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:

Dhruva S Rao

(1BM23CS092)

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 Dhruva S Rao(1BM23CS092) 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

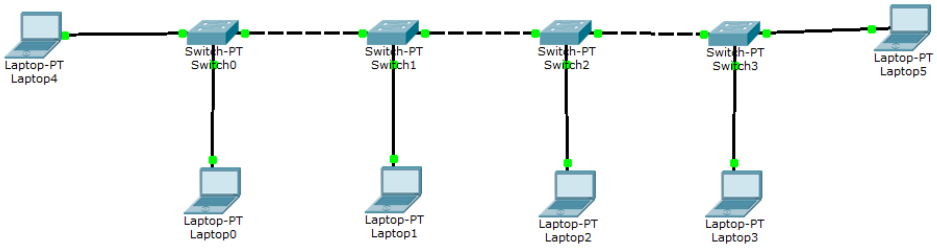
Table of Contents

PART - A

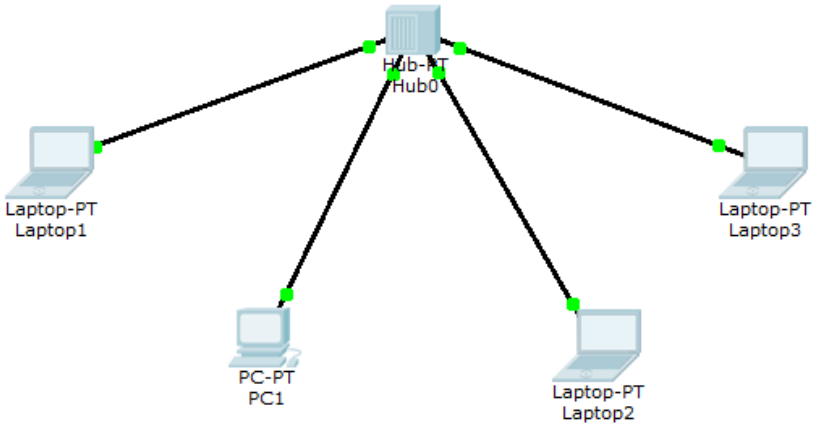
Program 1: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Network diagram:

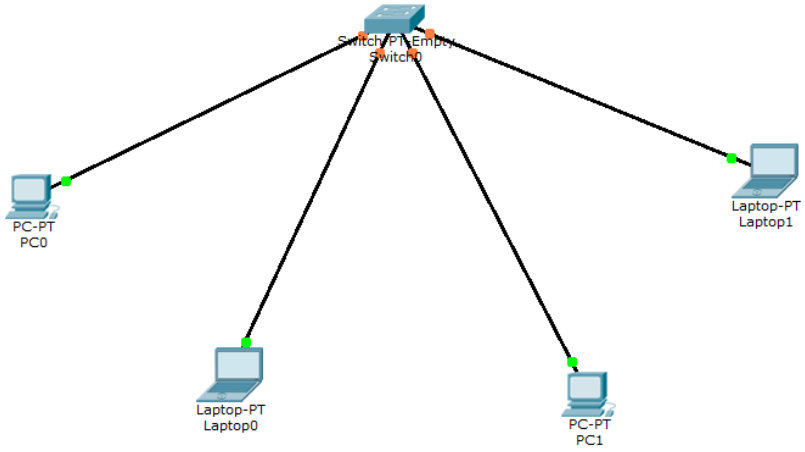
Bus



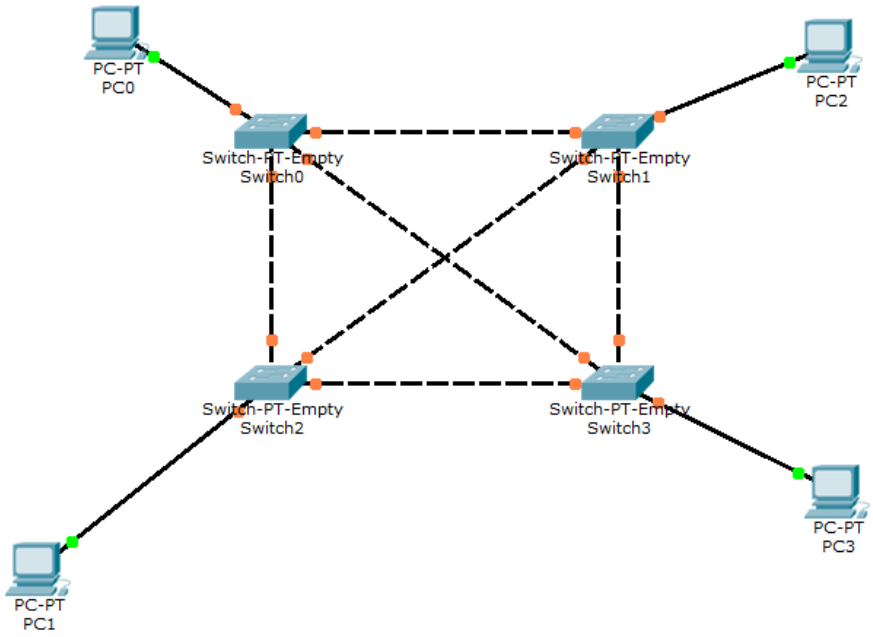
Hub



Switch

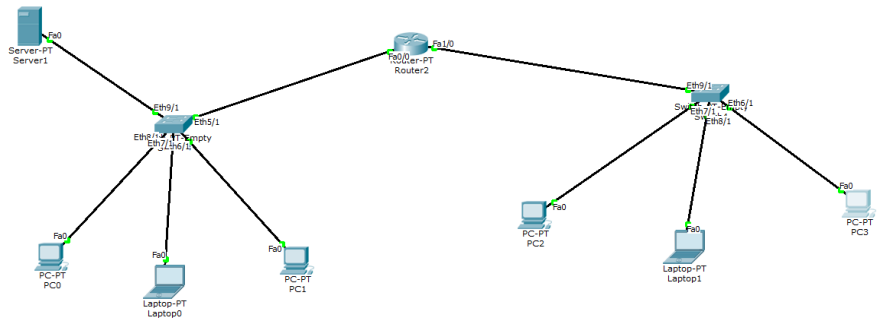

Switch-PT-Empty
Switch0
PC-PT
PC0
Laptop-PT
Laptop1
Laptop-PT
Laptop0
PC-PT
PC1

Mesh

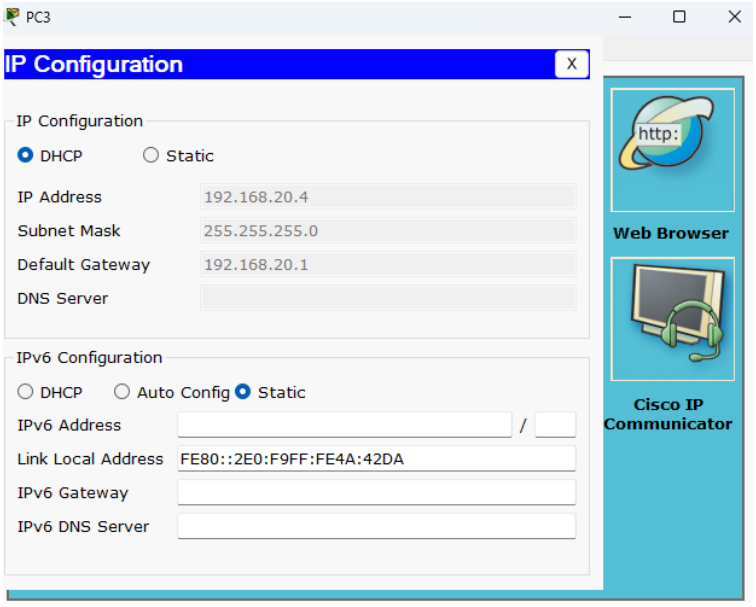

PC-PT
PC0
PC-PT
PC2
Switch-PT-Empty
Switch0
Switch-PT-Empty
Switch1
Switch-PT-Empty
Switch2
Switch-PT-Empty
Switch3
PC-PT
PC3
PC-PT
PC1

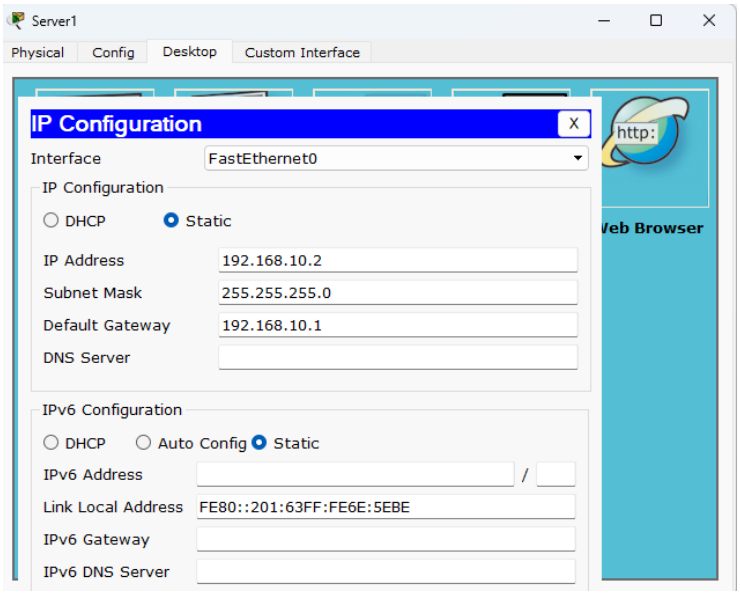
Program 2: Configure DHCP within a LAN and outside LAN.

Network diagram:



Configuration:

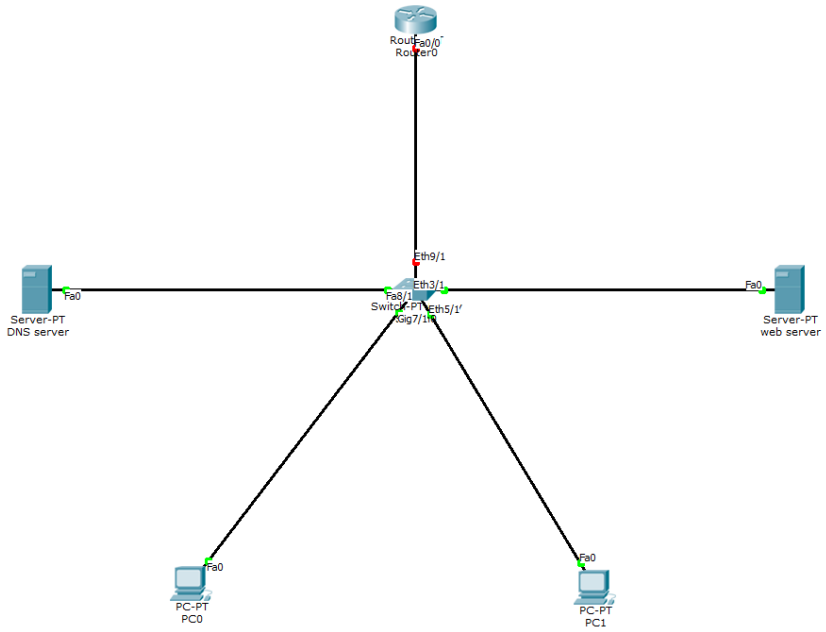




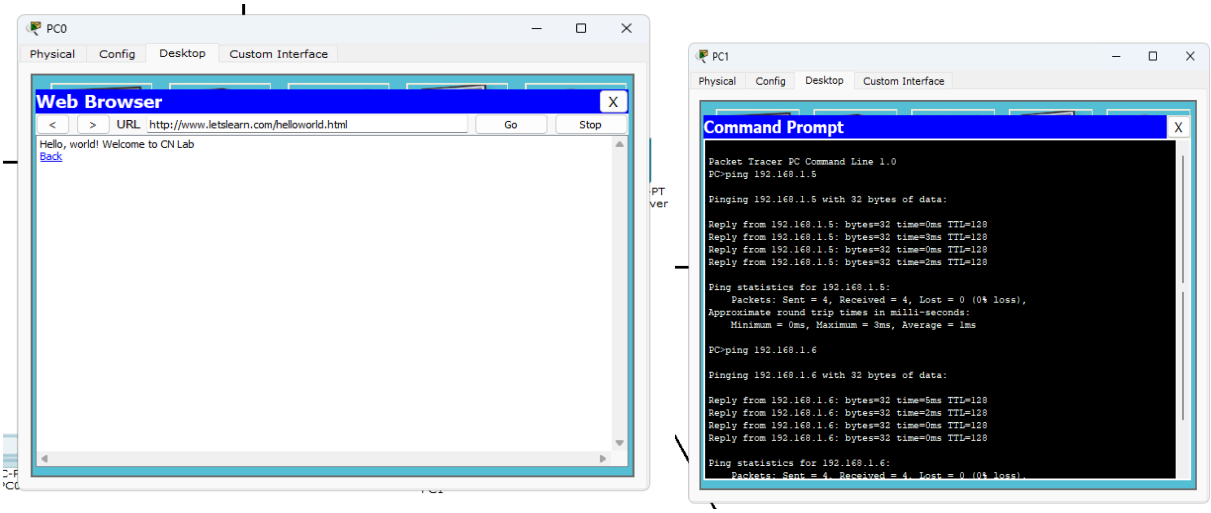
```
PT 1001 (PTSC2005) processor (revision 0x200) with 60416K/5120K bytes of memory
.
Processor board ID PT0123 (0123)
PT2005 processor: part number 0, mask 01
Bridging software.
X.25 software, Version 3.0.0.
4 FastEthernet/IEEE 802.3 interface(s)
2 Low-speed serial(sync/async) network interface(s)
32K bytes of non-volatile configuration memory.
63488K bytes of ATA CompactFlash (Read/Write)


         --- System Configuration Dialog ---

Continue with configuration dialog? [yes/no]: no


Press RETURN to get started!



Router>enable
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int Fa0/0
Router(config-if)#ip address 192.168.10.1 255.255.255.0
Router(config-if)#ip helper-address 192.168.10.2
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state t
o up
do write memory
Building configuration...
[OK]
Router(config-if)#exit
Router(config)#int Fa1/0
Router(config-if)#ip address 192.168.20.1 255.255.255.0
Router(config-if)#ip helper-address 192.168.10.2
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet1/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/0, changed state t
o up
do write memory
Building configuration...
[OK]
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
write memory
Building configuration...
[OK]
Router#%IP-4-DUPADDR: Duplicate address 192.168.10.1 on FastEthernet0/0, sourced
 by 000A.4166.1664
```

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

Network diagram:

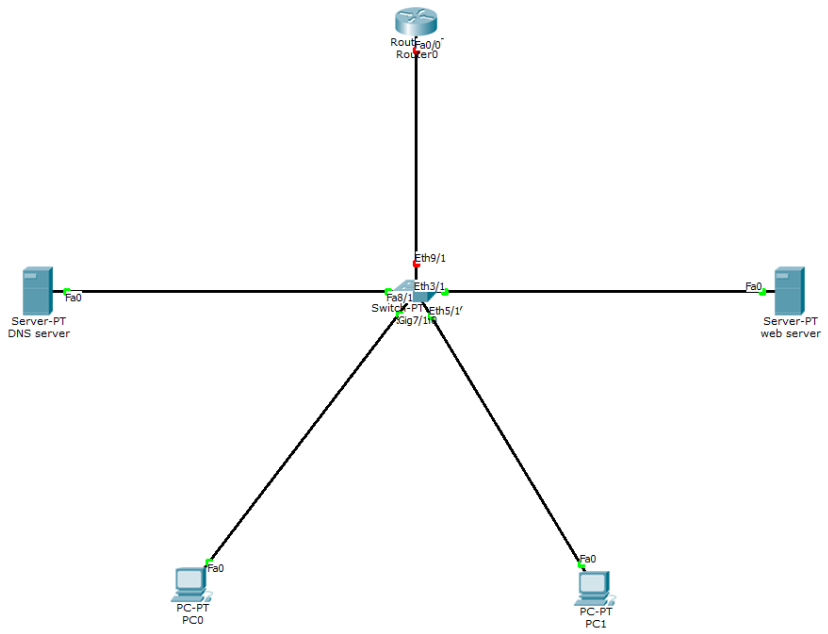


Configuration:



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

Network diagram:



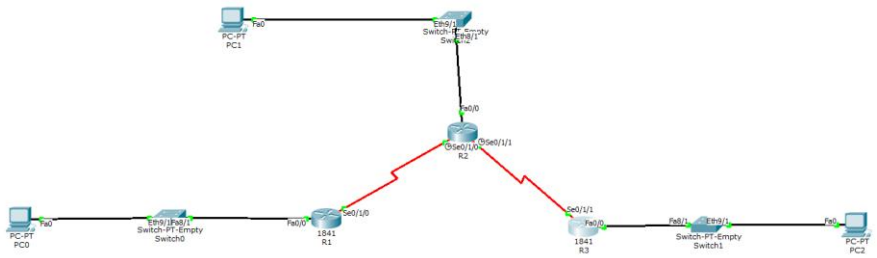
Configuration:

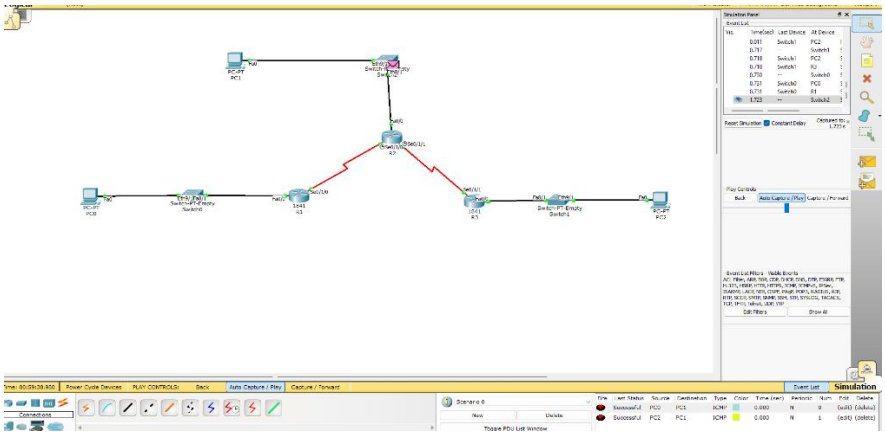
```
PC>ping 192.168.1.101

Pinging 192.168.1.101 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.101:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

PC>
```

```
PC>ping 192.168.1.6

Pinging 192.168.1.6 with 32 bytes of data:

Reply from 192.168.1.6: bytes=32 time=5ms TTL=128
Reply from 192.168.1.6: bytes=32 time=0ms TTL=128
Reply from 192.168.1.6: bytes=32 time=0ms TTL=128
Reply from 192.168.1.6: bytes=32 time=0ms TTL=128

Ping statistics for 192.168.1.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 5ms, Average = 1ms
```

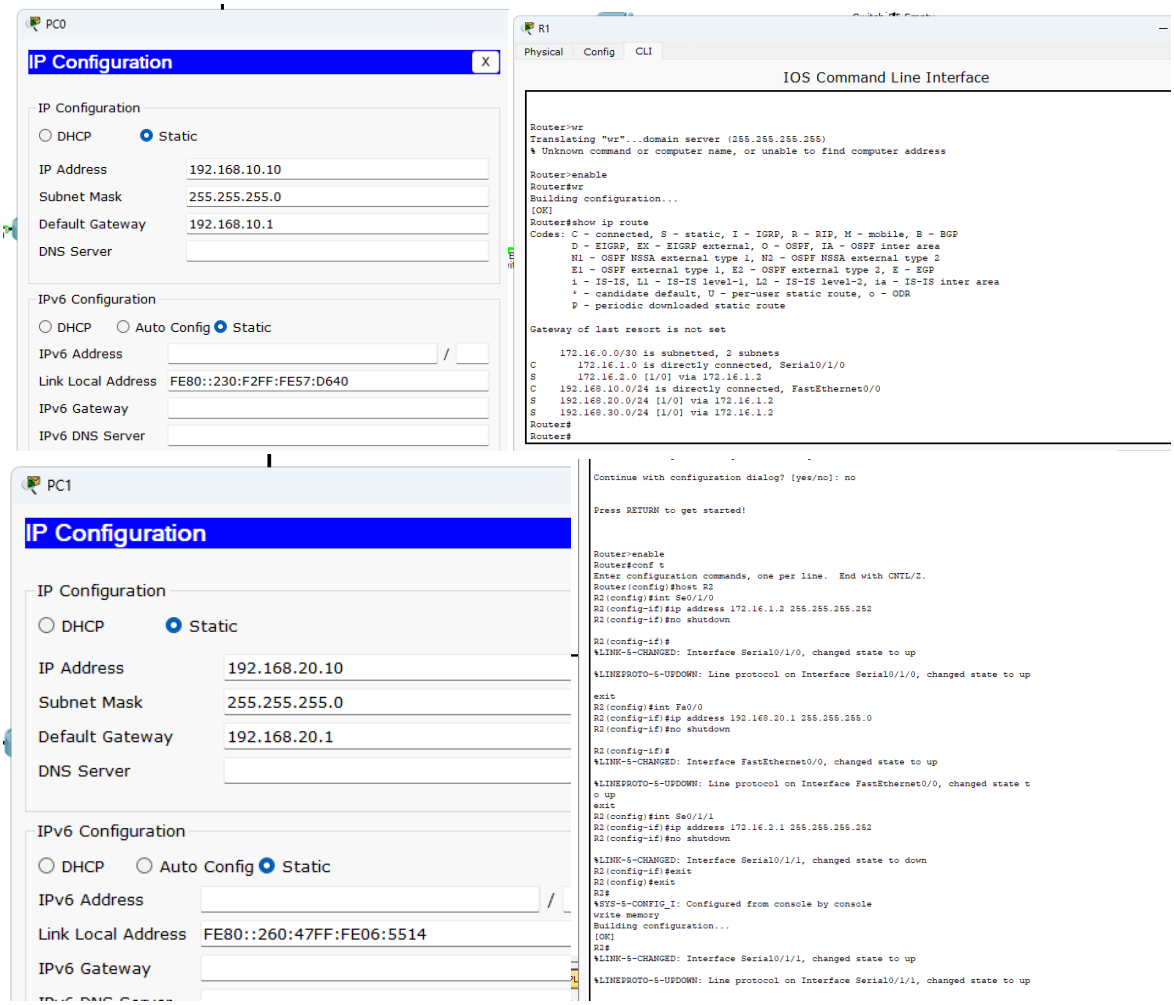
Program 5: Configure default route, static route to the Router.

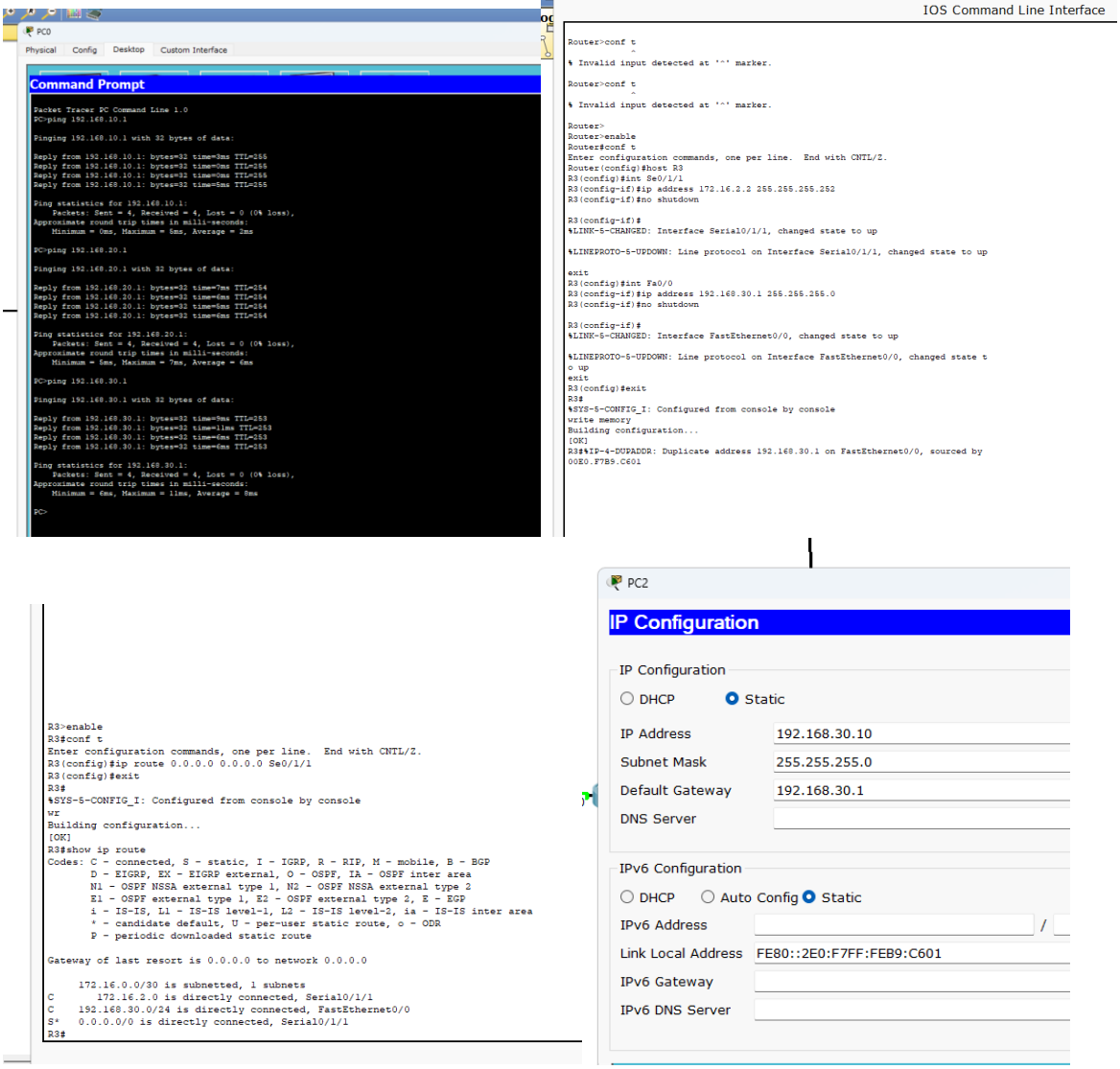
Network diagram:





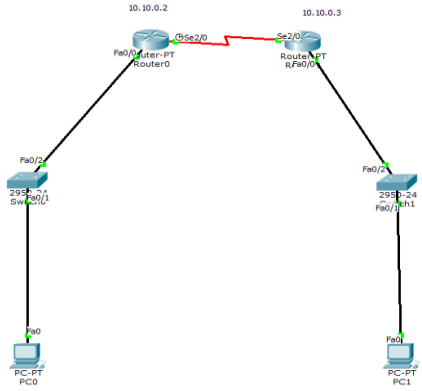
Configuration:



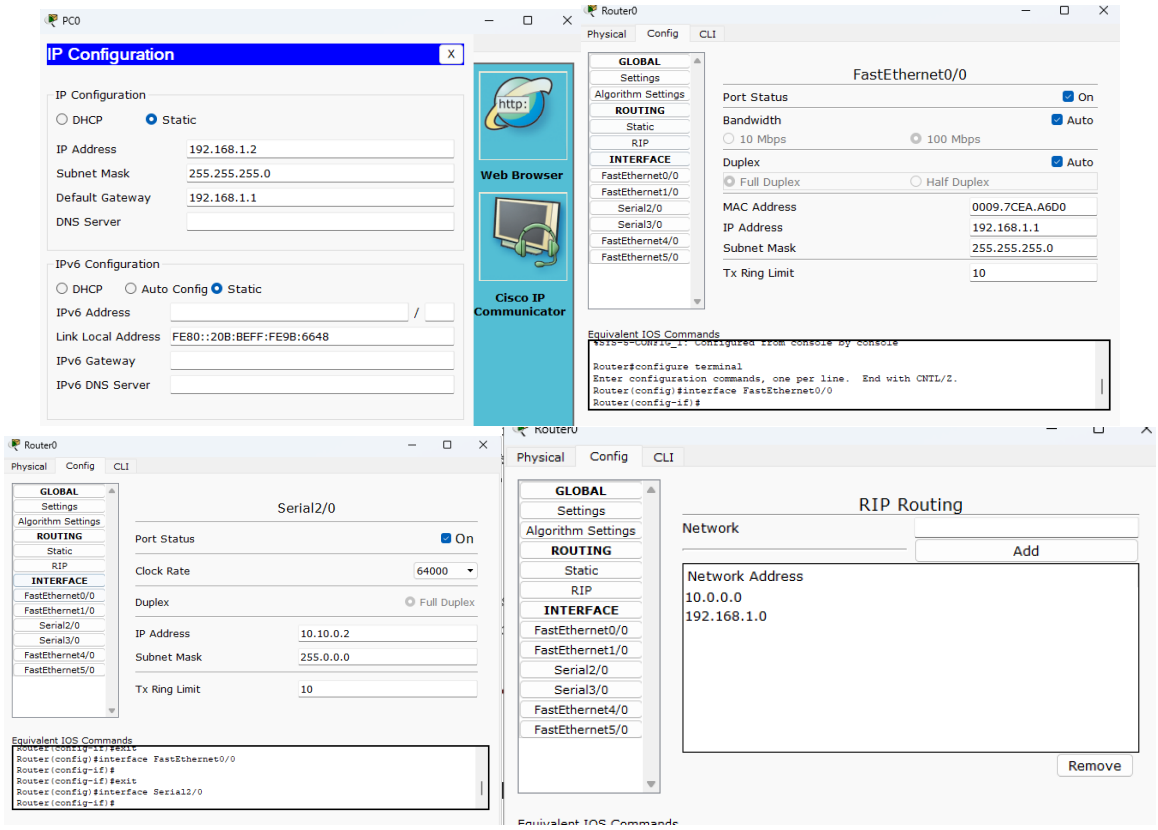

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 192.168.10.1
Pinging 192.168.10.1 with 32 bytes of data:
Reply from 192.168.10.1: bytes=32 time=3ms TTL=255
Reply from 192.168.10.1: bytes=32 time=0ms TTL=255
Reply from 192.168.10.1: bytes=32 time=0ms TTL=255
Reply from 192.168.10.1: bytes=32 time=5ms TTL=255
Ping statistics for 192.168.10.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 5ms, Average = 2ms
PC>ping 192.168.20.1
Pinging 192.168.20.1 with 32 bytes of data:
Reply from 192.168.20.1: bytes=32 time=7ms TTL=254
Reply from 192.168.20.1: bytes=32 time=6ms TTL=254
Reply from 192.168.20.1: bytes=32 time=6ms TTL=254
Reply from 192.168.20.1: bytes=32 time=6ms TTL=254
Ping statistics for 192.168.20.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 6ms, Maximum = 7ms, Average = 6ms
PC>ping 192.168.30.1
Pinging 192.168.30.1 with 32 bytes of data:
Reply from 192.168.30.1: bytes=32 time=9ms TTL=253
Reply from 192.168.30.1: bytes=32 time=11ms TTL=253
Reply from 192.168.30.1: bytes=32 time=6ms TTL=253
Reply from 192.168.30.1: bytes=32 time=6ms TTL=253
Ping statistics for 192.168.30.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 6ms, Maximum = 11ms, Average = 8ms
PC>
R3
Physical
Config
CLI
IOS Command Line Interface
Router>conf t
% Invalid input detected at '^' marker.
Router>conf t
% Invalid input detected at '^' marker.
Router>
Router>enable
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#host R3
R3(config)#int Se0/1/1
R3(config-if)#ip address 172.16.2.2 255.255.255.252
R3(config-if)#no shutdown
R3(config-if)#
%LINK-5-CHANGED: Interface Serial0/1/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1/1, changed state to up
exit
R3(config)#int Fa0/0
R3(config-if)#ip address 192.168.30.1 255.255.255.0
R3(config-if)#no shutdown
R3(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state t
o up
exit
R3(config)#exit
R3#
%SYS-5-CONFIG_I: Configured from console by console
write memory
Building configuration...
[OK]
R3#%IP-4-DUPADDR: Duplicate address 192.168.30.1 on FastEthernet0/0, sourced by
00E0.F7B9.C601
R3>enable
R3#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R3(config)#ip route 0.0.0.0 0.0.0.0 Se0/1/1
R3(config)#exit
R3#
%SYS-5-CONFIG_I: Configured from console by console
wr
Building configuration...
[OK]
R3#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route
Gateway of last resort is 0.0.0.0 to network 0.0.0.0
172.16.0.0/30 is subnetted, 1 subnets
C 172.16.2.0 is directly connected, Serial0/1/1
C 192.168.30.0/24 is directly connected, FastEthernet0/0
S* 0.0.0.0/0 is directly connected, Serial0/1/1
R3#
PC2
IP Configuration
IP Configuration
DHCP
Static
IP Address
192.168.30.10
Subnet Mask
255.255.255.0
Default Gateway
192.168.30.1
DNS Server
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address
FE80::2E0:F7FF:FEB9:C601
IPv6 Gateway
IPv6 DNS Server

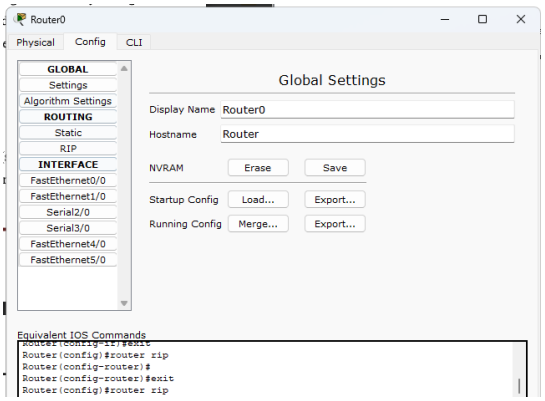
Program 6: Configure RIP routing Protocol in Routers.

Network diagram:



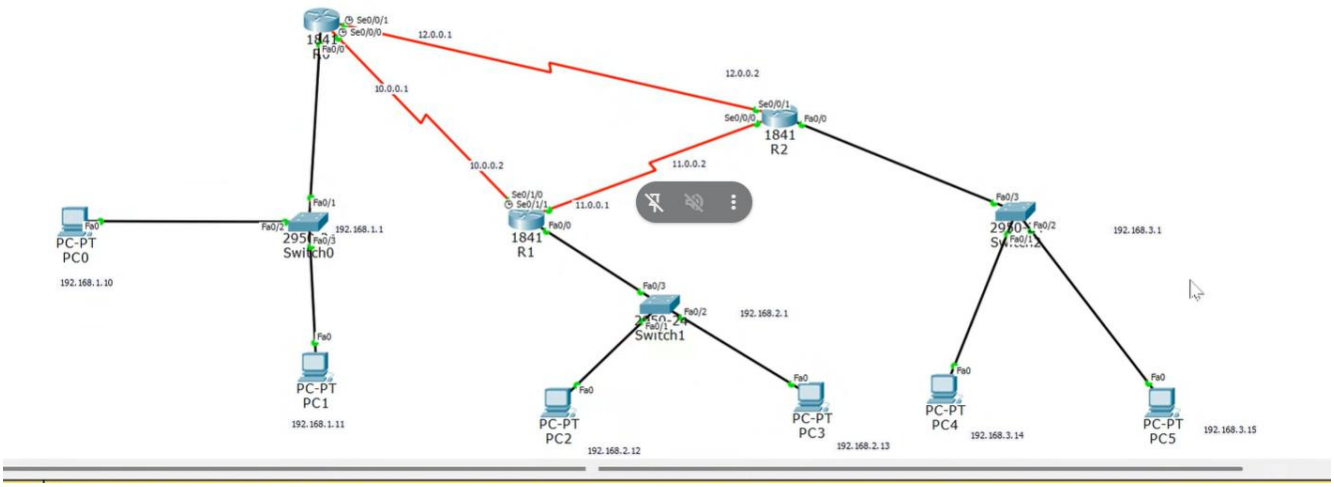
Configuration:





Program 7: Configure OSPF routing protocol.

Network diagram:



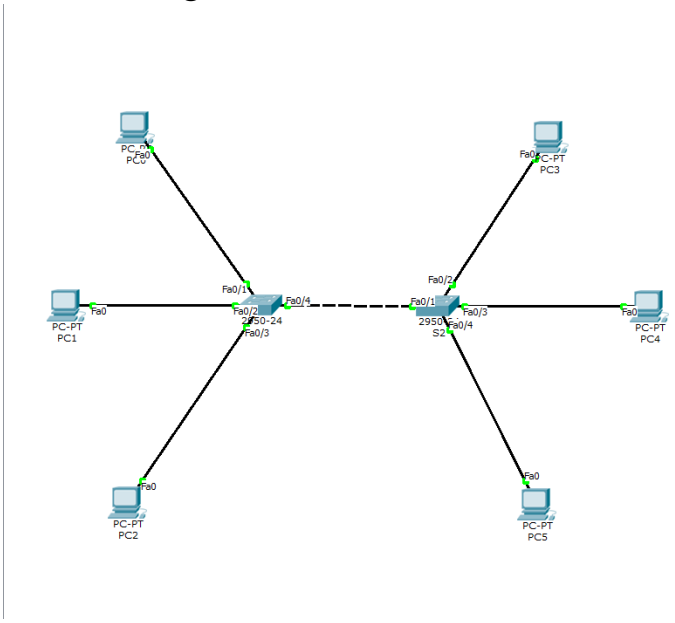
Configuration:

```
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
enable
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#router ospf 1
Router(config-router)#network 192.168.1.0 0.0.0.255 area0
                                                      ^
% Invalid input detected at '^' marker.

Router(config-router)#network 192.168.1.1 0.0.0.255 area0
                                                      ^
% Invalid input detected at '^' marker.

Router(config-router)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
enable
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#router ospfl
                          ^
% Invalid input detected at '^' marker.

Router(config)#router ospf 1
Router(config-router)#
```

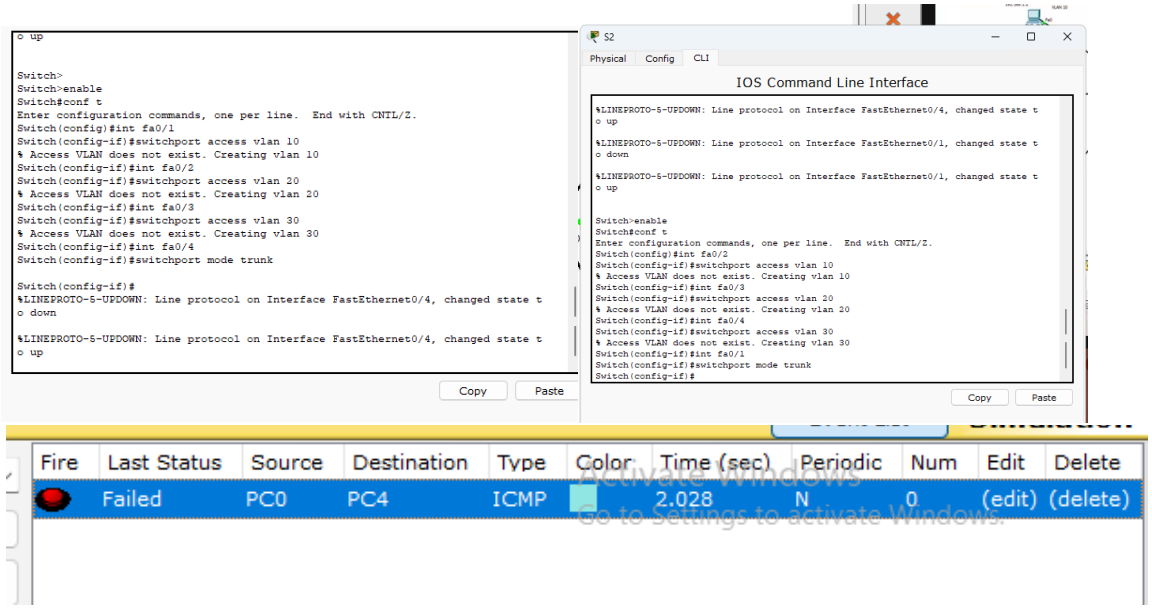
```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up

Router(config-if)#exit
Router(config)#interface Serial3/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial3/0
Router(config-if)#ip address 11.0.0.1 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface Serial3/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/0, changed state to up
exit
Router(config)#enable
% Incomplete command.
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
enable
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#router ospf 1
Router(config-router)#network 192.168.2.0 0.0.0.255 area 0
Router(config-router)#network 10.0.0.0 0.255.255.255 area 0
Router(config-router)#network 11.0.0.0 0.255.255.255 area 0
                                                      ^
% Invalid input detected at '^' marker.

Router(config-router)#network 11.0.0.0 0.255.255.255 area 0
Router(config-router)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
wr
Building configuration...
[OK]
Router#
00:18:31: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.3.1 on Serial3/0 from LOADING t
o FULL, Loading Done
```

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

Network diagram:

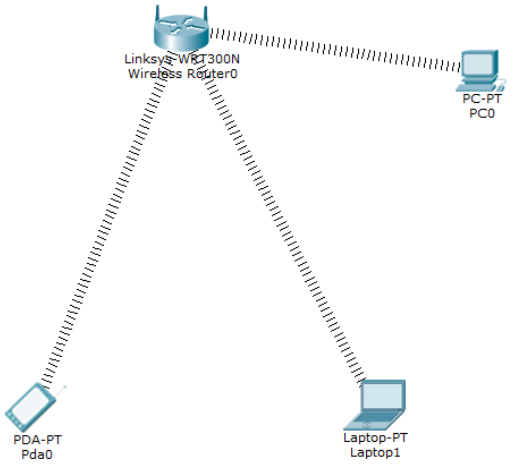


Configuration:

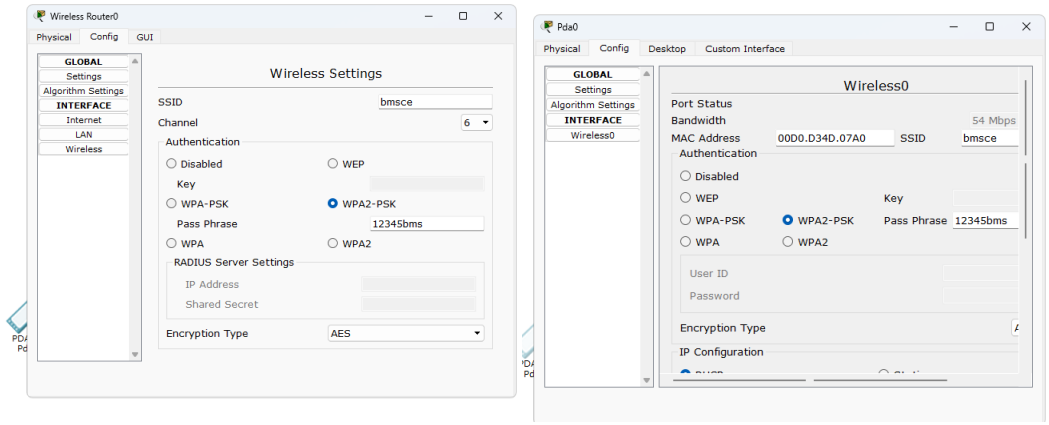


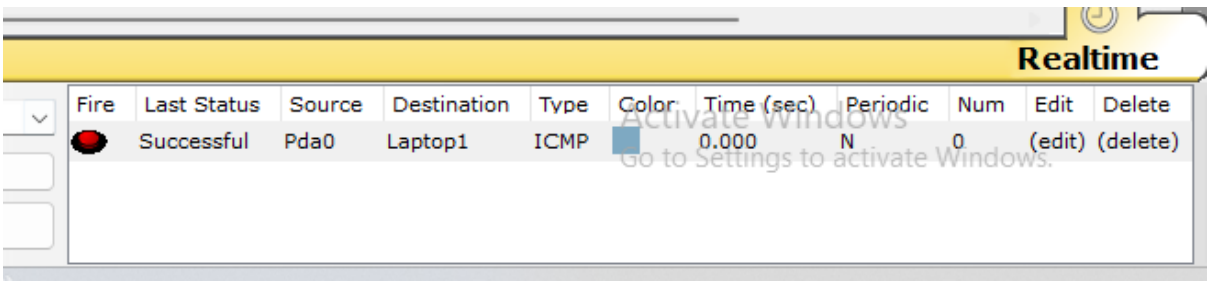
Program 9: To construct a WLAN and make the nodes communicate wirelessly.

Network diagram:



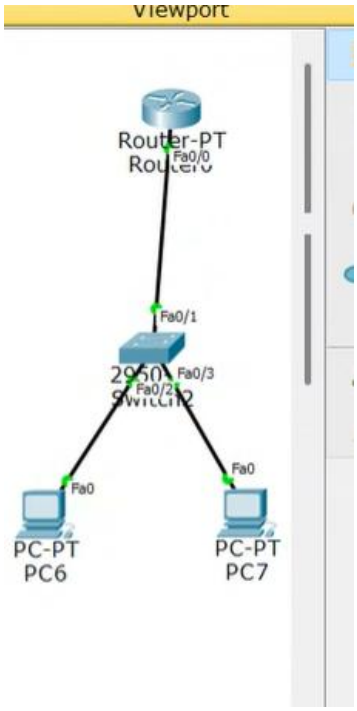
Configuration:



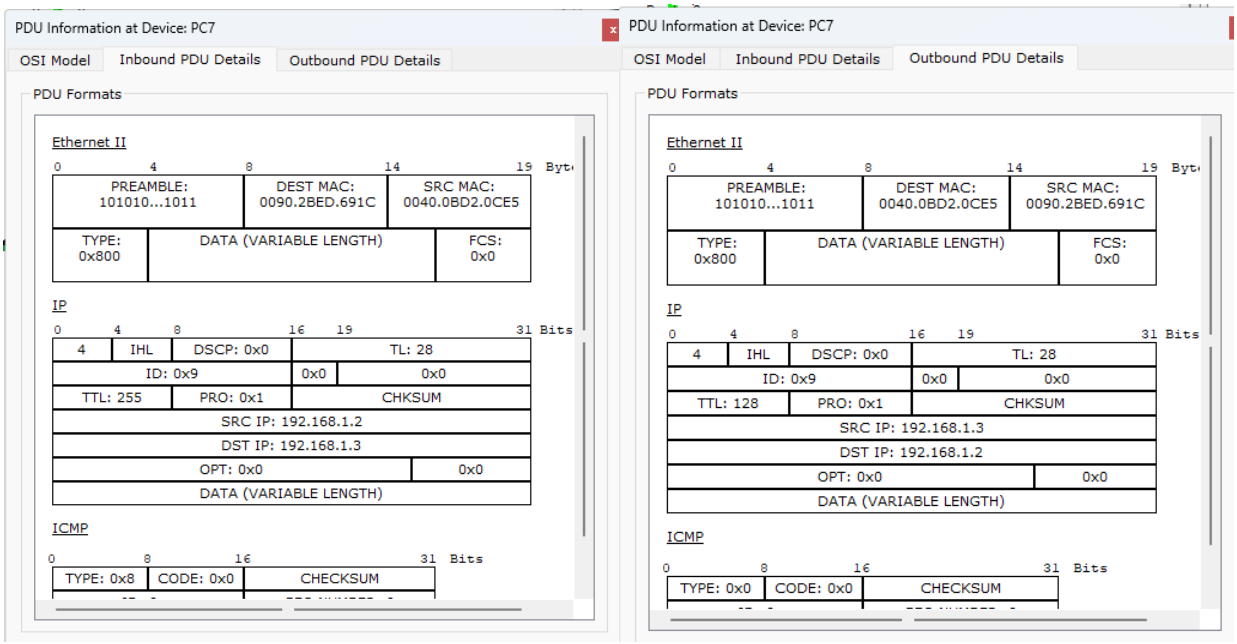


Program 10: Demonstrate the TTL/ Life of a Packet.

Network diagram:

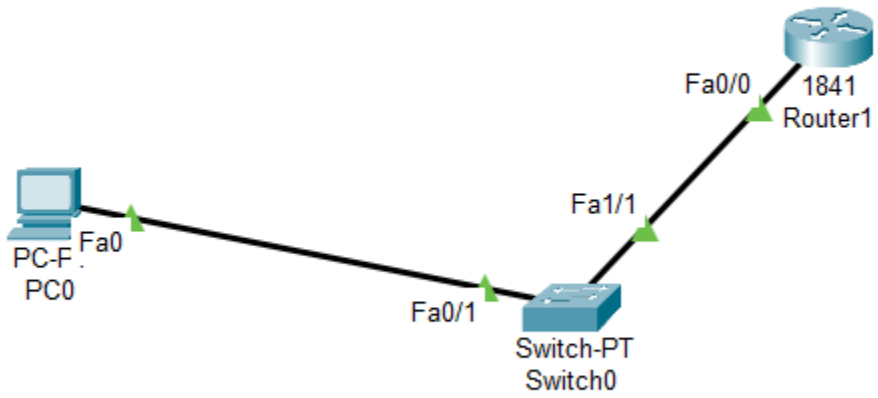


Configuration:



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

Network diagram:

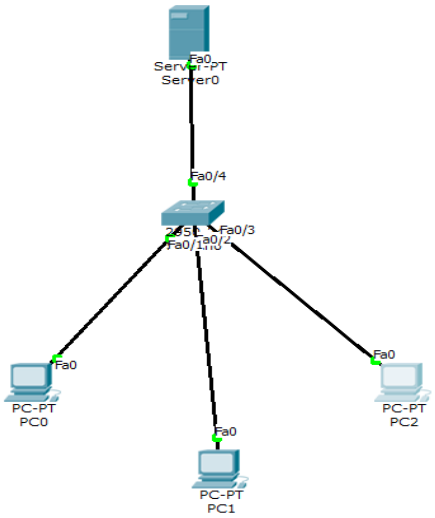


Configuration:

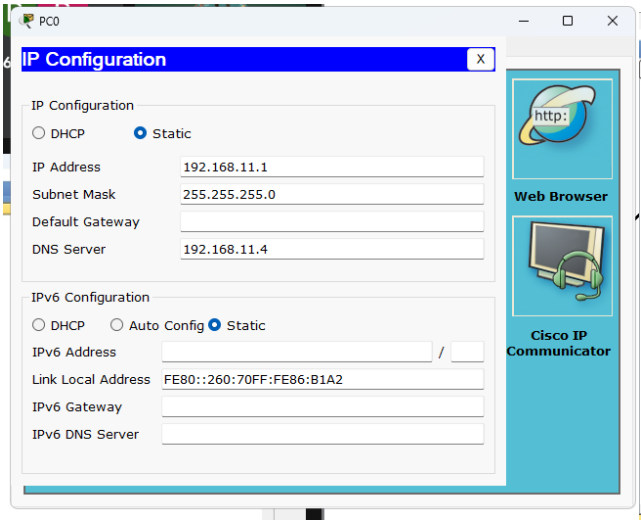
```
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#enable secret rp
Router(config)#int Fa 0/0
Router(config-if)#int address 192.168.1.1 255.255.255.0
                   ^
% Invalid input detected at '^' marker.

Router(config-if)#ip address 192.168.1.1 255.255.255.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up

Router(config-if)#
Router(config-if)#ine vty
                  ^
% Invalid input detected at '^' marker.

Router(config-if)#line vty
                  ^
% Invalid input detected at '^' marker.

Router(config-if)#line vty
                  ^
% Invalid input detected at '^' marker.

Router(config-if)#exit
Router(config)#line vty
% Incomplete command.
Router(config)#login
% Incomplete command.
Router(config)#enable secret rp
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
enable
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#enable secret rp
```

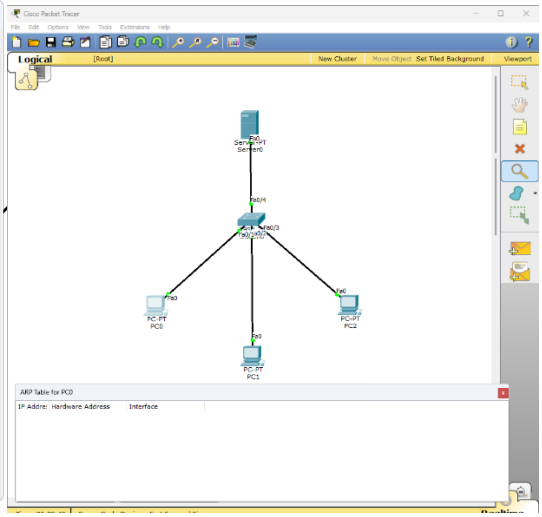
Program 12: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).

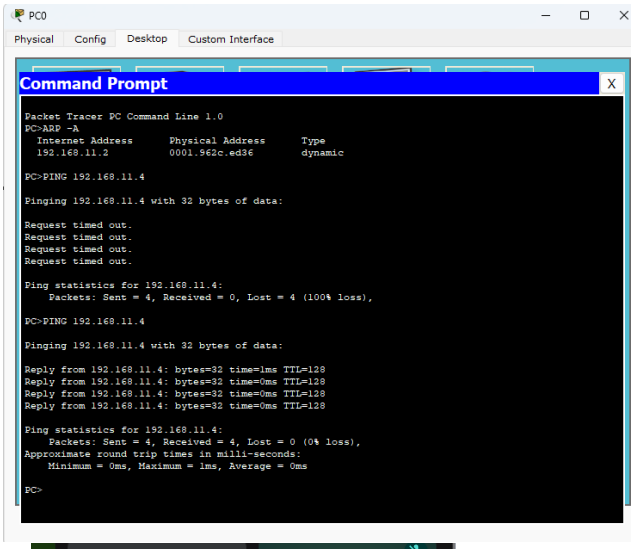
Network diagram:

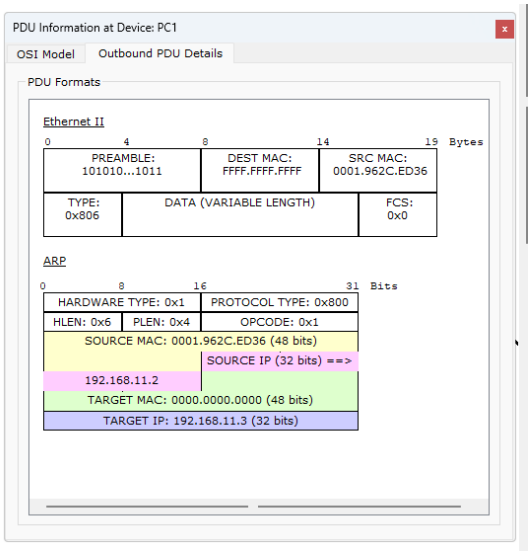


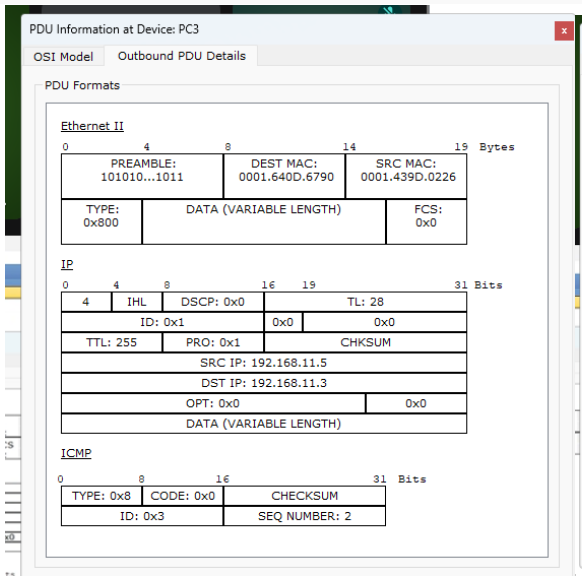
Configuration:

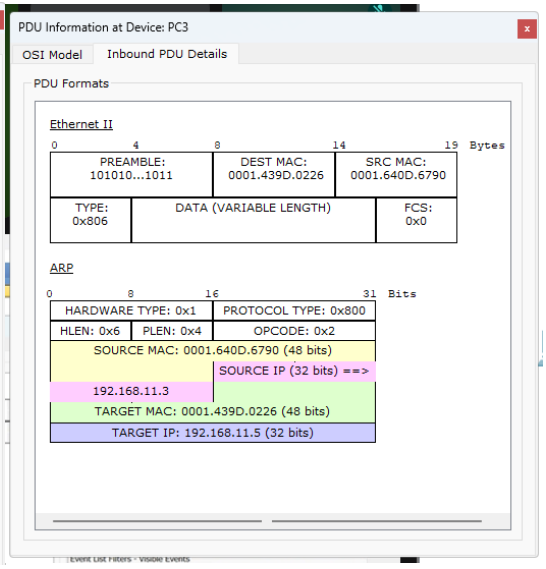





PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ARP -A
Internet Address Physical Address Type
192.168.11.2 0001.962c.ed36 dynamic
PC>PING 192.168.11.4
Pinging 192.168.11.4 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 192.168.11.4:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>PING 192.168.11.4
Pinging 192.168.11.4 with 32 bytes of data:
Reply from 192.168.11.4: bytes=32 time=1ms TTL=128
Reply from 192.168.11.4: bytes=32 time=0ms TTL=128
Reply from 192.168.11.4: bytes=32 time=0ms TTL=128
Reply from 192.168.11.4: bytes=32 time=0ms TTL=128
Ping statistics for 192.168.11.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
PC>


PDU Information at Device: PC1
OSI Model
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: FFFF.FFFF.FFFF
SRC MAC: 0001.962C.ED36
TYPE: 0x806
DATA (VARIABLE LENGTH)
FCS: 0x0
ARP
0 8 16 31 Bits
HARDWARE TYPE: 0x1
PROTOCOL TYPE: 0x800
HLEN: 0x6
PLEN: 0x4
OPCODE: 0x1
SOURCE MAC: 0001.962C.ED36 (48 bits)
SOURCE IP (32 bits) ==>
192.168.11.2
TARGET MAC: 0000.0000.0000 (48 bits)
TARGET IP: 192.168.11.3 (32 bits)


PDU Information at Device: PC3
OSI Model
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: 0001.640D.6790
SRC MAC: 0001.439D.0226
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 28
ID: 0x1
0x0
0x0
TTL: 255
PRO: 0x1
CHKSUM
SRC IP: 192.168.11.5
DST IP: 192.168.11.3
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
0 8 16 31 Bits
TYPE: 0x8
CODE: 0x0
CHECKSUM
ID: 0x3
SEQ NUMBER: 2


PDU Information at Device: PC3
OSI Model
Inbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: 0001.439D.0226
SRC MAC: 0001.640D.6790
TYPE: 0x806
DATA (VARIABLE LENGTH)
FCS: 0x0
ARP
0 8 16 31 Bits
HARDWARE TYPE: 0x1
PROTOCOL TYPE: 0x800
HLEN: 0x6
PLEN: 0x4
OPCODE: 0x2
SOURCE MAC: 0001.640D.6790 (48 bits)
SOURCE IP (32 bits) ==>
192.168.11.3
TARGET MAC: 0001.439D.0226 (48 bits)
TARGET IP: 192.168.11.5 (32 bits)

PART - B

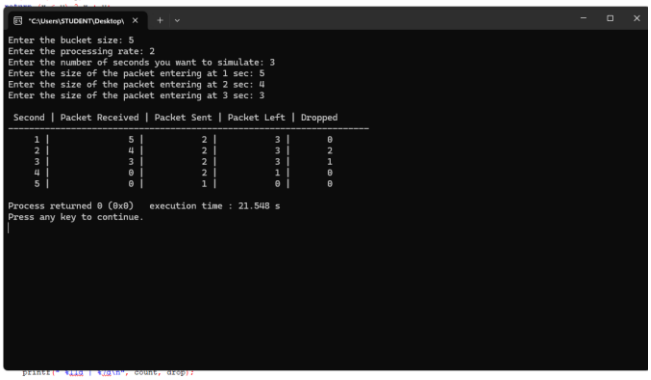
Program 1: Write a program for congestion control using Leaky bucket algorithm.

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: ");
   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("\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(" %11d | %7d\n", count, drop);

      drop = 0;
   }
   while (count != 0) {
      i++;
      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:



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:

Server:

```
#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, n;
   char buffer[256], line[2000], filedata[20000];
   struct sockaddr_in serv, cli;
   socklen_t len;
```

```
FILE *fp;
if (argc < 2) {
   printf("Error: No port number provided.\nUsage: ./server <port>\n");
   exit(1);
}
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0) {
   perror("Socket creation failed");
   exit(1);
}
memset(&serv, 0, sizeof(serv));
portno = atoi(argv[1]);
serv.sin_family = AF_INET;
serv.sin_addr.s_addr = INADDR_ANY;
serv.sin_port = htons(portno);
if (bind(sockfd, (struct sockaddr *)&serv, sizeof(serv)) < 0) {
   perror("Bind failed");
   exit(1);
}
listen(sockfd, 5);
printf("Server: Waiting for connection...\n");
len = sizeof(cli);
newsockfd = accept(sockfd, (struct sockaddr *)&cli, &len);
if (newsockfd < 0) {
   perror("Accept failed");
   exit(1);
}
```

```
memset(buffer, 0, sizeof(buffer));
n = read(newsockfd, buffer, sizeof(buffer) - 1);
if (n < 0) {
   perror("Error reading file name");
   exit(1);
}
printf("Server received file request: %s\n", buffer);
fp = fopen(buffer, "r");
if (fp == NULL) {
   printf("Server: File not found.\n");
   write(newsockfd, "File not found", 15);
   close(newsockfd);
   close(sockfd);
   return 0;
}
printf("Server: File found. Reading...\n");
memset(filedata, 0, sizeof(filedata));
while (fgets(line, sizeof(line), fp) != NULL) {
   strcat(filedata, line);
}
fclose(fp);
n = write(newsockfd, filedata, strlen(filedata));
if (n < 0)
   perror("Error writing to socket");
printf("Transfer complete.\n");
```

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

Client:

```
#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, portno, n;
    char filename[256], filedata[20000];
    struct sockaddr_in serv;
    if (argc < 2) {
        printf("Error: No port number provided.\nUsage: ./client <port>\n");
        exit(1);
    }
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        exit(1);
    }
    memset(&serv, 0, sizeof(serv));
```

```
    portno = atoi(argv[1]);
    serv.sin_family = AF_INET;
    serv.sin_port = htons(portno);
    serv.sin_addr.s_addr = INADDR_ANY;  // connecting to same machine
    if (connect(sockfd, (struct sockaddr *)&serv, sizeof(serv)) < 0) {
        perror("Server not responding");
        exit(1);
    }
    printf("Enter file path: ");
    scanf("%s", filename);
    n = write(sockfd, filename, strlen(filename));
    if (n < 0)
        perror("Error writing filename");
    memset(filedata, 0, sizeof(filedata));
    n = read(sockfd, filedata, sizeof(filedata) - 1);
    if (n < 0)
        perror("Error reading data from server");
    printf("\n--- File Content ---\n%s\n", filedata);
    close(sockfd);
    return 0;
}
```

Output:

```
$ cc socketserver.c
$ ./a.out 1025
server:
waiting for connection
server received:/home/aps/cse.txt
```

server:/home/aps/cse.txt found

opening and reading..

reading..

..reading complete

transfer complete

$ cc socketclient.c

$ ./a.out 1025

Enter the file with complete path

/home/aps/cse.txt

Reading..

..

client: display content of /home/aps/cse.txt

..

Welcome to the CSE department.....

2)

$ cc fserver.c

$ ./a.out

error:no port no

usage:

/server port no

$ cc fclient.c

$./a.out

Err:no port no.

usage:

./client portno

ex:./client 7777

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:

Server:

```
#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, portno, n;
   char buffer[1024];
   struct sockaddr_in servaddr, cliaddr;
   socklen_t len;

   if (argc < 2) {
      printf("Usage: ./server <port>\n");
      exit(1);
   }

   portno = atoi(argv[1]);
```

```
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if (sockfd < 0) {
   perror("Socket creation failed");
   exit(1);
}

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

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

if (bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
   perror("Bind failed");
   exit(1);
}

printf("UDP Server: Waiting for data...\n");

len = sizeof(cliaddr);
memset(buffer, 0, sizeof(buffer));

n = recvfrom(sockfd, buffer, sizeof(buffer)-1, 0,
            (struct sockaddr *)&cliaddr, &len);
if (n < 0) {
   perror("Receive failed");
```

```
        exit(1);
    }

    printf("Received from client: %s\n", buffer);

    // Echo back the same data
    n = sendto(sockfd, buffer, strlen(buffer), 0,
            (struct sockaddr *)&cliaddr, len);
    if (n < 0)
        perror("Send failed");

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

Client:

```
#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, portno, n;
    char buffer[1024];
```

```
struct sockaddr_in servaddr;
socklen_t len;

if (argc < 2) {
    printf("Usage: ./client <port>\n");
    exit(1);
}

portno = atoi(argv[1]);

sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if (sockfd < 0) {
    perror("Socket creation failed");
    exit(1);
}

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

servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(portno);
servaddr.sin_addr.s_addr = INADDR_ANY; // local machine

printf("Enter message to send: ");
scanf("%[^\n]", buffer);

len = sizeof(servaddr);
```

```
    n = sendto(sockfd, buffer, strlen(buffer), 0,
            (struct sockaddr *)&servaddr, len);
    if (n < 0) {
        perror("Send failed");
        exit(1);
    }

    memset(buffer, 0, sizeof(buffer));

    n = recvfrom(sockfd, buffer, sizeof(buffer)-1, 0,
                (struct sockaddr *)&servaddr, &len);
    if (n < 0) {
        perror("Receive failed");
        exit(1);
    }

    printf("Server replied: %s\n", buffer);
    close(sockfd);
    return 0;
}
```

Output:

1)

UDP Server: Waiting for data...

Received from client: Hello UDP Server!

2)

Enter message to send: Hello UDP Server!

Server replied: Hello UDP Server!

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: ");
   fgets(gen, sizeof(gen), stdin);
   gen[strcspn(gen, "\n")] = '\0';
   printf("Generator polynomial is CRC: %s\n", gen);

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

   printf("Enter the message: ");
   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++) {
            a[t] = (a[t] == gen[j]) ? '0' : '1';
            t++;
         }
      }
   }

   for (i = 0; i < k; i++) {
      rem[i] = a[n + i];
   }
```

```
    rem[k] = '\0';

    printf("The checksum appended:\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("Enter the received message: ");
    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++) {
                s[t] = (s[t] == gen[j]) ? '0' : '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 has an error.\n");
    }

    return 0;
}
```

OUTPUT:

```
Enter the generator polynomial: 10011
Generator polynomial is CRC: 10011
Enter the message: 1101011011

Message polynomial appended with zeros:
11010110110000
The checksum appended:
1110

The message with checksum appended:
11010110111110
Enter the received message: 11010110111110
Received polynomial is error-free.

Process returned 0 (0x0)   execution time : 20.417 s
Press any key to continue.
```