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:

DAIVYA PRIYANKKUMAR SHAH

(1BM23CS084)

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 Daivya Priyankkumar Shah(1BM23CS084) 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).

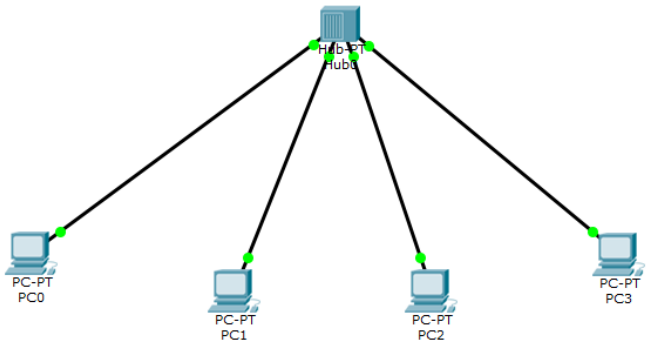
PART – B	
Serial No.	**Name of Expirement**
1.	Write a program for congestion control using Leaky bucket algorithm.
2.	Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.
3.	Using UDP sockets,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

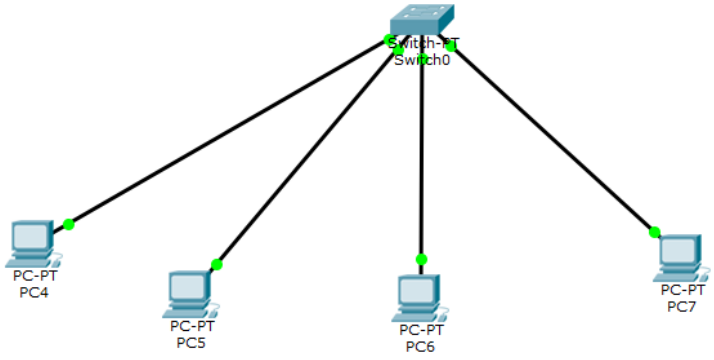
GitHub Link: https://github.com/daivya17/CN_lab

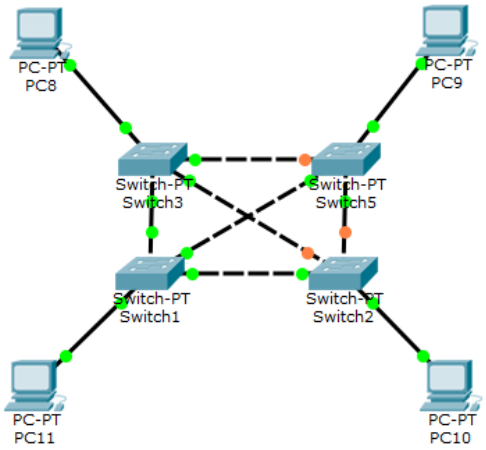
PART - A

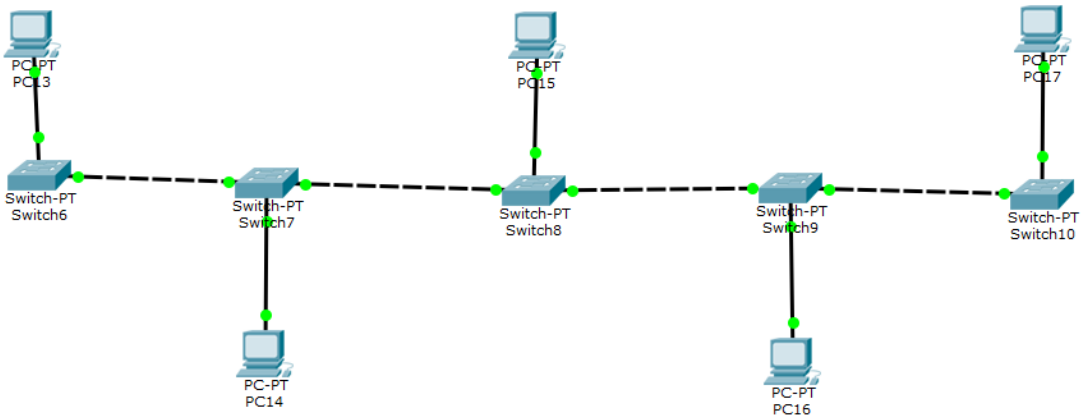
Program 1: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Network diagram:

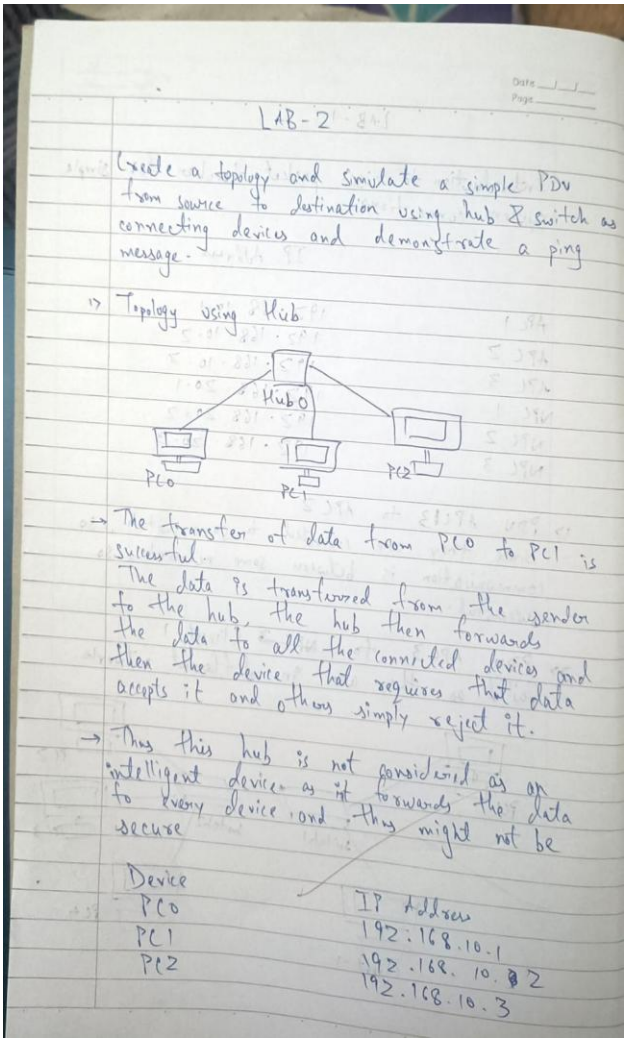





PC-PT
PC8
PC-PT
PC9
Switch-PT
Switch3
Switch-PT
Switch5
Switch-PT
Switch1
Switch-PT
Switch2
PC-PT
PC11
PC-PT
PC10


PC-PT
PC13
PC-PT
PC15
PC-PT
PC17
Switch-PT
Switch6
Switch-PT
Switch7
Switch-PT
Switch8
Switch-PT
Switch9
Switch-PT
Switch10
PC-PT
PC14
PC-PT
PC16

Configuration:



LAB-2

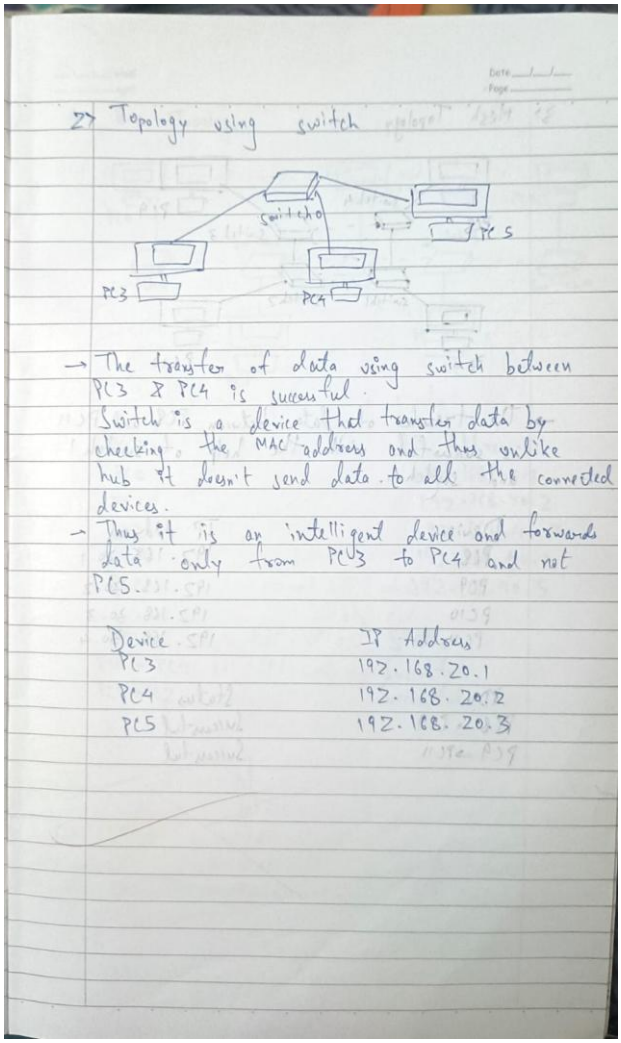
Create a topology and simulate a simple PDU from source to destination using hub & switch as connecting devices and demonstrate a ping message.

1) Topology using Hub

→ The transfer of data from PC0 to PC1 is successful.
The data is transferred from the sender to the hub, the hub then forwards the data to all the connected devices and then the device that requires that data accepts it and others simply reject it.

→ Thus this hub is not considered as an intelligent device as it forwards the data to every device and thus might not be secure

Device	IP Address
PC0	192.168.10.1
PC1	192.168.10.2
PC2	192.168.10.3

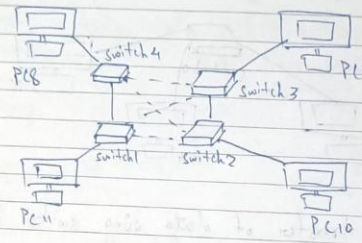


2) Topology using switch

→ The transfer of data using switch between PC3 & PC4 is successful.
Switch is a device that transfers data by checking the MAC address and thus unlike hub it doesn't send data to all the connected devices.
→ Thus it is an intelligent device and forwards data only from PC3 to PC4 and not PC5.

Device	IP Address
PC3	192.168.20.1
PC4	192.168.20.2
PC5	192.168.20.3

3) Mesh Topology

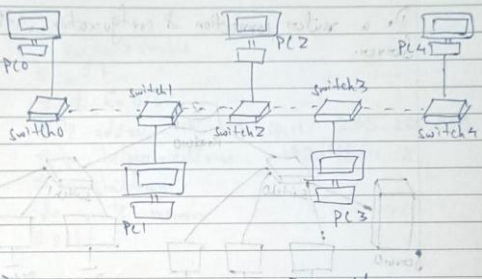


→ The transfer of data between PC8 and PC11 is successful with the help of switch 1 and switch 4.

Devices	IP Address
PC8	192.168.30.1
PC9	192.168.30.2
PC10	192.168.30.3
PC11	192.168.30.4

PDU	Status
PC8 → PC9	Successful
PC9 → PC11	Successful

4) Bus Topology

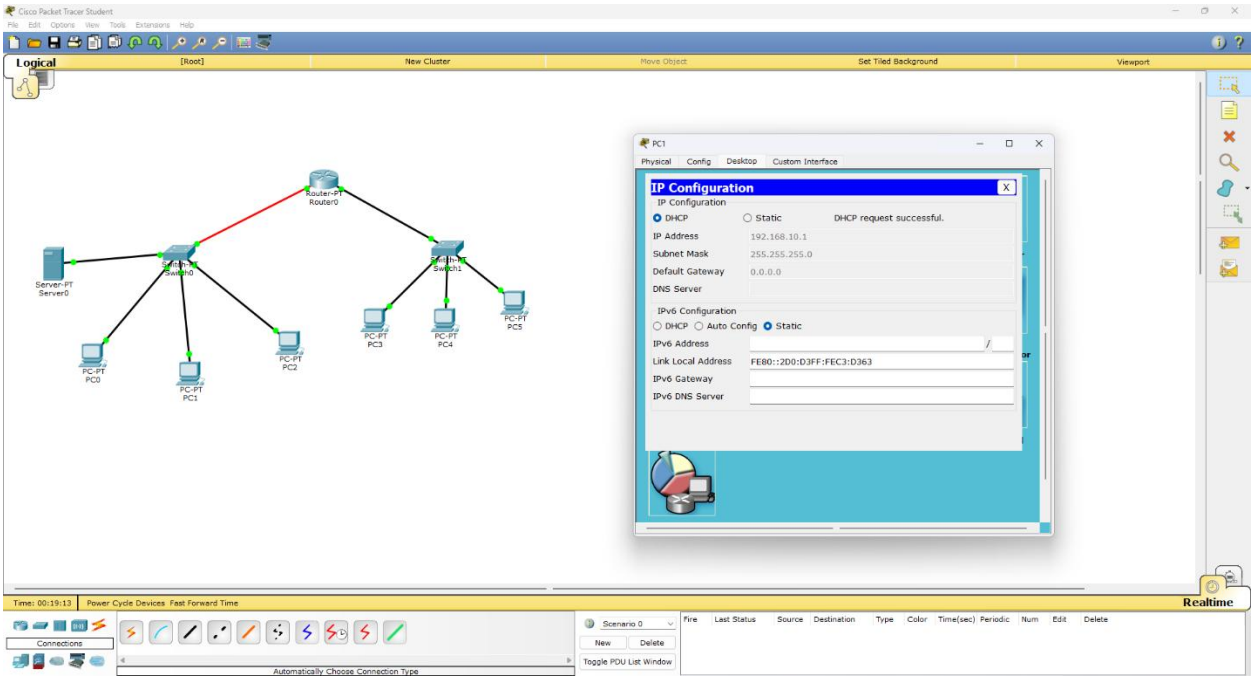


Devices	IP Address
PC0	192.168.40.1
PC1	192.168.40.2
PC2	192.168.40.3
PC3	192.168.40.4
PC4	192.168.40.5

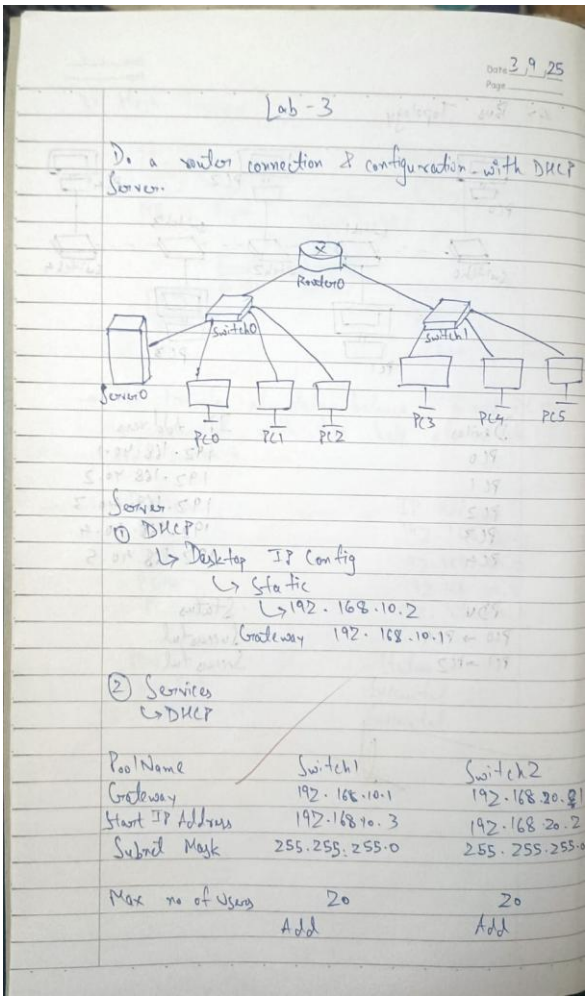
PDU	Status
PC0 → PC4	Successful
PC1 → PC2	Successful

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

Network diagram:



Configuration:



Date 3/9/25
Page

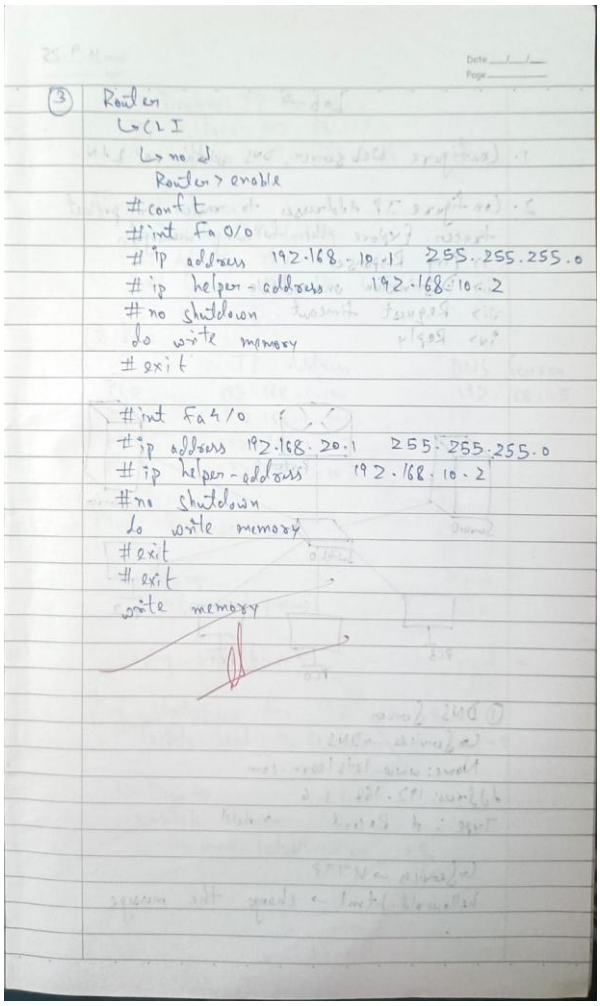
Lab - 3

Do a router connection & configuration with DHCP Server.

Server
① DHCP
↳ Desktop IP Config
↳ Static
↳ 192.168.10.2
Gateway 192.168.10.1

② Services
↳ DHCP

Pool Name	Switch1	Switch2
Gateway	192.168.10.1	192.168.20.1
Start IP Address	192.168.10.3	192.168.20.2
Subnet Mask	255.255.255.0	255.255.255.0
Max no of Users	20	20
	Add	Add



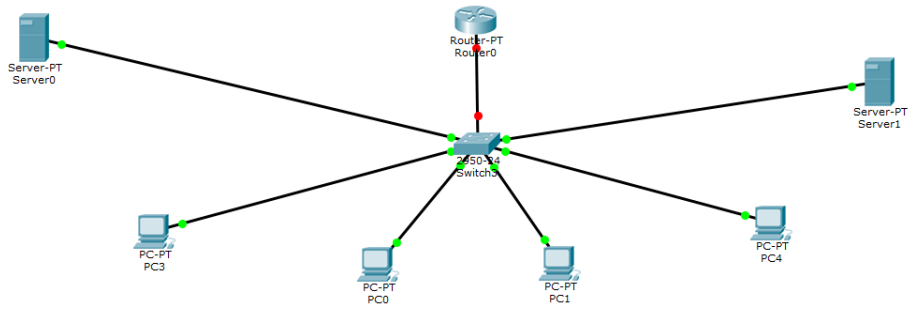
Date
Page

③ Router
↳ CLI
↳ no
Router > enable
conf t
int Fa 0/0
ip address 192.168.10.1 255.255.255.0
ip helper-address 192.168.10.2
no shutdown
do write memory
exit

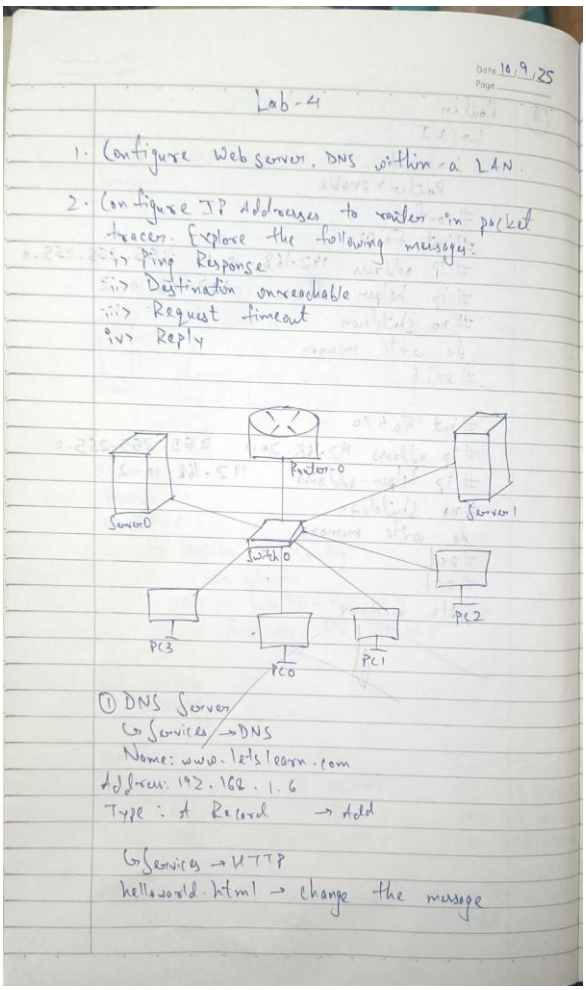
int Fa 4/0
ip address 192.168.20.1 255.255.255.0
ip helper-address 192.168.10.2
no shutdown
do write memory
exit
exit
write memory

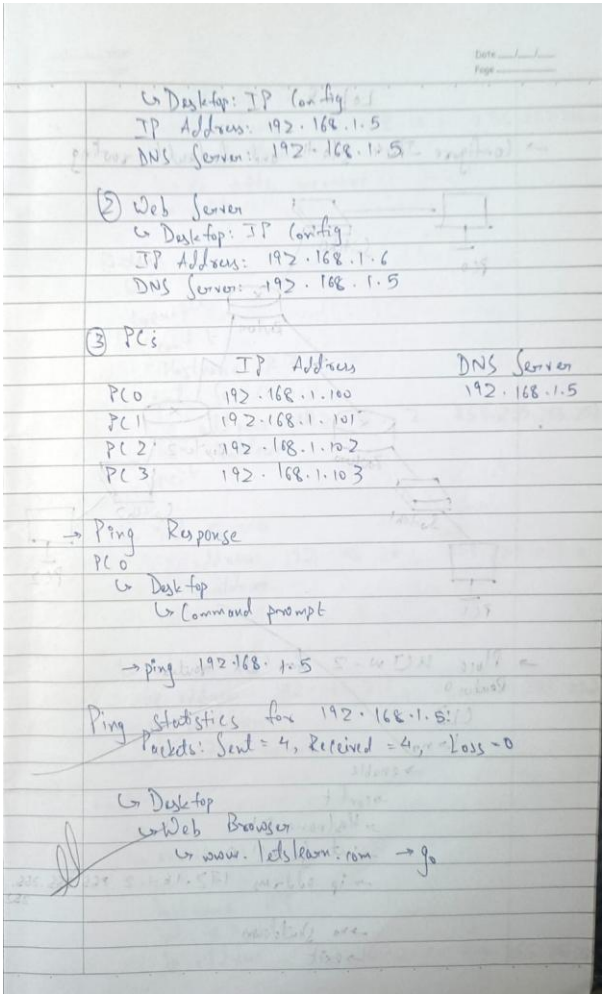
Program 3: Configure Web Server, DNS within a LAN.

Network diagram:



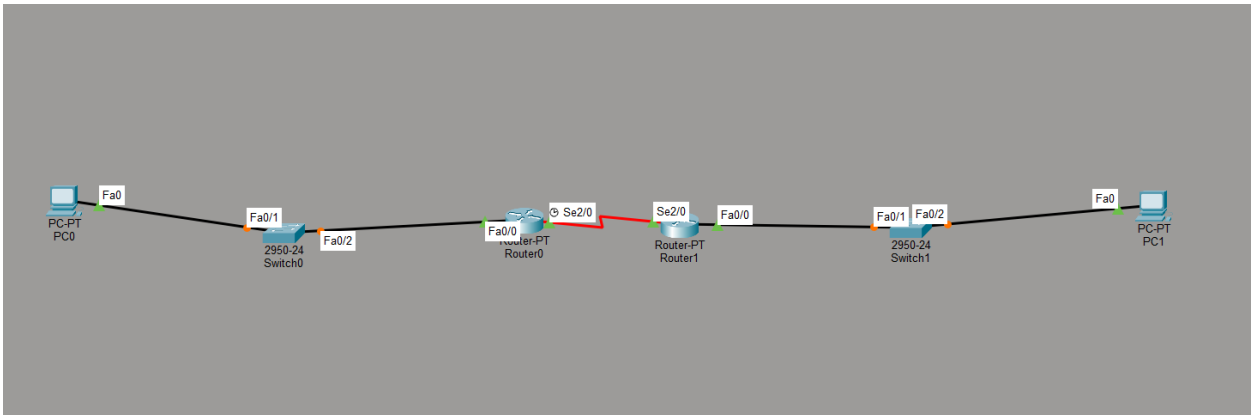
Configuration:



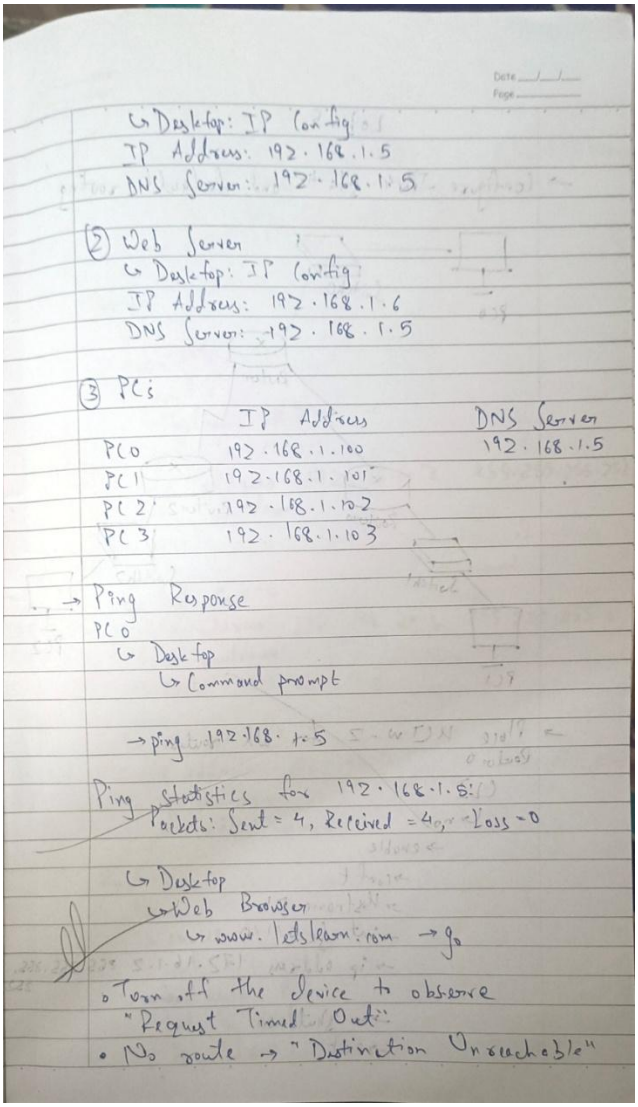


Program 4: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

Network diagram:



Configuration:



↳ Desktop: IP Config
IP Address: 192.168.1.5
DNS Server: 192.168.1.5

(2) Web Server
↳ Desktop: IP Config
IP Address: 192.168.1.6
DNS Server: 192.168.1.5

(3) PCs

	IP Address	DNS Server
PC0	192.168.1.100	192.168.1.5
PC1	192.168.1.101	
PC2	192.168.1.102	
PC3	192.168.1.103	

→ Ping Response
PC0
↳ Desktop
↳ Command prompt

→ ping 192.168.1.5

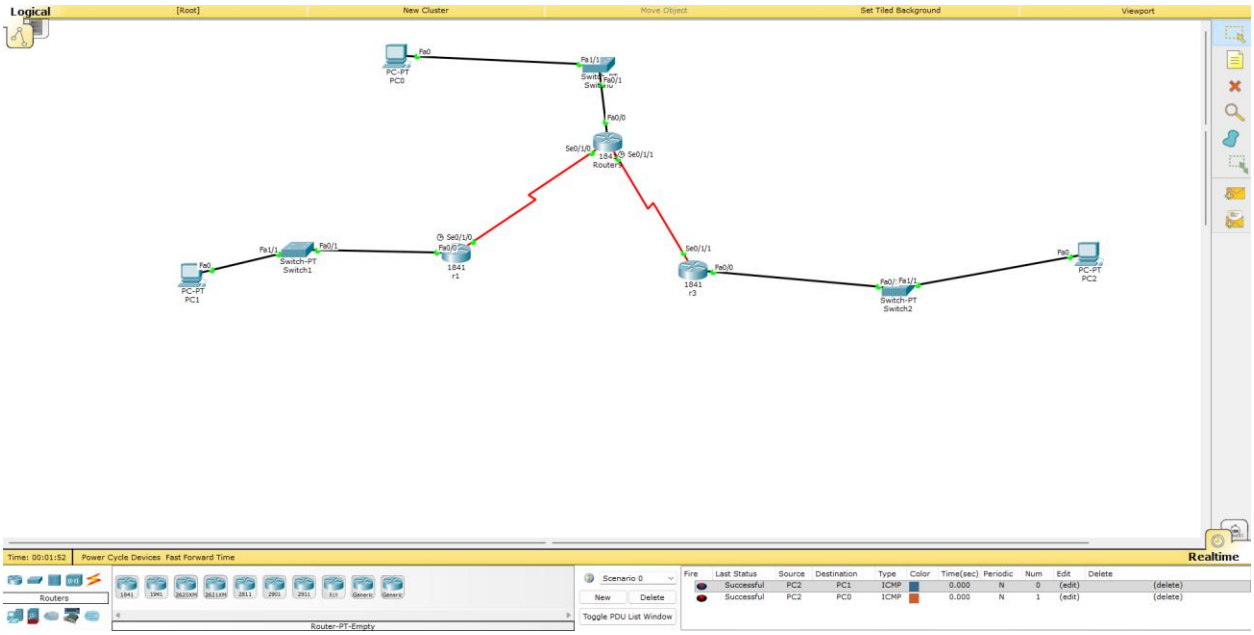
Ping statistics for 192.168.1.5:
Packets: Sent = 4, Received = 4, Loss = 0

↳ Desktop
↳ Web Browser
↳ www.letslearn.com → go

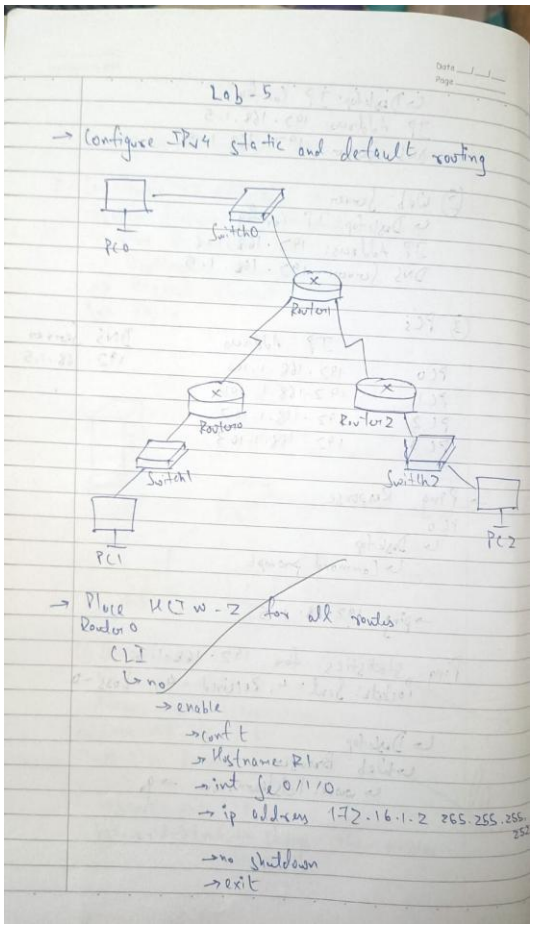
- Turn off the device to observe "Request Timed Out"
- No route → "Destination Unreachable"

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

Network diagram:



Configuration:



→ int Fa 0/0
→ ip address 192.168.10.1 255.255.255.0
→ no shutdown
→ do write memory
→ exit

Router1
↳ no
↳ enable
conf t
Hostname R2
int Se 0/1/0
ip address 172.16.1.2 255.255.255.252
no shutdown
exit

↳ int Fa 0/0
ip address 192.168.20.1 255.255.255.0
no shutdown
exit

↳ int Se 0/1/1
ip address 172.16.2.1 255.255.255.252
no shutdown
do write memory

Router2
↳ no
↳ enable
conf t
hostname R3
int Se 0/1/0
ip address 172.16.2.2 255.255.255.252
no shutdown
exit

int Fa 0/0
ip address 192.168.30.1 255.255.255.0
no shutdown
exit

Configure IP Address of all PC

	IP	Default
PC1	192.168.10.10	192.168.10.1
PC2	192.168.20.10	192.168.20.1
PC3	192.168.30.10	192.168.30.1

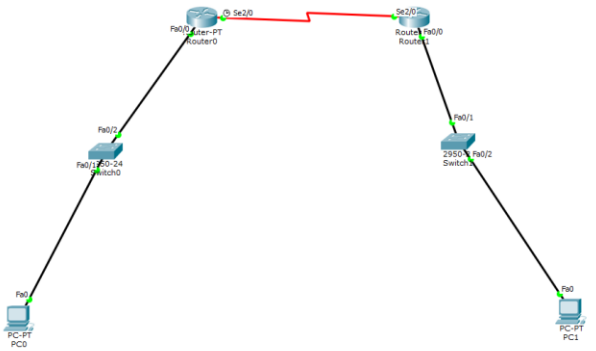
→ Router0
↳ enable
conf t
ip route 192.168.20.0 255.255.255.0 172.16.1.2
ip route 172.16.2.0 255.255.255.252 172.16.1.2
ip route 192.168.30.0 255.255.255.0 172.16.1.2

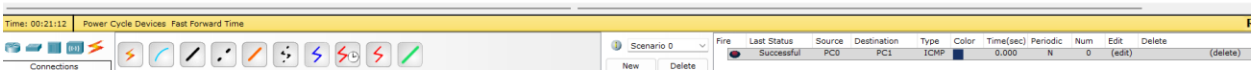
→ For router1
ip route 192.168.10.0 255.255.255.0 172.16.1.1
ip route 192.168.30.0 255.255.255.0 172.16.2.2

→ For router2
ip route 0.0.0.0 0.0.0.0 Se 0/1/0

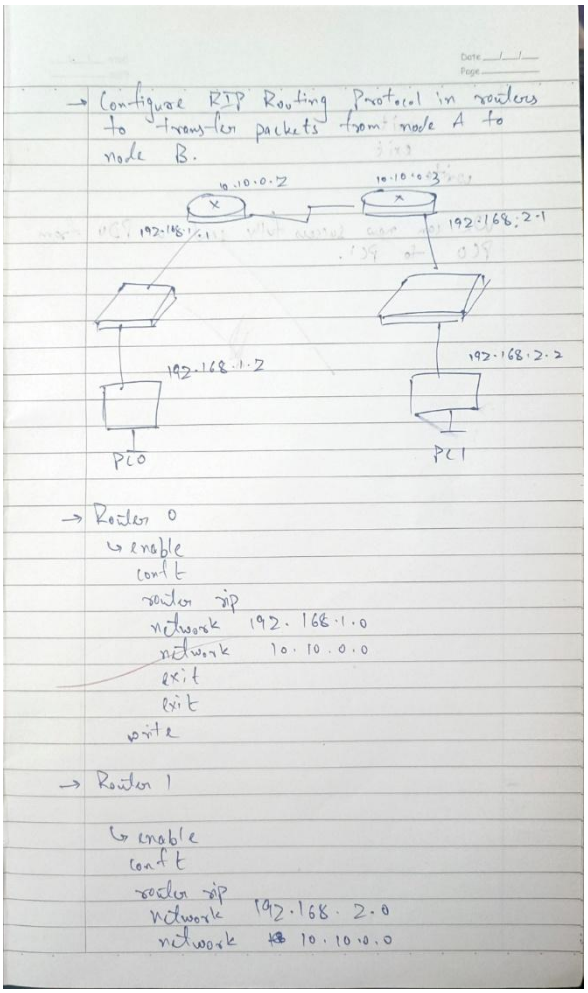
Program 6: Configure RIP routing Protocol in Routers.

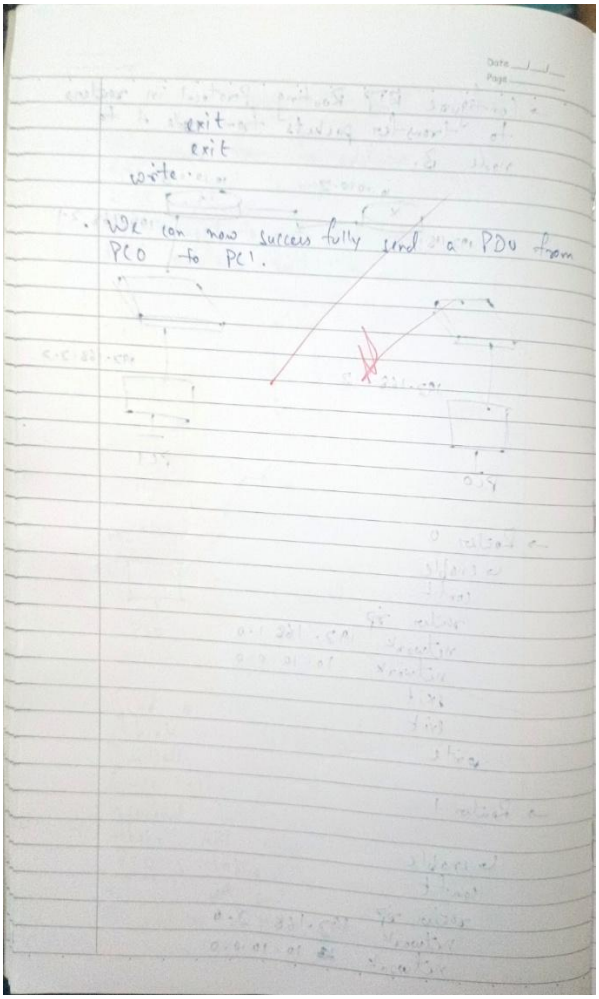
Network diagram:





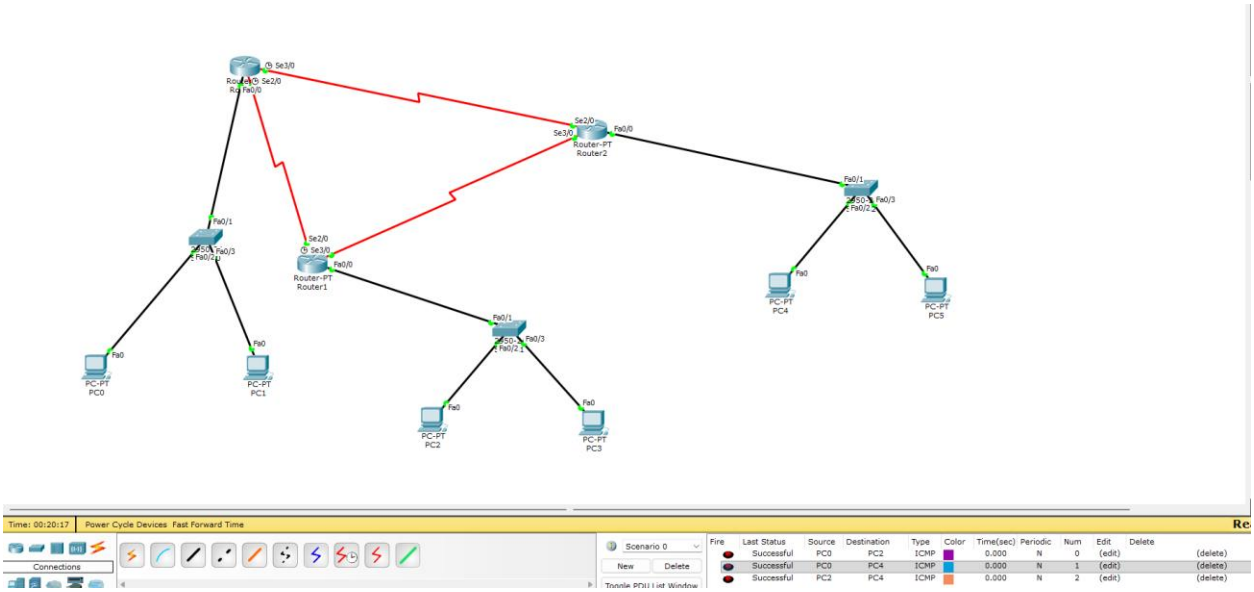
Configuration:



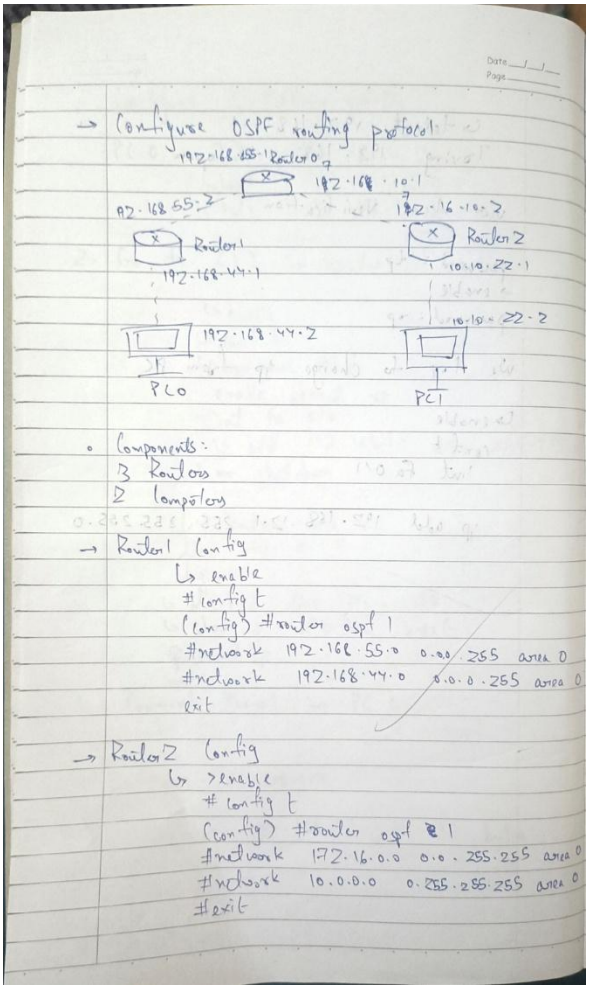


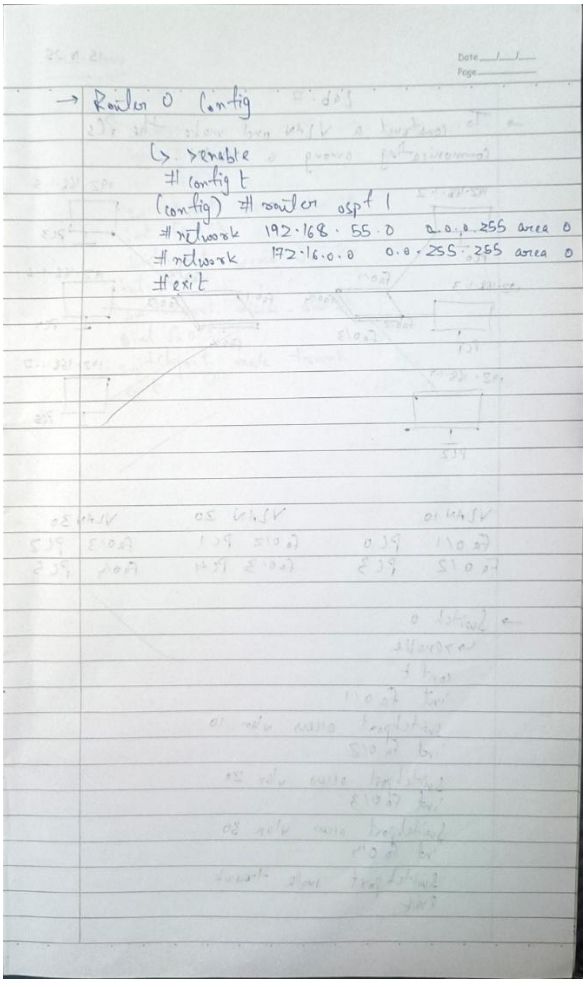
Program 7: Configure OSPF routing protocol.

Network diagram:



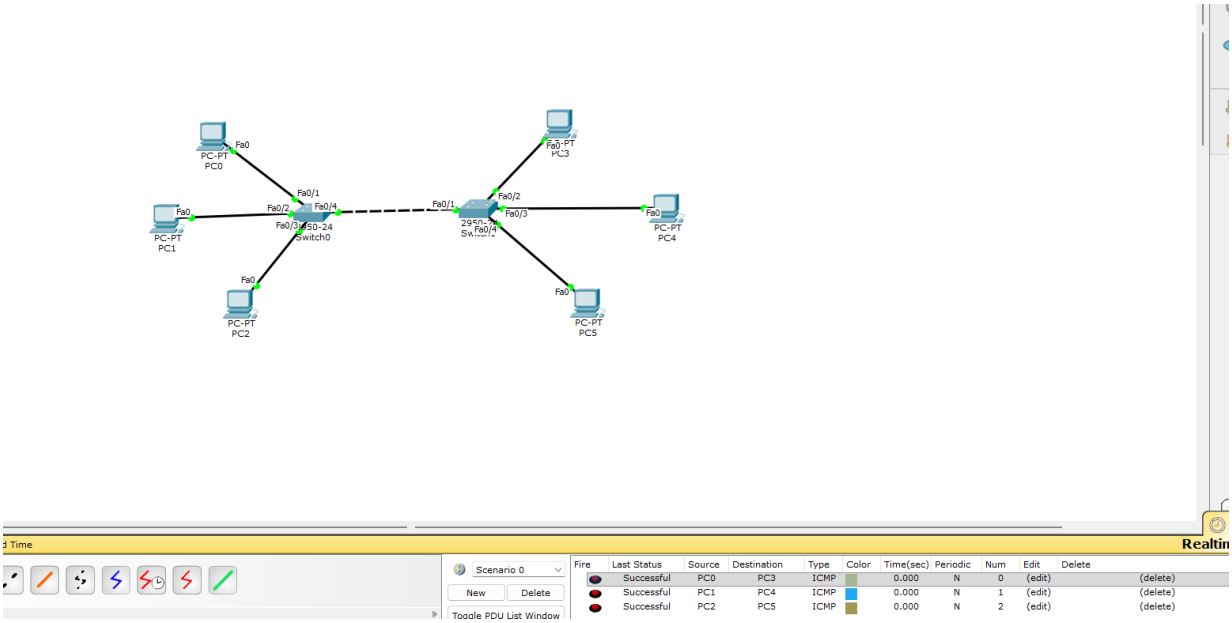
Configuration:



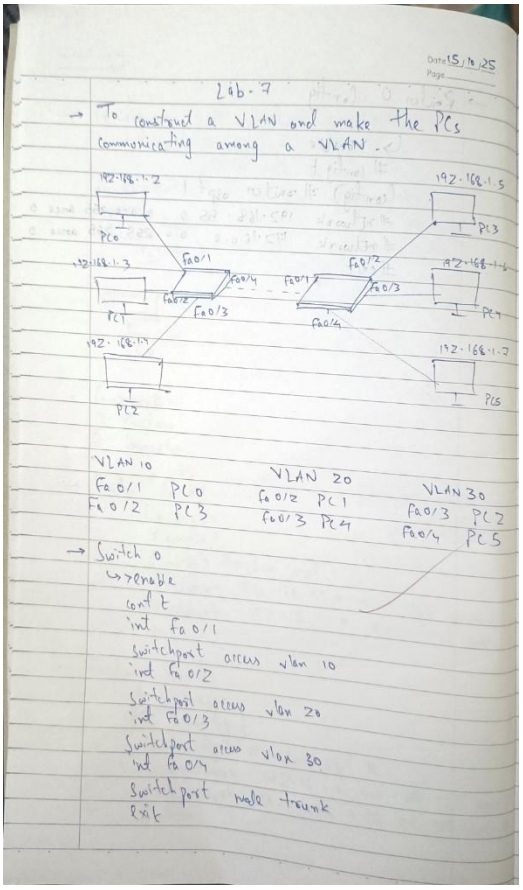


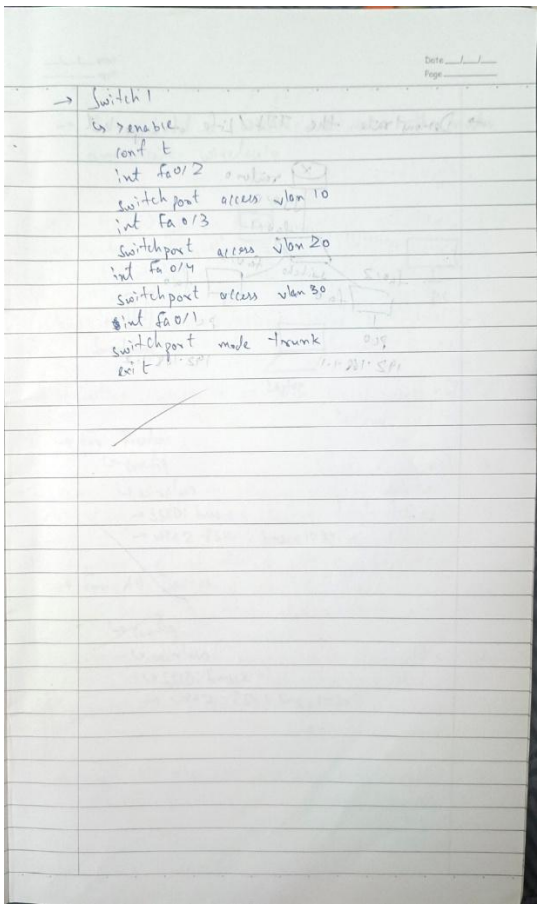
Program 8: To construct a VLAN and make the PC's communicate among a VLAN.

Network diagram:



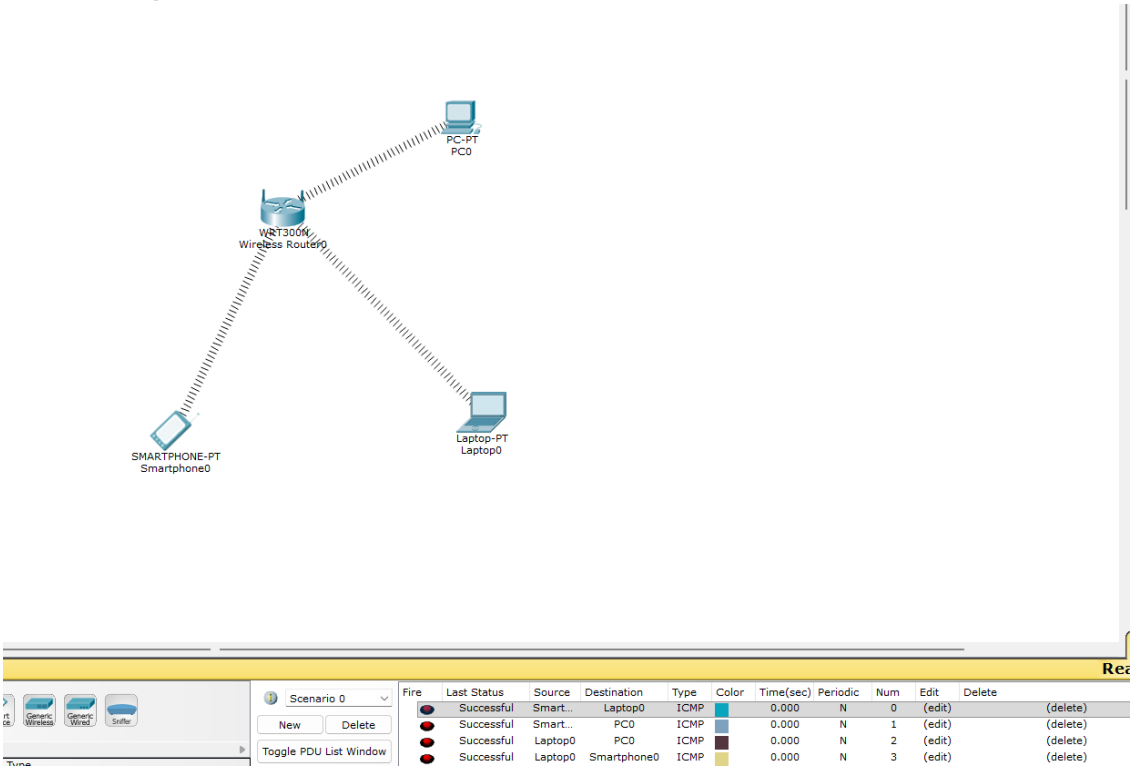
Configuration:



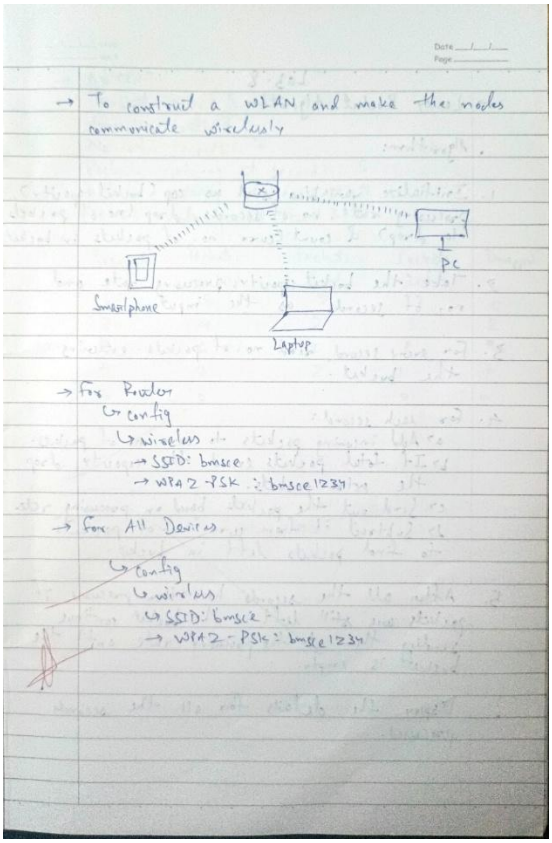


Program 9: To construct a WLAN and make the nodes communicate wirelessly.

Network diagram:

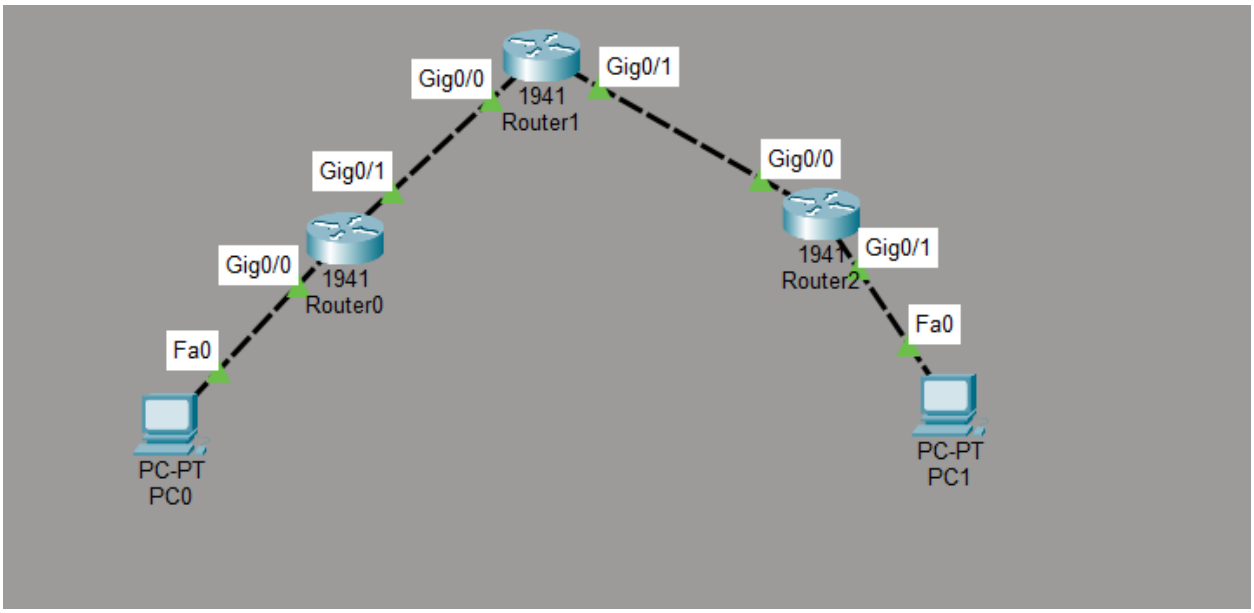


Configuration:

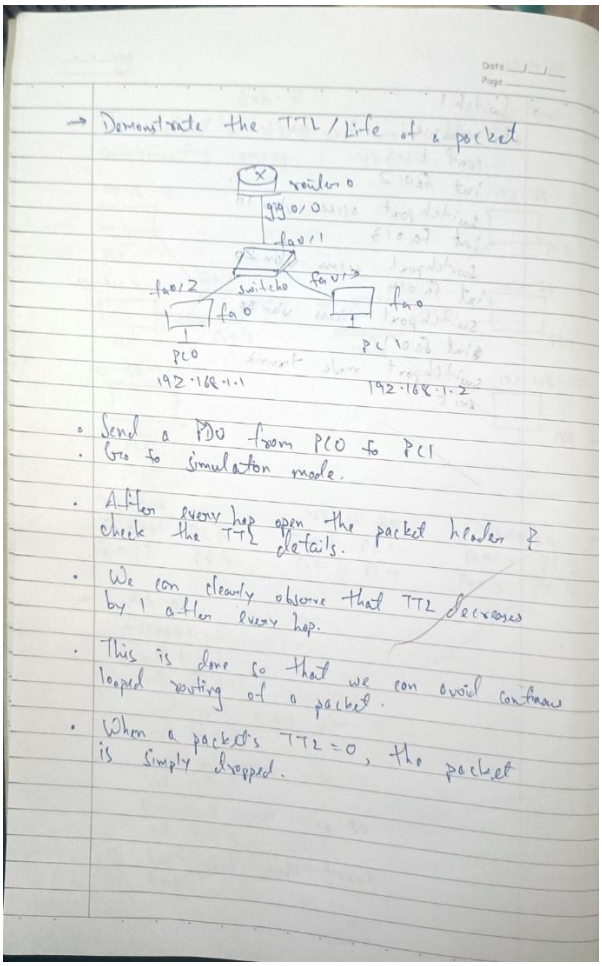


Program 10: Demonstrate the TTL/ Life of a Packet.

Network diagram:

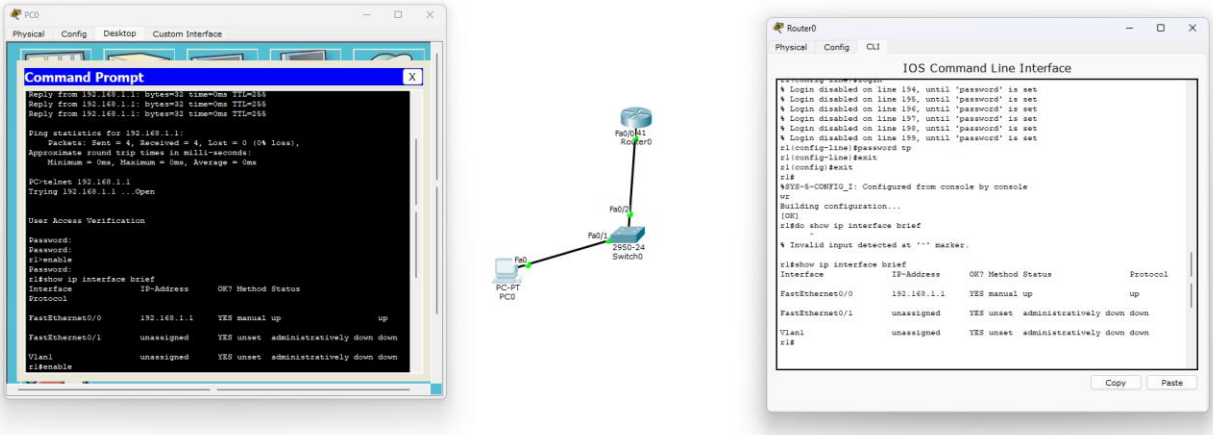


Configuration:

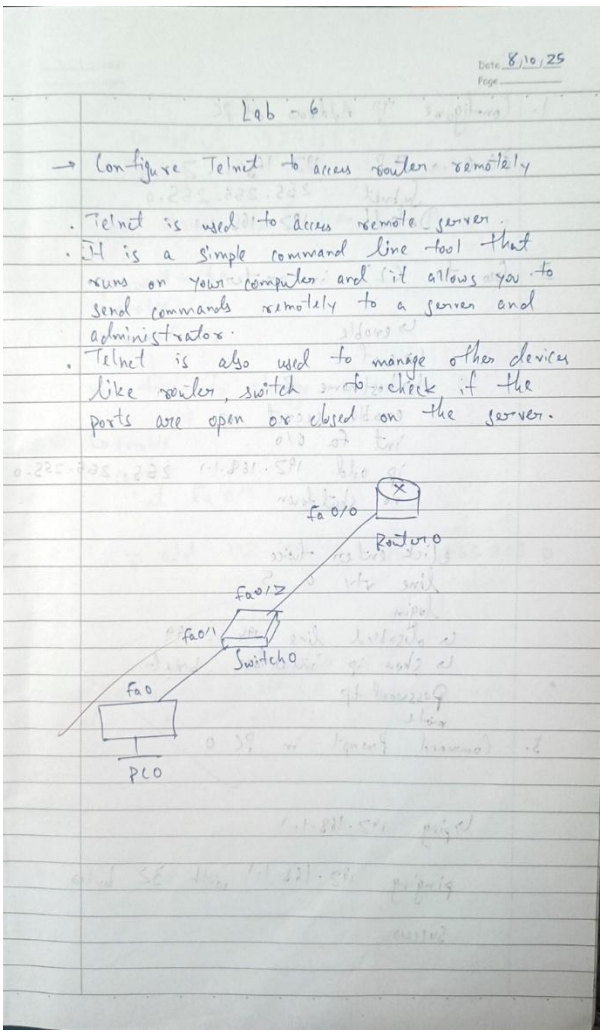


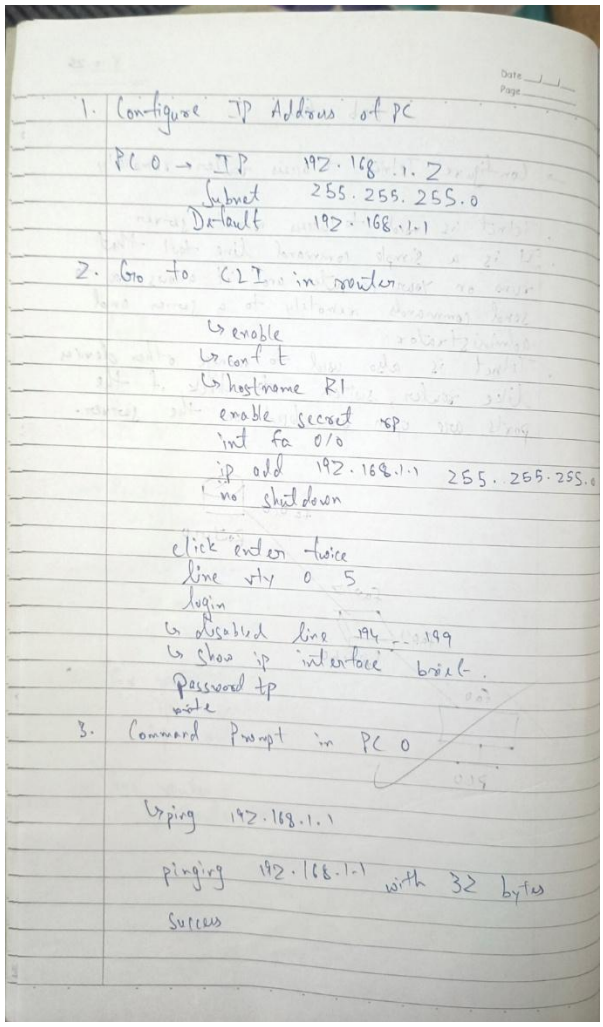
Program 11: To understand the operation of TELNET by accessing the router in server room from a PC in IT office.

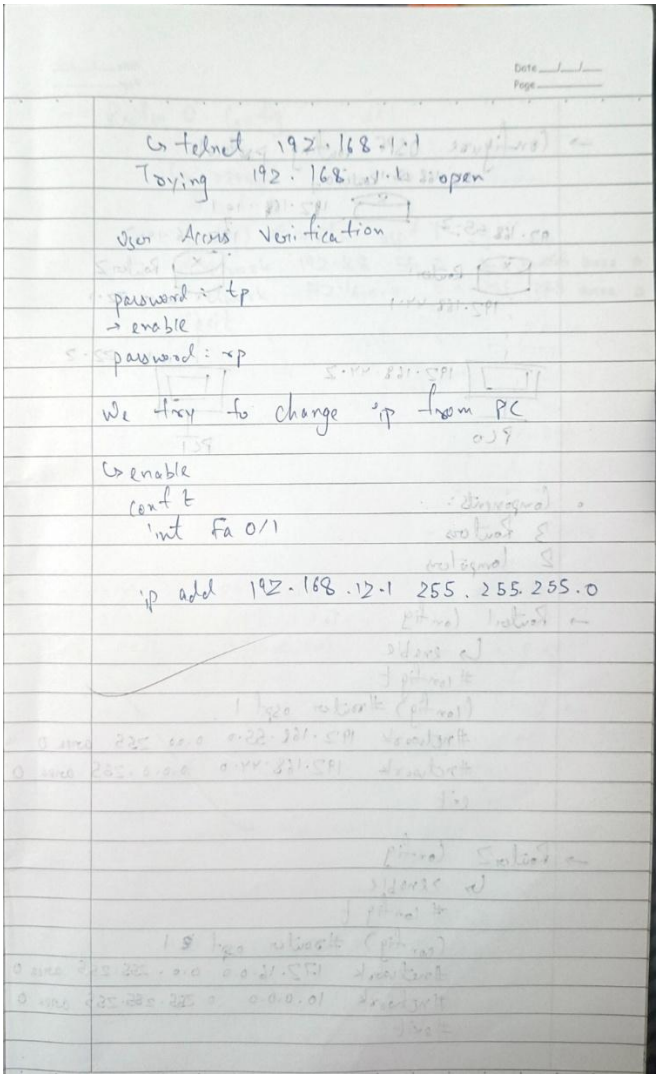
Network diagram:



Configuration:







Date __/__/__
Page _____

↳ telnet 192.168.1.1
Trying 192.168.1.1 ... open

User Access Verification

password : tp
→ enable
password : xp

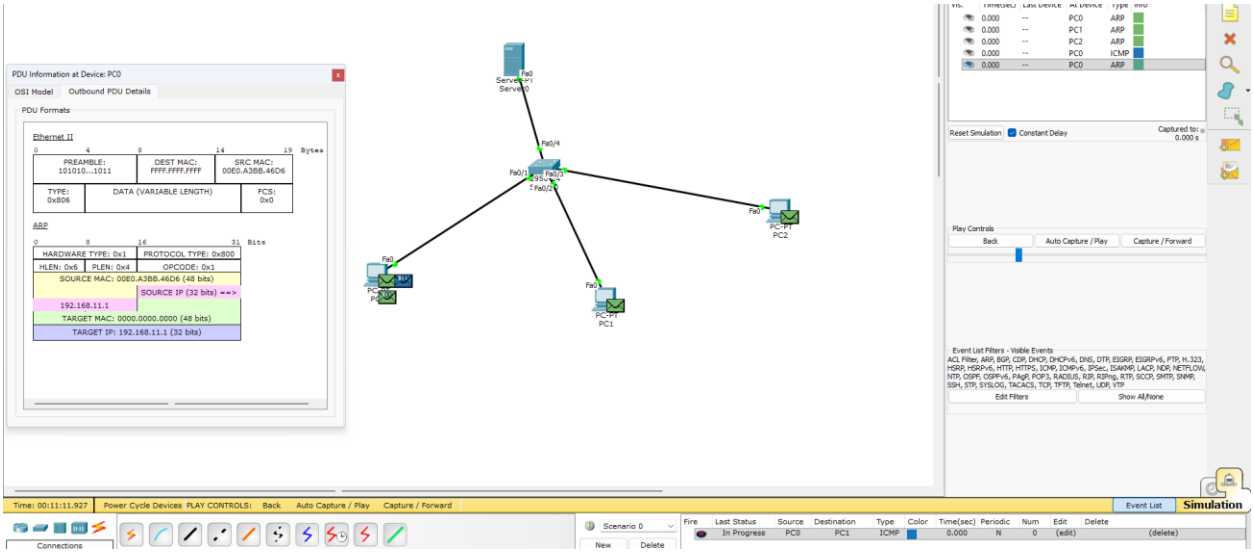
We try to change ip from PC

↳ enable
conf t
int fa 0/1

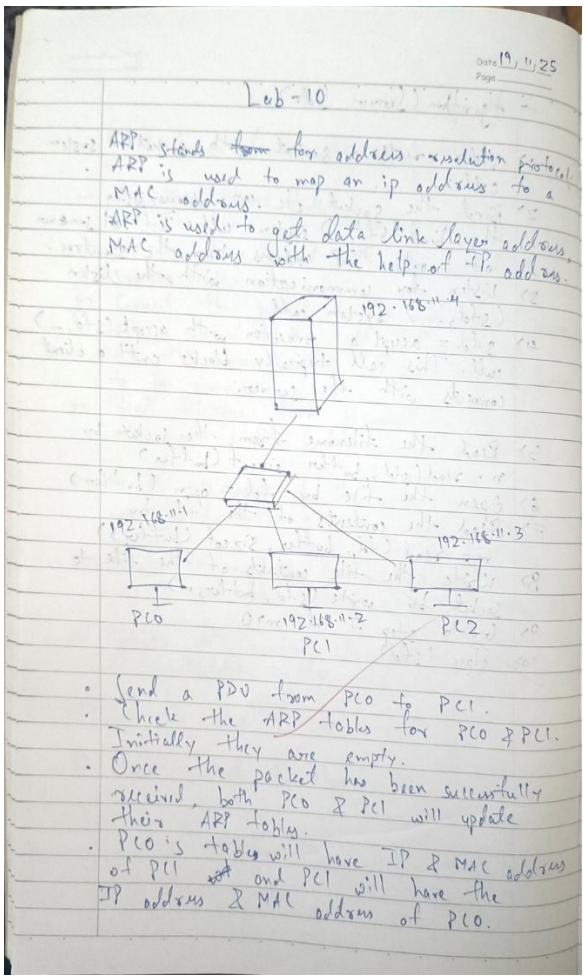
ip add 192.168.12.1 255.255.255.0

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

Network diagram:

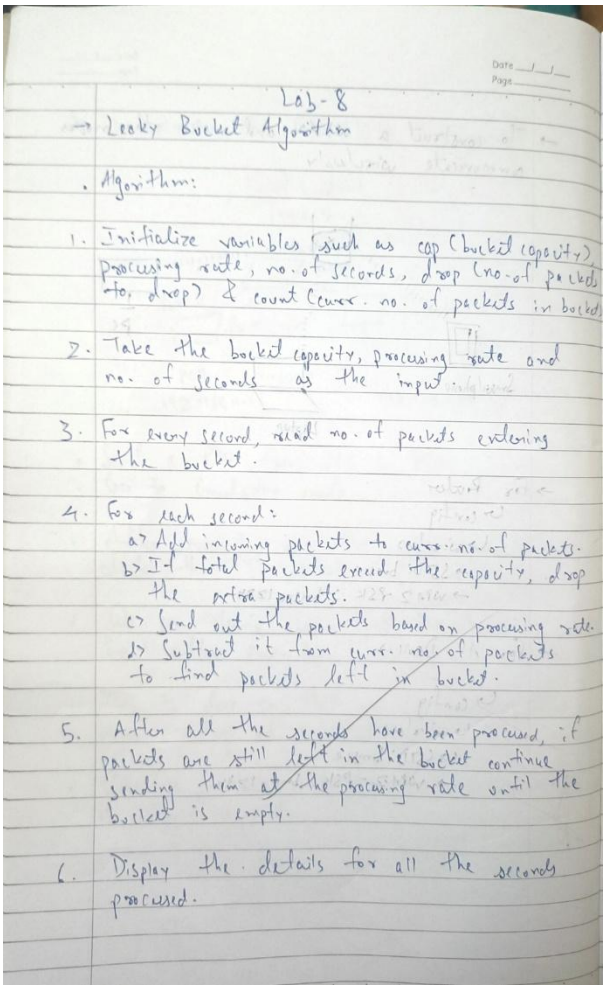


Configuration:



PART - B

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

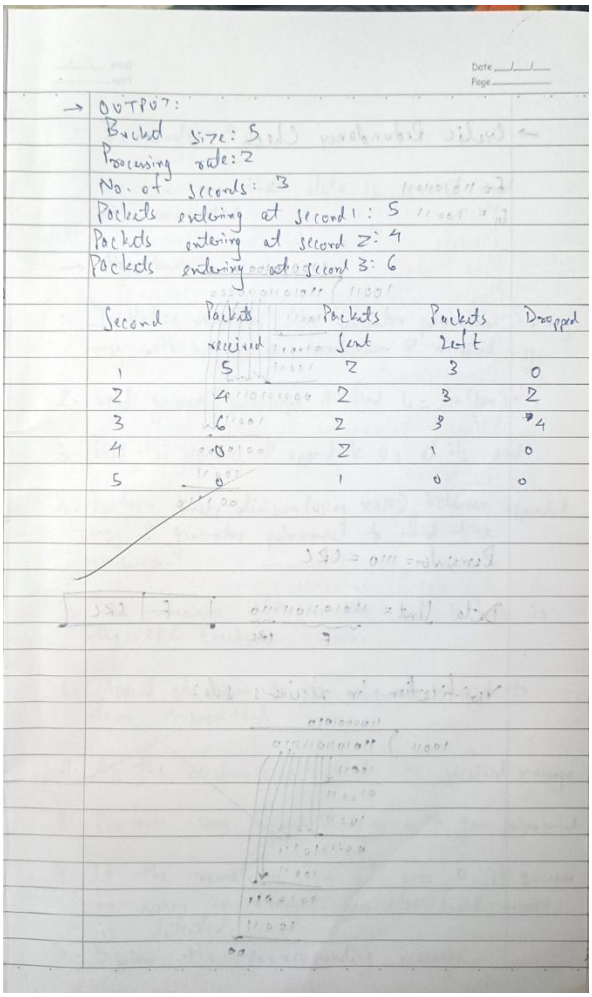


Lab-8

→ Leaky Bucket Algorithm

. Algorithm:

1. Initialize variables (such as cap (bucket capacity), processing rate, no. of seconds, drop (no. of packets to drop) & count (curr. no. of packets in bucket)
2. Take the bucket capacity, processing rate and no. of seconds as the input.
3. For every second, read no. of packets entering the bucket.
4. For each second:
   a) Add incoming packets to curr. no. of packets.
   b) If total packets exceed the capacity, drop the extra packets.
   c) Send out the packets based on processing rate.
   d) Subtract it from curr. no. of packets to find packets left in bucket.
5. After all the seconds have been processed, if packets are still left in the bucket continue sending them at the processing rate until the bucket is empty.
6. Display the details for all the seconds processed.



→ OUTPUT:

Bucket size: 5
Processing rate: 2
No. of seconds: 3
Packets entering at second 1: 5
Packets entering at second 2: 4
Packets entering at second 3: 6

Second	Packets received	Packets Sent	Packets Left	Dropped
1	5	2	3	0
2	4	2	3	2
3	6	2	3	4
4	0	2	1	0
5	0	1	0	0

Code:

```
#include <stdio.h>
int min(int x, int y)
{
   if (x < y)
      return x;
   else
      return y;
}
```

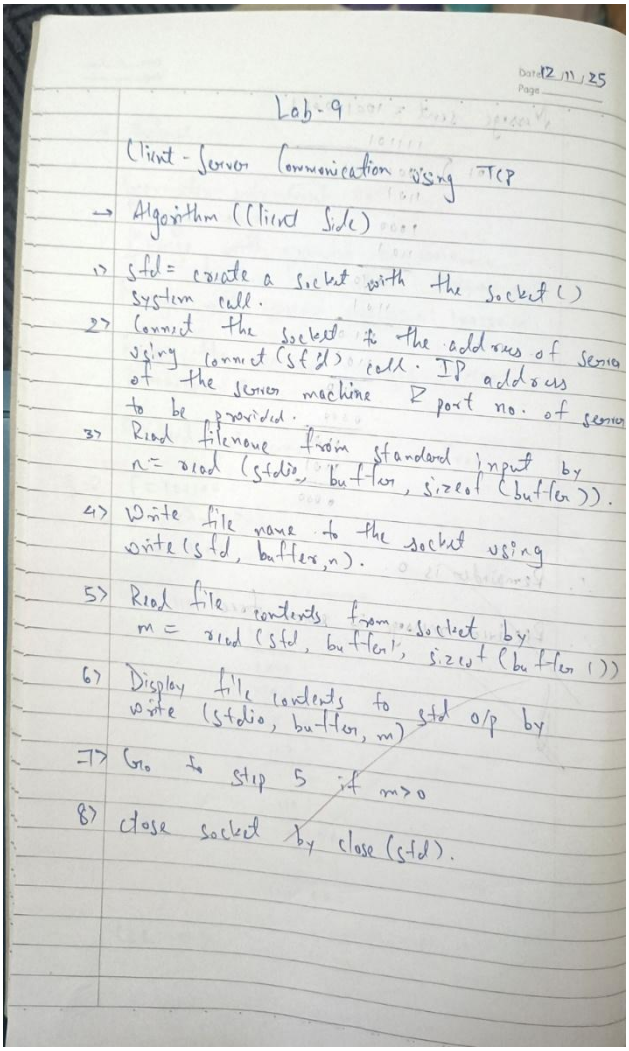
```
int main()
{
    int drop = 0, mini, nsec, cap, count = 0, i, inp[25], process;
    printf("Enter the bucket size: ");
    scanf("%d", &cap);
    printf("Enter the processing rate: ");
    scanf("%d", &process);
    printf("Enter the number of seconds you want to simulate: ");
    scanf("%d", &nsec);
    for (i = 0; i < nsec; i++)
    {
        printf("Enter the size of the packet entering at %d sec: ", 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("%6d | %15d |", i + 1, inp[i]);
        mini = min(count, process);
        printf(" %11d |", mini);
```

```
        count -= mini;
        printf(" %11d | %7d\n", count, drop);
        drop = 0;
    }
    while (count != 0)
    {
        if (count > cap)
        {
            drop = count - cap;
            count = cap;
        }
        printf("%6d | %15d |", ++i, 0);
        mini = min(count, process);
        printf(" %11d |", mini);
        count -= mini;
        printf(" %11d | %7d\n", count, drop);
    }
    return 0;
}
```

Output:

```
Enter the bucket size: 5
Enter the processing rate: 3
Enter the number of seconds you want to simulate: 3
Enter the size of the packet entering at 1 sec: 4
Enter the size of the packet entering at 2 sec: 6
Enter the size of the packet entering at 3 sec: 7

Second | Packet Received | Packet Sent | Packet Left | Dropped
-----------------------------------------------------------------
     1 |               4 |           3 |           1 |       0
     2 |               6 |           3 |           2 |       2
     3 |               7 |           3 |           2 |       4
     4 |               0 |           2 |           0 |       0
```

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.

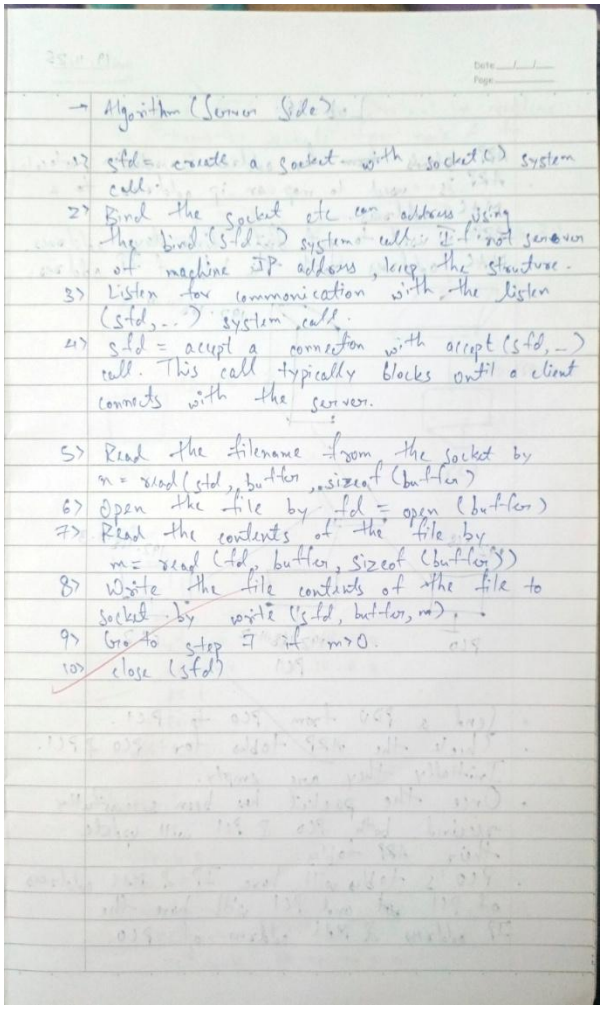


Lab-9

Client - Server communication using TCP

→ Algorithm (Client Side)

1) sfd = create a socket with the socket() system call.
2) Connect the socket to the address of server using connect(sfd) call. IP address of the server machine & port no. of server to be provided.
3) Read filename from standard input by n = read(stdio, buffer, sizeof(buffer)).
4) Write file name to the socket using write(sfd, buffer, n).
5) Read file contents from socket by m = read(sfd, buffer, sizeof(buffer))
6) Display file contents to std o/p by write(stdio, buffer, m)
7) Go to step 5 if m>0
8) close socket by close(sfd).



→ Algorithm (Server Side)

1) sfd = create a socket with socket() system call.
2) Bind the socket etc on address using the bind(sfd,..) system call. i.e. port no. of machine IP address, keep the structure.
3) Listen for communication with the listen(sfd,..) system call.
4) sfd = accept a connection with accept(sfd,..) call. This call typically blocks until a client connects with the server.
5) Read the filename from the socket by n = read(sfd, buffer, sizeof(buffer))
6) Open the file by fd = open(buffer)
7) Read the contents of the file by m = read(fd, buffer, sizeof(buffer))
8) Write the file contents of the file to socket by write(sfd, buffer, m).
9) Go to step 7 if m>0
10) close(sfd)

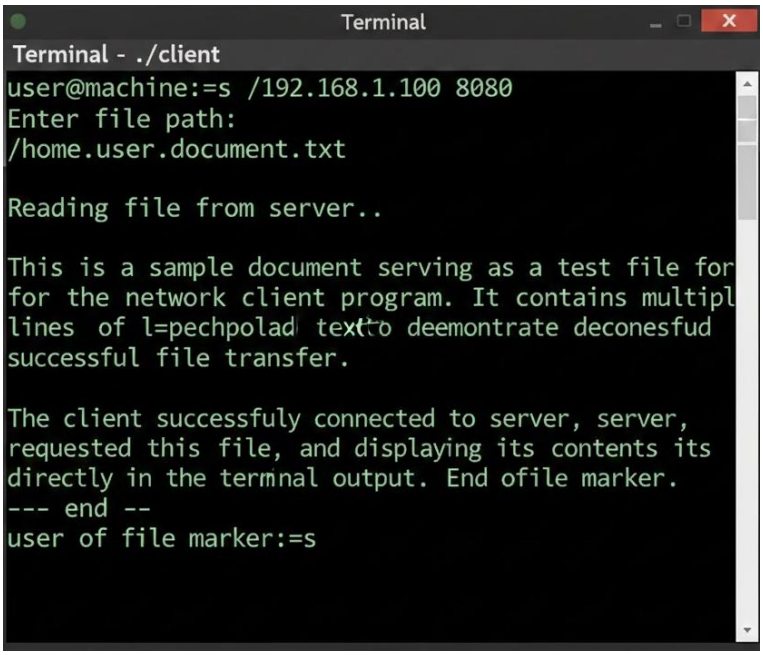
Code:

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

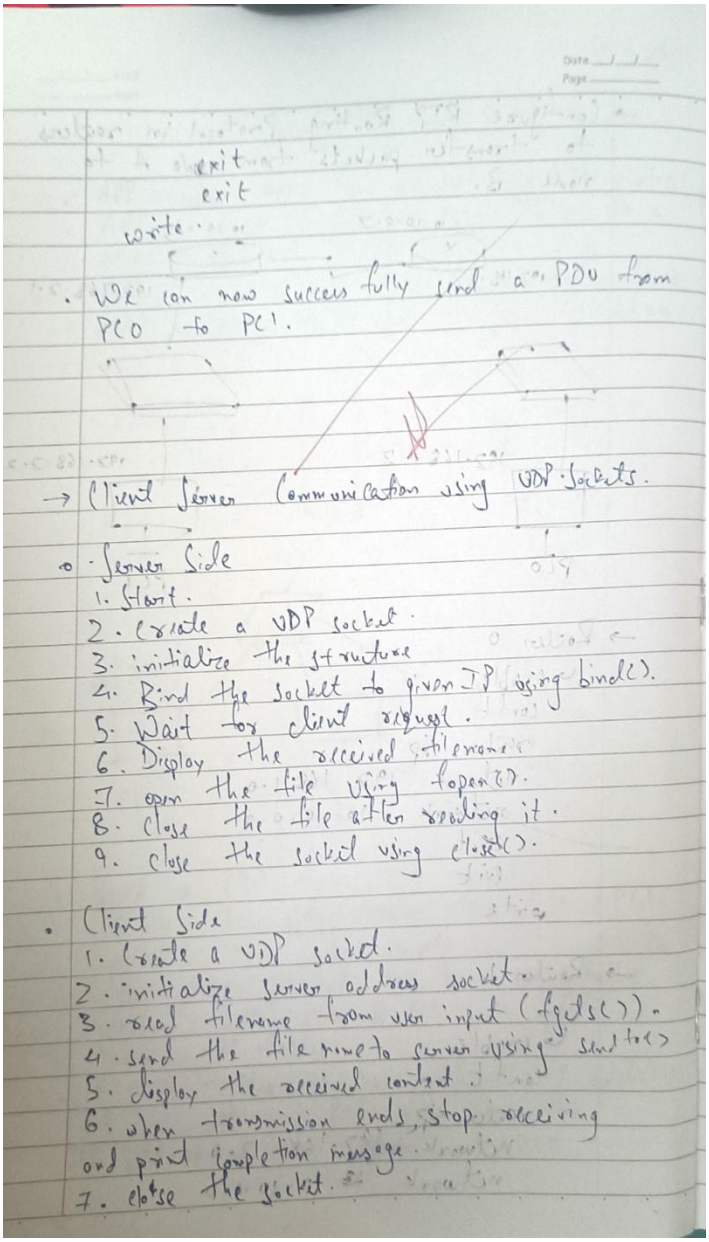
```
#include <arpa/inet.h>
#include <netdb.h>
#define BUF_SIZE 20000
int main(int argc, char *argv[])
{
   int sockfd, portno, n;
   char filename[256];
   char buffer[BUF_SIZE];
   struct sockaddr_in serv;
   if (argc < 3) {
      printf("Usage: %s <server-ip> <port>\n", argv[0]);
      exit(1);
   }
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd < 0) {
      perror("socket");
      exit(1);
   }
   memset(&serv, 0, sizeof(serv));
   portno = atoi(argv[2]);
   serv.sin_family = AF_INET;
   serv.sin_port = htons(portno);
   if (inet_pton(AF_INET, argv[1], &serv.sin_addr) <= 0) {
      perror("Invalid IP");
      exit(1);
   }
   if (connect(sockfd, (struct sockaddr *)&serv, sizeof(serv)) < 0) {
```

```
        perror("connect");
        exit(1);
    }
    printf("Enter file path: ");
    fgets(filename, sizeof(filename), stdin);
    filename[strcspn(filename, "\n")] = 0;
    if (write(sockfd, filename, strlen(filename) + 1) < 0) {
        perror("write");
        exit(1);
    }
    printf("Reading file from server...\n");
    while ((n = read(sockfd, buffer, sizeof(buffer) - 1)) > 0) {
        buffer[n] = '\0';
        fputs(buffer, stdout);
    }
    if (n < 0)
        perror("read");
    printf("\n--- end ---\n");
    close(sockfd);
    return 0;
}
```

Output:


Terminal
Terminal - ./client
user@machine:=s /192.168.1.100 8080
Enter file path:
/home.user.document.txt
Reading file from server..
This is a sample document serving as a test file for
for the network client program. It contains multipl
lines of l=pechpolad text to deemontrate deconesfud
successful file transfer.
The client successfuly connected to server, server,
requested this file, and displaying its contents its
directly in the terminal output. End ofile marker.
--- end --
user of file marker:=s

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:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>
```

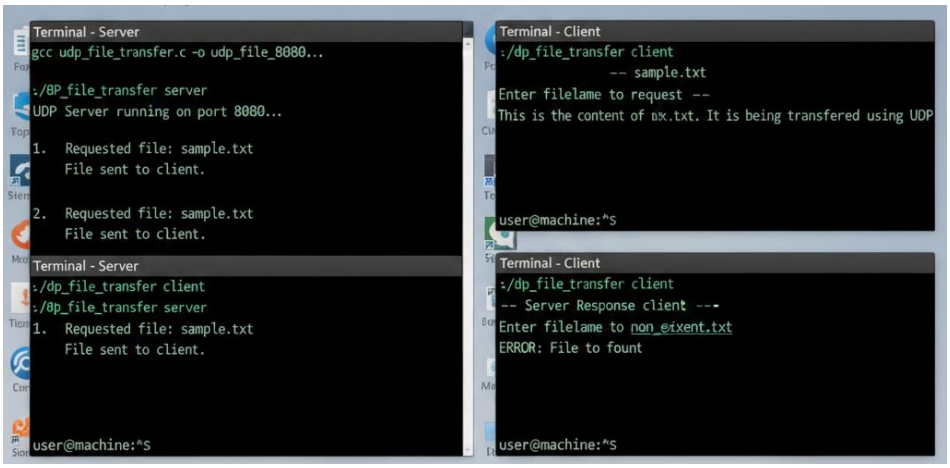
```
#include <arpa/inet.h>
#define PORT 8080
#define MAX 1024
void runServer() {
    int sockfd;
    char buffer[MAX];
    struct sockaddr_in serverAddr, clientAddr;
    socklen_t addr_size;
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        exit(1);
    }
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    if (bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr)) < 0) {
        perror("Bind failed");
        exit(1);
    }
    printf("UDP Server running on port %d...\n", PORT);
    while (1) {
        addr_size = sizeof(clientAddr);
        memset(buffer, 0, MAX);
        recvfrom(sockfd, buffer, MAX, 0,
                (struct sockaddr*)&clientAddr, &addr_size);
```

```
        printf("Requested file: %s\n", buffer);
        FILE *fp = fopen(buffer, "r");
        if (fp == NULL) {
            char *err = "ERROR: File not found";
            sendto(sockfd, err, strlen(err), 0,
                   (struct sockaddr*)&clientAddr, addr_size);
            continue;
        }
        char filedata[MAX];
        memset(filedata, 0, MAX);
        fread(filedata, 1, MAX, fp);
        fclose(fp);
        sendto(sockfd, filedata, strlen(filedata), 0,
               (struct sockaddr*)&clientAddr, addr_size);
        printf("File sent to client.\n");
    }
}
void runClient() {
    int sockfd;
    char filename[MAX], buffer[MAX];
    struct sockaddr_in serverAddr;
    socklen_t addr_size;
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        exit(1);
    }
```

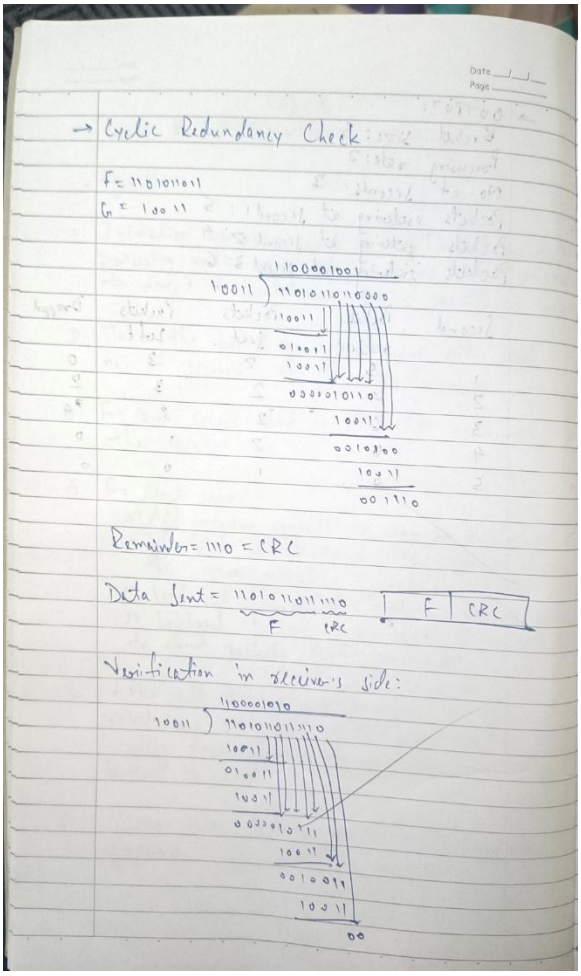
```
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    printf("Enter filename to request: ");
    scanf("%s", filename);

    sendto(sockfd, filename, strlen(filename), 0,
           (struct sockaddr*)&serverAddr, sizeof(serverAddr));
    addr_size = sizeof(serverAddr);
    memset(buffer, 0, MAX);

    recvfrom(sockfd, buffer, MAX, 0,
             (struct sockaddr*)&serverAddr, &addr_size);

    printf("\n--- Server Response ---\n");
    printf("%s\n", buffer);
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Usage: %s <server|client>\n", argv[0]);
        exit(1);
    }

    if (strcmp(argv[0], "./combined") != 0) {}
```

```
    if (strcmp(argv[1], "server") == 0)
        runServer();
    else if (strcmp(argv[1], "client") == 0)
        runClient();
    else
        printf("Invalid option. Use 'server' or 'client'.\n");

    return 0;
}
```

Output:



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



→ Cyclic Redundancy Check:

F = 11010110011

G = 10011

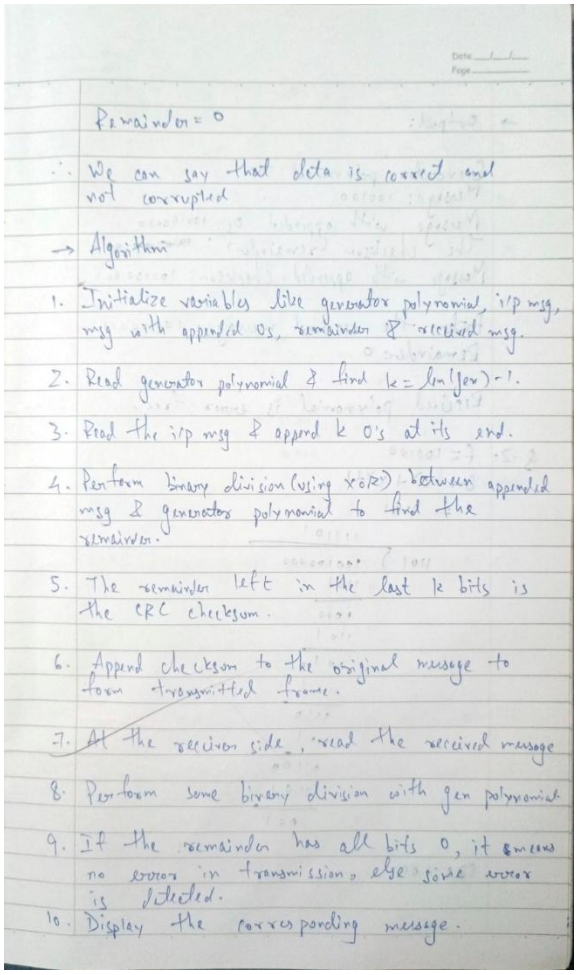
1100001001
10011) 11010110110000
10011
01001
10011
00001110
10011
00101100
10011
00111110

Remainder = 1110 = CRC

Data Sent = 11010110111110 (F | CRC)

Verification in receiver's side:

11000010 10
10011) 11010110111110
10011
01001 1
10011
00001 011
10011
00101 111
10011
00

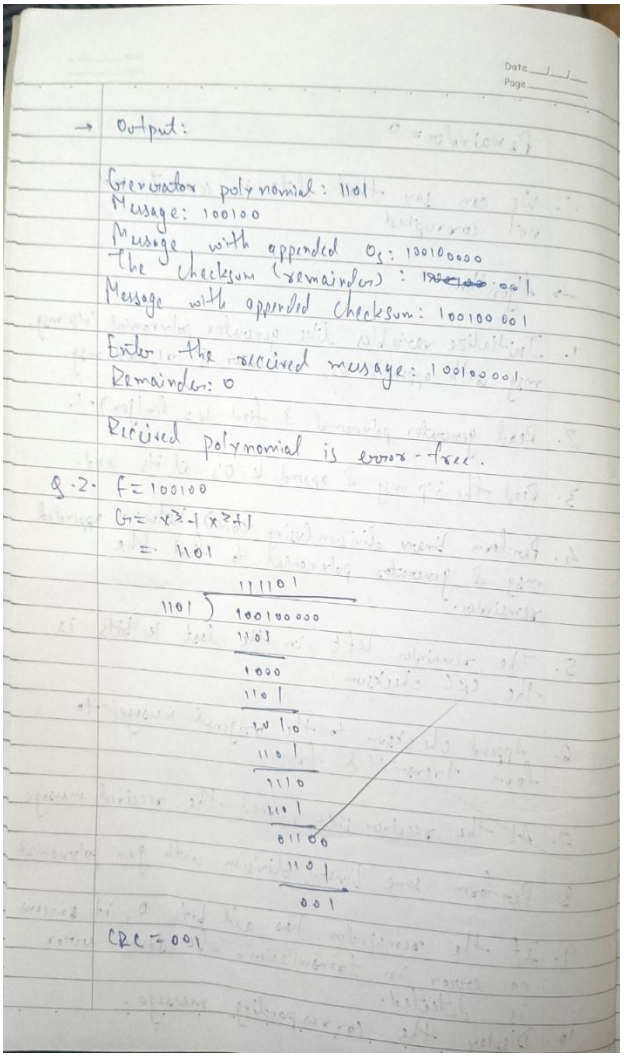


Remainder = 0

∴ We can say that data is correct and not corrupted

→ Algorithm

1. Initialize variables like generator polynomial, i/p msg, msg with appended 0s, remainder & received msg.
2. Read generator polynomial & find k = len(gen) - 1.
3. Read the i/p msg & append k 0's at its end.
4. Perform binary division (using XOR) between appended msg & generator polynomial to find the remainder.
5. The remainder left in the last k bits is the CRC checksum.
6. Append checksum to the original message to form transmitted frame.
7. At the receiver side, read the received message
8. Perform same binary division with gen polynomial
9. If the remainder has all bits 0, it means no error in transmission, else some error is detected.
10. Display the corresponding message.



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: ");
    fgets(gen, sizeof(gen), stdin);
    gen[strcspn(gen, "\n")] = '\0'; // remove newline
```

```
printf("Generator polynomial (CRC-CCITT): %s\n", gen);
genlen = strlen(gen);
k = genlen - 1;
printf("Enter the message: ");
n = 0;
while ((c = getchar()) != '\n' && c != EOF)
{
   msj[n++] = c;
}
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("\nThe checksum (remainder) is:\n");
puts(rem);
printf("\nThe message 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("\nEnter the received message: ");
while ((c = getchar()) != '\n' && c != EOF)
{
   s[n++] = c;
}
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 - k + i];
rem[k] = '\0';

flag = 0;
for (i = 0; i < k; i++)
{
    if (rem[i] == '1')
    {
        flag = 1;
        break;
    }
}
if (flag == 0)
    printf("\nReceived polynomial is ERROR-FREE ☑\n");
```

```
    else
        printf("\nReceived polynomial contains ERROR ❌\n");
    return 0;
}
```

Output: