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:

Himika Kakhani

(1BM23CS112)

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 Himika Kakhani (1BM23CS112) during the 5th Semester August 2025-December 2025.

Signature of the Faculty Incharge:

Rashmi H
Assistant Professor

Department of Computer Science and Engineering

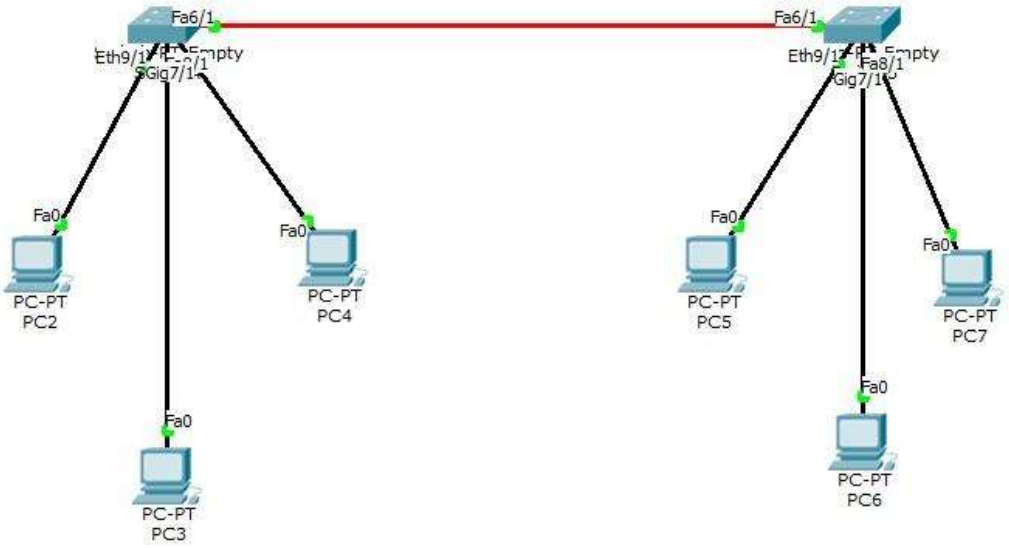
B.M.S. College of Engineering, Bangalore

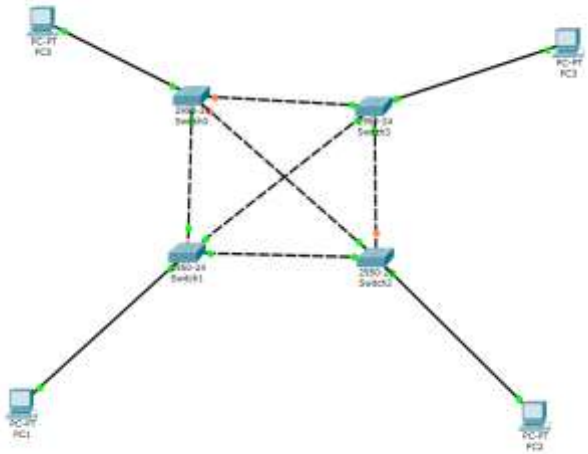
Table of Contents

PART - A

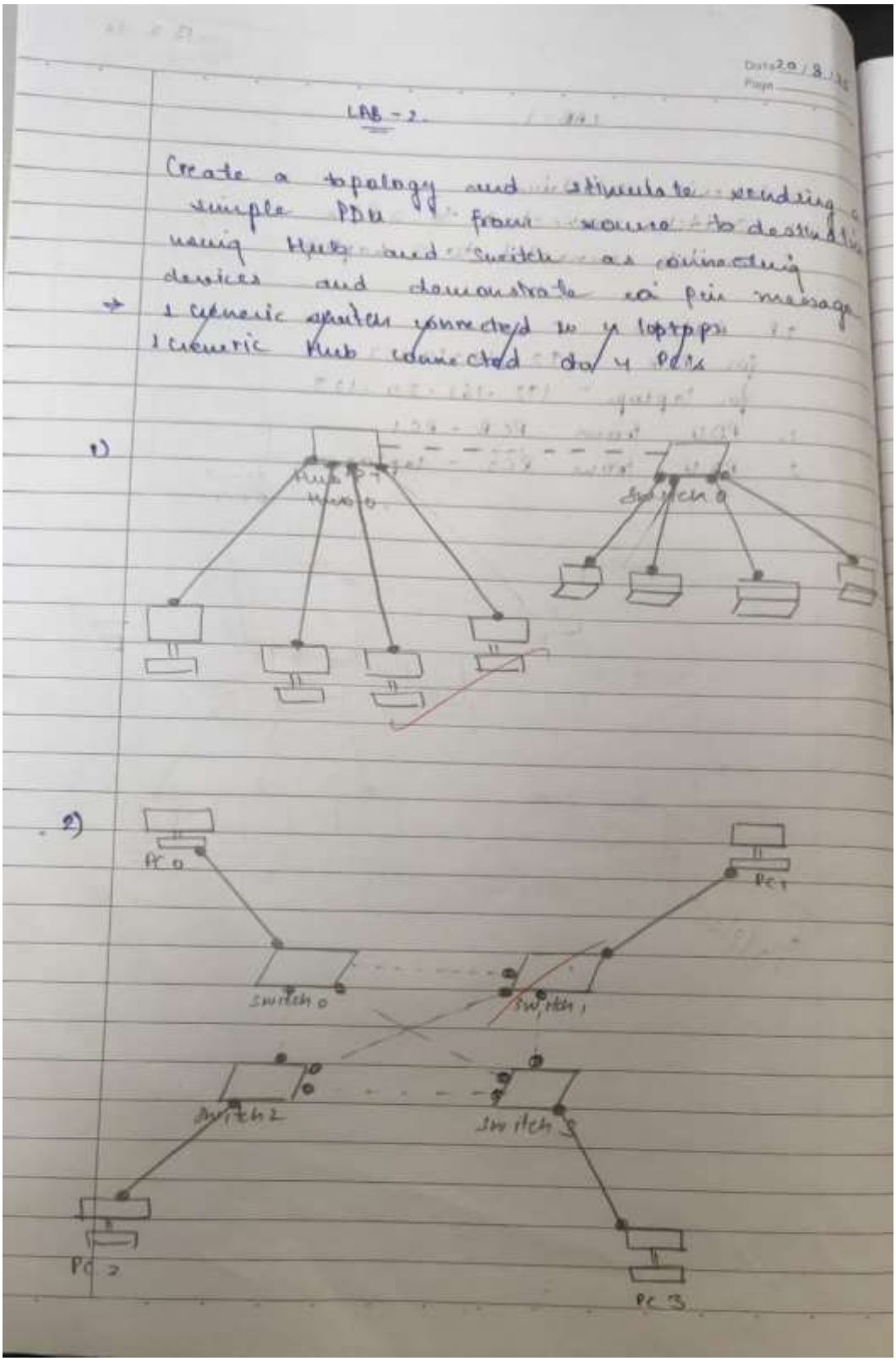
Program 1: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping messages.

Network diagram:



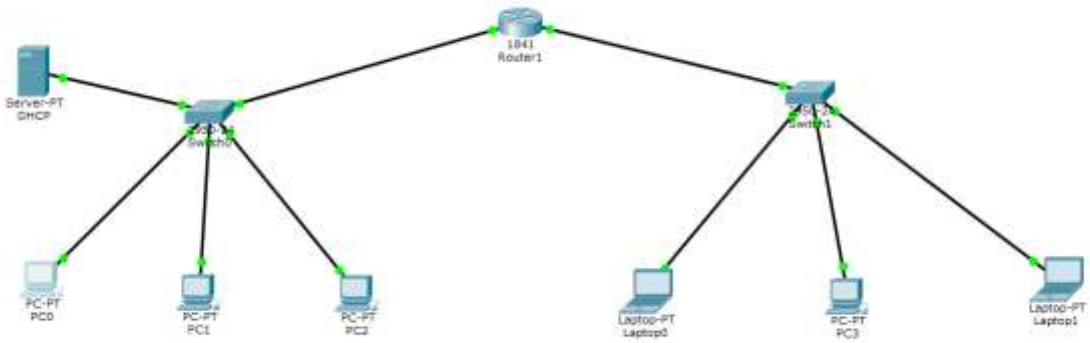


Configuration:

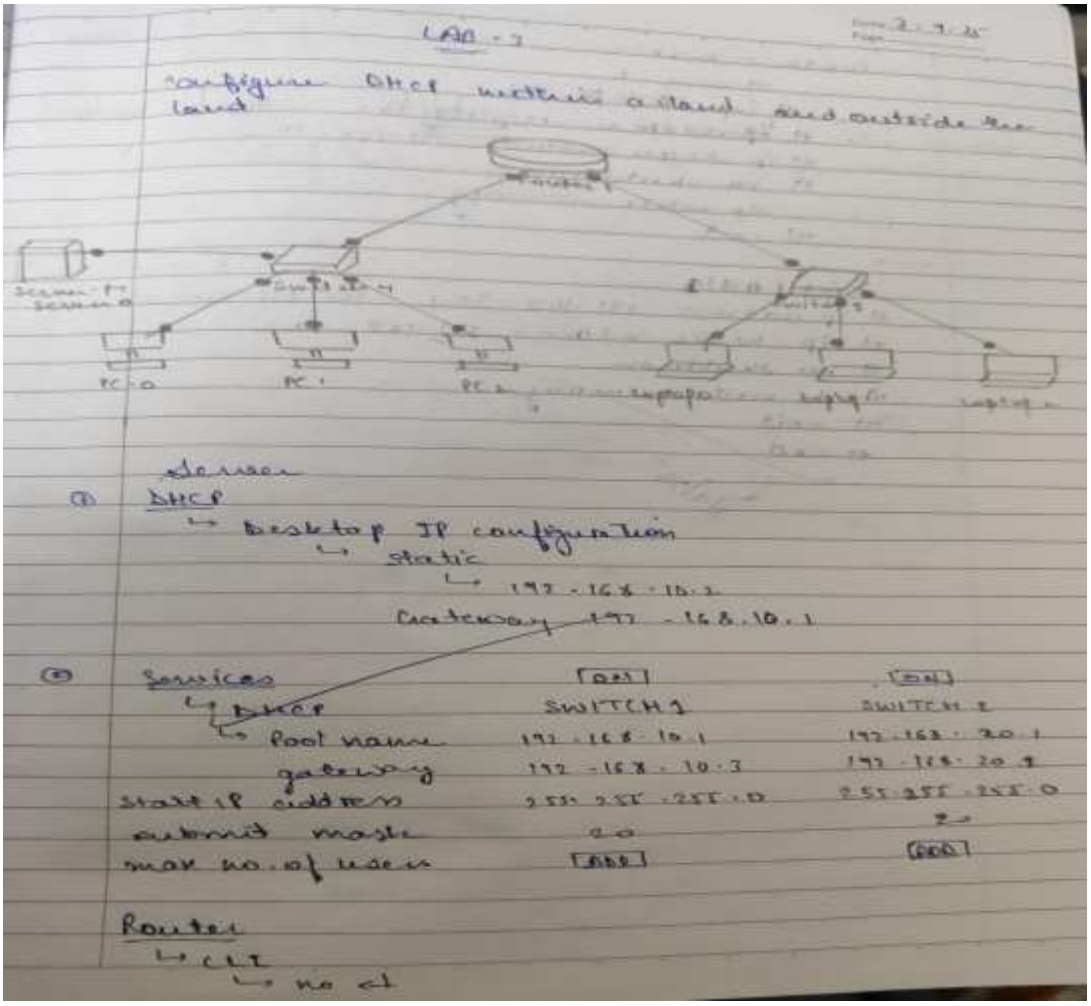


Program 2: Configure DHCP within a LAN and outside LAN.

Network diagram:

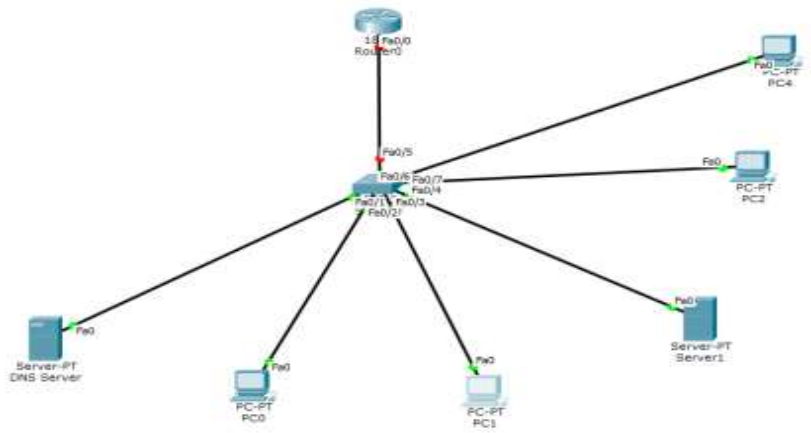


Configuration:

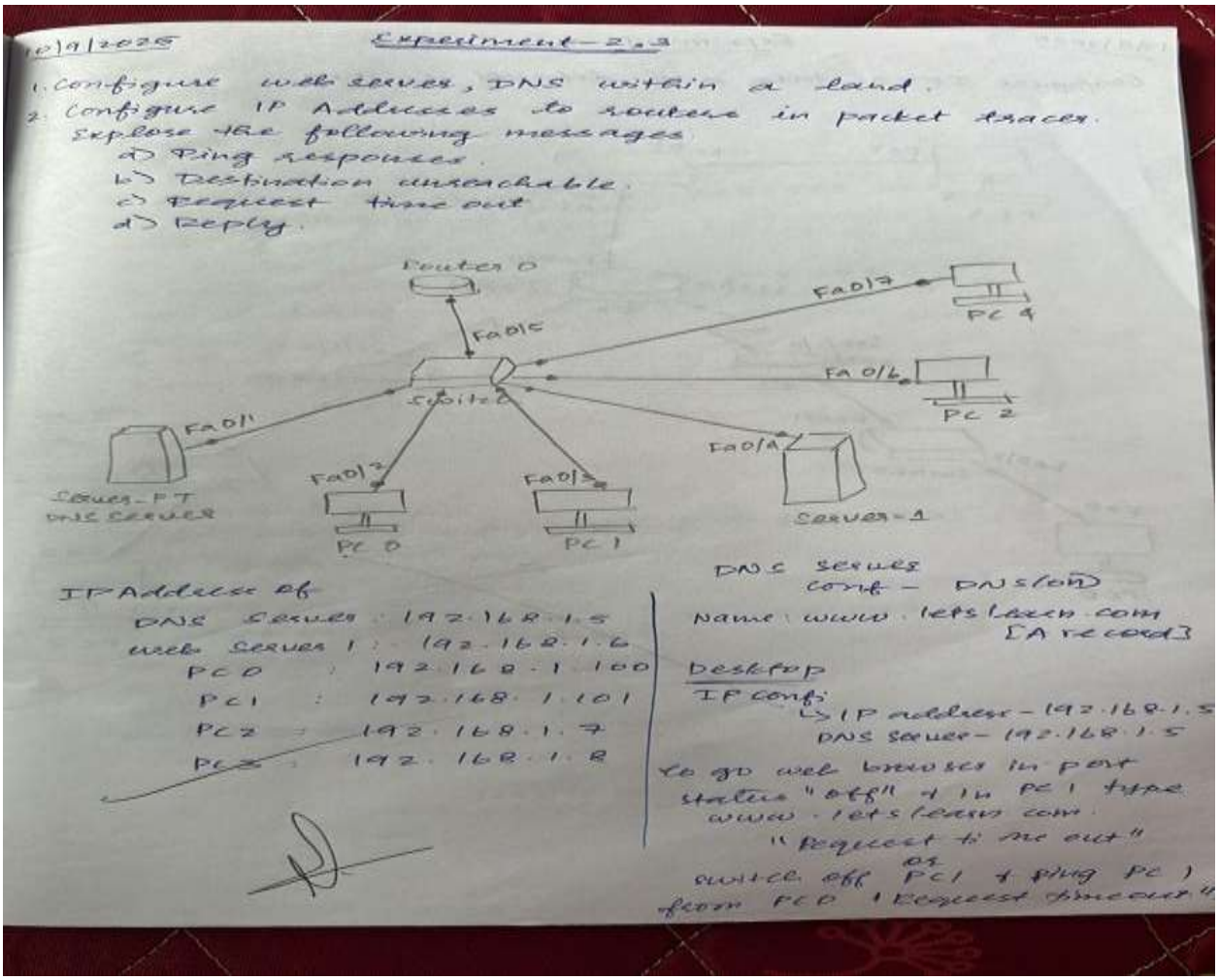


Program 3: Configure Web Server, DNS within a LAN.

Network diagram:

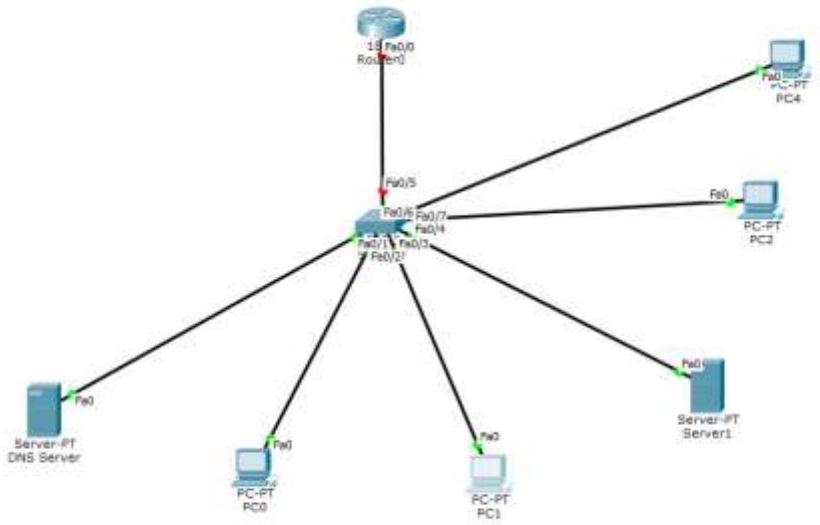


Configuration:



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

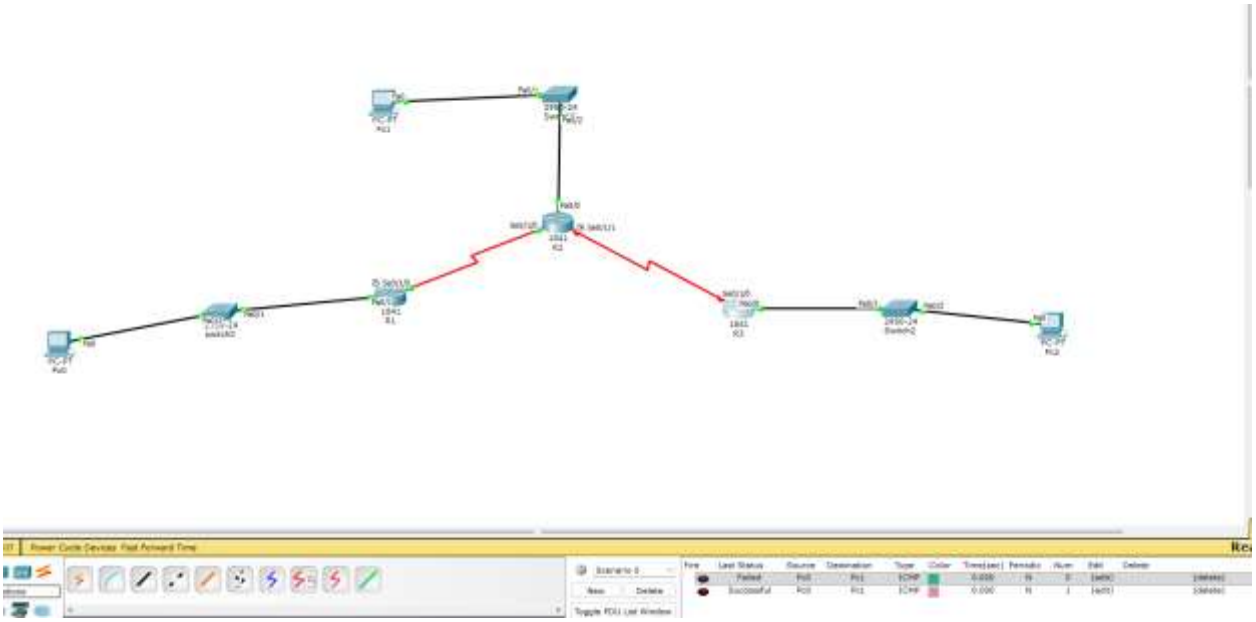
Network diagram:



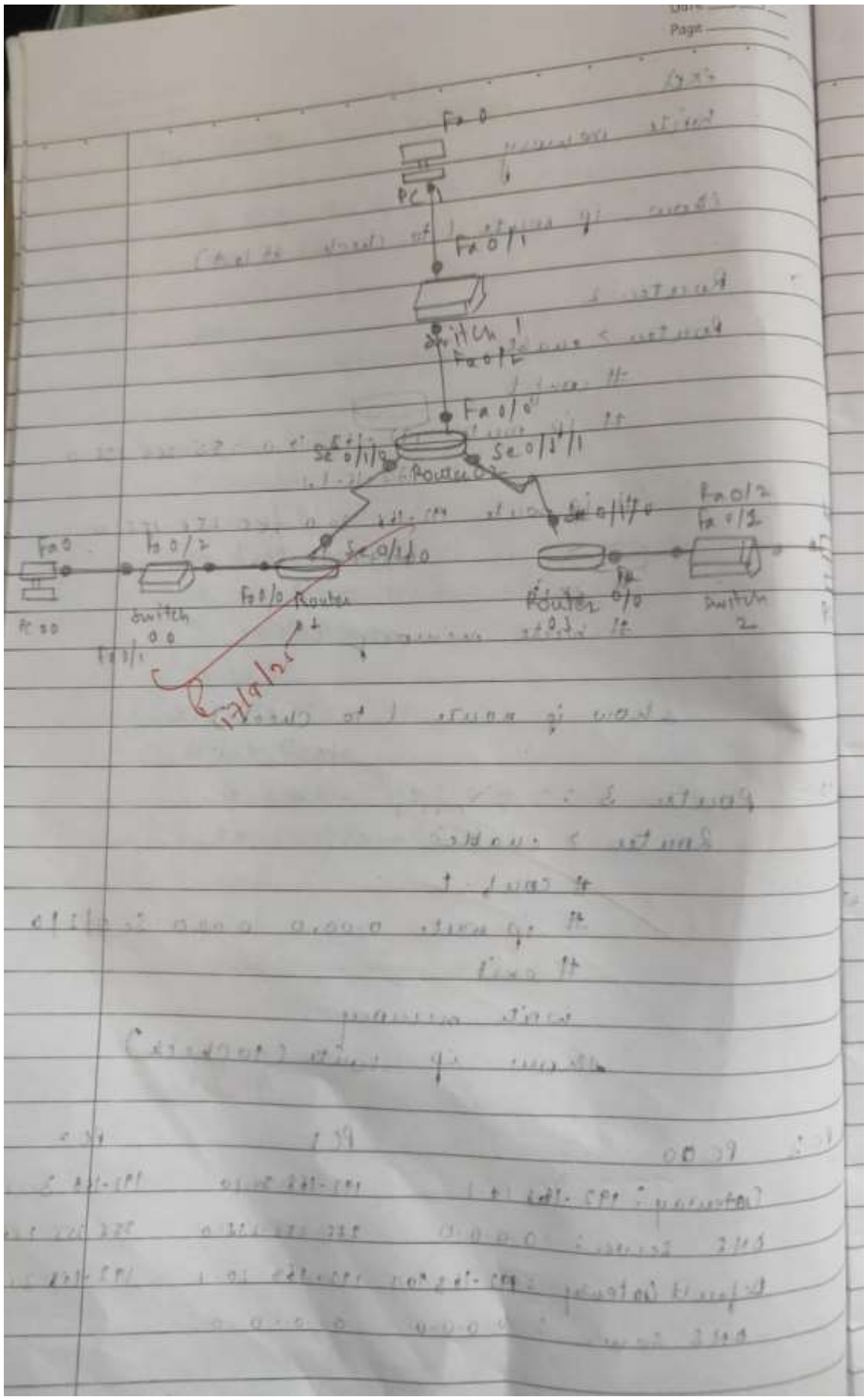
Configuration:

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

Network diagram:

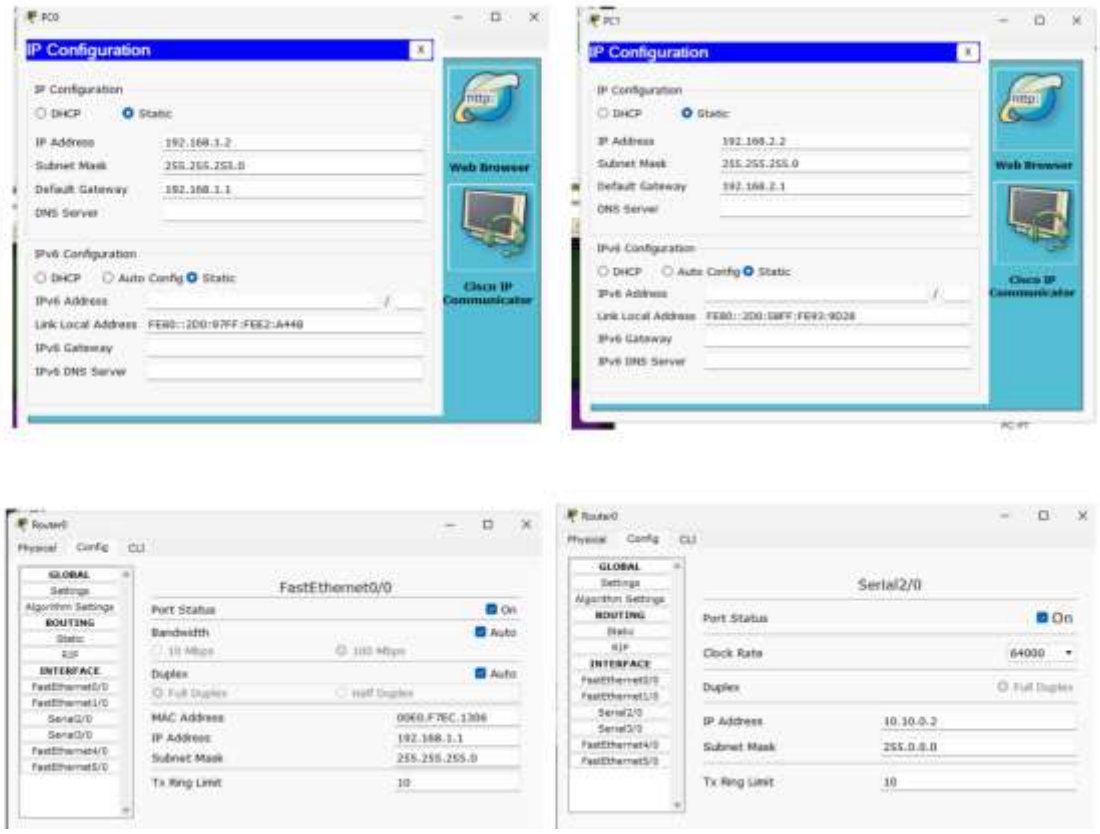


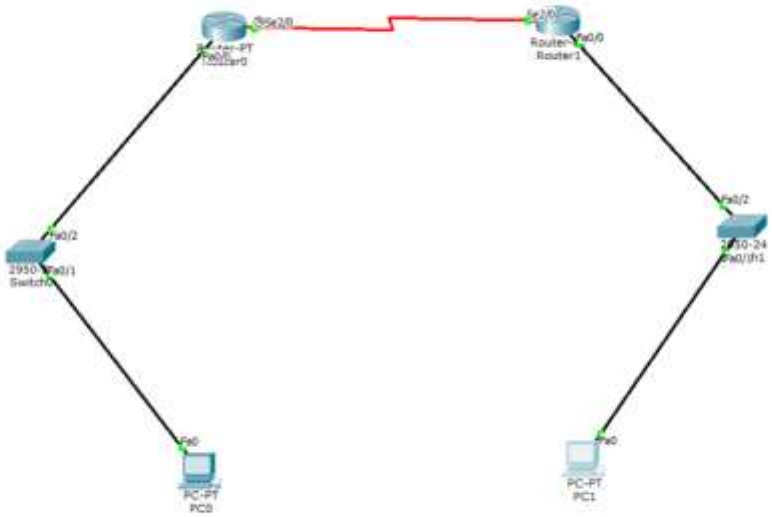
Configuration:


Fa 0
PC
Fa 0/1
Switch 1
Fa 0/2
Fa 0/0
Se 0/1/0
Se 0/1/1
Router 0
Se 0/1/0
Fa 0/2
Fa 0/3
Fa 0
Fa 0/2
Switch 0 0
Fa 0/1
Se 0/1/0
Fa 0/0
Router
Router 0/0
Switch 2
172.9.12.2

Program 6: Configure RIP routing Protocol in Routers.

Network diagram:





Configuration:

Date / /
Page

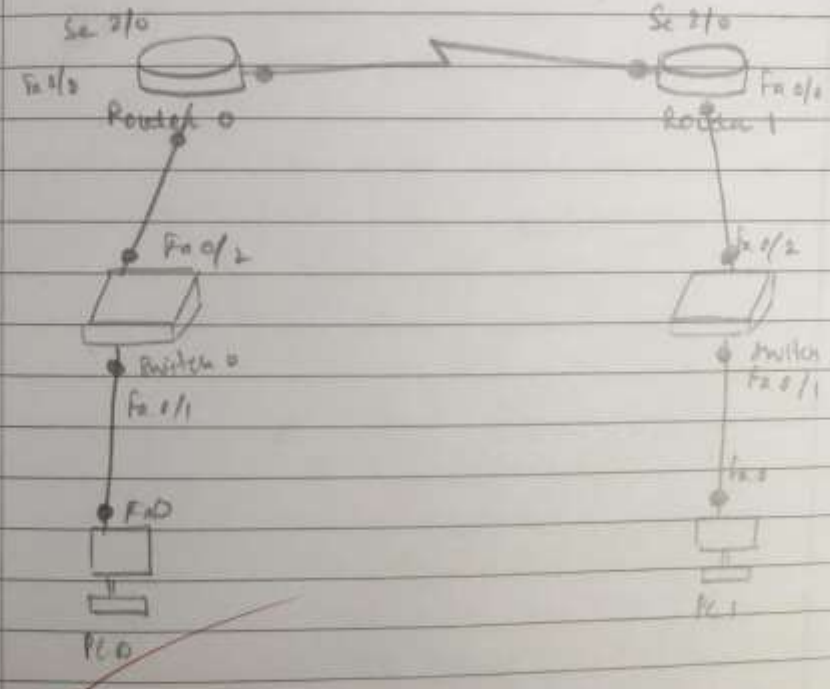
PC2

IP address : 192.168.11.3

DNS address : 192.168.11.4

Routing Information Protocol (RIP

Configure RIP routing protocol in router & transfer packet from node A to no



PC0 IP address : 192.168.1.2

Gateway : 192.168.1.1

PC IP address : 192.168.2.2

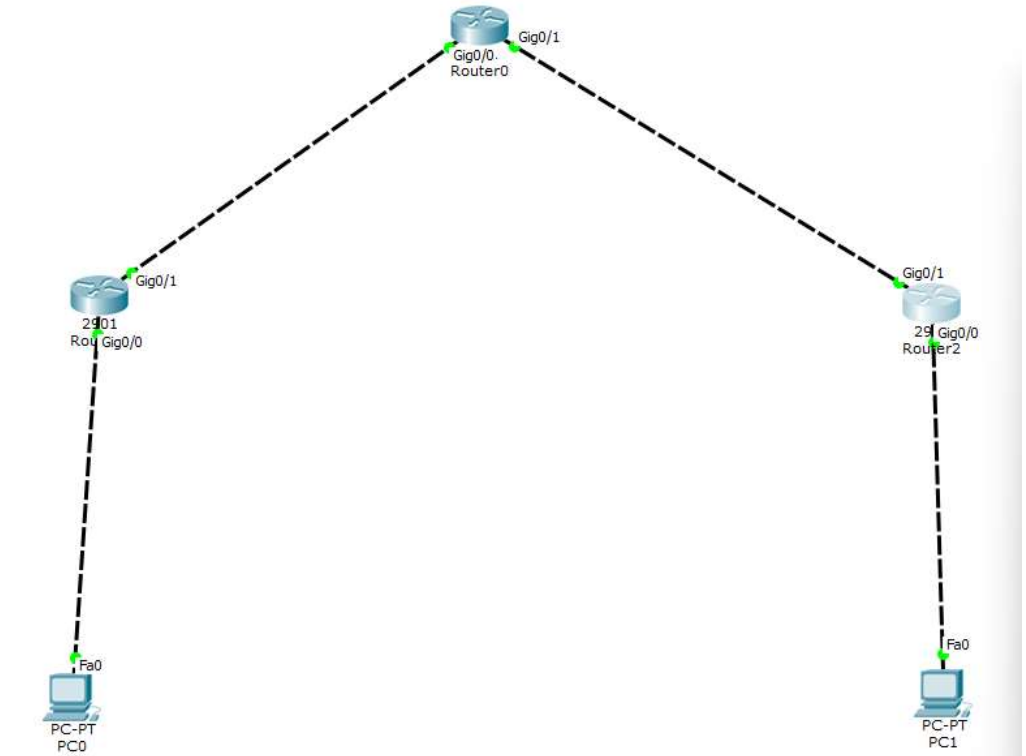
Gateway : 192.168.2.1

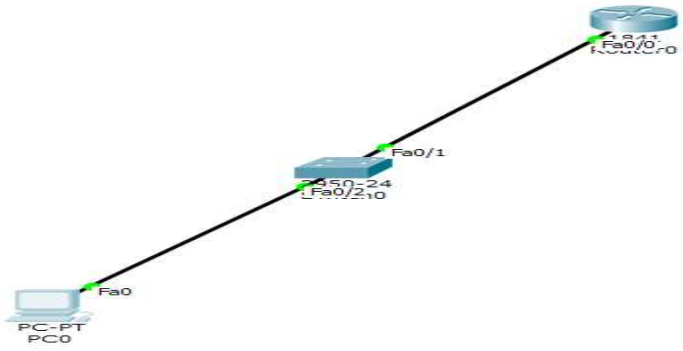
Router0 IP address 192.168.1.1 (fa 0/0)

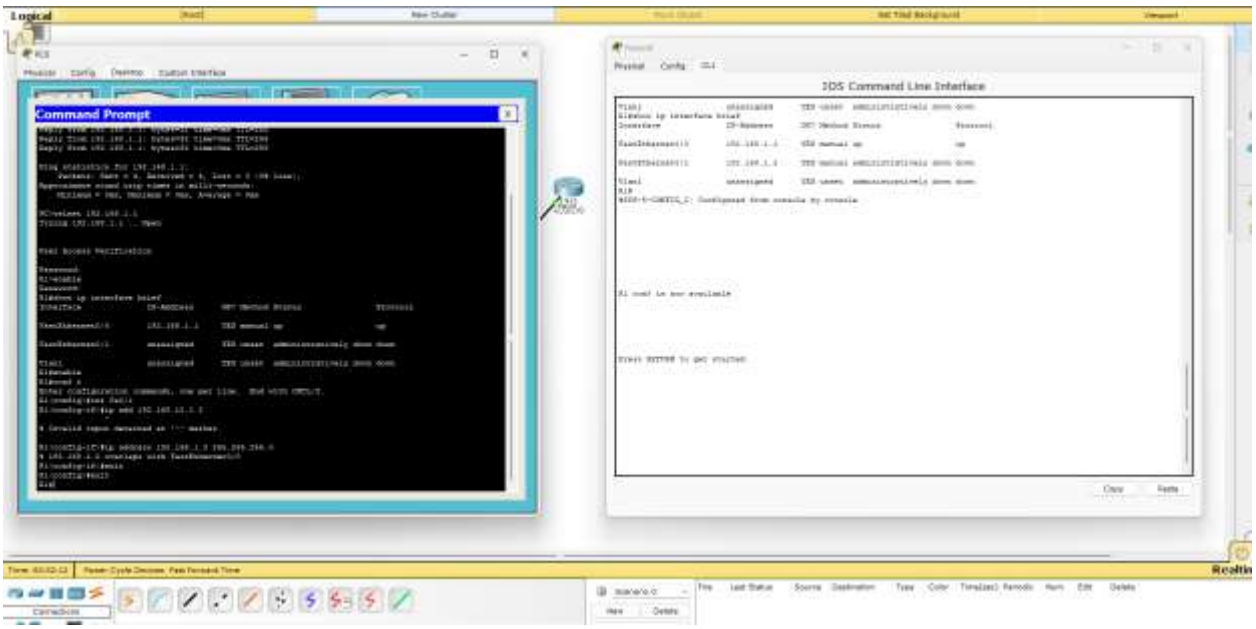
Serial 2/0 10.10.0.2

Program 7: Configure OSPF routing protocol.

Network diagram:





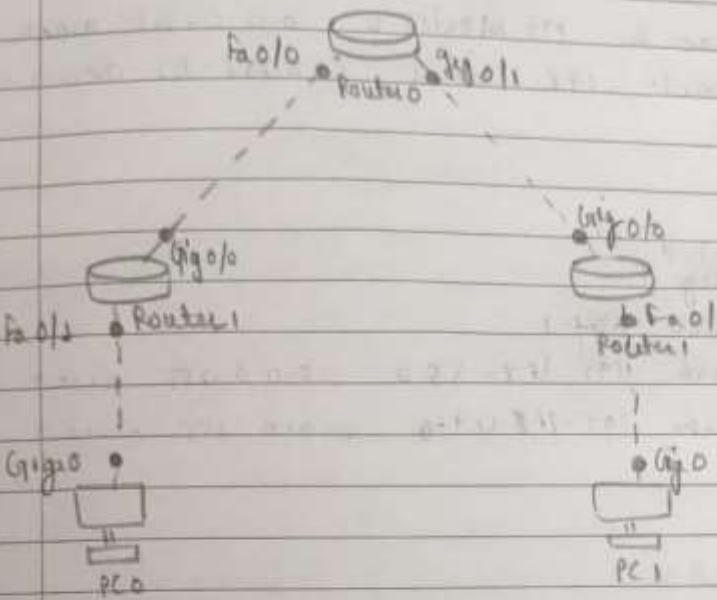


Configuration:

Date 8/10/25
Page

Exp 6

config OSPF protocol.



In Router 0
config
Gigabit ethernet 0/0
ip addres : 192.168.55.1 ON
Gigabite ethernet 0/1
ip addres : 176.16.10.1 ON

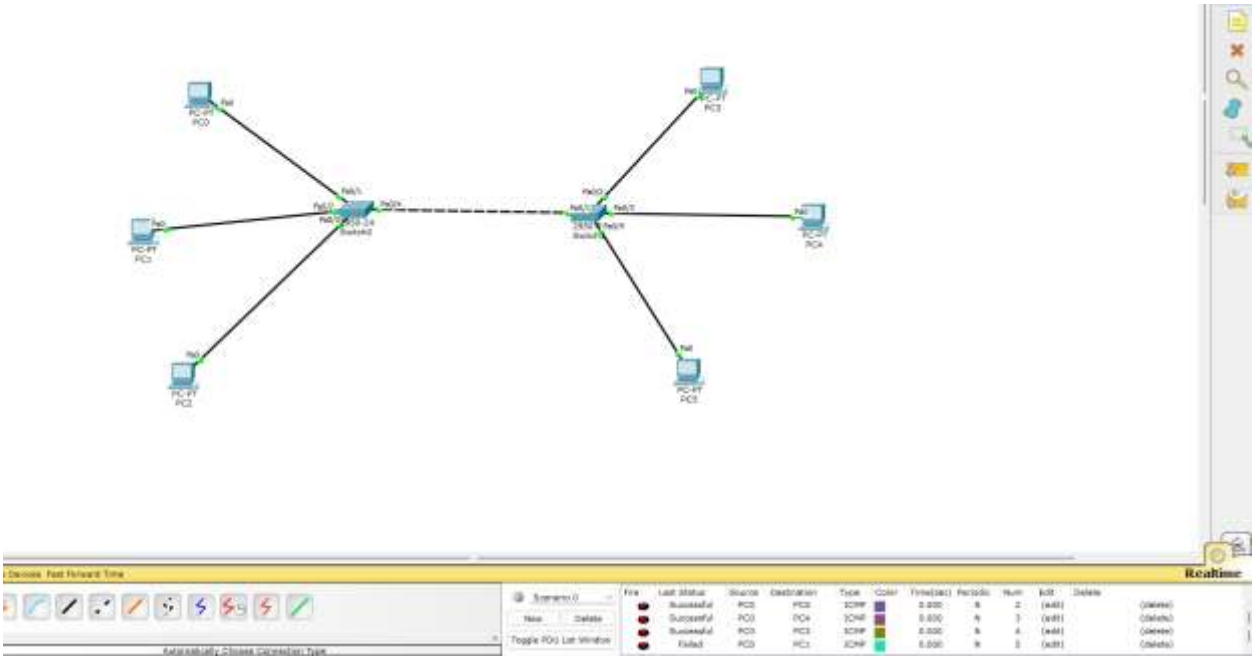
Router 1
0/0 : ip addres : 192.168.44.1
0/1 : ip addres : 192.168.55.2

Route 2:-
0/0 : ip addres : 176.16.10.2
0/1 : ip addres : 10.10.22.1

PC0 : IP : 192.168.44.2
default gateway : 192.168.55.2
PC1 : IP : 10.10.22.2
default gateway : 10.10.22.1

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

Network diagram:

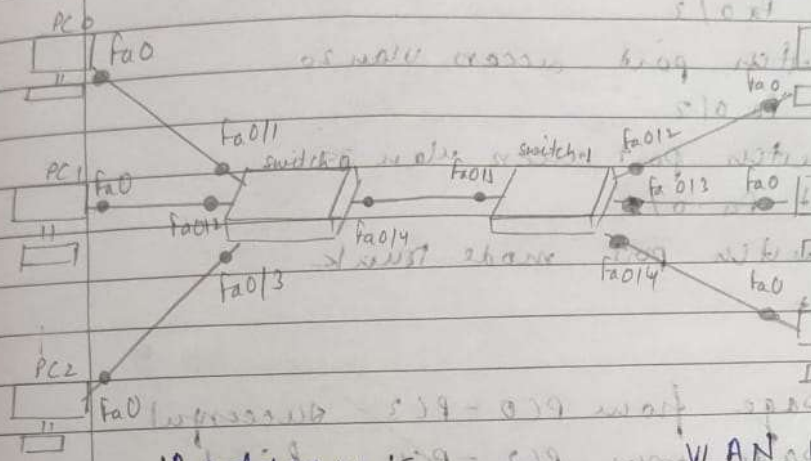


Configuration:

Date 15/10/25
Page

Exp 7

To construct a vlan and make the PCs communicating along VLAN



IP address :-

PC0 - 192.168.1.2

PC1 - 192.168.1.3

PC2 - 192.168.1.4

PC3 - 192.168.1.5

PC4 - 192.168.1.6

PC5 - 192.168.1.7

PC.

VLAN 10
Fa0/1 PC0
fa0/2 PC3

VLAN 20
fa0/2 PC1
fa0/3 PC4

VLAN 30
fa0/3 PC2
fa0/4 PC5

switch 0

> enable

conft

int fa 0/2

switch port acces vlan 10

int fa 0/2

switch port acces vlan 20

int fa 0/3

switch port acces vlan 30

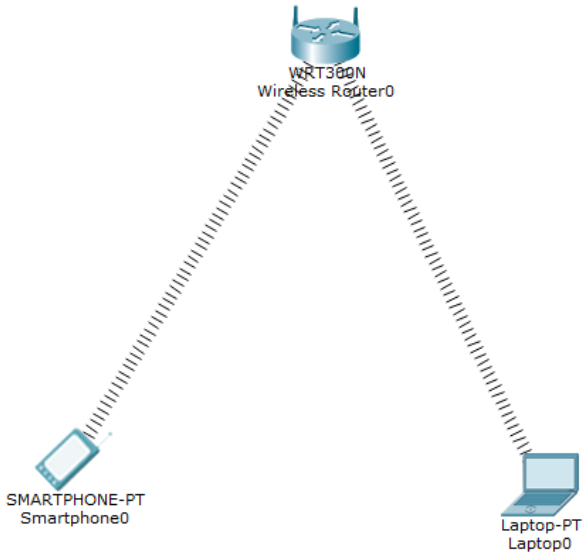
int fa 0/4

switch port mode trunk

exit

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

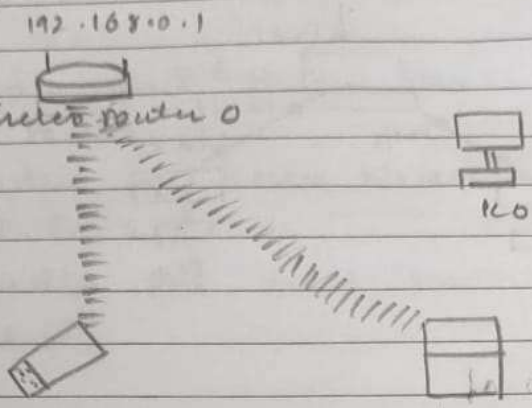
Network diagram:



Configuration:

Exp : 7

Construct a WLAN and make the router nodes communicate wirelessly.



for laptop:

turned off, take out the port and drop it

again drag and drop wireless port

Wireless Router:

↳ config → wireless,

SSID : BMSCE

authentication - WPA2 - PSK PSK

password : bmsce1234

in smartphone

↳ config - wireless

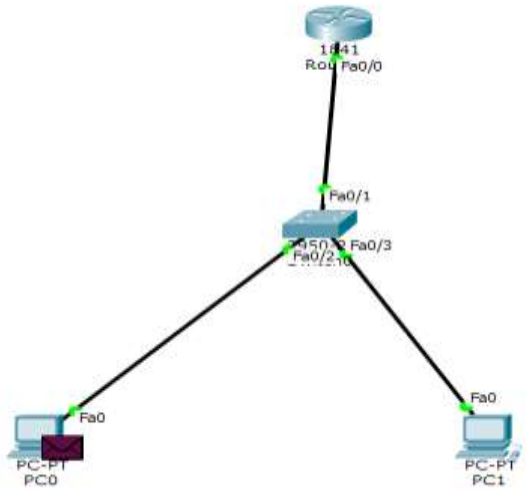
↳ SSID - BMSCE

↳ ~~authentication~~

↳ authentication : bmsce1234

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

Network diagram:

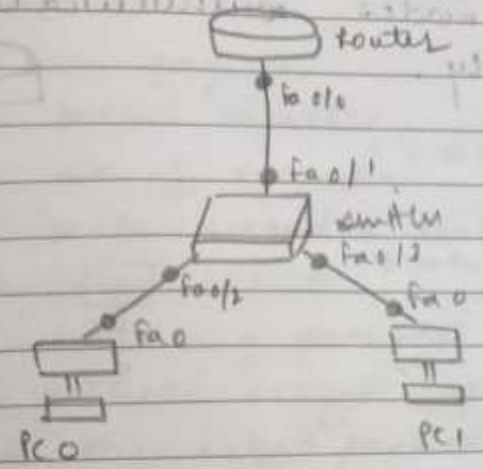


Configuration:

Date 11/10/25
Page

Exp. 8.

Demonstrate TTL life of a packet



IP addresses of

PC 0 : 192.168.1.2

PC 1 : 192.168.1.3

Router : 192.168.1.1

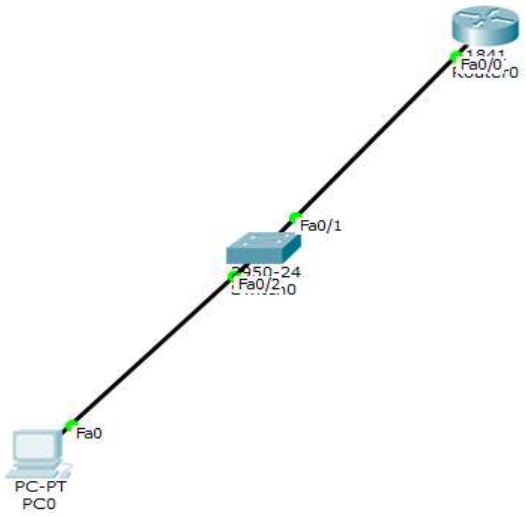
Default gateway for PC0 of PC1 } 192.168.1.1

Message from PC0 - PC1 = successful

TTL 255, 128

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

Network diagram:



Configuration:

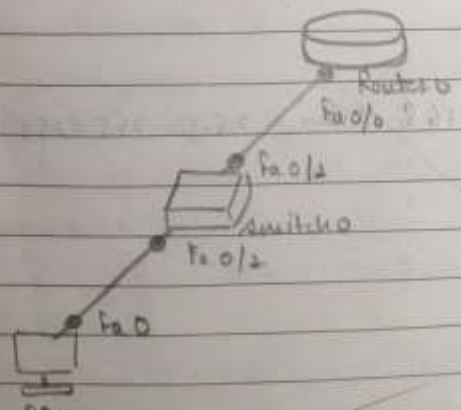
Date 6/10/25
Page

experiment - 5.

Configure TELNET to access router remotely.

Telnet is used to access remote server. It is a simple command line tool that runs on your computer and it allow to send commands remotely to a server and administrator.

It is also used to manage other devices like router, switch to check if ports are open or close on the server.



In PC0

IP address 192.168.1.2

default gateway : 255.255.255.0

In router

enable

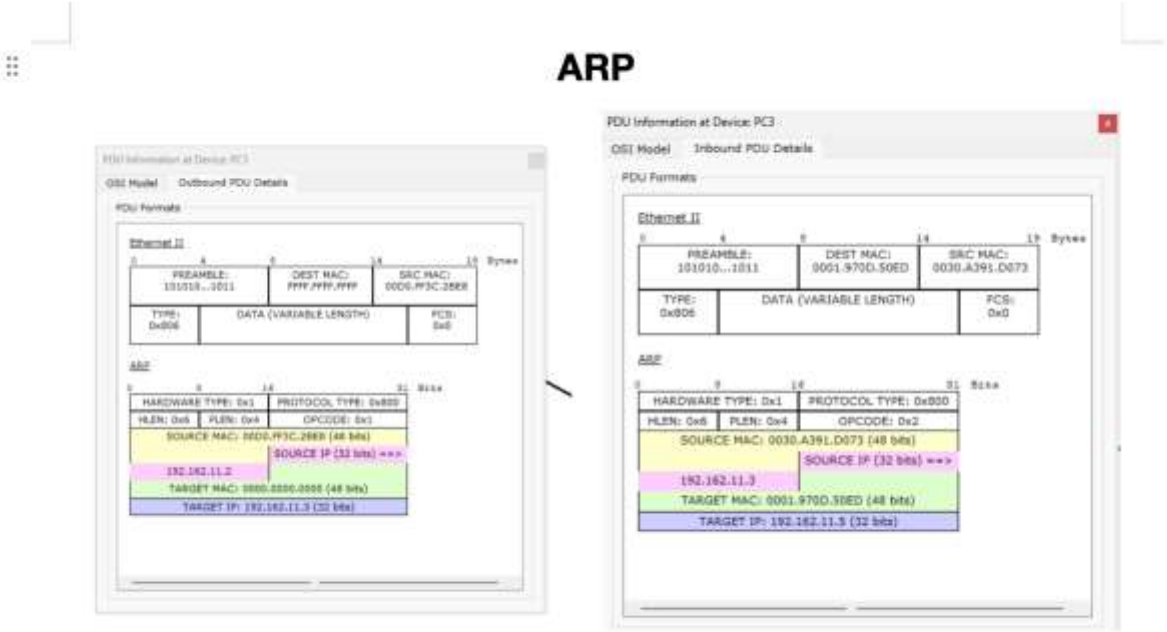
conf t

enable secret IP

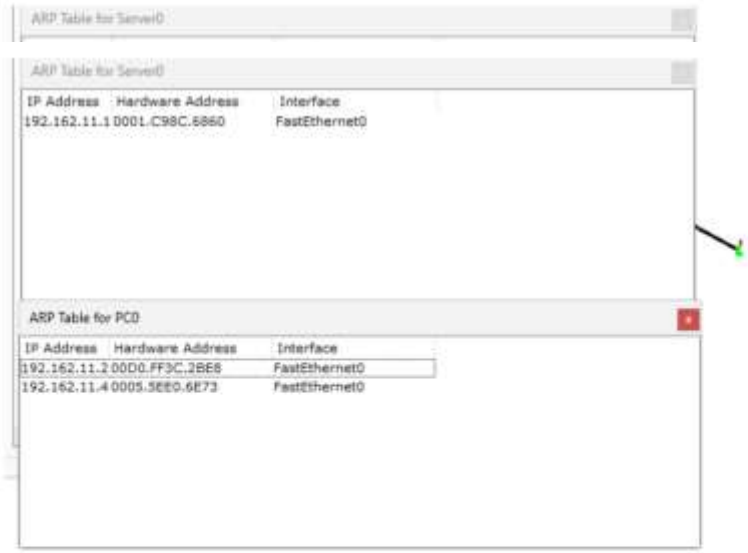
int fa 0/0

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

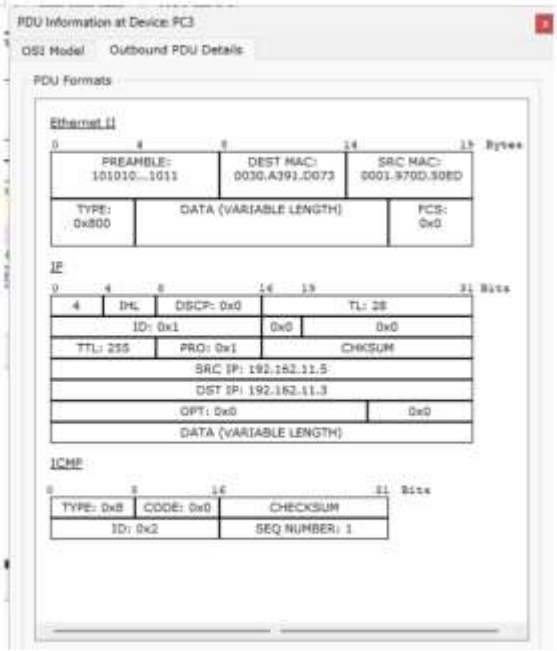
Network diagram:



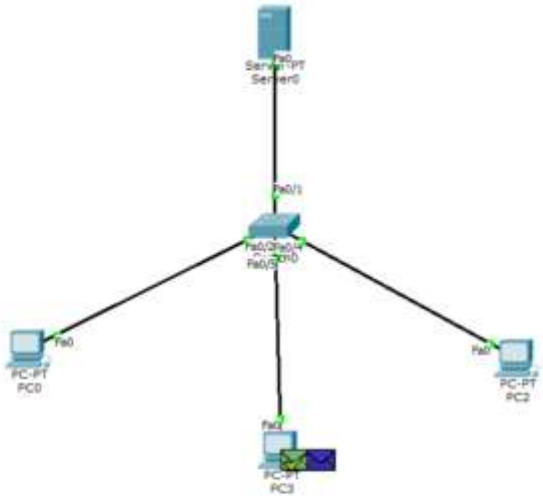
THE MAC ADDRESS before and after encrypting and sending



ARP table of pc0 and server after sending a simple PDU



The address of sent one



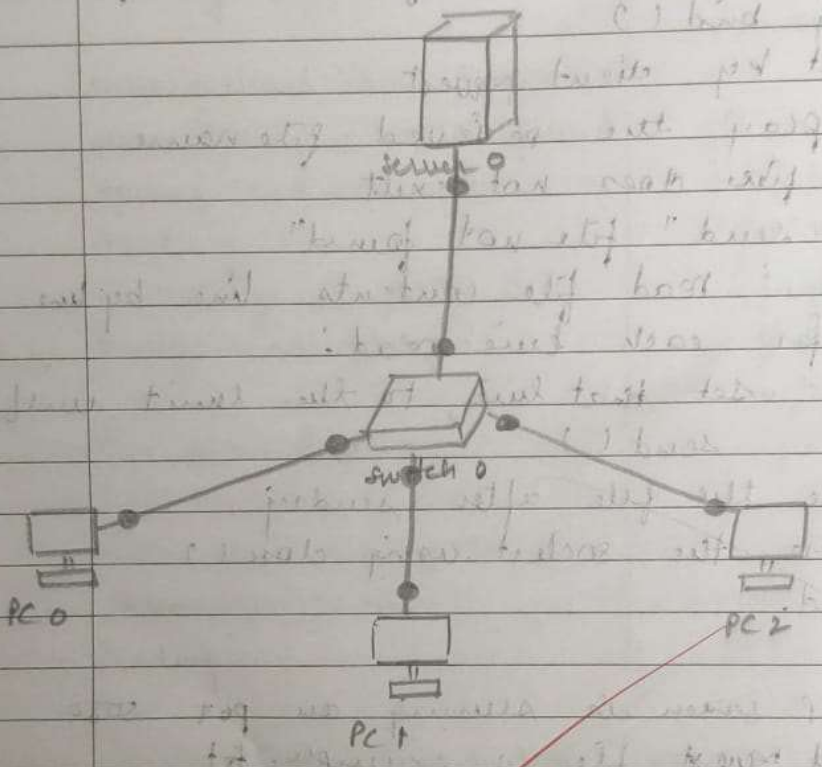
topology

Configuration:

Date 19/11/25
Page

Address Resolution Protocol (ARP)

- ARP is used to map an IP address to the MAC address
- ARP is used to get data link layer address MAC address with the help of IP address



PC0

IP address : 192.168.11.1

DNS server : 192.168.11.4

PC1

IP address : 192.168.11.2

DNS server : 192.168.11.4

PART - B

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

Code:

```
#include <stdio.h>
int min(int x, int y)
{
   return (x < y) ? x : 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("\n Second | Packet received | Packet sent | Packet left | Dropped\n");
   printf("-------------------------------------------------------------\n");
   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("\n Second | 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("%6d | %15d |", i + 1, inp[i]);

        mini = min(count, process);
        printf(" %11d |", mini);

        count -= mini;
        printf(" %12d | %7d\n", count, drop);

        drop = 0;
    }
    // Process remaining packets after all seconds
    for (; count != 0; i++)
    {
 if (count > cap)
        {
            drop = count - cap;
            count = cap;
        }

        printf("%6d | %15d |", i + 1, 0);

        mini = min(count, process);
        printf(" %11d |", mini);

        count -= mini;
        printf(" %12d | %7d\n", count, drop);
    }

    return 0;
}
```

Output:

Date 29/10/25
Page

Leaky Bucket Simulation

Algorithm :-

1. Start the program
2. Read bucket capacity, processing rate and no. of seconds.
3. For each second, read the no. of incoming packet
4. Add incoming packets to the bucket
5. If packets exceed capacity, drop the extra packets.
6. Send packets equal to minimum of (packets in bucket, processing rate)
7. Update the number of packet left in bucket
8. Repeat for all seconds
9. After all inputs, continue sending remaining packets until the bucket is empty
10. Stop the program

Output enter bucket size : 5
enter the processing rate : 2
enter the number of seconds you want to simulate : 3
enter the size of packet entering at 1 sec : 5
enter the size of packet entering at 2 sec : 4
enter the size of packet entering at 3 sec : 3

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

Code:

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

int main(int argc, char *argv[])
{
   int sockfd, portno, n;
   struct sockaddr_in serv;
   struct hostent *server;
   char buffer[256];
   char c[20000];

   if (argc < 3) {
      printf("Error: insufficient arguments.\n");
      printf("Usage: %s <hostname> <port>\nExample: %s 127.0.0.1 7777\n",
argv[0], argv[0]);
      exit(1);
   }
```

```
    portno = atoi(argv[2]);

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

    // Get server by name/IP
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr, "Error: no such host.\n");
        exit(1);
    }

    // Zero out the structure
    bzero((char *)&serv, sizeof(serv));
    serv.sin_family = AF_INET;
    bcopy((char *)server->h_addr, (char *)&serv.sin_addr.s_addr, server-
>h_length);
    serv.sin_port = htons(portno);

    // Connect to server
    if (connect(sockfd, (struct sockaddr *)&serv, sizeof(serv)) < 0) {
        perror("Error connecting");
```

```
    exit(1);
  }

  printf("Enter the file path (complete path): ");
  scanf("%s", buffer);

  // Send filename to server
  n = write(sockfd, buffer, strlen(buffer));
  if (n < 0) {
    perror("Error writing to socket");
    exit(1);
  }

  bzero(c, sizeof(c));
  printf("Reading file contents from server...\n");

  // Read file contents
  n = read(sockfd, c, sizeof(c) - 1);
  if (n < 0) {
    perror("Error reading from socket");
    exit(1);
  }

  printf("\nClient: Display content of %s\n----------------------------------\n",
buffer);
  fputs(c, stdout);
  printf("\n----------------------------------\n");
```

```
    close(sockfd);
    return 0;
}
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

int main(int argc, char *argv[])
{
    int sockfd, newsockfd, portno, len, n;
    char buffer[256], c[2000], cc[20000];
    struct sockaddr_in serv, cli;
    FILE *fd;

    if (argc < 2) {
        printf("Error: no port number provided.\n");
        printf("Usage: %s <port>\nExample: %s 7777\n", argv[0], argv[0]);
        exit(1);
    }

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

```
if (sockfd < 0) {
    perror("Error opening socket");
    exit(1);
}

// Initialize server address structure
bzero((char *)&serv, sizeof(serv));
portno = atoi(argv[1]);
serv.sin_family = AF_INET;
serv.sin_addr.s_addr = INADDR_ANY;
serv.sin_port = htons(portno);

// Bind socket
if (bind(sockfd, (struct sockaddr *)&serv, sizeof(serv)) < 0) {
    perror("Error on binding");
    close(sockfd);
    exit(1);
}

// Listen for incoming connections
listen(sockfd, 5);
len = sizeof(cli);
printf("Server: waiting for connection on port %d...\n", portno);

// Accept a client
newsockfd = accept(sockfd, (struct sockaddr *)&cli, (socklen_t *)&len);
if (newsockfd < 0) {
```

```
    perror("Error on accept");
    close(sockfd);
    exit(1);
}

// Read filename from client
bzero(buffer, 255);
n = read(newsockfd, buffer, 255);
if (n < 0) {
    perror("Error reading from socket");
    close(newsockfd);
    close(sockfd);
    exit(1);
}

printf("Server received filename: %s\n", buffer);

// Try to open the file
fd = fopen(buffer, "r");
if (fd != NULL) {
    printf("Server: file '%s' found, reading and sending...\n", buffer);
    bzero(cc, sizeof(cc));
    while (fgets(c, sizeof(c), fd) != NULL) {
        strcat(cc, c);
    }
    fclose(fd);
```

```
        // Send file content to client
        n = write(newsockfd, cc, strlen(cc));
        if (n < 0)
            perror("Error writing to socket");
        else
            printf("File transfer complete.\n");
    } else {
        printf("Server: file not found.\n");
        n = write(newsockfd, "Error: file not found.\n", 24);
        if (n < 0)
            perror("Error writing to socket");
    }

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

Output:

Date 12/11/25
Page

Part B Client Server Communication

Algorithm :- (Client Side)

1) Sfd = create a socket (....) system call
2) Connect the socket to the address of server using the connect (sfd, ...) system call. The IP address of the server machine and port number of the server service need to be provided
3) Read the file name from standard input by n = read (stdin, buffer, sizeof(buffer))
4) Write file name to the socket using write (sfd, buffer, n)
5) Read file name to the socket using write from socked by m = read (sfd, buffer, size of (buffer))
6) Display the file contents to standard output by write (stdin, buffer, m)
7) Go to step 5 if $m > 0$
8) Close socket by close (sfd)

Algorithm (Server side)

1) Sfd = create socket with socket system call
2) Bind the socket to an address using the bind system call. Keep the s_addr to INADDR_ANY Assign a port number b/w 3000 to 5000 to sin_port
3) Wait for connections with listen (sfd....) system call
4) Read filename from the socket by n = read (sfd, buffer, sizeof (buffer))
5) Write the file content to socked by write

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

Code:

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

#define PORT 8080
#define MAXLINE 1024

int main() {
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr, cliaddr;
    socklen_t len;
    ssize_t n;

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

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

    // Server info
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

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

    printf("UDP Server is running on port %d...\n", PORT);

    len = sizeof(cliaddr);

    // Receive filename from client
    n = recvfrom(sockfd, (char *)buffer, MAXLINE, 0, (struct sockaddr *)&cliaddr,
&len);
    buffer[n] = '\0';
    printf("Client requested file: %s\n", buffer);

    FILE *fp = fopen(buffer, "r");
    if (fp == NULL) {
```

```
        char *msg = "File not found!";
        sendto(sockfd, msg, strlen(msg), 0, (struct sockaddr *)&cliaddr, len);
        printf("File not found, message sent to client.\n");
    } else {
        // Read and send file content
        while (fgets(buffer, MAXLINE, fp) != NULL) {
            sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr *)&cliaddr, len);
        }
        fclose(fp);
        printf("File sent successfully.\n");
    }

    close(sockfd);
    return 0;
}
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define MAXLINE 1024

int main() {
    int sockfd;
    char buffer[MAXLINE];
```

```
    struct sockaddr_in servaddr;
    socklen_t len;

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

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

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

    printf("Enter the filename to request: ");
    fgets(buffer, MAXLINE, stdin);
    buffer[strcspn(buffer, "\n")] = '\0';  // remove newline

    // Send filename to server
    sendto(sockfd, buffer, strlen(buffer), 0, (const struct sockaddr *)&servaddr,
sizeof(servaddr));

    printf("Request sent. Waiting for file content...\n\n");

    len = sizeof(servaddr);
```

```
    // Receive file contents
    ssize_t n;
    while ((n = recvfrom(sockfd, buffer, MAXLINE, 0, (struct sockaddr
*)&servaddr, &len)) > 0) {
        buffer[n] = '\0';
        printf("%s", buffer);
        if (n < MAXLINE - 1) break;  // assume end of file
    }

    printf("\n\nFile transfer complete.\n");

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

Output:

Date 12/11/25
Page

Client Server Using UDP socket

1) Start
2) Use socket (AF_INET, SOCK_DGRAM, 0) to create datagram (UDP) socket
3) Initialise
   - set sin_family = AF_INET
   - Set sin_addr.s_addr = INADDR_ANY
4) Bind the socket to given IP and port using bind()
5) wait by client request
6) Display the received file name
7) If file does not exist
   send "file not found"
   else : read file contents line by line
   for each line read:
   set that line to the limit with send()
8) close the file after sending
9) close the socket using close()
10) end.

output UDP server is running on port 5080
Until request file : example.txt

This is a sample file
This denotes UDP file transfer
File transfer complete

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 generator polynomial:\n");
   fgets(gen, sizeof(gen), stdin);
   gen[strcspn(gen, "\n")] = '\0';  // remove newline if present
   printf("Generator polynomial (CRC-CCITT): %s\n", gen);

   genlen = strlen(gen);
   k = genlen - 1;

   printf("Enter the message:\n");
   fgets(msj, sizeof(msj), stdin);
   msj[strcspn(msj, "\n")] = '\0';  // remove newline

   n = strlen(msj);

   // Append k zeros to the message
```

```
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);

// Division (XOR)
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++;
        }
    }
}
// Get remainder
for (i = 0; i < k; i++)
```

```
    rem[i] = a[n + i];
rem[k] = '\0';
printf("\nChecksum (Remainder):\n");
puts(rem);
// Append checksum to message
printf("\nTransmitted message (with checksum):\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);
// Receiver side
printf("\nEnter the received message:\n");
fgets(s, sizeof(s), stdin);
s[strcspn(s, "\n")] = '\0';  // remove newline
n = strlen(s);
// Division on received message
for (i = 0; i < n - k; 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 - k + i];
rem[k] = '\0';
// Check for error
flag = 0;
for (i = 0; i < k; i++)
{
    if (rem[i] == '1')
        flag = 1;
}
if (flag == 0)
    printf("\nReceived message is error-free ✅\n");
else
    printf("\nReceived message contains errors ❌\n");
return 0;
}
```

Output:

Output

```
Enter the generator polynomial:
101
Generator polynomial (CRC-CCITT): 101
Enter the message:
110101

Message polynomial appended with zeros:
11010100

Checksum (Remainder):
11

Transmitted message (with checksum):
11010111

Enter the received message:
11010111

Received message is error-free ✅


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

Date 29/10/25
Page

CRC Error detection

Algorithm :-

1. Start the program
2. enter the generator polynomial, and the message
3. Append zeroes to the message (generator length - 1)
4. Divide the appended message by the generator using XOR to find the reminder.
5. Append the reminder to the original message to form the transmitted one
6. enter the recieved message
7. Divide the recieved message by the generator again
8. If the remainder is all zeroes : message is error-free, else - error detected
9. Stop the program

Output

enter the generator polynomial : 10 1
generator polynomial (CRC - CCITT) : 101
enter the message : 1101 01
Message Polynomial appended with zeroes :
1101 0100
checksum (remainder) : 11
Message with checksum appended : 1101 0111
enter recieved message : 1101 0111
Recieved message is error-free