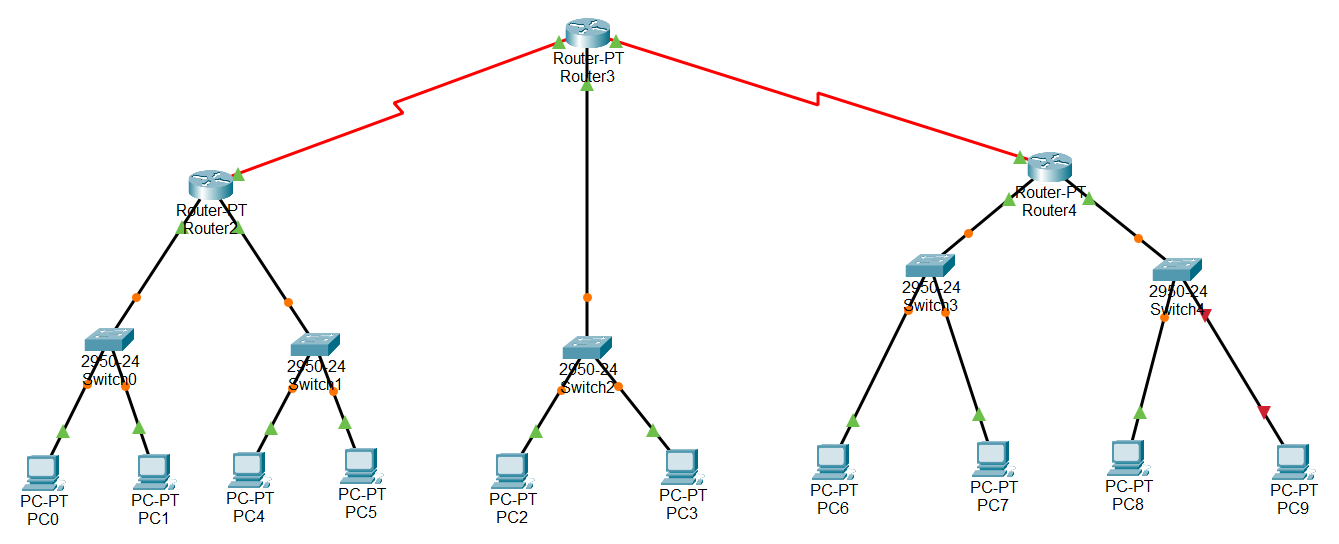


Configuration:

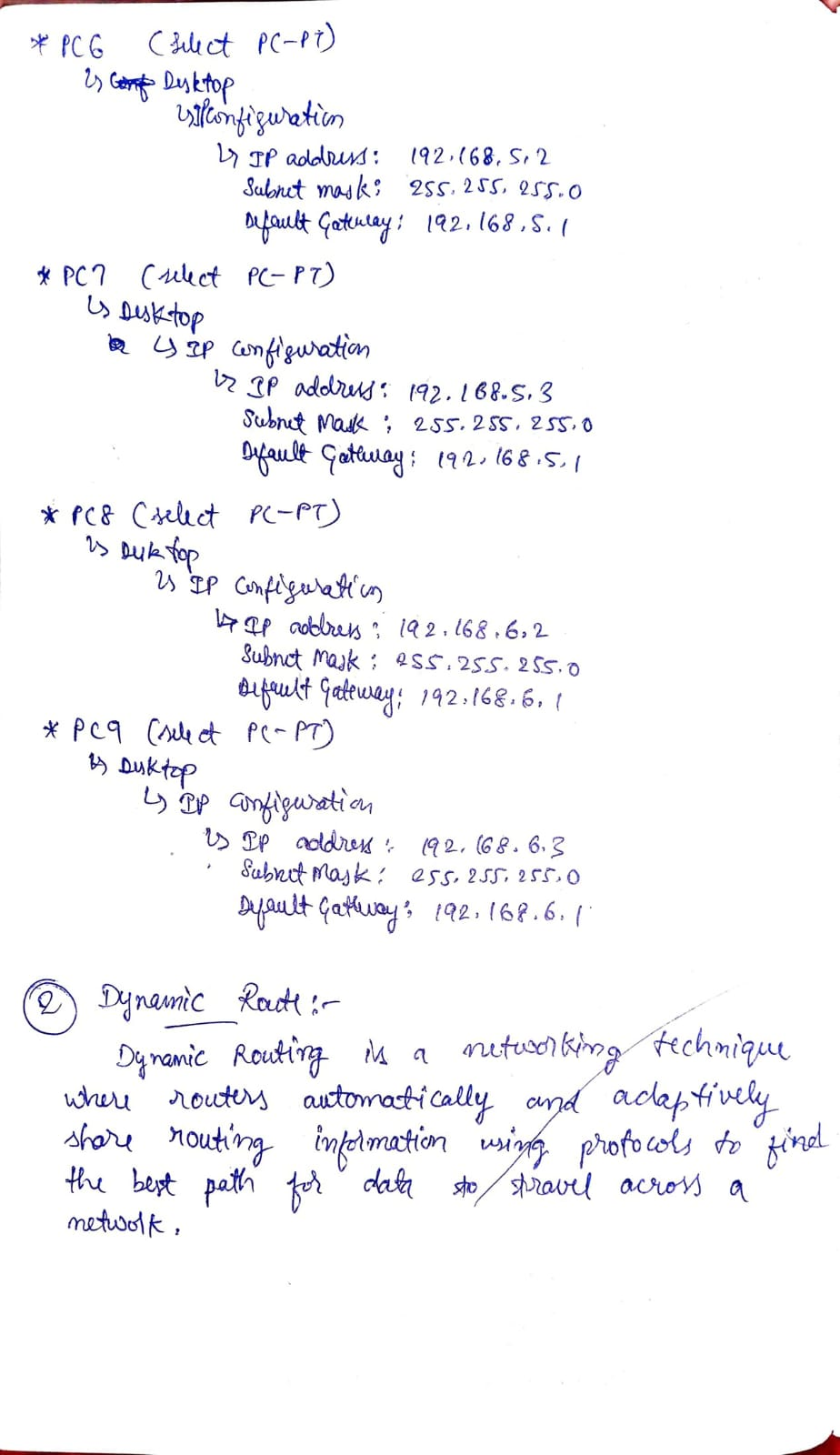
|  |  |
| --- | --- |
|  |  |
|  |  |

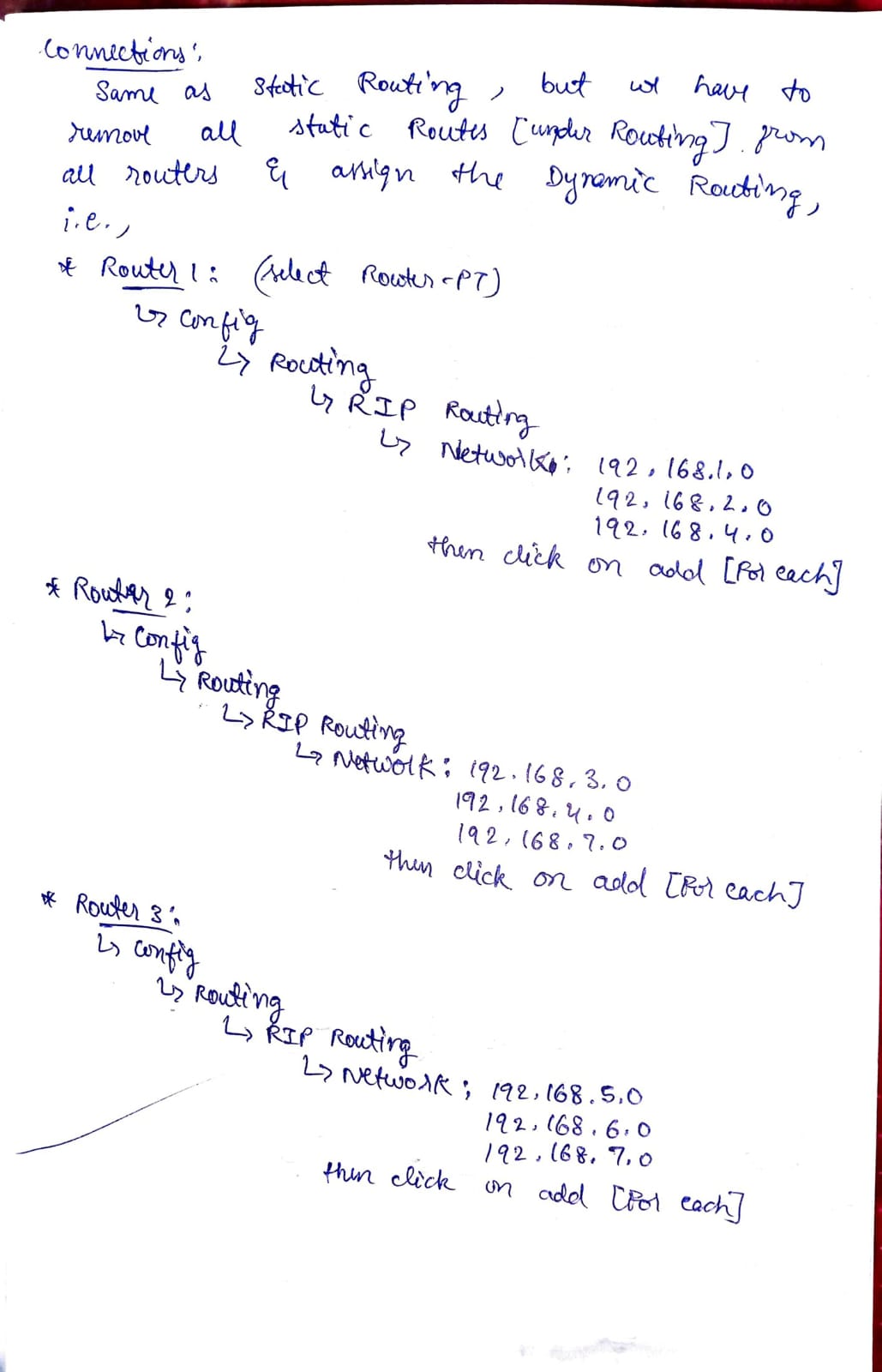
Program 6: Configure RIP routing Protocol in Routers.

Network diagram:



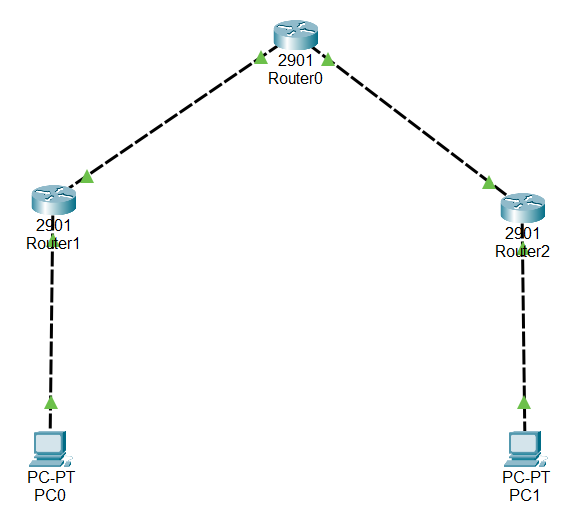
Configuration:





Program 7: Configure OSPF routing protocol.

Network diagram:

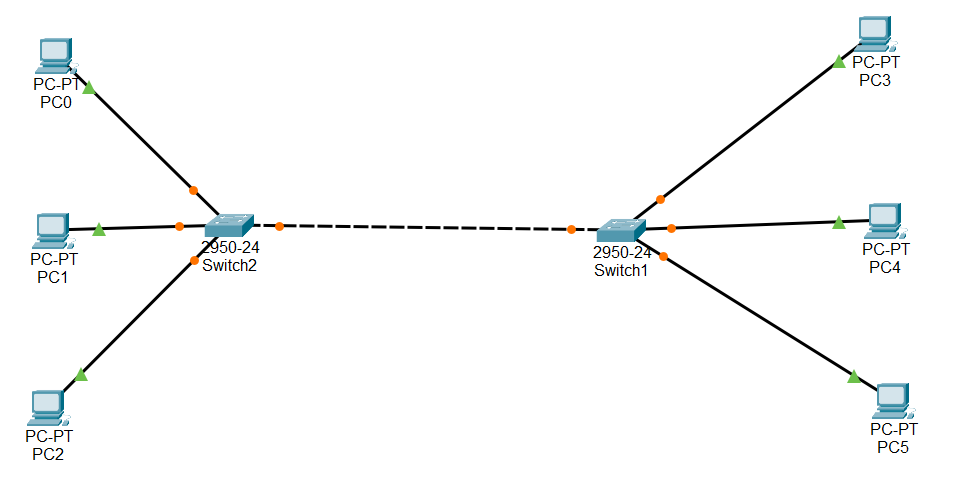


Configuration:

|  |  |
| --- | --- |
|  |  |

Program 8: To construct a VLAN and make the PC’s communicate among a VLAN.

Network diagram:



Configuration:

|  |  |
| --- | --- |
|  |  |

Program 9: To construct a WLAN and make the nodes communicate wirelessly.

Network diagram:

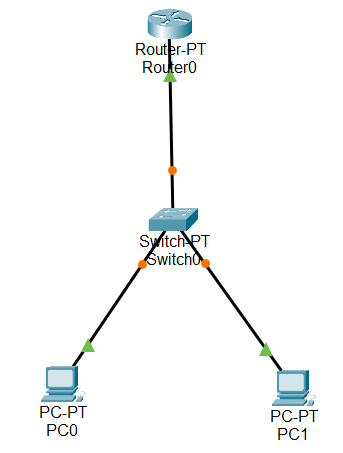
|  |  |
| --- | --- |
| Configuration 1 | Configuration 2 |

Configuration:

|  |  |
| --- | --- |
|  |  |

Program 10: Demonstrate the TTL/ Life of a Packet.

Network diagram:

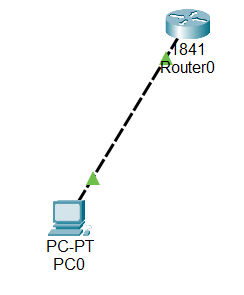


Configuration:

|  |  |
| --- | --- |
|  |  |

Program 11: To understand the operation of TELNET by accessing the router in server room from a PC in IT office.

Network diagram:

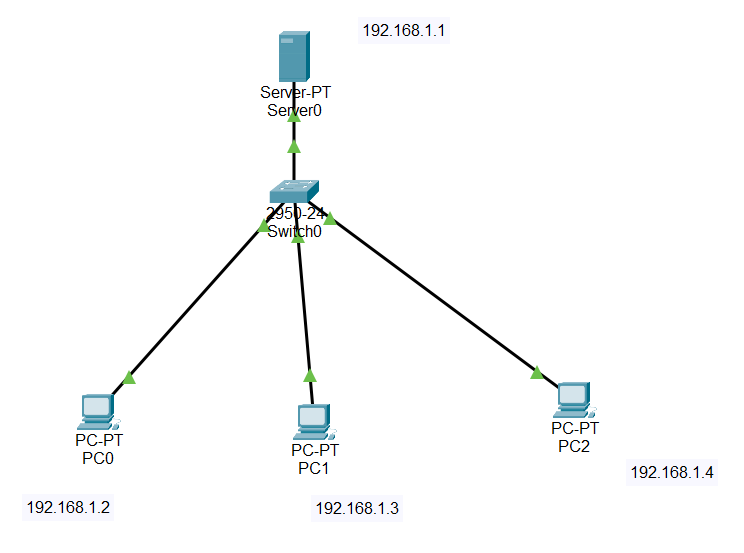


Configuration:

|  |  |
| --- | --- |
|  |  |

Program 12: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).

Network diagram:



Configuration:

|  |  |
| --- | --- |
|  |  |

**PART - B**

Program 1: Write a program for congestion control using Leaky bucket algorithm.

Code:

#include <stdio.h>

int min(int x, int y) {

if (x < y)

return x;

else

return y;

}

int main() {

int drop = 0, mini, nsec, cap, count = 0, i, inp[25], process;

printf("Enter the bucket size:\n");

scanf("%d", &cap);

printf("Enter the processing rate:\n");

scanf("%d", &process);

printf("Enter the number of seconds you want to simulate:\n");

scanf("%d", &nsec);

for (i = 0; i < nsec; i++) {

printf("Enter the size of the packet entering at %d sec:\n", i + 1);

scanf("%d", &inp[i]);

}

printf("\nSecond | Packet Received | Packet Sent | Packet Left | Dropped\n");

printf("--------------------------------------------------------------------\n");

for (i = 0; i < nsec; i++) {

count += inp[i];

if (count > cap) {

drop = count - cap;

count = cap;

}

printf("%d\t %d\t\t", i + 1, inp[i]);

mini = min(count, process);

printf("%d\t\t", mini);

count = count - mini;

printf("%d\t\t %d\n", count, drop);

drop = 0;

}

// Remaining packets after time ends

for (; count != 0; i++) {

if (count > cap) {

drop = count - cap;

count = cap;

}

printf("%d\t 0\t\t", i + 1);

mini = min(count, process);

printf("%d\t\t", mini);

count = count - mini;

printf("%d\t\t %d\n", count, drop);

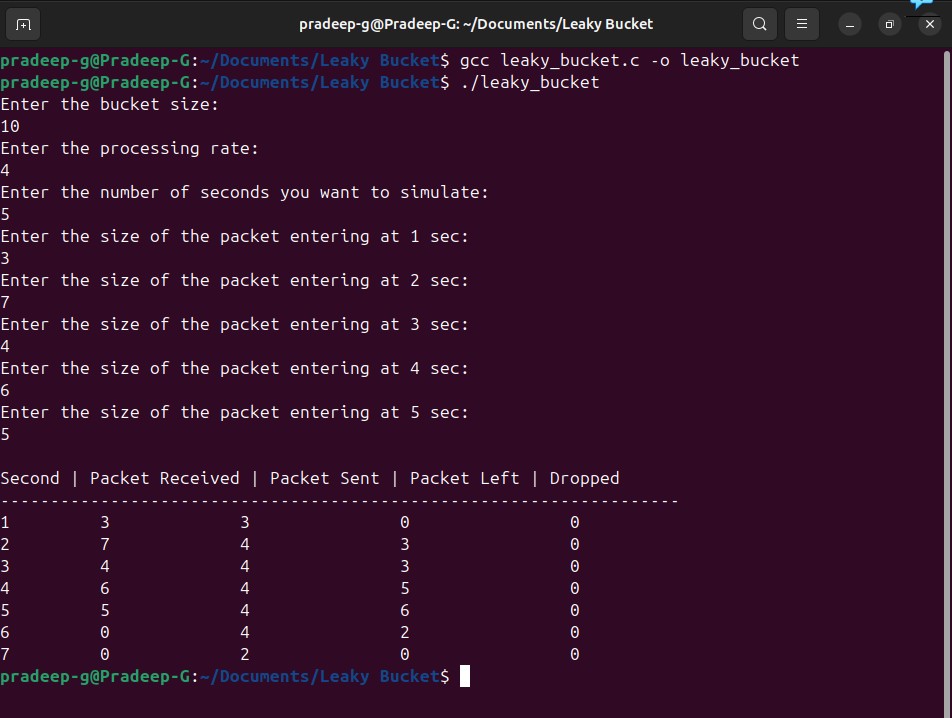
drop = 0;

}

return 0;

}

Output:



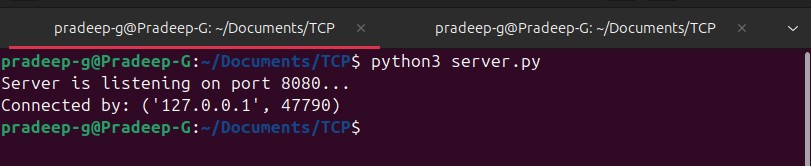
Program 2: Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Code:

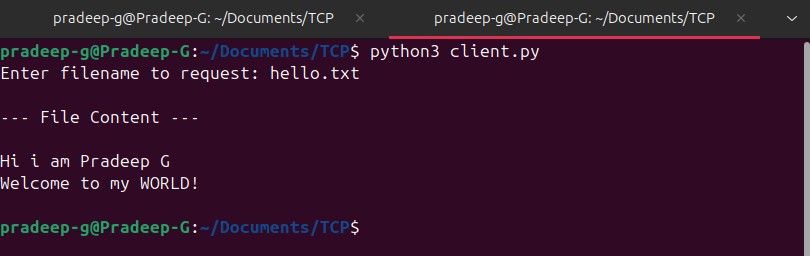
|  |  |
| --- | --- |
| # tcp\_client.py  import socket  # Step 1: Create TCP socket  client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  # Step 2: Connect to server  client\_socket.connect(('localhost', 8080))  # Step 3: Send filename  filename = input("Enter filename to request: ")  client\_socket.send(filename.encode())  # Step 4: Receive file contents  data = client\_socket.recv(4096).decode()  print("\n--- File Content ---\n")  print(data)  # Step 5: Close connection  client\_socket.close() | # tcp\_server.py  import socket  # Step 1: Create a TCP socket  server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  # Step 2: Bind to address and port  server\_socket.bind(('localhost', 8080))  # Step 3: Listen for client connections  server\_socket.listen(1)  print("Server is listening on port 8080...")  # Step 4: Accept connection  conn, addr = server\_socket.accept()  print("Connected by:", addr)  # Step 5: Receive file name  filename = conn.recv(1024).decode().strip()  try:      # Step 6: Open and read file      with open(filename, 'r') as f:          data = f.read()      conn.send(data.encode())  # Send file contents  except FileNotFoundError:      conn.send(b"File not found on server.")  # Step 7: Close connection  conn.close()  server\_socket.close() |

Output:

Server side Terminal:



Client side Terminal:



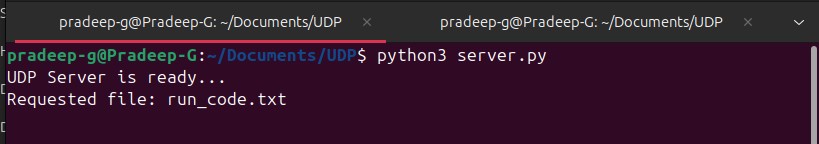
Program 3: Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Code:

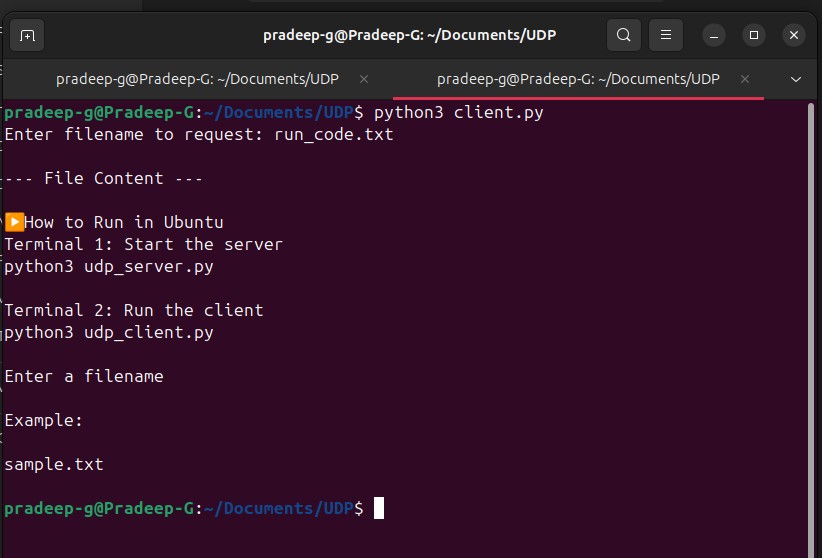
|  |  |
| --- | --- |
| # udp\_client.py  import socket  # Step 1: Create UDP socket  client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  server\_address = ('localhost', 8081)  filename = input("Enter filename to request: ")  # Step 2: Send filename to server  client\_socket.sendto(filename.encode(), server\_address)  # Step 3: Receive response  data, addr = client\_socket.recvfrom(4096)  print("\n--- File Content ---\n")  print(data.decode())  # Step 4: Close socket  client\_socket.close() | # udp\_server.py  import socket  # Step 1: Create UDP socket  server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  # Step 2: Bind to address and port  server\_socket.bind(('localhost', 8081))  print("UDP Server is ready...")  while True:      # Step 3: Receive filename from client      filename, addr = server\_socket.recvfrom(1024)      filename = filename.decode().strip()      print(f"Requested file: {filename}")      try:          # Step 4: Open file and send content          with open(filename, 'r') as f:              data = f.read()            server\_socket.sendto(data.encode(), addr)      except FileNotFoundError:          server\_socket.sendto(b"File not found on server.", addr) |

Output:

Server side Terminal:



Client side Terminal:



Program 4: Write a program for error detecting code using CRC-CCITT (16-bits).

Code:

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main() {

char rem[50], a[50], s[50], c, msj[50], gen[30];

int i, genlen, t, j, flag = 0, k, n;

printf("Enter the generation polynomial:\n");

gets(gen);

printf("Generator polynomial is CRC-CCITT: %s\n", gen);

genlen = strlen(gen);

k = genlen - 1;

printf("Enter the message:\n");

n = 0;

while ((c = getchar()) != '\n') {

msj[n] = c;

n++;

}

msj[n] = '\0';

for (i = 0; i < n; i++)

a[i] = msj[i];

for (i = 0; i < k; i++)

a[n + i] = '0';

a[n + k] = '\0';

printf("\nMessage polynomial appended with zeros:\n");

puts(a);

for (i = 0; i < n; i++) {

if (a[i] == '1') {

t = i;

for (j = 0; j <= k; j++) {

if (a[t] == gen[j])

a[t] = '0';

else

a[t] = '1';

t++;

}

}

}

for (i = 0; i < k; i++)

rem[i] = a[n + i];

rem[k] = '\0';

printf("Checksum (remainder):\n");

puts(rem);

printf("\nMessage with checksum appended:\n");

for (i = 0; i < n; i++)

a[i] = msj[i];

for (i = 0; i < k; i++)

a[n + i] = rem[i];

a[n + k] = '\0';

puts(a);

n = 0;

printf("Enter the received message:\n");

while ((c = getchar()) != '\n') {

s[n] = c;

n++;

}

s[n] = '\0';

for (i = 0; i < n; i++) {

if (s[i] == '1') {

t = i;

for (j = 0; j <= k; j++, t++) {

if (s[t] == gen[j])

s[t] = '0';

else

s[t] = '1';

}

}

}

for (i = 0; i < k; i++)

rem[i] = s[n + i];

rem[k] = '\0';

for (i = 0; i < k; i++) {

if (rem[i] == '1')

flag = 1;

}

if (flag == 0)

printf("Received polynomial is error-free \n");

else

printf("Received polynomial contains error \n");

return 0;

}

Output:

