Bengaluru, India

SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY

COMPUTER NETWORKS LABORATORY MANUAL
[B20CI0506]
B.TECH III YEAR – V SEM
[A.Y:2022-2023]

Fifth Semester

B. Tech in AIML

INDEX

VISION OF THE UNIVERSITY

- "REVA University aspires to become an innovative university by developing excellent human resources with leadership qualities, ethical and moral values, research culture and innovative skills through higher education of global standards".

MISSION OF THE UNIVERSITY

- To create excellent infrastructure facilities and state-of-the-art laboratories and incubation centers
- To provide student-centric learning environment through innovative pedagogy and education reforms
- To encourage research and entrepreneurship through collaborations and extension activities
- To promote industry-institute partnerships and share knowledge for innovation and development
- To organize society development programs for knowledge enhancement in thrust areas
- To enhance leadership qualities among the youth and enrich personality traits, promote patriotism and moral values.

Program: B.Tech in CSE [Artificial Intelligence and Machine Learning]

VISION OF THE SCHOOL

To produce excellent quality technologists and researchers of global standards in computing and Information technology who have potential to contribute to the development of the nation and the society with their expertise, skills, innovative problem-solving abilities, strong moral and ethical values.

MSSION OF THE SCHOOL

- To create state of the art computing labs infrastructure and research facilities in information technology.
- To provide student-centric learning environment in Computing and Information technology through innovative pedagogy and education reforms.
- To encourage research, innovation and entrepreneurship in computing and information technology through industry/academia collaborations and extension activities
- Organize programs through club activities for knowledge enhancement in thrust areas of information technology.
- To enhance leadership qualities among the youth and enrich personality traits, promote patriotism, moral and ethical values.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

After few years of graduation, the graduates of B. Tech Computer Science & Engineering (AI & ML) will:

PEO-1: Demonstrate technical skills, competency in AI & ML and exhibit team management capability with proper communication in a job environment.

PEO-2: Support the growth of economy of a country by starting enterprise with a lifelong learning attitude

PEO-3: Carry out research in the advanced areas of AI & ML and address the basic needs of the society.

PROGRAM SPECIFIC OUTCOMES (PSO's)

On successful completion of the program, the graduates of B. Tech CSE(AIML) program will be able to:

- **PSO-1:** Demonstrate the knowledge of human cognition, Artificial Intelligence, Machine Learning and data engineering for designing intelligent systems.
- **PSO-2:** Apply computational knowledge and project development skills to provide innovative solutions.
- **PSO-3:** Use tools and techniques to solve problems in AI & ML.

PROGRAM OUTCOMES (PO'S)

On successful completion of the program, the graduates of B. Tech CSE (AIML) program will be able to:

PO-1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals for the solution of complex problems in Computer Science and Engineering.

PO-2: Problem analysis: Identify, formulate, research literature, and analyze engineering problems

To arrive at substantiated conclusions using first principles of mathematics, natural, and engineering sciences.

PO-3: Design/development of solutions: Design solutions for complex engineering problems and design system components, processes to meet the specifications with consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO-4: Conduct investigations of complex problems: Use research-based knowledge including design of Experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO-5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO-6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO-7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO-8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice

PO-9: Individual and team work: Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.

PO-10: Communication: Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations, and give and receive clear instructions.

PO-11: Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments.

PO-12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

1. Course Objectives:

The main objectives of this course are:

1. Explain the protocol stacks (OSI and TCP/IP) for data communication
2. Discuss the error detection & correction strategies for data transmission.
3. Design the connection establishment of network computing devices.
4. Illustrate the TCP, UDP protocols and explain Domain Name System.
5. Emphasis the management of local area networks
6. Learning about computer network organization and implementation

2. Lab Requirements:

Following are the required hardware and software for this lab, which is availablein the laboratory.

Minimum System requirements:

- Processors: Intel® Core™ i5 Core processor.
- Disk space: 1 GB.
- Operating systems: Windows and Linux.
- Wireshark, CISCO Packet Tracer,NS3

About the Lab:

The main emphasis of this course is on the organization and management of local area networks(LANs).The course description include learning about computer network organization and implementation, obtaining a the or etical understanding of data communication and computer networks, and about Open Systems Interconnection(OSI) communication model with TCP/IP protocol; This course provides knowledge of error detection and recovery; local area networks; bridges, routers and gateways; network naming and addressing; and local and remote procedures. This course also emphasis on User Datagram Protocol, TCP Congestion Control; DNS Message Formatting and Remote Login. Protocols

3. Guidelines to Students

- ➢ Equipment in the lab for the use of student community. Students need to maintaina proper decorum in the computer lab. Students must use the equipment with care. Any damage is caused is punishable.
- ➢ Students are required to carry their observation / programs book with completed

exercises while entering the lab.

- Students are supposed to occupy the machines allotted to them and are not supposed to talk or make noise in the lab. The allocation is put up on the lab noticeboard.
- Lab can be used in free time / lunch hours by the students who need to use the systemsshould take prior permission from the lab in-charge.
- Lab records need to be submitted on or before date of submission.
- Students are not supposed to use flash drives.

Instructions to maintain the record

- Before start of the first lab they have to buy the record and bring the record to the lab.
- Regularly (Weekly) update the record after completion of the experiment and get itcorrected with concerned lab in-charge for continuous evaluation.
- In case the record is lost inform the same day to the faculty in charge and get the new record within 2 days the record has to be submitted and get it corrected by the faculty.
- If record is not submitted in time or record is not written properly, the evaluation marks(5M) will be deducted.

General laboratory instructions

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with: a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session. b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab. c. Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty. 5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
5. All the students should be polite and cooperative with the laboratory staff, must maintainthe discipline and decency in the laboratory.\
6. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
7. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
8. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

Course Outcomes (COs)

After the completion of the course, the student will be able to:

COs	Course Outcomes
B20CI0506.1	Make use of the architectural principles of computer networking and compare different approaches to organizing networks.
B20CI0506.2	Identify the good network design with simplicity, scalability, performance and the end-to-end principle.
B20CI0506.3	Appraise the working principles of Internet.
B20CI0506.4	Develop applications using network protocols.
B20CI0506.5	Emphasis the management of local area networks
B20CI0506.6	Learning about computer network organization and implementation

Conduction of Practical Examination:

1. All laboratory experiments (Part A No. 1 to 4 and Part B No. 1 to 5) are included for the syllabus of practicalexamination.
2. Students are allowed to pick one experiment from the lot.
3. Strictly follow the instructions as printed on the cover page of answer script.
4. Marks distribution:

Procedure + Conduction + Viva: 08 + 35 +07 = 50 Marks

Change of experiment is allowed only once and marks allotted to the procedure part tobe made zero.

CO-PO-PSO MAPPING

Course	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
B20CI0506.1	3	3	3	3	2	1						1	3		
B20CI0506.2	3	3	3	3	2	2						1	3		
B20CI0506.3	3	3	3	3	2	1						2	3		
B20CI0506.4	3	3	3	2	2	1						1	3	3	3

B20CI0506.5	3	3	3	2	2	1						2		3	3
B20CI0506.6	3	3	3	2	2	1						2	2	3	3

WEEKWISE EVALUATION OF EACH PROGRAM

ACTIVITY	**MARKS**
Observation +Viva	20
Record	10
TOTAL	30

INTERNAL ASSESSMENT EVALUATION (End of Semester)		
Sl No	**ACTIVITY**	**MARKS**
01	Procedure	7
02	Conduction	8
03	Viva Voce	5
	Total	20

FINAL INTERNAL ASSESSMENT CALCULATION		
Sl. No	**ACTIVITY**	**MARKS**
01	Average of weekly Entries	30
02	Internal Assessment Reduced to	20
	Total	50

Program 1

1. **Write a program for error detecting code using CRC-CCITT (16- bits).**

Whenever digital data is stored or interfaced, data corruption might occur. Since the beginning of computer science, developers have been thinking of ways to deal with this type of problem. For serial data they came up with the solution to attach a parity bit to each sent byte. This simple detection mechanism works if an odd number of bits in a byte changes, but an even number of false bits in one byte will not be detected by the parity check. To overcome this problem developers have searched for mathematical sound mechanisms todetect multiple false bits. The CRC calculation or cyclic redundancy check was the result of this. Nowadays CRC calculations are used in all types of communications. All packets sent over a network connection are checked with a CRC. Also each data block on your hard disk has a CRC value attached to it. Modern computer world cannot do without these CRC calculations. So let's see why they are so widely used. The answer is simple; they are powerful, detect many types of errors and are extremely fast to calculate especially when dedicated hardware chips are used.The idea behind CRC calculation is to look at the data as one large binary number. This number is divided by a certain value and the remainder of the calculation is called the CRC. Dividing in the CRC calculation at first looks to cost a lot of computing power, but it can be performed very quickly if we use a method similar to the one learned at school. We will as an example calculate the remainder for the character 'm'—which is 1101101 in binary notation— by dividing it by 19 or 10011. Please note that 19 is an odd number. This is necessary as we will see further on. Please refer to your schoolbooks as the binary calculation method here is not very different from the decimal method you learned when you were young. It might only look a little bit strange. Also notations differ between countries, but the method is similar.

```
1 0 1
----------------
10011 / 1 1 0 1 1 0 1
1 0 0 1 1 | |
--------- –---| |
1 0 0 0 0 |
0 0 0 0 0 |
-------------- |
1 0 0 0 0 1
1 0 0 1 1
-----------------
1 1 1 0 = 14 remainder
```

• The message bits are appended with c zero bits; this augmented message is the dividend
• A predetermined c+1-bit binary sequence, called the generator polynomial, is the divisor
• The checksum is the c-bit remainder that results from the division operation

With decimal calculations you can quickly check that 109 divided by 19 gives a quotient of 5 with 14 as the remainder. But what we also see in the scheme is that every bit extra to check only costs one binary comparison and in 50% of the cases one binary subtraction. You can easily increase the number of bits of the test data string—for example to 56 bits if we use our example value "Lammert"—and the result can be calculated with 56 binary comparisons and an average of 28 binary subtractions. This can be implemented in hardware directly with only very few transistors involved. Also software algorithms can be very efficient.

All of the CRC formulas you will encounter are simply checksum algorithms based on modulo-2 binary division where we ignore carry bits and in effect the subtraction will be equal to an exclusive or operation. Though some differences exist in the specifics across different CRC formulas, the basic mathematical process is always the same:

Table 1 lists some of the most commonly used generator polynomials for 16- and 32-bit CRCs. Remember that the width of the divisor is always one bit wider than the remainder.

So, for example, you‘d use a 17-bit generator polynomial whenever a 16-bit checksum is required.

	CRC - CCITT	**CRC – 16**	**CRC - 32**
Checksum Width	16 bits	16 bits	32 bits
Generator Polynomial	10001000000100001	11000000000000101	100000100110000010001110110110111

International Standard CRC Polynomials

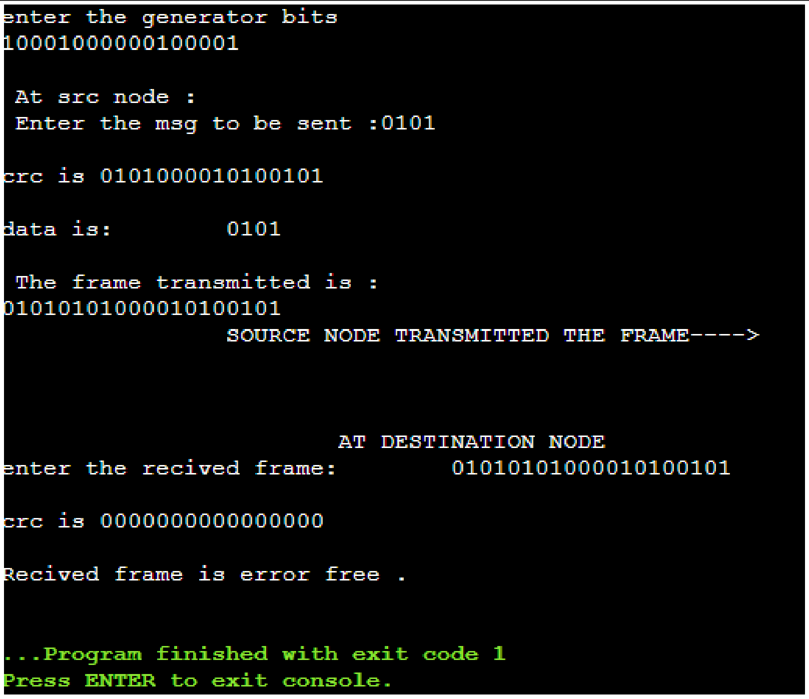
Program No: 1

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

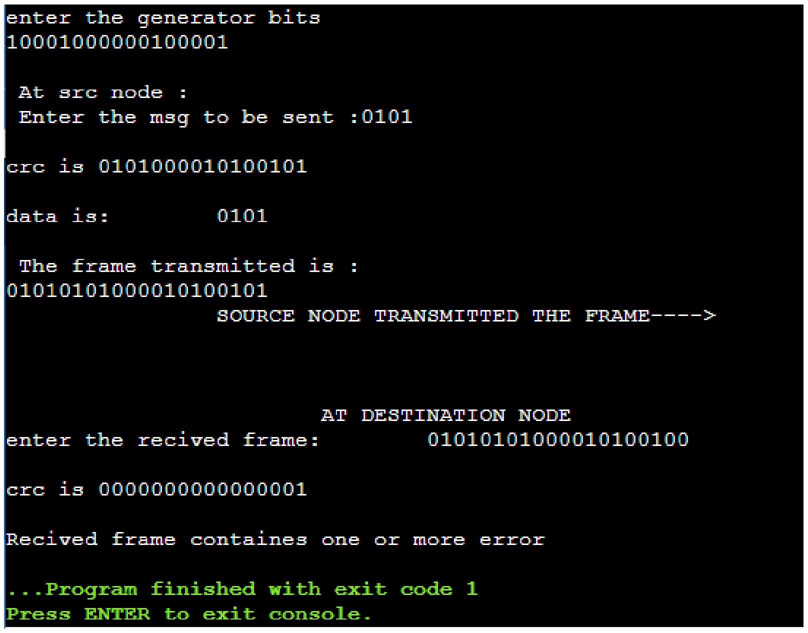
```
#include<stdio.h>
#include<string.h>

char data[100],concatdata[117],src_crc[17],dest_crc[17],frame[120],divident[18];
char divisor[18];
char res[17]="0000000000000000";

void crc_cal(int node)
{
int i,j;
for(j=17;j<=strlen(concatdata);j++)
{
        if(divident[0]=='1')
        {
        for(i=1;i<=16;i++)
        if(divident[i]!=divisor[i])
        divident[i-1]='1';
        else
        divident[i-1]='0';
        }

        else
        {
        for(i=1;i<=16;i++)
        divident[i-1]=divident[i];
        }
        if(node==0)
        divident[i-1]=concatdata[j];
        else
        divident[i-1]=frame[j];
}
divident[i]='\0';
printf("\ncrc is %s\n",divident);
if(node==0)
{
strcpy(src_crc,divident);
}
else
strcpy(dest_crc,divident);
}
int main()
{
```

```
int i;
printf("enter the generator bits\n");
gets(divisor);
if(strlen(divisor)<17 || strlen(divisor)>17)
{
printf("please enter the geneartor length min of 17 bits\n");
exit(0);
}
printf("\n At src node :\n Enter the msg to be sent :");
gets(data);
strcpy(concatdata,data);
strcat(concatdata,"0000000000000000");
for(i=0;i<=16;i++)
divident[i]=concatdata[i];
divident[i]='\0';
crc_cal(0);
printf("\ndata is:\t");
puts(data);
printf("\n The frame transmitted is :\t");
printf("\n%s%s",data,src_crc);
printf("\n\t\tSOURCE NODE TRANSMITTED THE FRAME---->");
printf("\n\n\n\n\t\t\tAT DESTINATION NODE\nenter the recived frame:\t");
gets(frame);
for(i=0;i<=16;i++)
divident[i]=frame[i];
divident[i]='\0';
crc_cal(1);
if((strcmp(dest_crc,res))==0)
printf("\nRecived frame is error free .\n ");
else
printf("\nRecived frame containes one or more error ");
return 1;
}
```

Output:

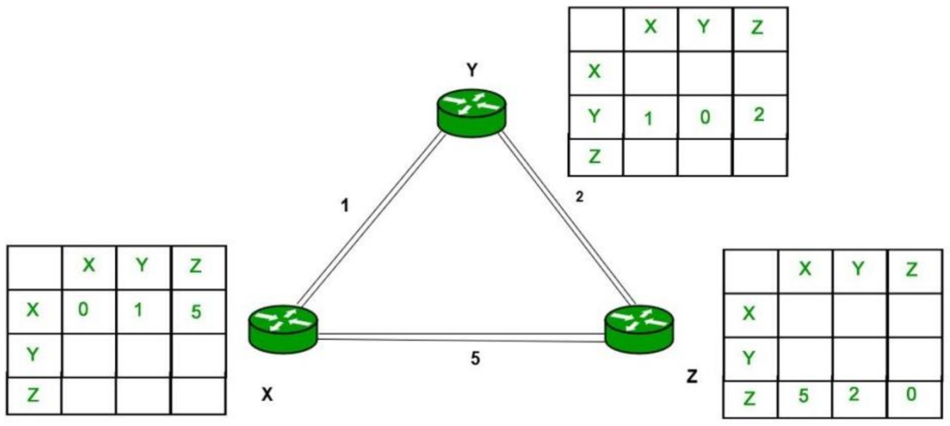


```
enter the generator bits
10001000000100001

 At src node :
 Enter the msg to be sent :0101

crc is 0101000010100101

data is:         0101

 The frame transmitted is :
01010101000010100101
                SOURCE NODE TRANSMITTED THE FRAME---->



                        AT DESTINATION NODE
enter the recived frame:        01010101000010100101

crc is 0000000000000000

Recived frame is error free .


...Program finished with exit code 1
Press ENTER to exit console.
```



```
enter the generator bits
10001000000100001

 At src node :
 Enter the msg to be sent :0101

crc is 0101000010100101

data is:         0101

 The frame transmitted is :
01010101000010100101
                SOURCE NODE TRANSMITTED THE FRAME---->



                        AT DESTINATION NODE
enter the recived frame:        01010101000010100100

crc is 0000000000000001

Recived frame containes one or more error

...Program finished with exit code 1
Press ENTER to exit console.
```

Write a program to find the shortest path between vertices using Distance vector algorithm.Consider the below network:



Algorithm:

At each node x,

Initialization

for all destinations y in N:

$D_X(y) = c(x,y)$ // If y is not a neighbor then $c(x,y) = \infty$

for each neighbor w

$D_W(y) = ?$ for all destination y in N.for each neighbor w

send distance vector $D_X = [\ D_X(y)\ : y \text{ in } N\ ]$ to w

loop

wait(until I receive any distance vector from some neighbor w)for each y in N:

$D_X(y) = \min_v\{c(x,v)+D_v(y)\}$

If $D_X(y)$ is changed for any destination y

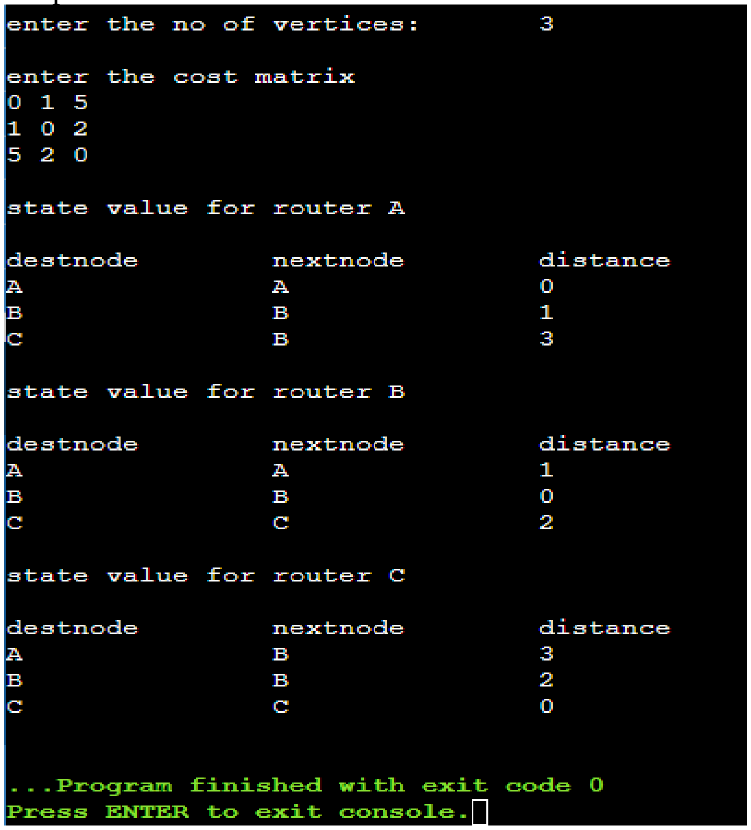
Send distance vector $D_X = [\ D_X(y) : y \text{ in } N\ ]$ to all neighbors

Program No: 2

Write a program to find the shortest path between vertices using Distance vector algorithm.Consider the below network:

```
#include<stdio.h>

struct rtable
{
int dist[20],nextnode[20];
}table[20];
 int cost[10][10],n;
 void distvector()
{
int i,j,k,count=0;
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
table[i].dist[j]=cost[i][j];
table[i].nextnode[j]=j;
}       }
do
{
count=0;
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
for(k=0;k<n;k++)
{
if(table[i].dist[j]>cost[i][k]+table[k].dist[j])
{
table[i].dist[j]=table[i].dist[k]+table[k].dist[j];
table[i].nextnode[j]=k;
count++;
}
}
}
}
}while(count!=0);
}
int main()
{
int i,j;
printf("\nenter the no of vertices:\t");
scanf("%d",&n);
printf("\nenter the cost matrix\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&cost[i][j]);
distvector();
for(i=0;i<n;i++)
{
printf("\nstate value for router %c \n",i+65);
printf("\ndestnode\tnextnode\tdistance\n");
for(j=0;j<n;j++)
{
if(table[i].dist[j]==99)
printf("%c\t\t-\t\t infinite\n",j+65);
else
printf("%c\t\t%c\t\t%d\n",j+65,table[i].nextnode[j]+65,table[i].dist[j]);
}
}
return 0;

 }
```

Output:



```
enter the no of vertices:       3

enter the cost matrix
0 1 5
1 0 2
5 2 0

state value for router A

destnode        nextnode        distance
A               A               0
B               B               1
C               B               3

state value for router B

destnode        nextnode        distance
A               A               1
B               B               0
C               C               2

state value for router C

destnode        nextnode        distance
A               B               3
B               B               2
C               C               0


...Program finished with exit code 0
Press ENTER to exit console.
```

Using TCP sockets, write a client – server program to make the interaction between the client and the server.

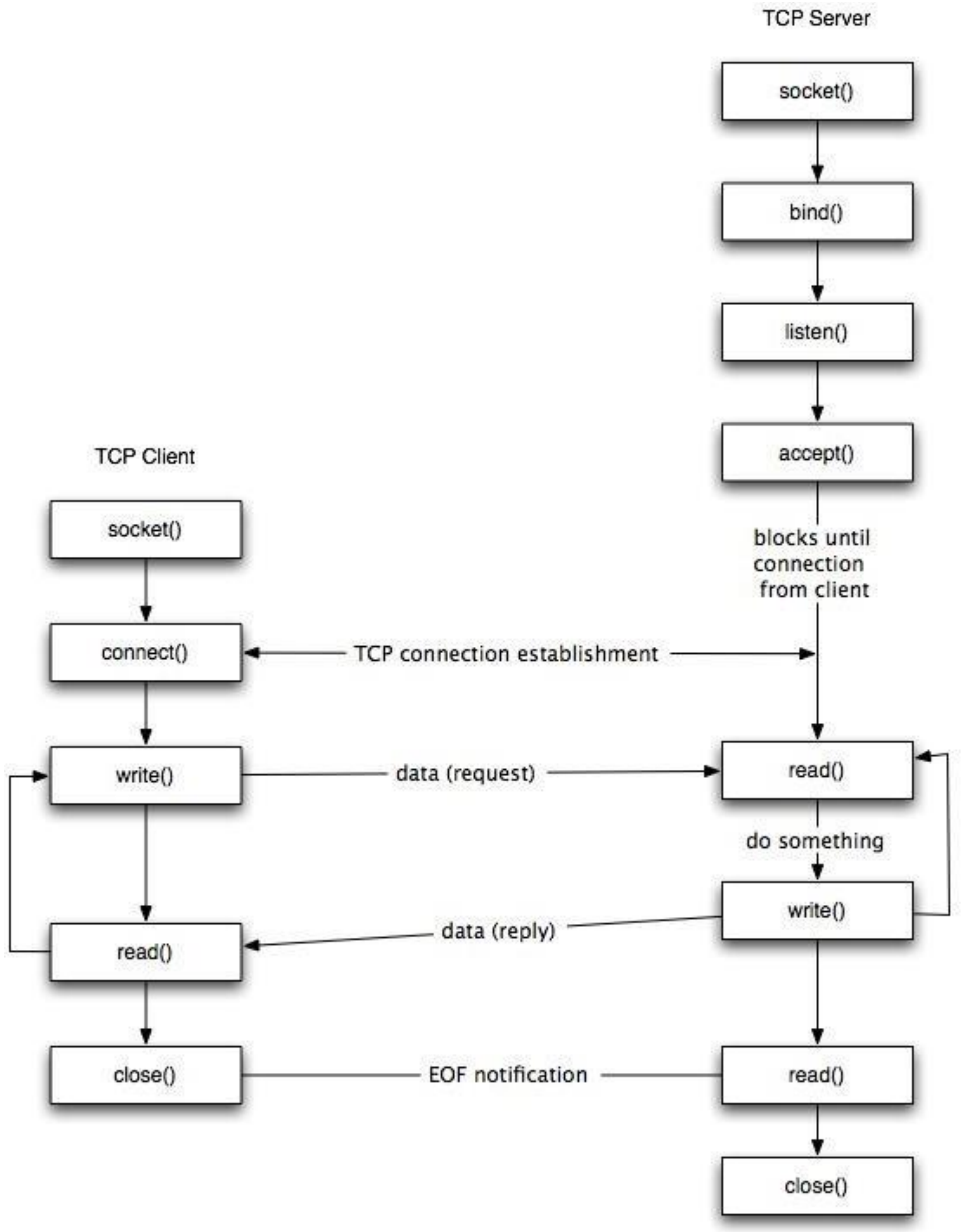


Fig: TCP Socket lifecycle diagram.

Stages for server

1. **Socket creation:**
   int sockfd = socket(domain, type, protocol)

- **sockfd:** socket descriptor, an integer (like a file-handle)
- **domain:** integer, specifies communication domain. We use AF_ LOCAL as defined in the POSIX standard for communication between processes on the same host. For communicating between processes on different hosts connected by IPV4, we use AF_INET and AF_I NET 6for processes connected by IPV6.
- **type:** communication type
  SOCK_STREAM: TCP(reliable, connection oriented) SOCK_DGRAM: UDP(unreliable, connectionless)
- **protocol:** Protocol value for Internet Protocol(IP), which is 0. This is the same number which appears on protocol field in the IP header of a packet.(man protocols for more details)

2. **Setsockopt:** This helps in manipulating options for the socket referred by the file descriptor sockfd. This is completely optional, but it helps in reuse of address and port. Prevents error such as: "address already in use".
   *int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen);*

3. **Bind:**
   *int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);*

   After creation of the socket, bind function binds the socket to the address and port number specified in addr(custom data structure). In the example code, we bind the server to the localhost, hence we use INADDR_ANY to specify the IP address.

4. **Listen:**
   int listen(int sockfd, int backlog);

   It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection. The backlog, defines the maximum length to which the queue of pending connections for sockfd may grow. If a connection request arrives when the queue is full, the client may receive an error with an indication of ECONNREFUSED.

5. **Accept:**
   *int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);*

   It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket. At this point, connection is established between client and server, and they are ready to transfer data.

Code for TCP socket Server

Program No: 3(a) Server

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/fcntl.h>
#include<stdlib.h>
int main(int argc,char *argv[])
{
int fd,sockfd,newsockfd,clilen,portno,n;
struct sockaddr_in seradd,cliadd;
char buffer[4096];
if(argc<2)
{
fprintf(stderr,"\n\n No port\n");
exit(1);
}
portno=atoi(argv[1]);
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0)
error("\n error opening socket.\n");
bzero((char *)&seradd,sizeof(seradd));
seradd.sin_family=AF_INET;
seradd.sin_addr.s_addr=(htonl)INADDR_ANY;
seradd.sin_port=htons(portno);
if(bind(sockfd,(struct sockaddr *)&seradd,sizeof(seradd))<0)
perror("\n IP addr cannt bind");
listen(sockfd,5);
clilen=sizeof(cliadd);
printf("\n Server waiting for clint request\n");
while(1)
{
newsockfd=accept(sockfd,(struct sockaddr *)&cliadd,&clilen);
if(newsockfd<0)
perror("\n Server cannot accept connection request ");
bzero(buffer,4096);
read(newsockfd,buffer,4096);
fd=open(buffer,O_RDONLY);
if(fd<0)
perror("\n File  doesnot exist");
while(1)
{
n=read(fd,buffer,4096);
if(n<=0)
exit(0);
write(newsockfd,buffer,n);
printf("\n File transfer completet\n");
}
close(fd);
close(newsockfd);
}
return 0;}
```

Stages for Client

- **Socket connection:** Exactly same as that of server's socket creation
- **Connect:** The connect() system call connects the socket referred to by the file Descriptor sockfd to the address specified by addr. Server's address and port is specified in addr.

Int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

Program No: 3(b) Client

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

int main(int argc,char *argv[])
{
int sockfd,portno,n;
struct sockaddr_in seradd;
char buffer[4096],*serip;
if(argc<4)
{
fprintf(stderr,"usage %s serverip filename port",argv[0]);
exit(0);
}
serip=argv[1];
portno=atoi(argv[3]);
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0)
perror("\n Error in creating socket.\n");
perror("\n Client on line.");
bzero((char *)&seradd,sizeof(seradd));
seradd.sin_family=AF_INET;
seradd.sin_addr.s_addr=inet_addr(serip);
seradd.sin_port=htons(portno);
if(connect(sockfd,(struct sockaddr *)&seradd,sizeof(seradd))<0)
perror("\n Error in connection setup \n");
write(sockfd,argv[2],strlen(argv[2])+1);
bzero(buffer,4096);
n=read(sockfd,buffer,4096);
if(n<=0)
{
perror("\n File not found");
exit(0);
}
write (1,buffer,n);
}
```

Output:

cc client.c

./a.out

```
Client:Hello message sentHello from server
```

cc server.c

./a.out

```
Server:Hello from clientHello message sent
```

Implement the above program using as message queues or FIFOs as IPC channels.

Algorithm (Client Side)

1. Start.
2. Open well known server FIFO in write mode.
3. Write the pathname of the file in this FIFO and send the request.
4. Open the client specified FIFO in read mode and wait for reply.
5. When the contents of the file are available in FIFO, display it on the terminal
6. Stop.

Algorithm (Server Side)

1. Start.
2. Create a well known FIFO using mkfifo()
3. Open FIFO in read only mode to accept request from clients.
4. When client opens the other end of FIFO in write only mode, then read the contents and store it in buffer.
5. Create another FIFO in write mode to send replies.
6. Open the requested file by the client and write the contents into the client specified FIFO and terminate theconnection.
7. Stop.

Program No: 4(a) Server

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#define FIFO1 "fifo1"
#define FIFO2 "fifo2"

int main()
{
char p[100],c[5000,ch;
int num,fd,fd2,f1;
mknod(FIFO1,S_IFIFO|0666,0);
mknod(FIFO2,S_IFIFO|0666,0);
printf("\n Server online...\n");
fd=open(FIFO1,O_RDONLY);
fd2=open(FIFO2,O_WRONLY);
printf("Server online\n waiting for client \n\n");
if((num=read(fd,p,100))==-1)
perror("\n Read Error ");
else
{
p[num]='\0';
printf("\n File is %s .\n",p);
if((f1=open(p,O_RDONLY))<0)
{
write(fd2,"File not found",15);
return 1;
}
else
{
stdin=fdopen(f1,"r");
num=0;
while((ch=fgetc(stdin))!=EOF)
c[num++]=ch;
c[num]=0;
printf(" Server: Transfering the contents of :%s ",p);
if(num=write(fd2,c,strlen(c))==-1)
printf("\n Error in writting to FIFO\n");
else
printf("\n File transfer completed \n");
}}}
```

Program No: 4(b) Client

```
#include<stdio.h>

#include<stdlib.h>
#include<string.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#define FIFO1 "fifo1"
#define FIFO2 "fifo2"

int main()
{
char p[100],c[5000];
int num,fd,fd2,f1;
mknod(FIFO1,S_IFIFO|0666,0);
mknod(FIFO2,S_IFIFO|0666,0);
printf("\n Client online...\n");
fd=open(FIFO1,O_WRONLY);
fd2=open(FIFO2,O_RDONLY);
printf("Client : Enter the filename . \n\n");
scanf("%s",p);
num=write(fd,p,strlen(p));
if(num==-1)
{
perror("\nWrite Error.\n");
return 1;
}
else
{
printf("\n Waiting for reply\n");
if((num=read(fd2,c,5000))==-1)
perror("\nError while reading from fifo \n");
else
{
c[num]=0;
printf("%s",c);
}}
return 1;
}
```

SERVER OUTPUT

```
[root@localhost ~]# vi fifo_server.c

[root@localhost ~]# cc -o fifo_server fifo_server.c



[root@localhost ~]# ./fifo_serverWaiting for connection Request...Connection Established...Client has
requested file dvr.c

[root@localhost ~]#
```

CLIENTOUTPUT

```
[root@localhost ~]# vi fifo_client.c

[root@localhost ~]# cc -o fifo_client fifo_client.c

[root@localhost ~]# ./fifo_clientTrying to Connect to Server...Connected...

Enter the filename to request from server: dvr.c

Waiting for Server to reply...
```

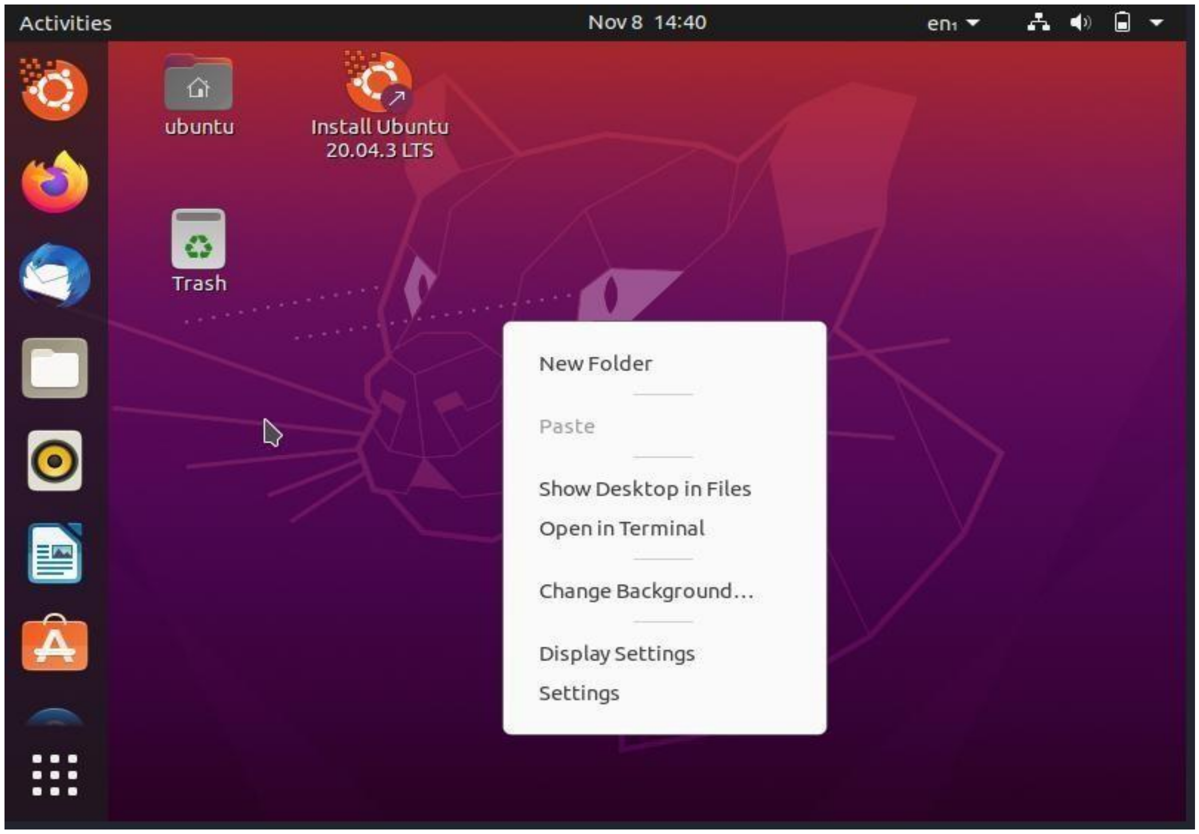
Part B- Network Simulation

Program No: 1

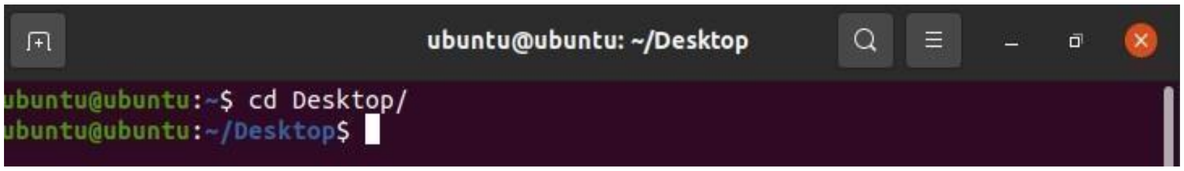
1. **Using WIRESHARK observe the data transferred in client server communicationusing UDP and identify the UDP datagram.**

UDP Program Execution Steps:Steps :

- Open Terminal using **CTRL+ALT+T** or right-click and select **Open in Terminal**.



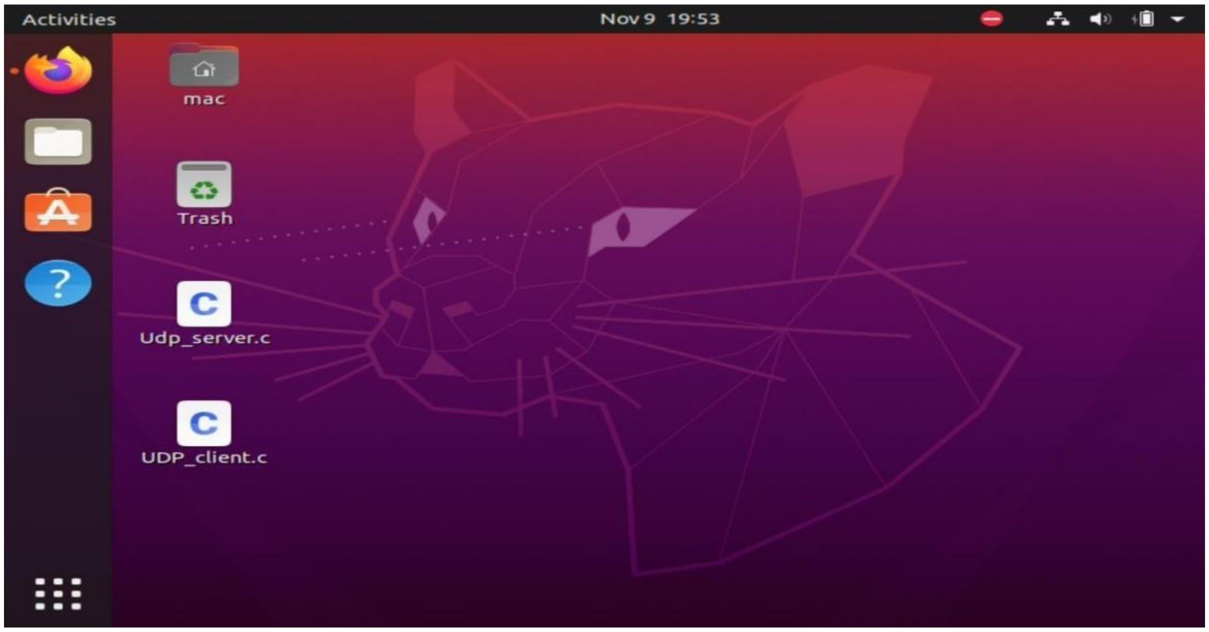
- Once opened, type the following command in the terminal:
cd Desktop/



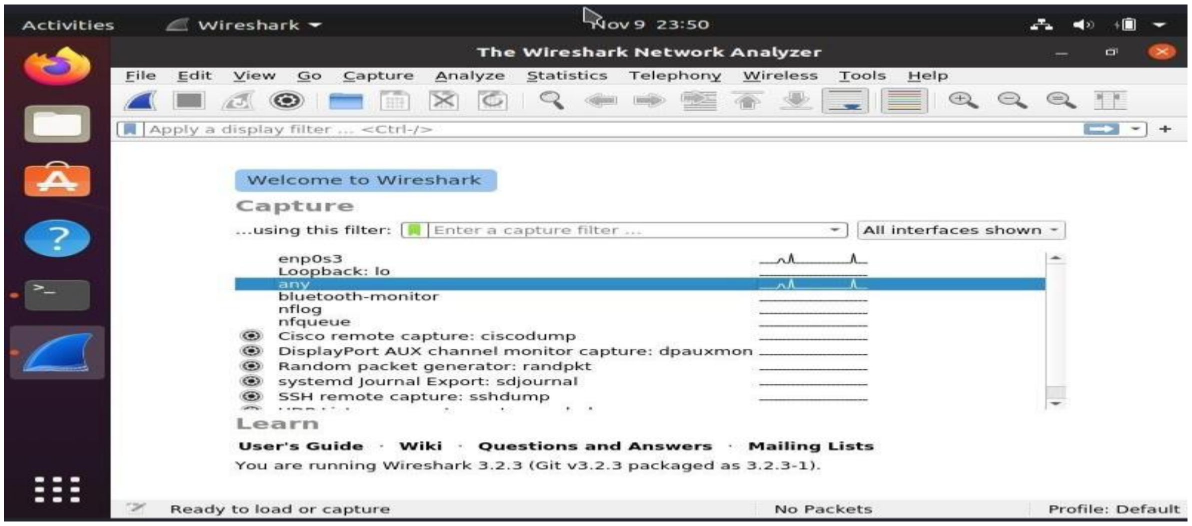
```
ubuntu@ubuntu:~$ cd Desktop/
ubuntu@ubuntu:~/Desktop$
```

This command will change your default terminal path to desktop. **(Note: Linux iscase-sensitive, so type the command as it is)**

[Before going through the next sequence of commands, make sure that both theUDP programs are located on the desktop]



Before executing the program, open **Wireshark** and double-clickon **any-interface** for packet capture process to start.

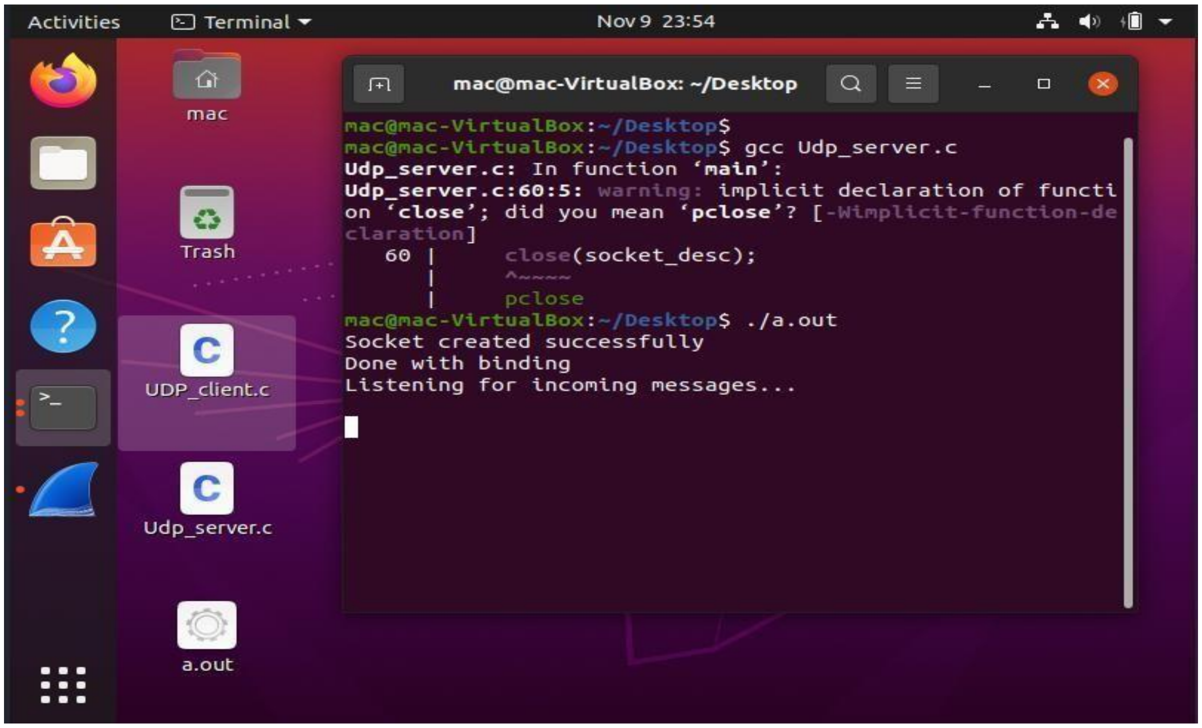


```
enp0s3
Loopback: lo
any
bluetooth-monitor
nflog
nfqueue
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
```

You are running Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1).

Ready to load or capture   No Packets   Profile: Default

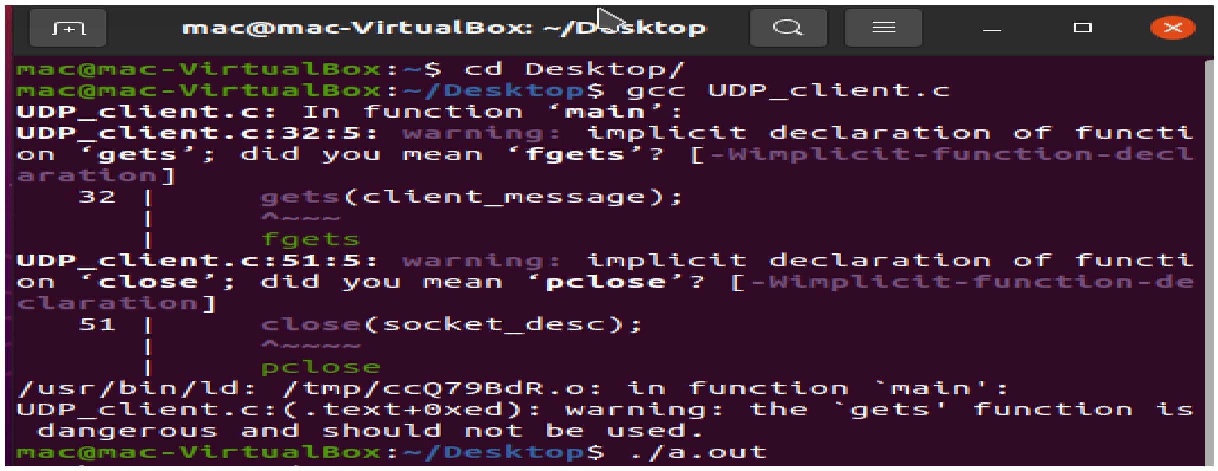
- To compile the C program, type the following command in the same order asfollows inthe terminal:
  gcc Udp_server.c
- To execute the program type, object code file name in the terminal as follows :
  ./a.out



```
mac@mac-VirtualBox:~/Desktop$
mac@mac-VirtualBox:~/Desktop$ gcc Udp_server.c
Udp_server.c: In function 'main':
Udp_server.c:60:5: warning: implicit declaration of functi
on 'close'; did you mean 'pclose'? [-Wimplicit-function-de
claration]
   60 |     close(socket_desc);
      |     ^~~~~
      |     pclose
mac@mac-VirtualBox:~/Desktop$ ./a.out
Socket created successfully
Done with binding
Listening for incoming messages...
```

- Open another terminal and change its path to Desktop using Cd command and run the ccode as shown below:
  cd Desktop/
  gcc UDP_client.c

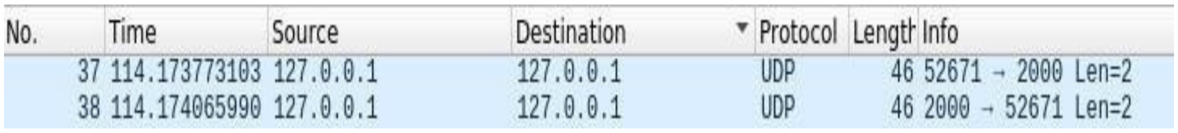
  ./a.out



```
mac@mac-VirtualBox:~$ cd Desktop/
mac@mac-VirtualBox:~/Desktop$ gcc UDP_client.c
UDP_client.c: In function 'main':
UDP_client.c:32:5: warning: implicit declaration of functi
on 'gets'; did you mean 'fgets'? [-Wimplicit-function-decl
aration]
   32 |     gets(client_message);
      |     ^~~~
      |     fgets
UDP_client.c:51:5: warning: implicit declaration of functi
on 'close'; did you mean 'pclose'? [-Wimplicit-function-de
claration]
   51 |     close(socket_desc);
      |     ^~~~~
      |     pclose
/usr/bin/ld: /tmp/ccQ79BdR.o: in function `main':
UDP_client.c:(.text+0xed): warning: the `gets' function is
 dangerous and should not be used.
mac@mac-VirtualBox:~/Desktop$ ./a.out
```

- Once the client program is executed, it will ask you to enter the appropriate message thatwill be sent to the server.
- After the execution of the program, go to Wireshark and stop the capturing process and

check for the packet that has the source and the destination address as **127.0.0.1.**

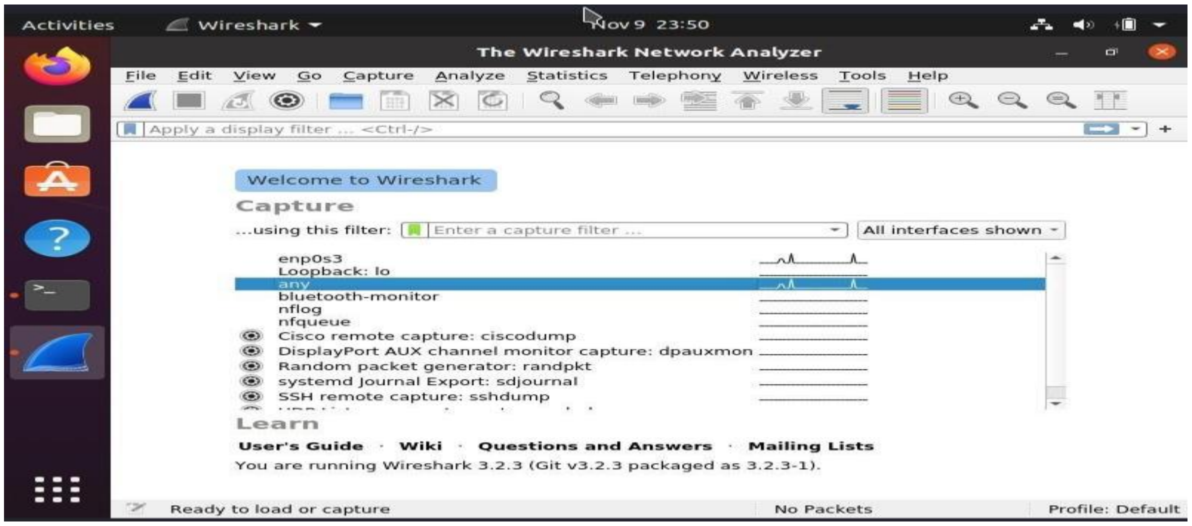


No.	Time	Source	Destination	Protocol	Length	Info
37	114.173773103	127.0.0.1	127.0.0.1	UDP	46	52671 → 2000 Len=2
38	114.174065990	127.0.0.1	127.0.0.1	UDP	46	2000 → 52671 Len=2

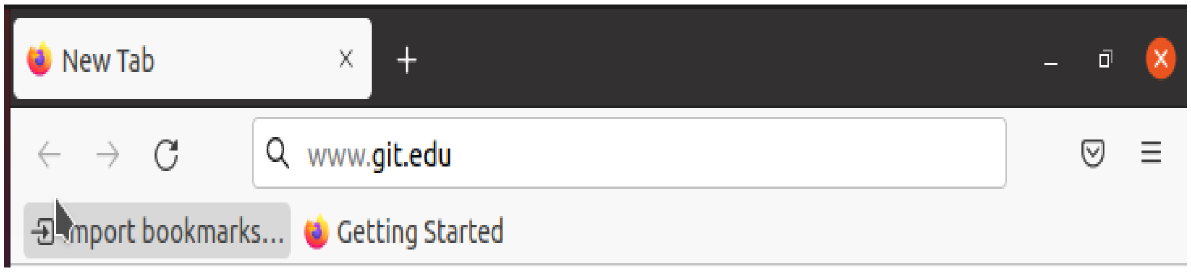
- Analyze the UDP Packets.

Capturing UDP Packets with browser :Steps:

- Open Wireshark and double-click on **any-interface** to start the packet captureprocess.

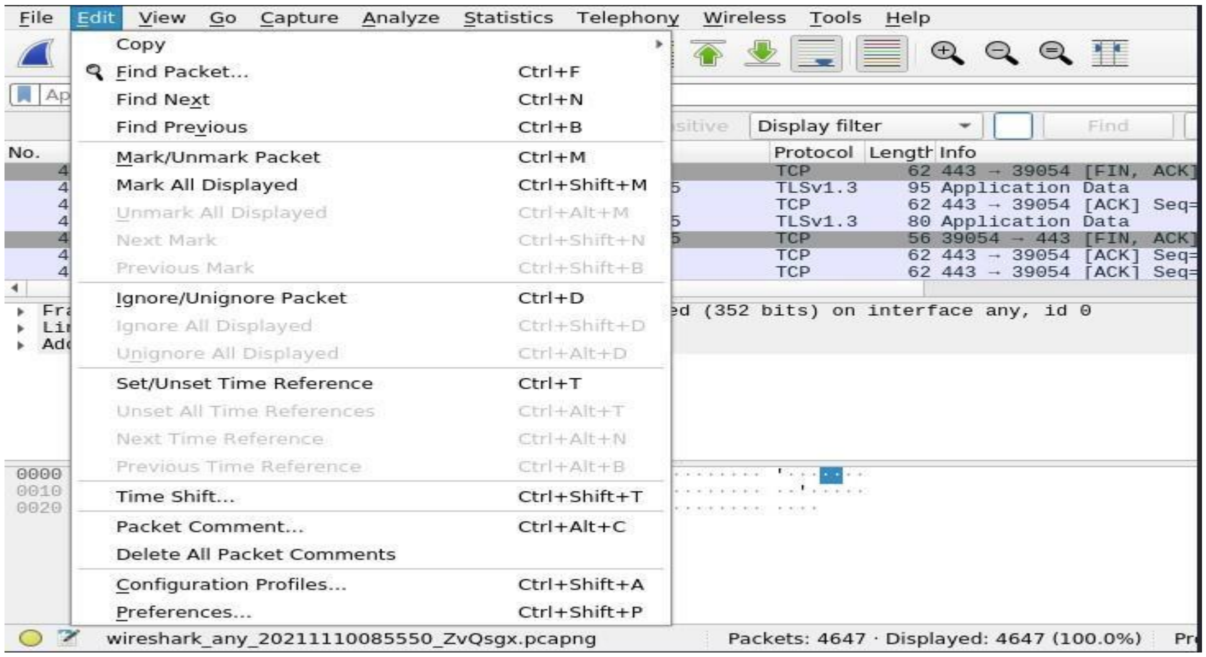

Activities
Wireshark
Nov 9 23:50
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
enp0s3
Loopback: lo
any
bluetooth-monitor
nflog
nfqueue
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
You are running Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1).
Ready to load or capture
No Packets
Profile: Default

- Open the browser and enter any website's fully qualified domain name in thebrowseraddress bar and



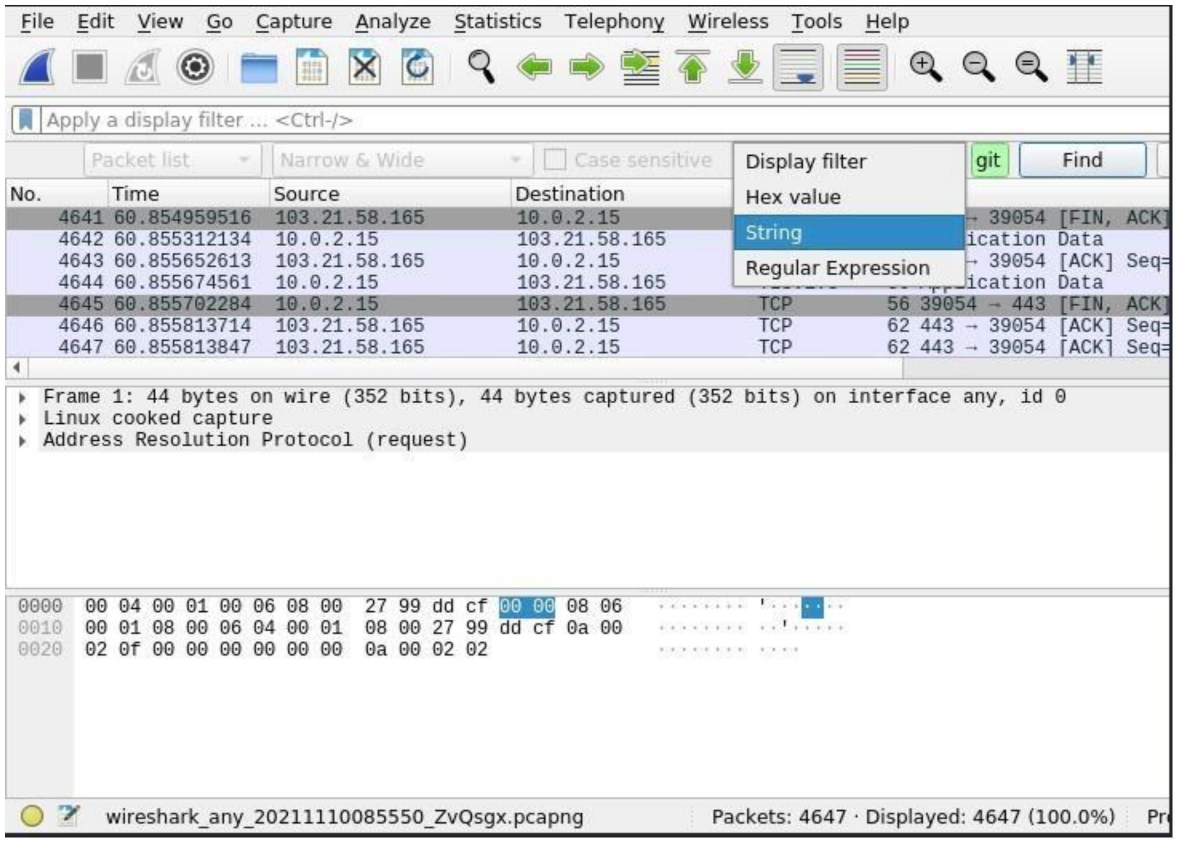
hit enter.

- After the site is fully loaded, stop the capturing process in Wireshark go to edit inthemenu bar and select find packet option or just press CTRL+F.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Copy
Find Packet... Ctrl+F
Find Next Ctrl+N
Find Previous Ctrl+B
Mark/Unmark Packet Ctrl+M
Mark All Displayed Ctrl+Shift+M
Unmark All Displayed Ctrl+Alt+M
Next Mark Ctrl+Shift+N
Previous Mark Ctrl+Shift+B
Ignore/Unignore Packet Ctrl+D
Ignore All Displayed Ctrl+Shift+D
Unignore All Displayed Ctrl+Alt+D
Set/Unset Time Reference Ctrl+T
Unset All Time References Ctrl+Alt+T
Next Time Reference Ctrl+Alt+N
Previous Time Reference Ctrl+Alt+B
Time Shift... Ctrl+Shift+T
Packet Comment... Ctrl+Alt+C
Delete All Packet Comments
Configuration Profiles... Ctrl+Shift+A
Preferences... Ctrl+Shift+P
Display filter
Find

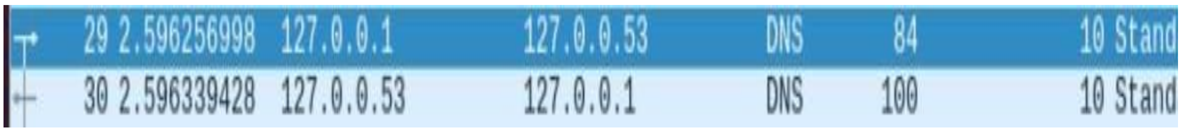
```
wireshark_any_20211110085550_ZvQsgx.pcapng    Packets: 4647 · Displayed: 4647 (100.0%)
```

- In Find Packet menu bar, select the **String** option in the **display filter drop-down** menuand enter the name of the website in the next box and click on find.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Packet list
Narrow & Wide
Case sensitive
Display filter
Hex value
String
Regular Expression
git
Find

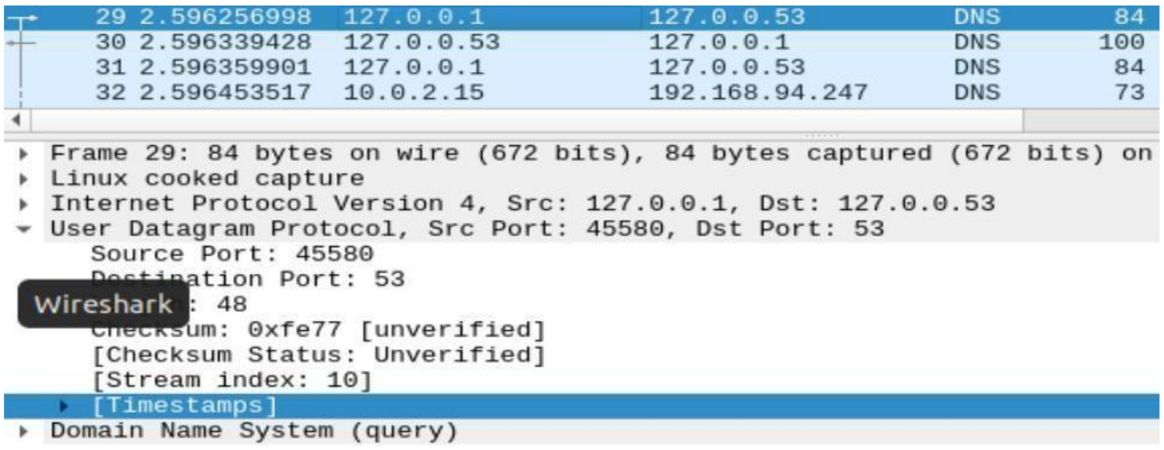
```
No.  Time           Source          Destination
4641 60.854959516   103.21.58.165   10.0.2.15                  → 39054 [FIN, ACK]
4642 60.855312134   10.0.2.15       103.21.58.165               ication Data
4643 60.855652613   103.21.58.165   10.0.2.15                  → 39054 [ACK] Seq=
4644 60.855674561   10.0.2.15       103.21.58.165               ication Data
4645 60.855702284   10.0.2.15       103.21.58.165   TCP     56 39054 → 443 [FIN, ACK]
4646 60.855813714   103.21.58.165   10.0.2.15       TCP     62 443 → 39054 [ACK] Seq=
4647 60.855813847   103.21.58.165   10.0.2.15       TCP     62 443 → 39054 [ACK] Seq=

▸ Frame 1: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0
▸ Linux cooked capture
▸ Address Resolution Protocol (request)

0000  00 04 00 01 00 06 08 00  27 99 dd cf 00 00 08 06
0010  00 01 08 00 06 04 00 01  08 00 27 99 dd cf 0a 00
0020  02 0f 00 00 00 00 00 00  0a 00 02 02

wireshark_any_20211110085550_ZvQsgx.pcapng    Packets: 4647 · Displayed: 4647 (100.0%)
```

- The arrow indicating towards the packet is the **request packet**, and the arrowcoming out from the packet is the **response packet**.



```
29 2.596256998  127.0.0.1    127.0.0.53   DNS   84    10 Stand
30 2.596339428  127.0.0.53   127.0.0.1    DNS   100   10 Stand
```

Click on any request or response DNS packet and exam



```
29 2.596256998   127.0.0.1      127.0.0.53       DNS    84
30 2.596339428   127.0.0.53     127.0.0.1        DNS   100
31 2.596359901   127.0.0.1      127.0.0.53       DNS    84
32 2.596453517   10.0.2.15      192.168.94.247   DNS    73

▸ Frame 29: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53
▾ User Datagram Protocol, Src Port: 45580, Dst Port: 53
    Source Port: 45580
    Destination Port: 53
    Wireshark : 48
    Checksum: 0xfe77 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 10]
  ▸ [Timestamps]
▸ Domain Name System (query)
```

Using WIRESHARK analyse three way handshaking connection establishment, data transferand connection termination in client server communication using TCP.

TCP Program Execution Steps:Steps :

● Open Terminal using **CTRL+ALT+T** or right-click and select **Open in Terminal**. ● Once opened, type the following command in the terminal:

cd Desktop/

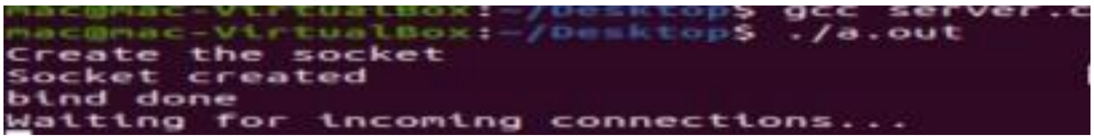
[Before going through the next sequence of commands, make sure that both the TCP programs are located on the desktop]

- Before executing the program, open **Wireshark** and double-click on **any-interface** for packet capture process to start.

- To compile the C program, type the following command in the same order as follows in the terminal:

gcc server.c

To execute the program type, object code file name in the terminal as follows : **./a.out**



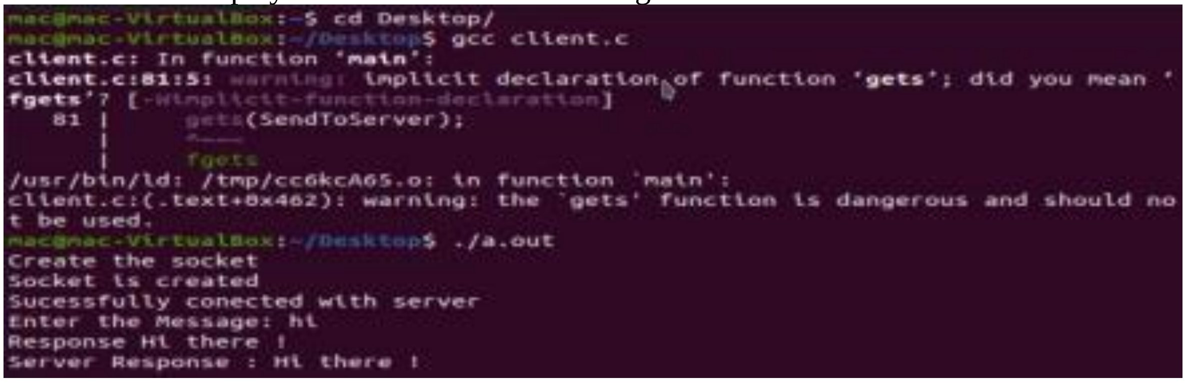
```
mac@mac-VirtualBox:~/Desktop$ gcc server.c
mac@mac-VirtualBox:~/Desktop$ ./a.out
Create the socket
Socket created
bind done
Waiting for incoming connections...
```

- Open another terminal and change its path to Desktop using Cd command and run
  the c code as shown below:

  cd Desktop/gcc client.c

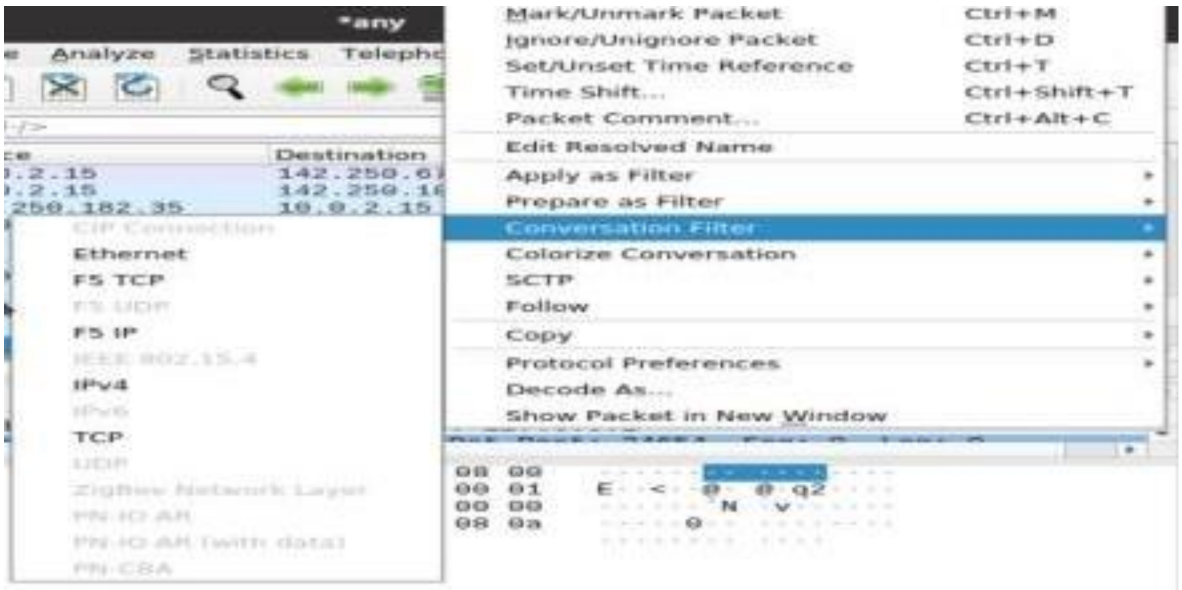
  ./a.out

- Once the client program is executed, it will ask you to enter a message type **hi without any space** and hit enter.
- The server will replay back with **hi there!** Message.



```
mac@mac-VirtualBox:~$ cd Desktop/
mac@mac-VirtualBox:~/Desktop$ gcc client.c
client.c: In function 'main':
client.c:81:5: warning: implicit declaration of function 'gets'; did you mean '
fgets'? [-Wimplicit-function-declaration]
   81 |     gets(SendToServer);
      |     ^~~~
      |     fgets
/usr/bin/ld: /tmp/cc6kcA65.o: in function `main':
client.c:(.text+0x462): warning: the `gets' function is dangerous and should no
t be used.
mac@mac-VirtualBox:~/Desktop$ ./a.out
Create the socket
Socket is created
Sucessfully conected with server
Enter the Message: hi
Response Hi there !
Server Response : Hi there !
```

- Once the program execution is completed, stop the capturing process in Wireshark.
- In Wireshark, check for the packets that has the source and destination address as
  127.0.0.1 and select any one of these packets, right-click and hover on conversation filter and



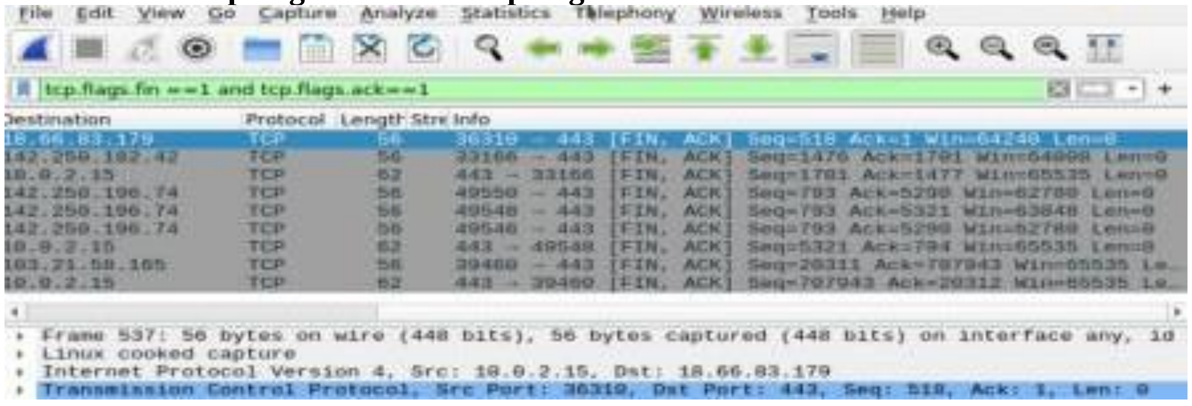
select TCP.

- Once done analyze the TCP Packets.

2. Capturing TCP Packets withbrowser: Steps:

- Open Wireshark and double-click on **any-interface** to start the packet capture process.
- Open the browser and enter any website's fully qualified domain name in the browser address bar and hit enter.
- After the site is fully loaded, stop the capturing process, in Wireshark.
- Type the following in, apply a filter column and hit-enter :

tcp.flags.fin==1 and tcp.flags.ack ==1

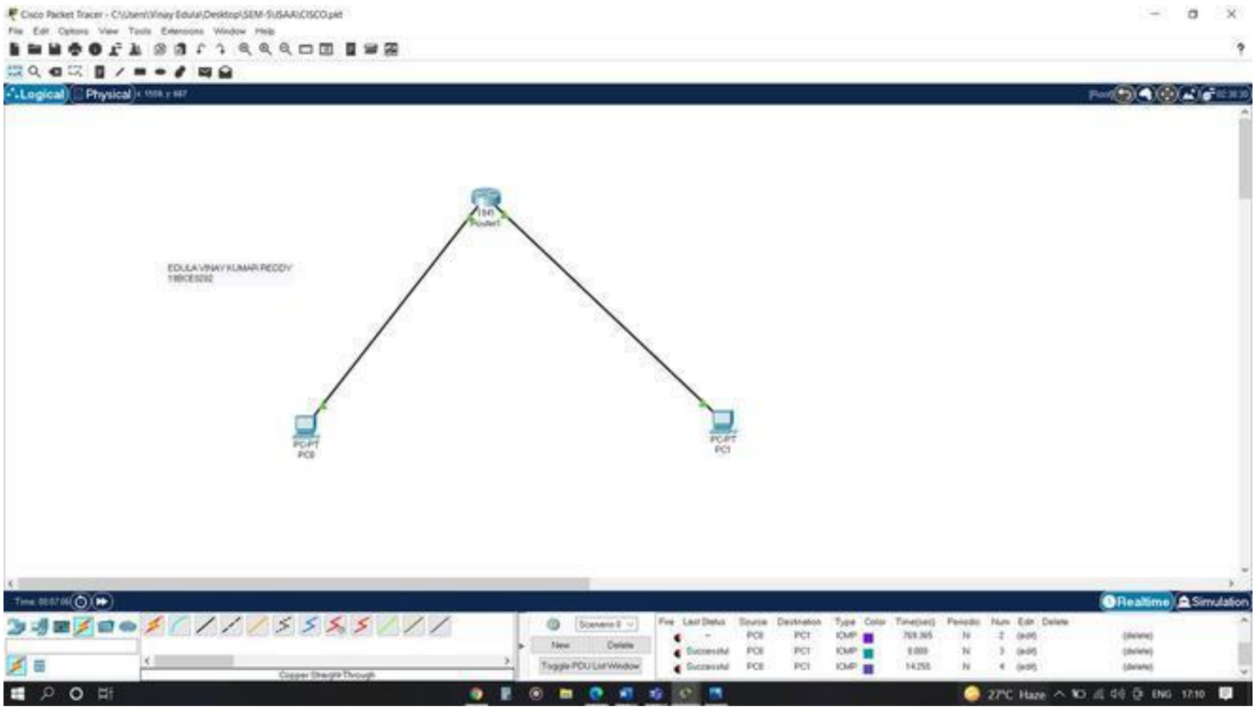

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.flags.fin ==1 and tcp.flags.ack==1

Destination          Protocol  Length Str Info
18.66.83.179         TCP       56     36310 → 443 [FIN, ACK] Seq=518 Ack=1 Win=64240 Len=0
142.250.182.42       TCP       56     33166 → 443 [FIN, ACK] Seq=1476 Ack=1701 Win=64000 Len=0
10.0.2.15            TCP       62     443 → 33166 [FIN, ACK] Seq=1701 Ack=1477 Win=65535 Len=0
142.250.196.74       TCP       56     49550 → 443 [FIN, ACK] Seq=793 Ack=5290 Win=62780 Len=0
142.250.196.74       TCP       56     49548 → 443 [FIN, ACK] Seq=793 Ack=5321 Win=63840 Len=0
142.250.196.74       TCP       56     49546 → 443 [FIN, ACK] Seq=793 Ack=5290 Win=62780 Len=0
10.0.2.15            TCP       62     443 → 49548 [FIN, ACK] Seq=5321 Ack=794 Win=65535 Len=0
103.21.58.165        TCP       56     39460 → 443 [FIN, ACK] Seq=20311 Ack=707943 Win=65535 Le…
10.0.2.15            TCP       62     443 → 39460 [FIN, ACK] Seq=707943 Ack=20312 Win=65535 Le…

▸ Frame 537: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface any, id
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 18.66.83.179
▸ Transmission Control Protocol, Src Port: 36310, Dst Port: 443, Seq: 518, Ack: 1, Len: 0
```

- Select any one of these listed packets, right-click and hover onconversation filter and select TCP.
- Once done analyze the TCP Packets.

Program No: 2

Demonstrate the router and pc configuration using packet tracer and verify it using real-time
PDU transmission.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	–	PC0	PC1	ICMP	🟪	769.365	N	2	(edit)	(delete)
	Successful	PC0	PC1	ICMP	🟩	8.000	N	3	(edit)	(delete)
	Successful	PC0	PC1	ICMP	🟪	14.255	N	4	(edit)	(delete)

Procedure:

Step-1(Configuring Router1):
1. Select the router and Open CLI.
2. Press ENTER to start configuring Router1.
3. Type enable to activate the privileged mode.
4. Type config t(configure terminal) to access the configuration menu.
5. Configure interfaces of Router1:

Router1 Command Line Interface:

```
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#ip address 192.168.10.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up

Router(config-if)#interface FastEthernet0/1
Router(config-if)#ip address 192.168.20.1 255.255.255.0
Router(config-if)#no shutdown
```

Step-2(Configuring PCs):
1. Assign IP Addresses to every PC in the network.
2. Select the PC, Go to the desktop and select IP Configuration and assign an IP address, Default gateway, Subnet Mask
3. Assign the default gateway of PC0 as 192.168.10.1.
4. Assign the default gateway of PC1 as 192.168.20.1.

Step-3(Connecting PCs with Router):
1. Connect FastEthernet0 port of PC0 with FastEthernet0/0 port of Router1 using a copper straight-through cable.
2. Connect FastEthernet0 port of PC1 with FastEthernet0/1 port of Router1 using a copper straight-through cable.

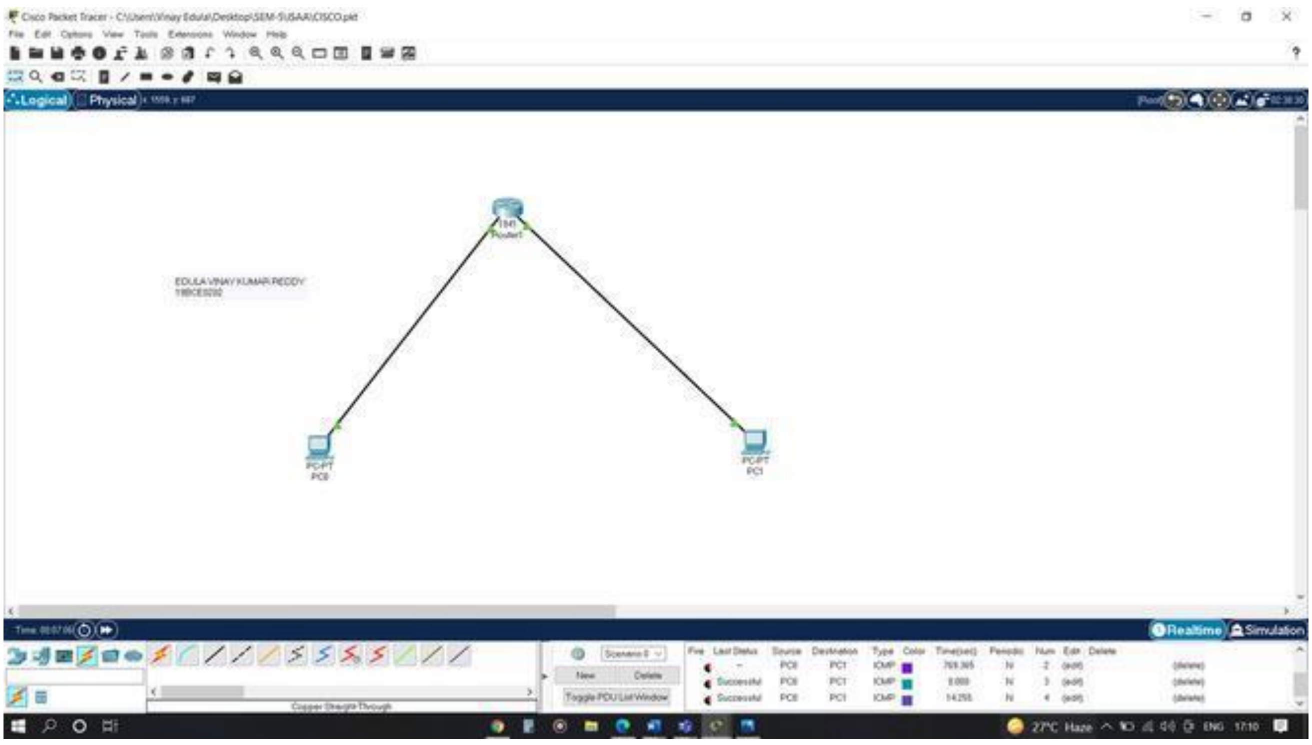
Router Configuration Table:

Device Name	IP address FastEthernet0/0	Subnet Mask	IP Address FastEthernet0/1	Subnet Mask
Router1	192.168.10.1	255.255.255.0	192.168.20.1	255.255.255.0

PC Configuration Table:

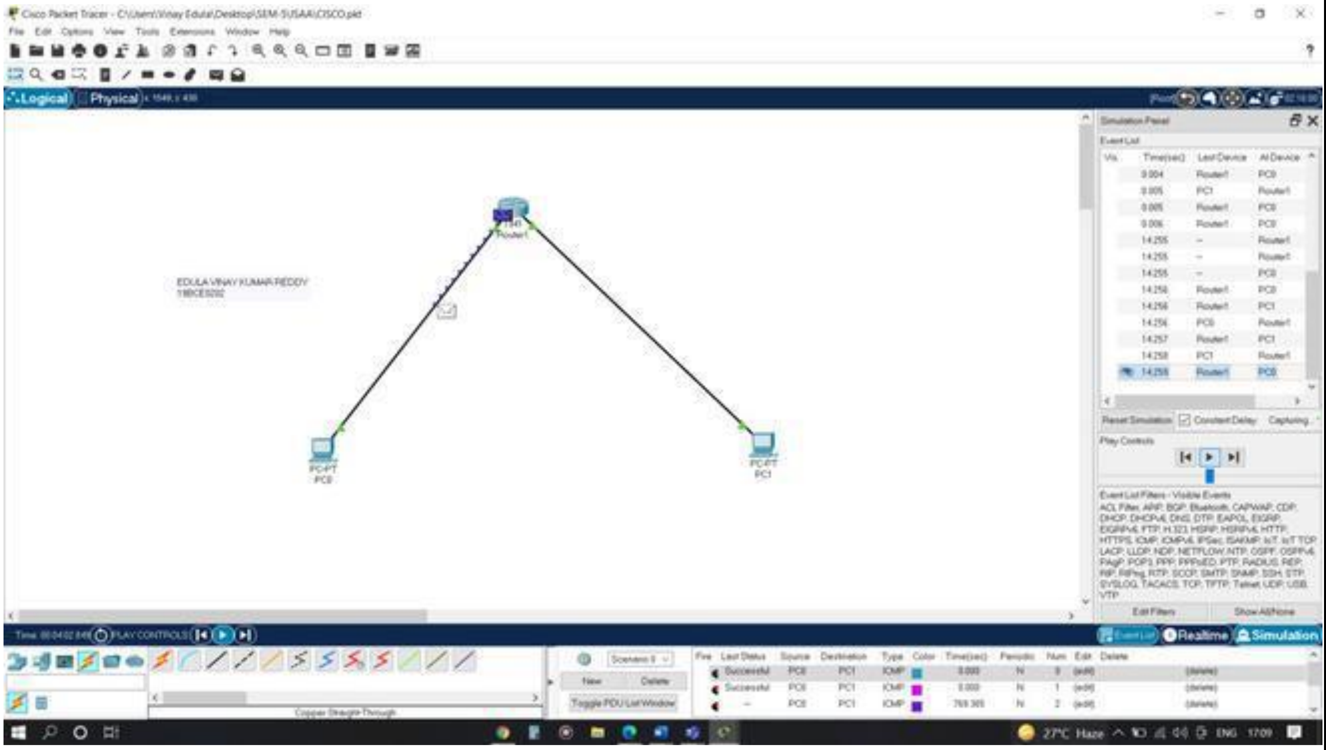
Device Name	IP address	Subnet Mask	Gateway
PC 0	192.168.10.2	255.255.255.0	192.168.10.1
PC 1	192.168.20.2	255.255.255.0	192.168.20.1

Designed Network topology:



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	--	PC0	PC1	ICMP	🟪	769.365	N	2	(edit)	(delete)
●	Successful	PC0	PC1	ICMP	🟦	8.000	N	3	(edit)	(delete)
●	Successful	PC0	PC1	ICMP	🟪	14.255	N	4	(edit)	(delete)

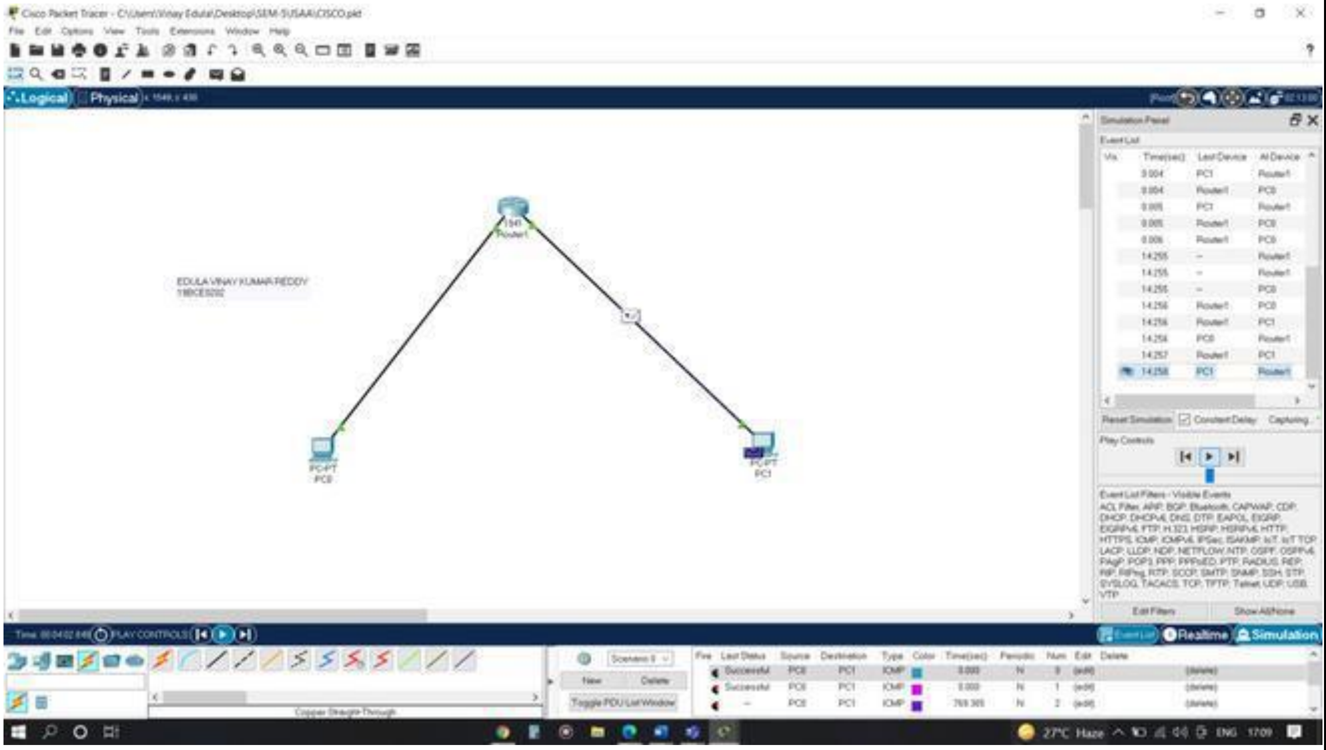
Simulation of Designed Network Topology:
Sending a PDU From PC0 to PC1:



Vis.	Time(sec)	Last Device	At Device
	0.004	Router1	PC0
	0.005	PC1	Router1
	0.005	Router1	PC0
	0.006	Router1	PC0
	14.255	--	Router1
	14.255	--	Router1
	14.255	--	PC0
	14.256	Router1	PC0
	14.256	Router1	PC1
	14.256	PC0	Router1
	14.257	Router1	PC1
	14.258	PC1	Router1
	14.258	Router1	PC0

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC1	ICMP	🟦	0.000	N	0	(edit)	(delete)
●	Successful	PC0	PC1	ICMP	🟪	0.000	N	1	(edit)	(delete)
●	--	PC0	PC1	ICMP	🟪	769.365	N	2	(edit)	(delete)

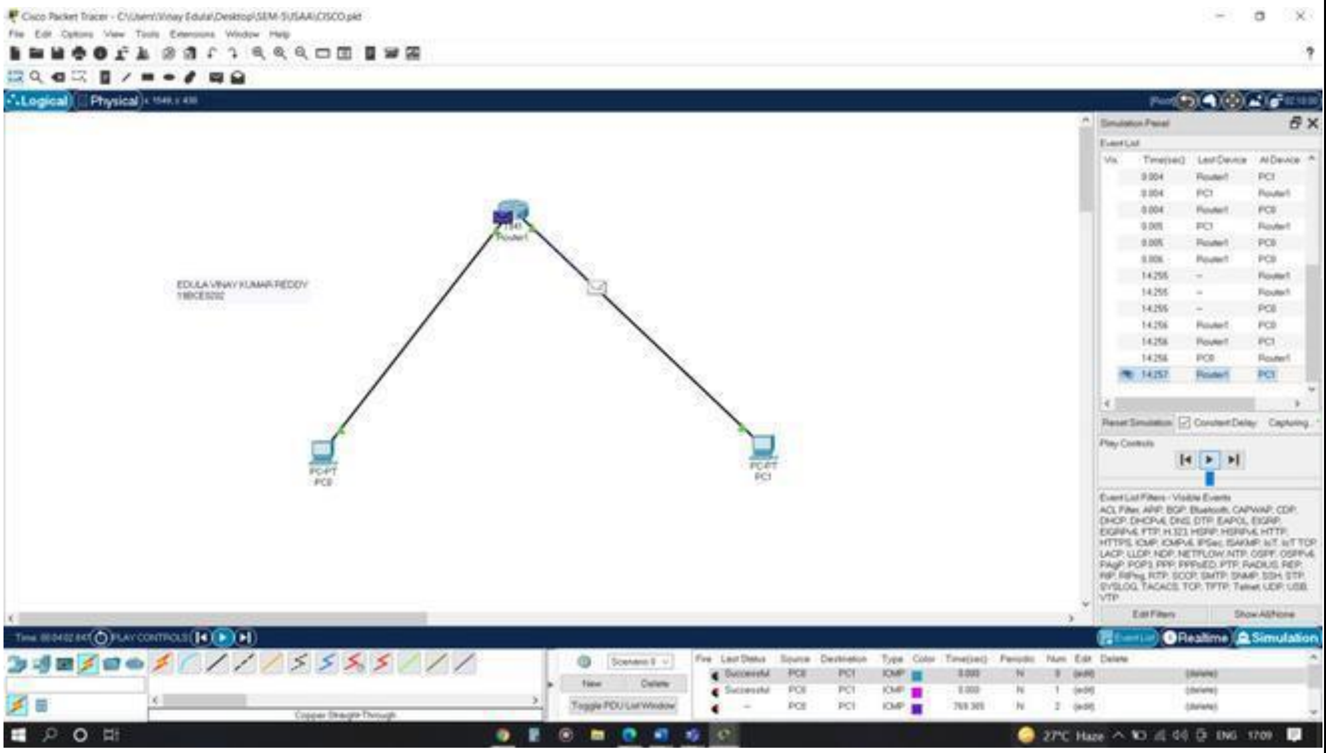


Vis.	Time(sec)	Last Device	At Device
	0.004	PC1	Router1
	0.004	Router1	PC0
	0.005	PC1	Router1
	0.005	Router1	PC0
	0.006	Router1	PC0
	14.255	--	Router1
	14.255	--	Router1
	14.255	--	PC0
	14.256	Router1	PC0
	14.256	Router1	PC1
	14.256	PC0	Router1
	14.257	Router1	PC1
	14.258	PC1	Router1

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

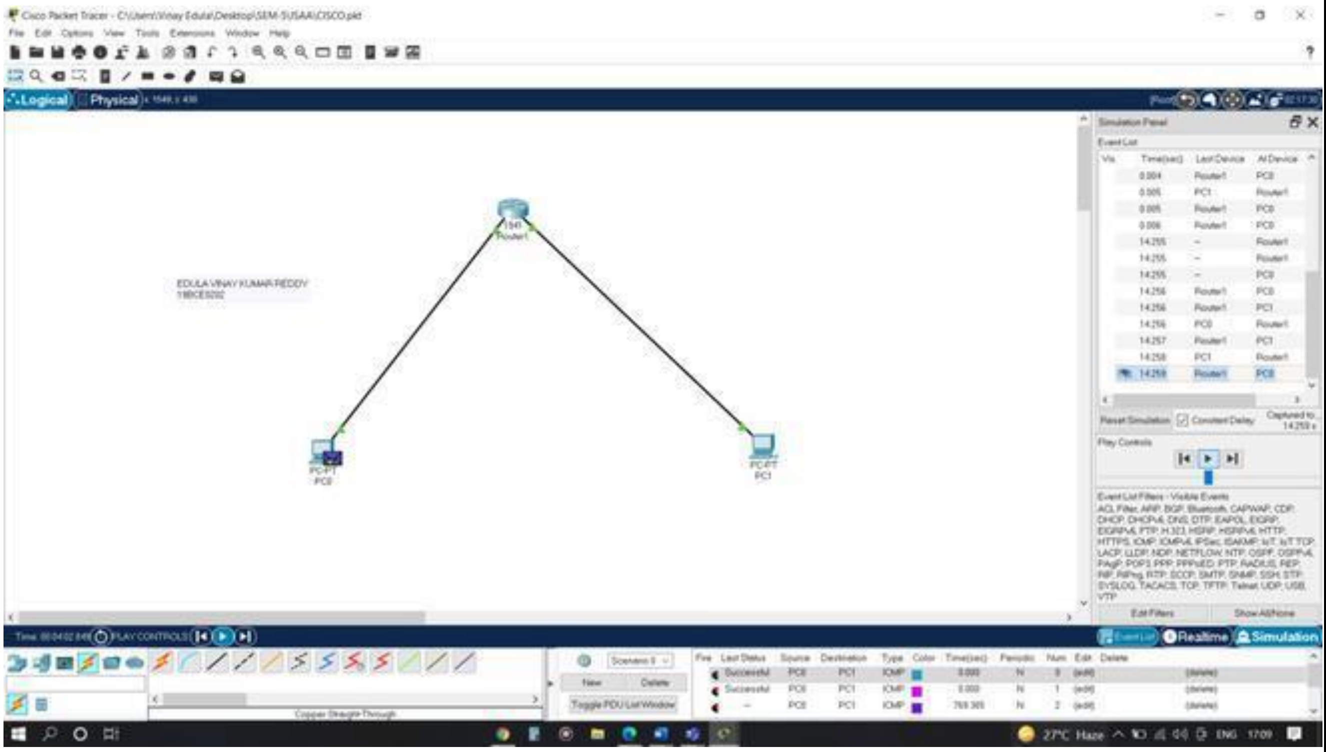
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC1	ICMP	🟦	0.000	N	0	(edit)	(delete)
●	Successful	PC0	PC1	ICMP	🟪	0.000	N	1	(edit)	(delete)
●	--	PC0	PC1	ICMP	🟪	769.365	N	2	(edit)	(delete)

Acknowledgment From PC1 to PC0:



```
Event List
Vis.     Time(sec)   Last Device   At Device
         0.004       Router1       PC1
         0.004       PC1           Router1
         0.004       Router1       PC0
         0.005       PC1           Router1
         0.005       Router1       PC0
         0.006       Router1       PC0
         14.255      --            Router1
         14.255      --            Router1
         14.255      --            PC0
         14.256      Router1       PC0
         14.256      Router1       PC1
         14.256      PC0           Router1
         14.257      Router1       PC1

Reset Simulation   Constant Delay   Capturing
```

```
Fire  Last Status   Source  Destination  Type  Color  Time(sec)  Periodic  Num  Edit    Delete
      Successful    PC0     PC1          ICMP  🟦     0.000      N         0    (edit)  (delete)
      Successful    PC0     PC1          ICMP  🟪     0.000      N         1    (edit)  (delete)
      --            PC0     PC1          ICMP  🟪     769.365    N         2    (edit)  (delete)
```

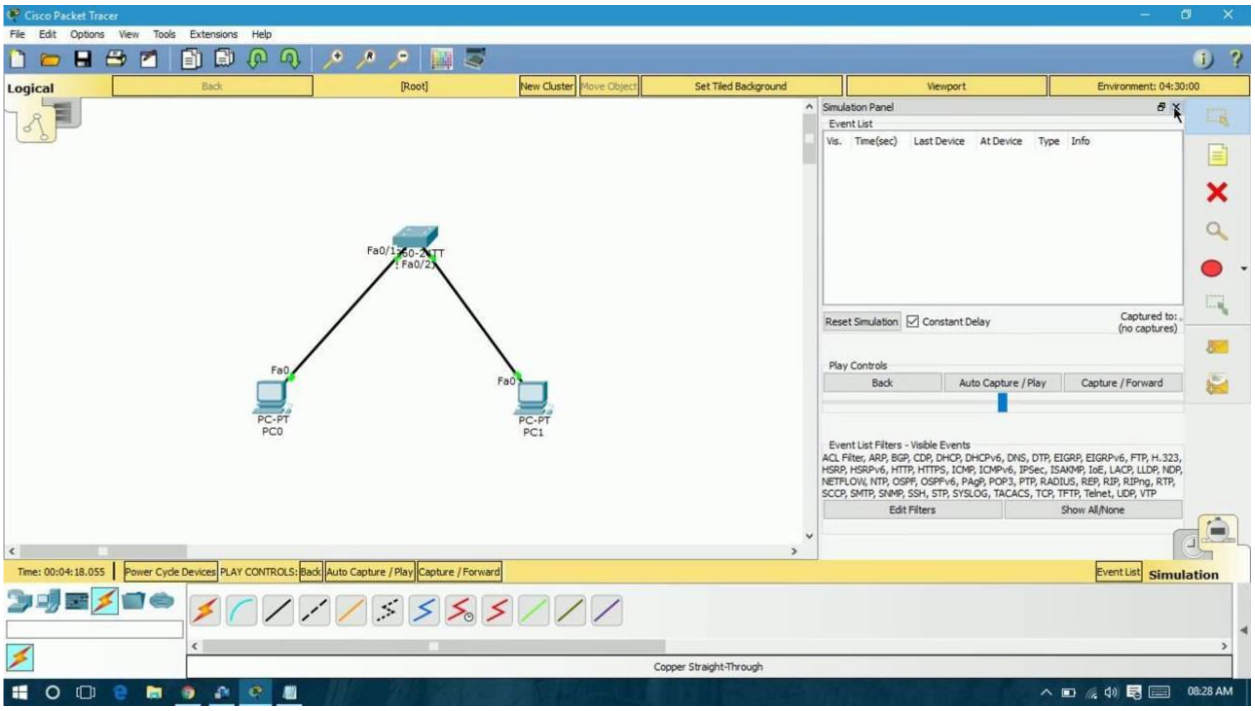


```
Event List
Vis.     Time(sec)   Last Device   At Device
         0.004       Router1       PC0
         0.005       PC1           Router1
         0.005       Router1       PC0
         0.006       Router1       PC0
         14.255      --            Router1
         14.255      --            Router1
         14.255      --            PC0
         14.256      Router1       PC0
         14.256      Router1       PC1
         14.256      PC0           Router1
         14.257      Router1       PC1
         14.258      PC1           Router1
         14.259      Router1       PC0

Reset Simulation   Constant Delay   Captured to: 14.259 s
```

```
Fire  Last Status   Source  Destination  Type  Color  Time(sec)  Periodic  Num  Edit    Delete
      Successful    PC0     PC1          ICMP  🟦     0.000      N         0    (edit)  (delete)
      Successful    PC0     PC1          ICMP  🟪     0.000      N         1    (edit)  (delete)
      --            PC0     PC1          ICMP  🟪     769.365    N         2    (edit)  (delete)
```

Program No: 3

Demonstrate the switch configuration using packet tracer and verify using realtime PDU
Transmission


Cisco Packet Tracer
File Edit Options View Tools Extensions Help
Logical
Back
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Environment: 04:30:00
Fa0/1
Fa0/2
Fa0
Fa0
PC-PT
PC0
PC-PT
PC1
Simulation Panel
Event List
Vis. Time(sec) Last Device At Device Type Info
Reset Simulation
Constant Delay
Captured to: (no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ACL Filter, ARP, BGP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoE, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All/None
Time: 00:04:18.055
Power Cycle Devices
PLAY CONTROLS: Back Auto Capture / Play Capture / Forward
Event List
Simulation
Copper Straight-Through
08:28 AM

Cisco Switch IOS Commands Enter global configuration mode.
S1# configure terminal Enter interface configuration mode.
S1(config)# interface vlan 99 Configure the management interface IP address.
S1(config-if)# ip address 172.17.99.11 255.255.255.0
Enable the management interface.

S1(config-if)# no shutdown Return to the privileged EXEC mode.
S1(config-if)# end Save the running configuration file to the startup configuration file.
S1# copy running-config startup-config

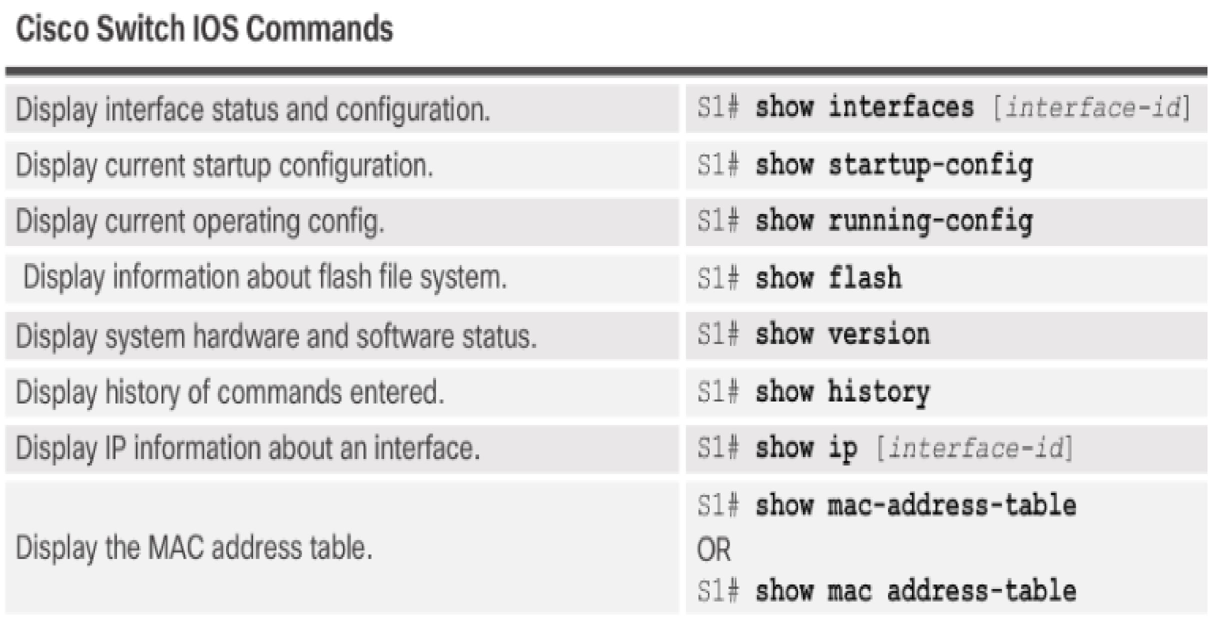
Examine the startup configuration file. To view the contents of the startup configuration f ile, issue the show startup-config command in privileged EXEC mode.
Switch#show startup-config startup-config is not present

Let's make one configuration change to the switch and then save it. Type the following commands:

Switch#configure terminal Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#hostname S1 S1(config)#exit S1#

To save the contents of the running configuration file to non-volatile RAM (NVRAM), issue the
the command copy running-config startup-config.

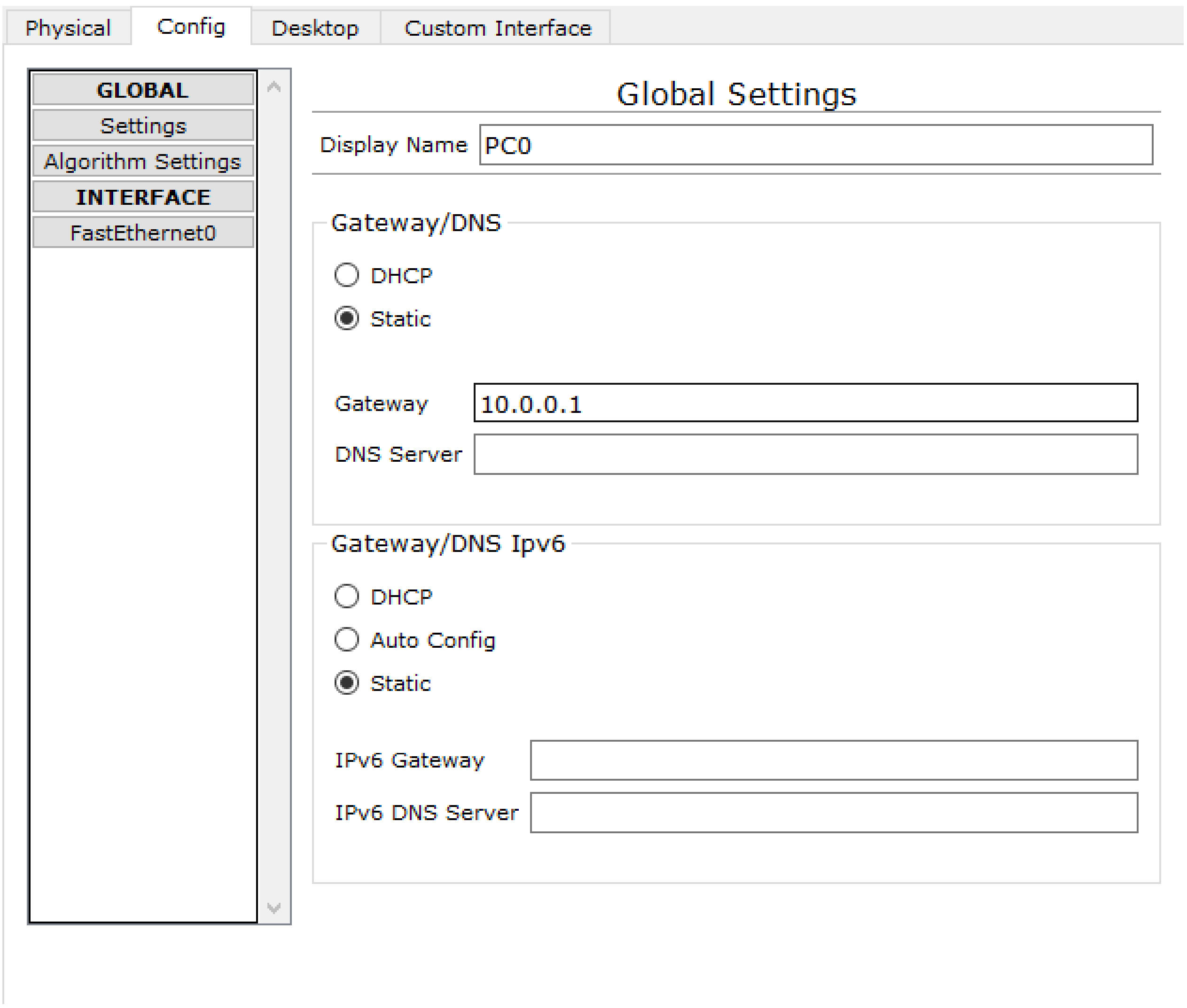
Switch#copy running-config startup-config Destination filename [startup-config]? (enter)
Building configuration... [OK]



Cisco Switch IOS Commands

Display interface status and configuration.	S1# **show interfaces** [*interface-id*]
Display current startup configuration.	S1# **show startup-config**
Display current operating config.	S1# **show running-config**
Display information about flash file system.	S1# **show flash**
Display system hardware and software status.	S1# **show version**
Display history of commands entered.	S1# **show history**
Display IP information about an interface.	S1# **show ip** [*interface-id*]
Display the MAC address table.	S1# **show mac-address-table** OR S1# **show mac address-table**

Configure an IP Address to PC
There are following steps involved to configure an IP Address to PC:
Step1: Open the Cisco Packet Tracer.
Step2: Drag and drop PC from the bottom of the interface into the middle of the working area.
Step3: Click on PC ->Config Gateway like 10.0.0.1



Global Settings

Display Name: PC0

Gateway/DNS

DHCP [✓ Static]

Gateway: 10.0.0.1

DNS Server:

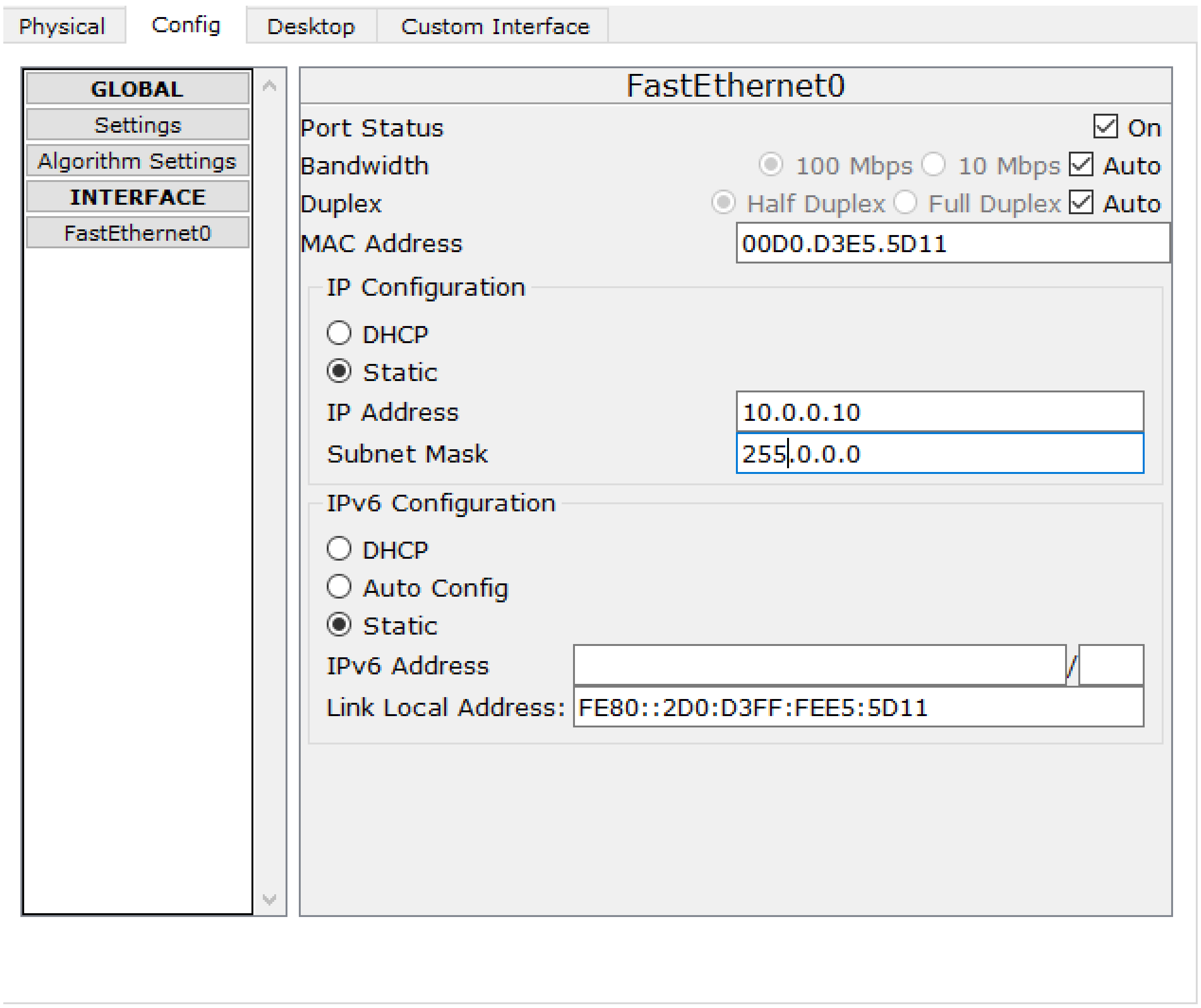
Gateway/DNS Ipv6

DHCP / Auto Config [✓ Static]

IPv6 Gateway:

IPv6 DNS Server:

Click on FastEthernet to assign an IP address and subnetmask to the PC, and close PC window. In our case IP is 10.0.0.10 and subnet mask is 255.0.0.0.0,



FastEthernet0

Port Status [✓ On]
Bandwidth ● 100 Mbps ○ 10 Mbps [✓ Auto]
Duplex ● Half Duplex ○ Full Duplex [✓ Auto]
MAC Address 00D0.D3E5.5D11

IP Configuration
○ DHCP
● Static
IP Address 10.0.0.10
Subnet Mask 255.0.0.0

IPv6 Configuration
○ DHCP
○ Auto Config
● Static
IPv6 Address /
Link Local Address: FE80::2D0:D3FF:FEE5:5D11

Basics of NS3 simulation
Introduction In this lab, we will be using the Network Simulator, NS3, available from www.nsnam.org. NS3 is a powerful program, however we will only be looking at some basic features. NS3 simulations are built in C++.

Installation:
Following are the basic steps which must be followed for installing NS3
1. Install prerequisite packages
2. Download ns3 codes
3. Build ns3
4. Validate ns3

Prerequisite packages for Linux are as follows:
1. Minimal requirements for Python: gcc g++ python
2. Debugging and GNU Scientific Library (GSL) support: gdbpython-dev
3. valgrind gsl-bin libgsl0-dev libgsl0ldbl Network Simulation Cradle (nsc): flex bison Reading pcap packet traces: tcpdump
4. Database support for statistics framework: sqlite sqlite3

5. Xml-based version of the config store: libxml2
6. A GTK-based configuration system: libgtk2.0-0
7. Experimental with virtual machines and ns-3: vtun lxc

Detail steps are as follows:

1. sudo apt-get update / dnf update
2. sudo apt-get upgrade / dnf upgrade
3. Once ubuntu/fedora is installed run following command opening the terminal(ctrl+alt+T) window.
4. To install prerequisites dependancy packages- Type the following command in terminal window.

sudo apt-get/ dnf install gcc g++ python python-dev mercurial bzr gdb valgrind gsl-bin libgsl0-dev
libgsl0ldbl flex bison tcpdump sqlite sqlite3 libsqlite3-dev libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-
dev uncrustify doxygen graphviz imagemagick texlive texlive-latex-extra texlive-generic-extra texlive-generic-recommended texinfo dia texlive texlive-latex-extra texlive-extra-utils texlivegeneric-
recommended texi2html python-pygraphviz python-kiwi python-pygoocanvas libgoocanvasdev
python-pygccxml

5. After downloading NS3 on the drive, extract all the files in the NS3 folder, which you have created.
6. Then you can find build.py along with other files in NS3 folder.
   Then to build the examples in ns-3 run :
   ./build.py --enable-examples –enable-tests
   If the build is successful then it will give output
   "Build finished successfully".
7. Now run the following command on the terminal window,to configure with waf(build tool)
   ./waf -d debug --enable-examples --enable-tests configure

To build with waf(optional)
./waf
8.To test everything allright run the following command on the terminal window,
./test.py
If the tests are ok the installation is done
9. after installing ns3 and testing it run some programs first to be ns3 user:
make sure you are in directory where waf script is available then run

Program in NS3 to connect two nodes

Node

Because in any network simulation, we will need nodes. So ns-3 comes with NodeContainer that you
can use to manage all the nodes (Add, Create, Iterate, etc.).

// Create two nodes to hold.
NodeContainer nodes;
nodes.Create (2);

Channel and NetDevice

In the real world, they correspond to network cables (or wireless media) and peripheral cards (NIC).
Typically these two things are intimately tied together. In the first example, we are usingPointToPointHelper that wraps the Channel and NetDevice.

// Channel: PointToPoint, a direct link with `DataRate` and `Delay` specified.
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

Then we need to install the devices. The internal of Install is actually more complicated, but for now,
let's just skip the magic behind the scene.

// NetDevice: installed onto the channel
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);

Protocols

Internet and IPv4. Since Internet is the current largest network to study, ns-3 has a particular focus on
it. The InternetStackHelper will install an Internet Stack (TCP, UDP, IP, etc.) on each of the nodes in
the node container.

// Protocol Stack: Internet Stack
InternetStackHelper stack;
stack.Install (nodes);

To assign IP addresses, use a helper and set the base. The low level ns-3 system actually remembers
all of the IP addresses allocated and will generate a fatal error if you accidentally cause the same
address to be generated twice.

// Since IP Address assignment is so common, the helper does the dirty work!
// You only need to set the base.
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
// Assign the address to devices we created above
Ipv4InterfaceContainer interfaces = address.Assign (devices);

Applications

Every application needs to have Start and Stop function so that the simulator knows how to schedule it. Other functions are application-specific. We will use UdpEchoServer and UdpEchoClientfor now.

```
// Application layer: UDP Echo Server and Client
// 1, Server:
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
// 2, Client:
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
```

Simulation

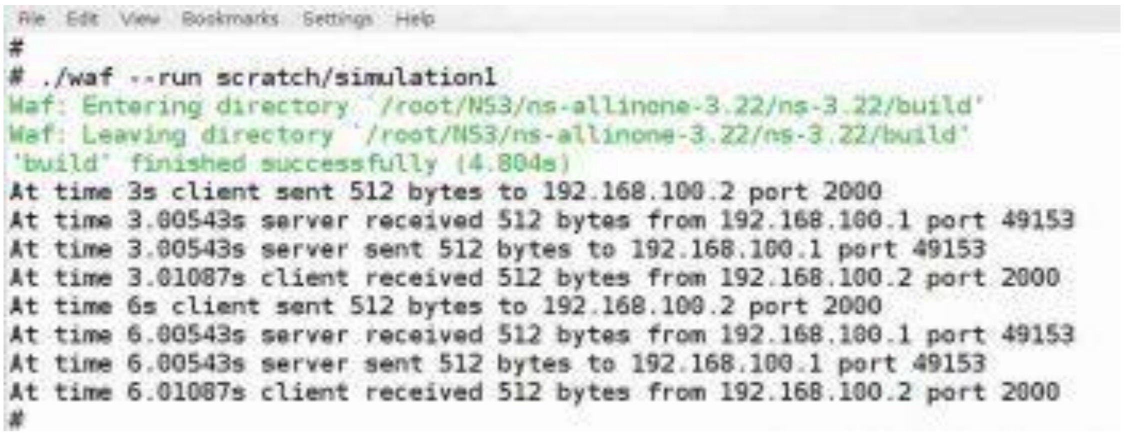
```
// Start Simulation
Simulator::Run ();
Simulator::Destroy ();
return 0;
```

Program No: 4

Simulate a Full duplex connection in an wired network using NS3.

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include   "ns3/point-to-point-module.h"
#include   "ns3/applications-module.h"
#include "ns3/netanim-module.h"
// Default Network Topology
//
// 10.1.1.0
// n0     n1
// point-to-point
//
using namespace ns3;
NS_LOG_COMPONENT_DEFINE    ("FirstScriptExample");
int
main (int argc, char *argv[])
{
//CommandLine cmd (__FILE__);
//cmd.Parse (argc, argv);
bool tracing = true;
Time::SetResolution (Time::NS);
LogComponentEnable   ("UdpEchoClientApplication",   LOG_LEVEL_INFO);
LogComponentEnable   ("UdpEchoServerApplication",   LOG_LEVEL_INFO);
NodeContainer nodes;
nodes.Create (2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("10Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
InternetStackHelper stack;
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign   (devices);
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (4));
echoClient.SetAttribute  ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
AnimationInterface anim("1.xml");
anim.SetConstantPosition(nodes.Get (0), 10.0, 10.0);
anim.SetConstantPosition(nodes.Get (1), 20.0, 30.0);
if (tracing==true)
{
pointToPoint.EnablePcapAll("p2p");
}

Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```

OUTPUT:



```
#
# ./waf --run scratch/simulation1
Waf: Entering directory `/root/NS3/ns-allinone-3.22/ns-3.22/build'
Waf: Leaving directory `/root/NS3/ns-allinone-3.22/ns-3.22/build'
'build' finished successfully (4.804s)
At time 3s client sent 512 bytes to 192.168.100.2 port 2000
At time 3.00543s server received 512 bytes from 192.168.100.1 port 49153
At time 3.00543s server sent 512 bytes to 192.168.100.1 port 49153
At time 3.01087s client received 512 bytes from 192.168.100.2 port 2000
At time 6s client sent 512 bytes to 192.168.100.2 port 2000
At time 6.00543s server received 512 bytes from 192.168.100.1 port 49153
At time 6.00543s server sent 512 bytes to 192.168.100.1 port 49153
At time 6.01087s client received 512 bytes from 192.168.100.2 port 2000
#
```

Program No: 5
Simulate a simple Wireless UDP application using NS3.

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma -module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
// Default Network Topology
//
// Wifi 10.1.3.0
// AP
// * * * *
// | | | | 10.1.1.0
// n5 n6 n7 n0 -------------- n1 n2 n3 n4
// point-to-point | | | |
// ================
// LAN 10.1.2.0
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");
int
main (int argc, char *argv[])
{
bool verbose = true;
uint32_t nCsma = 3;
uint32_t nWifi = 3;
bool tracing = false;
CommandLine cmd (__FILE__);
cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
cmd.AddValue ("tracing", "Enable pcap tracing", tracing);
cmd.Parse (argc,argv);
// The underlying restriction of 18 is due to the grid position
// allocator's configuration; the grid layout will exceed the
// bounding box if more than 18 nodes are provided.
if (nWifi > 18)
{
std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding box" <<
std::endl;
return 1;
}
if (verbose)
{
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
}
NodeContainer p2pNodes;
p2pNodes.Create (2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);
NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsma);
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);
NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy;
phy.SetChannel (channel.Create ());
WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
WifiHelper wifi;
NetDeviceContainer staDevices;
mac.SetType ("ns3::StaWifiMac",
"Ssid", SsidValue (ssid),
"ActiveProbing", BooleanValue (false));
staDevices = wifi.Install (phy, mac, wifiStaNodes);
NetDeviceContainer apDevices;
mac.SetType ("ns3::ApWifiMac",
"Ssid", SsidValue (ssid));
apDevices = wifi.Install (phy, mac, wifiApNode);
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
"MinX", DoubleValue (0.0),
"MinY", DoubleValue (0.0),
"DeltaX", DoubleValue (5.0),
"DeltaY", DoubleValue (10.0),
"GridWidth", UintegerValue (3),
"LayoutType", StringValue ("RowFirst"));
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
"Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);
InternetStackHelper stack;
```

```
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps =
echoClient.Install (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
Simulator::Stop (Seconds (10.0));
if (tracing)
{
phy.SetPcapDataLinkType (WifiPhyHelper::DLT_IEEE802_11_RADIO);
pointToPoint.EnablePcapAll ("third");
phy.EnablePcap ("third", apDevices.Get (0));
csma.EnablePcap ("third", csmaDevices.Get (0), true);
}
Simulator::Run ();
Simulator::Destroy ();
return 0;}
```

Output:

```
cp examples/third.cc scratch/mythird.cc
./waf
./waf --run scratch/mythird

Waf: Entering directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'
Waf: Leaving directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'
'build' finished successfully (0.407s)
Sent 1024 bytes to 10.1.2.4
Received 1024 bytes from 10.1.3.3
Received 1024 bytes from 10.1.2.4
third-0-0.pcap third-0-1.pcap third-1-0.pcap third-1-1.pcap
```

You should see some wifi-looking contents you haven’t seen here before:

```
reading from file third-0-1.pcap, link-type IEEE802_11 (802.11)
0.000025 Beacon () [6.0* 9.0 12.0 18.0 24.0 36.0 48.0 54.0 Mbit] IBSS
0.000263 Assoc Request () [6.0 9.0 12.0 18.0 24.0 36.0 48.0 54.0 Mbit]
0.000279 Acknowledgment RA:00:00:00:00:00:07
0.000357 Assoc Response AID(0) :: Succesful
0.000501 Acknowledgment RA:00:00:00:00:00:0a
0.000748 Assoc Request () [6.0 9.0 12.0 18.0 24.0 36.0 48.0 54.0 Mbit]
0.000764 Acknowledgment RA:00:00:00:00:00:08
0.000842 Assoc Response AID(0) :: Succesful
0.000986 Acknowledgment RA:00:00:00:00:00:0a
0.001242 Assoc Request () [6.0 9.0 12.0 18.0 24.0 36.0 48.0 54.0 Mbit]
0.001258 Acknowledgment RA:00:00:00:00:00:09
0.001336 Assoc Response AID(0) :: Succesful
0.001480 Acknowledgment RA:00:00:00:00:00:0a
2.000112 arp who-has 10.1.3.4 (ff:ff:ff:ff:ff:ff) tell 10.1.3.3
2.000128 Acknowledgment RA:00:00:00:00:00:09
2.000206 arp who-has 10.1.3.4 (ff:ff:ff:ff:ff:ff) tell 10.1.3.3
2.000487 arp reply 10.1.3.4 is-at 00:00:00:00:00:0a
2.000659 Acknowledgment RA:00:00:00:00:00:0a
2.002169 IP 10.1.3.3.49153 > 10.1.2.4.9: UDP, length 1024
2.002185 Acknowledgment RA:00:00:00:00:00:09
2.009771 arp who-has 10.1.3.3 (ff:ff:ff:ff:ff:ff) tell 10.1.3.4
2.010029 arp reply 10.1.3.3 is-at 00:00:00:00:00:09
2.010045 Acknowledgment RA:00:00:00:00:00:09
2.010231 IP 10.1.2.4.9 > 10.1.3.3.49153: UDP, length 1024
2.011767 Acknowledgment RA:00:00:00:00:00:0a
2.500000 Beacon () [6.0* 9.0 12.0 18.0 24.0 36.0 48.0 54.0 Mbit] IBSS
5.000000 Beacon () [6.0* 9.0 12.0 18.0 24.0 36.0 48.0 54.0 Mbit] IBSS
7.500000 Beacon () [6.0* 9.0 12.0 18.0 24.0 36.0 48.0 54.0 Mbit] IBSS
```

ADDITIONAL QUESTIONS

1. **What is a computer network?**

A computer network is a group of computers and devices that are connected together to share resources such as data, printers, and internet access.

2. **What are the different types of computer networks?**

There are several types of computer networks, including:

- Local Area Network (LAN): A LAN is a network that is limited to a small geographical area, such as a single building or campus.
- Wide Area Network (WAN): A WAN is a network that spans a large geographical area, such as a city or country.
- Metropolitan Area Network (MAN): A MAN is a network that spans a large metropolitan area, such as a city.
- Personal Area Network (PAN): A PAN is a network that is used to connect devices in close proximity, such as a computer and a printer.

3. **What is a network topology?**

A network topology is the arrangement of the various components (links, nodes, etc.) of a computer network. Various types of network topologies exist, including:

- Bus topology: All devices in a bus topology are linked to one central cable, or backbone.
- Star topology: The hub or switch in the middle of a star topology serves as the connection point for all devices.
- Ring topology: Each device among a ring topology is connected to two more devices in a circular pattern.

4. **Describe a router.**

A router is a piece of hardware that connects multiple networks and routes data between them. Routers consult routing tables and protocols to decide which network to send data to.

5. **What is meant by Router ?**

A router is an electronic device that interconnects two ormore computer networks, and selectively interchanges packets of data between them . A router is a networking device whose software and hardware are customized to thetasks of routing and forwarding information.

6. **What do u mean by NIC (Network Interface Card) ?**

   A network card, network adapter, or NIC (network interface card) is a piece of computer hardware designed to allow computers to communicate over a computer network. It provides physical access to a networking medium and often provides a low-level addressing system through the use of MAC address

7. **List the layers of OSI**

   a. Physical Layer
   b. Data Link Layer
   c. Network Layer
   d. Transport Layer
   e. Session Layer
   f. Presentation Layer
   g. Application Layer

8. **What are the responsibilities of Transport Layer?**

   The Transport Layer is responsible for source-to-destination delivery of the entire message.
   a. Service-point Addressing
   b. Segmentation and reassembly
   c. Connection Control
   d. Flow Control
   e. Error Control

9. **What is CRC? What is Checksum?**

   CRC, is the most powerful of the redundancy checking techniques, is based on binary division.Checksum is used by the higher layer protocols (TCP/IP) for error detection

10. **What is Redundancy?**

    The concept of including extra information in the transmission solely for the purpose of comparison. This technique is called redundancy.

11. **What is socket programming?**

    Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection.

12. Differences between TCP and UDP

Basis	Transmission Control Protocol (TCP)	User Datagram Protocol (UDP)
Type of Service	TCP is a connection-oriented protocol. Connection orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data.	UDP is the Datagram-oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, or terminating a connection. UDP is efficient for broadcast and multicast types of network transmission.
Reliability	TCP is reliable as it guarantees the delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
Error checking mechanism	TCP provides extensive error-checking mechanisms. It is because it provides flow control and acknowledgment of data.	UDP has only the basic error-checking mechanism using checksums.
Acknowledgment	An acknowledgment segment is present.	No acknowledgment segment.
Sequence	Sequencing of data is a feature of Transmission Control Protocol (TCP). this means that packets arrive in order at the receiver.	There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer.
Speed	TCP is comparatively slower than UDP.	UDP is faster, simpler, and more efficient than TCP.

Basis	Transmission Control Protocol (TCP)	User Datagram Protocol (UDP)
Retransmission	Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in the User Datagram Protocol (UDP).
Header Length	TCP has a (20-60) bytes variable length header.	UDP has an 8 bytes fixed-length header.
Weight	TCP is heavy-weight.	UDP is lightweight.
Handshaking Techniques	Uses handshakes such as SYN, ACK, SYN-ACK	It’s a connectionless protocol i.e. No handshake
Broadcasting	TCP doesn’t support Broadcasting.	UDP supports Broadcasting.
Protocols	TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet.	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP.
Stream Type	The TCP connection is a byte stream.	UDP connection is a message stream.
Overhead	Low but higher than UDP.	Very low.
Applications	This protocol is primarily utilized in situations when a safe and trustworthy communication procedure is necessary, such as in email, on the web surfing, and in military services.	This protocol is used in situations where quick communication is necessary but where dependability is not a concern, such as VoIP, game streaming, video, and music streaming, etc.

13. What is a network simulator?

In a controlled setting, a ***network simulator*** is a tool that is employed to imitate the behavior and
functionality of a network. Before implementing network configurations, protocol, and security
procedures in a real environment, network simulators are frequently used to test them.