REPORT ON COMPUTER NETWORKS LAB(U18CSI5201L)

Submitted by

GOKULNATHAN B
(22BCS029)

B.E-COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY COIMBATORE-641 049
(An Autonomous Institution Affiliated to Anna University, Chennai)

OCTOBER 2024

TABLE OF CONTENTS

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 1.a

Lab Code / Lab : **U18CSI5201L/ COMPUTER NETWORKS LABORATORY**

Course / Branch : **III BE CSE**

Title of the exercise : Develop a TCP Echo Client and Server Program using UNIX Socket Programming

STEP 1: INTRODUCTION

a) OBJECTIVE OF THE EXERCISE/EXPERIMENT

To develop echo client server application using TCP

STEP 2: ACQUISITION

b) Facilities/material required to do the exercise/experiment:

Sl.No.	Facilities/material required	Quantity
1. 2.	PC with Linux Platform LAN connection	1/Student

c) Procedure for doing the exercise/experiment:

SERVER:

1) Start the program.
2) Declare the variables for the socket.
3) Specify the family, IPaddress and port number.
4) Create a socket using socket() function.
5) Bind the IP address and port number.
6) Listen and accept the client's request for the connection.
7) Read the client's message.
8) Display the client's message.
9) Close the socket.
10) Stop the program.

CLIENT:

1) Start the program.
2) Declare the variable for the socket.
3) Specify the family, protocol, IP address and port number.
4) Create a socket using socket() function.
5) Call the connect() function.
6) Read the input message.
7) Send the input message to the server.
8) Display the server's echo message.
9) close the socket.

10) Stop the program.

Program

Echo client server application

using TCP

SERVER

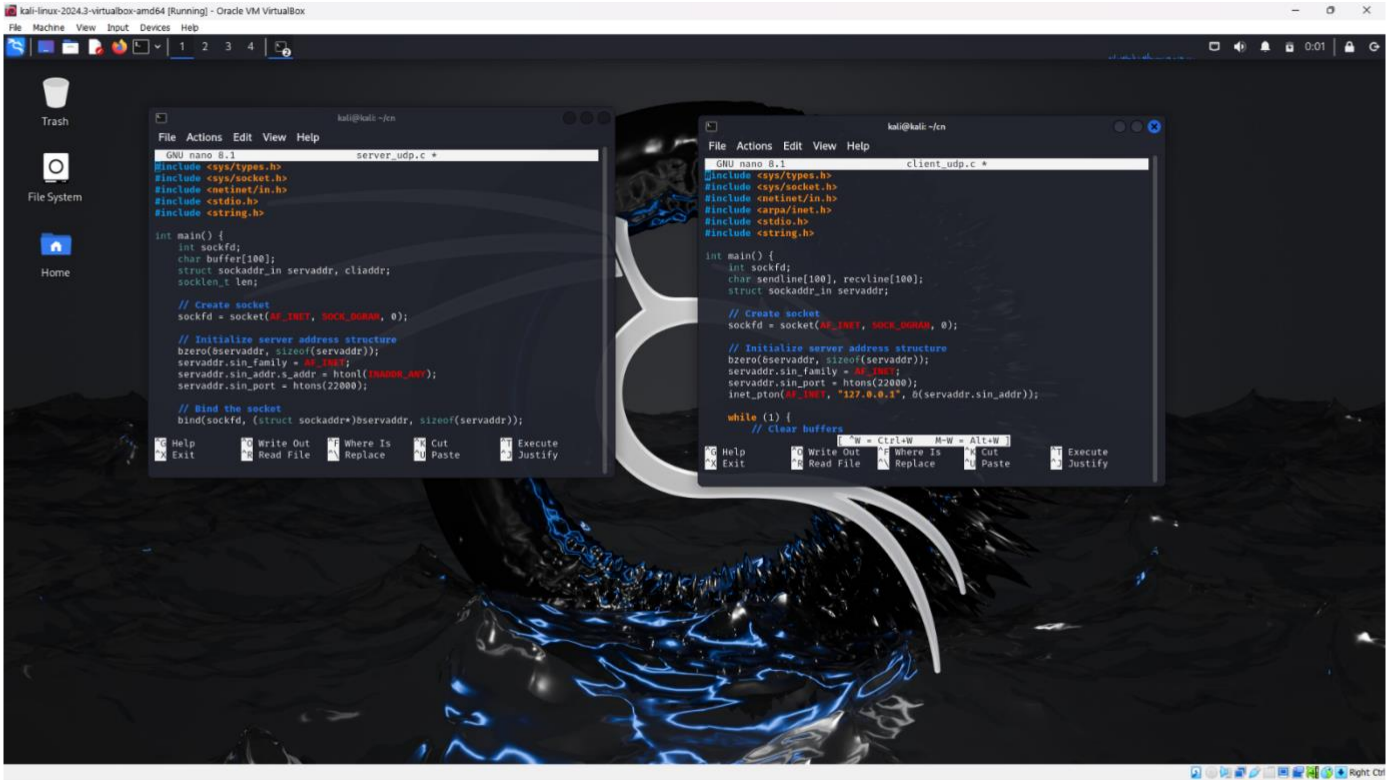
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<signal.h>

#define PORT 12345
#define BUFFER_SIZE 1024

int server_socket;

void handle_sigint(int sig){
printf("\nClosing server socket...\n");
close(server_socket);
exit(0);
}

int main(){
int client_socket;
struct sockaddr_in server_addr,client_addr;
socklen_t client_addr_len=sizeof(client_addr);
char buffer[BUFFER_SIZE];

signal(SIGINT,handle_sigint);

server_socket=socket(AF_INET,SOCK_STREAM,0);
if(server_socket==-1){
perror("Error creating socket");
exit(1);
}

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

if(bind(server_socket,(struct sockaddr *)&server_addr,sizeof(server_addr))==-1){
perror("Error binding socket");
close(server_socket);
exit(1);
}

if(listen(server_socket,5)==-1){
perror("Error listening for connections");
close(server_socket);
exit(1);
}

printf("Server listening on port %d...\n",PORT);

client_socket=accept(server_socket,(struct sockaddr *)&client_addr,&client_addr_len);
if(client_socket==-1){
perror("Error accepting connection");
close(server_socket);
exit(1);
}

printf("Client connected.\n");

while(1){
int bytes_received=recv(client_socket,buffer,sizeof(buffer)-1,0);
if(bytes_received<=0){
printf("Connection closed by client.\n");
break;
}

buffer[bytes_received]='\0';
printf("Received: %s",buffer);

int bytes_sent=send(client_socket,buffer,strlen(buffer),0);
if(bytes_sent==-1){
perror("Error sending data");
break;
}
}

close(client_socket);
close(server_socket);

return 0;
}
```

CLIENT:

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

```
#include<arpa/inet.h>
#define SERVER_IP "127.0.0.1"
#define PORT 12345
#define BUFFER_SIZE 1024

int main(){
int client_socket;
struct sockaddr_in server_addr;
char buffer[BUFFER_SIZE];

client_socket=socket(AF_INET,SOCK_STREAM,0);
if(client_socket==-1){
perror("Error creating socket");
exit(1);
}

server_addr.sin_family=AF_INET;
server_addr.sin_port=htons(PORT);
server_addr.sin_addr.s_addr=inet_addr(SERVER_IP);

if(connect(client_socket,(struct sockaddr *)&server_addr,sizeof(server_addr))==-1){
perror("Error connecting to server");
close(client_socket);
exit(1);
}

printf("Connected to server at %s:%d\n",SERVER_IP,PORT);

while(1){
printf("Enter a message to send (or 'quit' to exit): ");
fgets(buffer,sizeof(buffer),stdin);

if(strcmp(buffer,"quit\n")==0){
break;
}

send(client_socket,buffer,strlen(buffer),0);

int bytes_received=recv(client_socket,buffer,sizeof(buffer)-1,0);
if(bytes_received<=0){
printf("Connection closed by server.\n");
break;
}
buffer[bytes_received]='\0';
printf("Received: %s",buffer);
}

close(client_socket);

return 0;
}
```

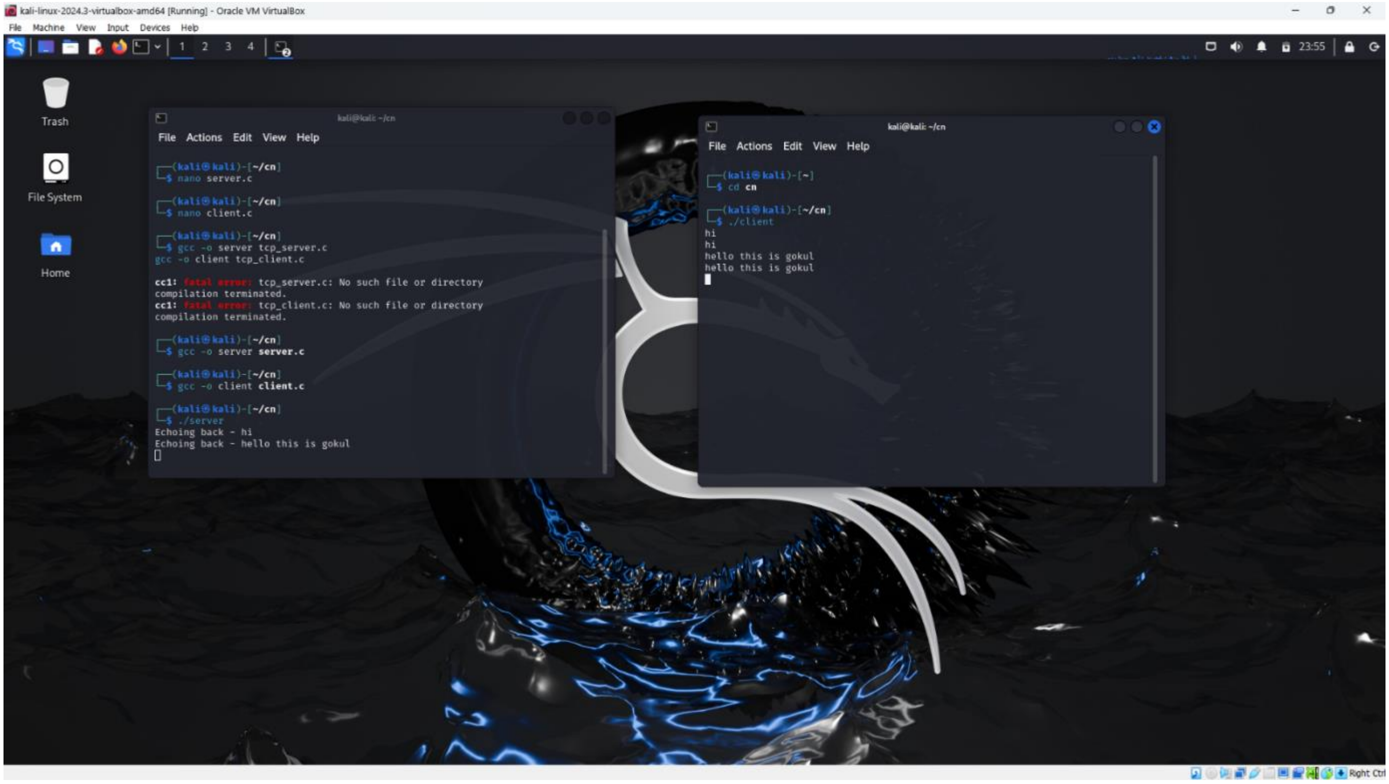
OUTPUT



kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
File System
Home
kali@kali: ~/cn
File Actions Edit View Help
kali@kali: ~/cn
File Actions Edit View Help
Right Ctrl

```
  GNU nano 8.1                    server_udp.c *
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <string.h>

int main() {
    int sockfd;
    char buffer[100];
    struct sockaddr_in servaddr, cliaddr;
    socklen_t len;

    // Create socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    // Initialize server address structure
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(22000);

    // Bind the socket
    bind(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr));

^G Help      ^O Write Out  ^F Where Is   ^K Cut       ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify
```

```
  GNU nano 8.1                    client_udp.c *
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>

int main() {
    int sockfd;
    char sendline[100], recvline[100];
    struct sockaddr_in servaddr;

    // Create socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    // Initialize server address structure
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(22000);
    inet_pton(AF_INET, "127.0.0.1", &(servaddr.sin_addr));

    while (1) {
        // Clear buffers
                       [ ^W = Ctrl+W    M-W = Alt+W ]
^G Help      ^O Write Out  ^F Where Is   ^K Cut       ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify
```


kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
23:55
Trash
File System
Home
kali@kali: ~/cn
File Actions Edit View Help
Right Ctrl

```
┌──(kali㉿kali)-[~/cn]
└─$ nano server.c

┌──(kali㉿kali)-[~/cn]
└─$ nano client.c

┌──(kali㉿kali)-[~/cn]
└─$ gcc -o server tcp_server.c
gcc -o client tcp_client.c

cc1: fatal error: tcp_server.c: No such file or directory
compilation terminated.
cc1: fatal error: tcp_client.c: No such file or directory
compilation terminated.

┌──(kali㉿kali)-[~/cn]
└─$ gcc -o server server.c

┌──(kali㉿kali)-[~/cn]
└─$ gcc -o client client.c

┌──(kali㉿kali)-[~/cn]
└─$ ./server
Echoing back - hi
Echoing back - hello this is gokul
```

```
┌──(kali㉿kali)-[~]
└─$ cd cn

┌──(kali㉿kali)-[~/cn]
└─$ ./client
hi
hi
hello this is gokul
hello this is gokul
```

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 1.b

Lab Code / Lab : **U18CSI5201L/ COMPUTER NETWORKS LABORATORY**

Course / Branch : **III BE CSE**

Title of the exercise : Develop a UDP Echo Client and Server Program using UNIX Socket Programming

STEP 1: INTRODUCTION

d) OBJECTIVE OF THE EXERCISE/EXPERIMENT

To develop echo client server application using UDP

STEP 2: ACQUISITION

e) Facilities/material required to do the exercise/experiment:

Sl.No.	Facilities/material required	Quantity
1. 2.	PC with Linux Platform LAN connection	1/Student

f) Procedure for doing the exercise/experiment:

SERVER:

1) Start the program.

2) Declare the variables for the socket.

3) Specify the family, IPaddress and port number.

4) Create a socket using socket() function.

7) Bind the IP address and port number.

8) Listen and accept the client's request for the connection.

7) Read the client's message.

8) Display the client's message.

9) Close the socket.

10) Stop the program.

CLIENT:

6) Start the program.

7) Declare the variable for the socket.

8) Specify the family, protocol, IP address and port number.

9) Create a socket using socket() function.

10) Call the connect()

function.6)Read the input

message.

7) Send the input message to the server.

8) Display the server's echo message.

11) close the socket.

12) Stop the program.

Program

Echo client server application using UDP

SERVER:

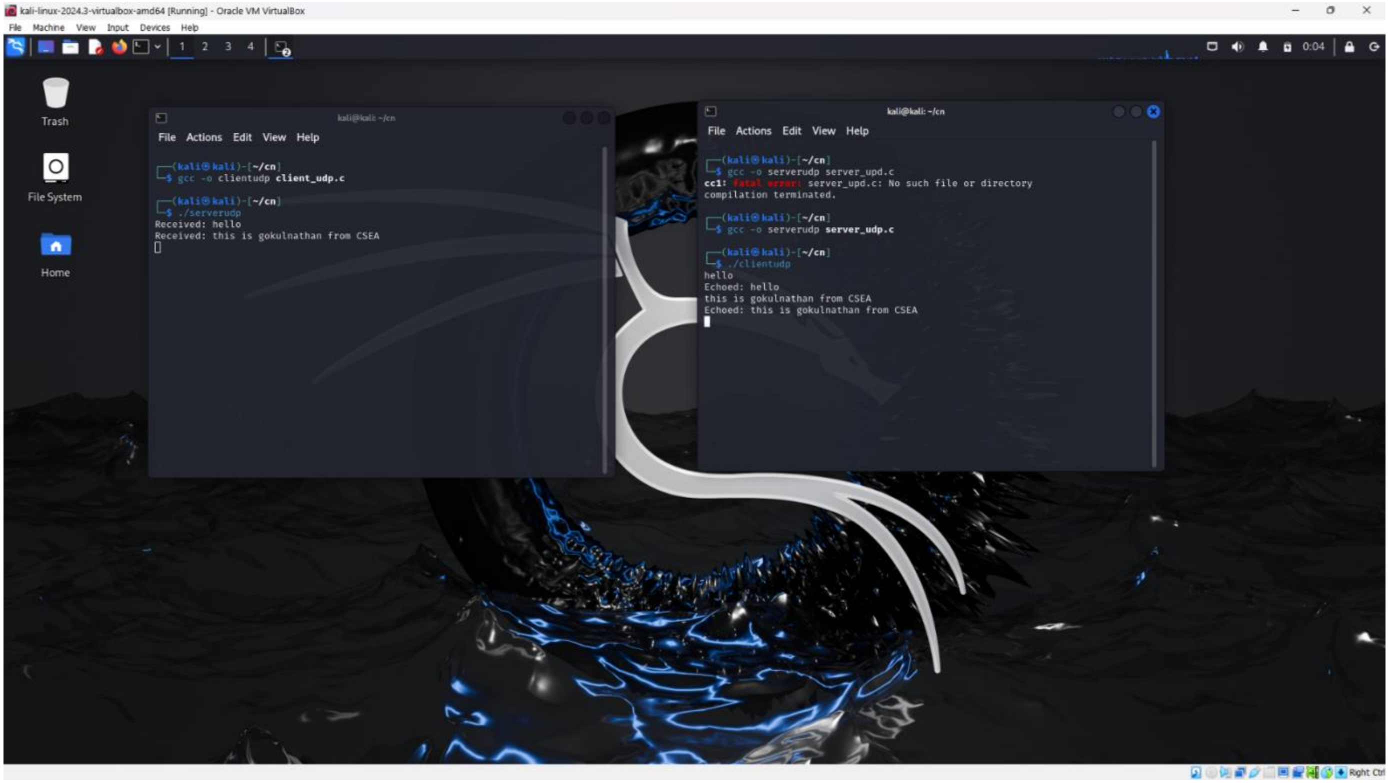
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>

#define PORT 12345
#define BUFFER_SIZE 1024

int main(){
int server_socket;
struct sockaddr_in server_addr,client_addr;
socklen_t client_addr_len=sizeof(client_addr);
char buffer[BUFFER_SIZE];

server_socket=socket(AF_INET,SOCK_DGRAM,0);
if(server_socket==-1){
perror("Error creating socket");
exit(1);
}

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

if(bind(server_socket,(struct sockaddr *)&server_addr,sizeof(server_addr))==-1){
perror("Error binding socket");
close(server_socket);
exit(1);
}

printf("Server listening on port %d...\n",PORT);

while(1){
int bytes_received=recvfrom(server_socket,buffer,sizeof(buffer)-1,0,(struct sockaddr *)&client_addr,&client_addr_len);
if(bytes_received<=0){
perror("Error receiving data");
break;
}

buffer[bytes_received]='\0';
printf("Received: %s",buffer);

sendto(server_socket,buffer,strlen(buffer),0,(struct sockaddr *)&client_addr,client_addr_len);
}

close(server_socket);

return 0;
}
```

CLIENT:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>
#define SERVER_IP "127.0.0.1"
#define PORT 12345
#define BUFFER_SIZE 1024
```

```
int main(){
int client_socket;
struct sockaddr_in server_addr;
char buffer[BUFFER_SIZE];

client_socket=socket(AF_INET,SOCK_DGRAM,0);
if(client_socket==-1){
perror("Error creating socket");
exit(1);
}

server_addr.sin_family=AF_INET;
server_addr.sin_port=htons(PORT);
server_addr.sin_addr.s_addr=inet_addr(SERVER_IP);

while(1){
printf("Enter a message to send (or 'quit' to exit): ");
fgets(buffer,sizeof(buffer),stdin);

if(strcmp(buffer,"quit\n")==0){
break;
}

sendto(client_socket,buffer,strlen(buffer),0,(struct sockaddr *)&server_addr,sizeof(server_addr));

int bytes_received=recvfrom(client_socket,buffer,sizeof(buffer)-1,0,NULL,NULL);
if(bytes_received<=0){
perror("Error receiving data");
break;
}

buffer[bytes_received]='\0';
printf("Received: %s",buffer);
}

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

OUTPUT



kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
File System
Home
kali@kali: ~/cn
File Actions Edit View Help

```
┌──(kali㉿kali)-[~/cn]
└─$ gcc -o clientudp client_udp.c

┌──(kali㉿kali)-[~/cn]
└─$ ./serverudp
Received: hello
Received: this is gokulnathan from CSEA
```

kali@kali: ~/cn
File Actions Edit View Help

```
┌──(kali㉿kali)-[~/cn]
└─$ gcc -o serverudp server_upd.c
cc1: fatal error: server_upd.c: No such file or directory
compilation terminated.

┌──(kali㉿kali)-[~/cn]
└─$ gcc -o serverudp server_udp.c

┌──(kali㉿kali)-[~/cn]
└─$ ./clientudp
hello
Echoed: hello
this is gokulnathan from CSEA
Echoed: this is gokulnathan from CSEA
```

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 2.a

Lab Code / Lab : **U18CSI5201L/ COMPUTER NETWORKS LABORATORY**

Course / Branch : **III BE CSE**

Title of the exercise **:** Develop a TCP Chat Client and Server Program.

STEP 1: INTRODUCTION

a) OBJECTIVE OF THE EXERCISE/EXPERIMENT

To develop CHAT client server application using TCP

STEP 2: ACQUISITION

b) Facilities/material required to do the exercise/experiment:

Sl.No.	Facilities/material required	Quantity
1. 2.	PC with Linux Platform LAN connection	1/Student

c) Procedure for doing the exercise/experiment:

- Start the program, declare the variables
- Create a socket using the socket structure socket(AF_INET, SOCK_STREAM,0)
- Set the socket family, IP address and the port using the server address
- Set the socket address of 8 bytes to zero using the memset() function
- Establish the connection to the server, and then create a child process
- The child process send a message to the server using send function and receive themessage from the server
- The client terminate the connection whenever it receive the bye message from theserver
- Compile and execute the program

SERVER

- Start the program, declare the variables
- Create a socket using the socket structure socket(AF_INET, SOCK_STREAM,0)
- Set the socket family, IP address and the port using the server address
- Set the socket address of 8 bytes to zero using the memset() function
- Bind and listen the socket structure
- Accept the client connection using the socket descriptor and the server address
- The child process receive the message from the client using the socket descriptor
- The child process send the response to the client, and terminate the connectionwhenever it receive the bye message from the client
- Compile and execute the program
- Start the program, declare the variables

Program

SERVER:

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

#define PORT 12345
#define BUFFER_SIZE 1024

int main(){
int server_socket,client_socket;
struct sockaddr_in server_addr,client_addr;
socklen_t client_addr_len=sizeof(client_addr);
char buffer[BUFFER_SIZE];

server_socket=socket(AF_INET,SOCK_STREAM,0);
if(server_socket==-1){
perror("Error creating socket");
exit(1);
}

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

if(bind(server_socket,(struct sockaddr *)&server_addr,sizeof(server_addr))==-1){
perror("Error binding socket");
close(server_socket);
exit(1);
}

if(listen(server_socket,5)==-1){
perror("Error listening for connections");
close(server_socket);
exit(1);
}

printf("Server listening on port %d...\n",PORT);

client_socket=accept(server_socket,(struct sockaddr *)&client_addr,&client_addr_len);
if(client_socket==-1){
perror("Error accepting connection");
close(server_socket);
exit(1);
}

printf("Client connected.\n");

while(1){
int bytes_received=recv(client_socket,buffer,sizeof(buffer),0);
if(bytes_received<=0){
printf("Connection closed by client.\n");
break;
}
buffer[bytes_received]='\0';
printf("Client: %s",buffer);

printf("Server (Type 'quit' to exit): ");
fgets(buffer,sizeof(buffer),stdin);

send(client_socket,buffer,strlen(buffer),0);

if(strcmp(buffer,"quit\n")==0){
break;
}
}
```

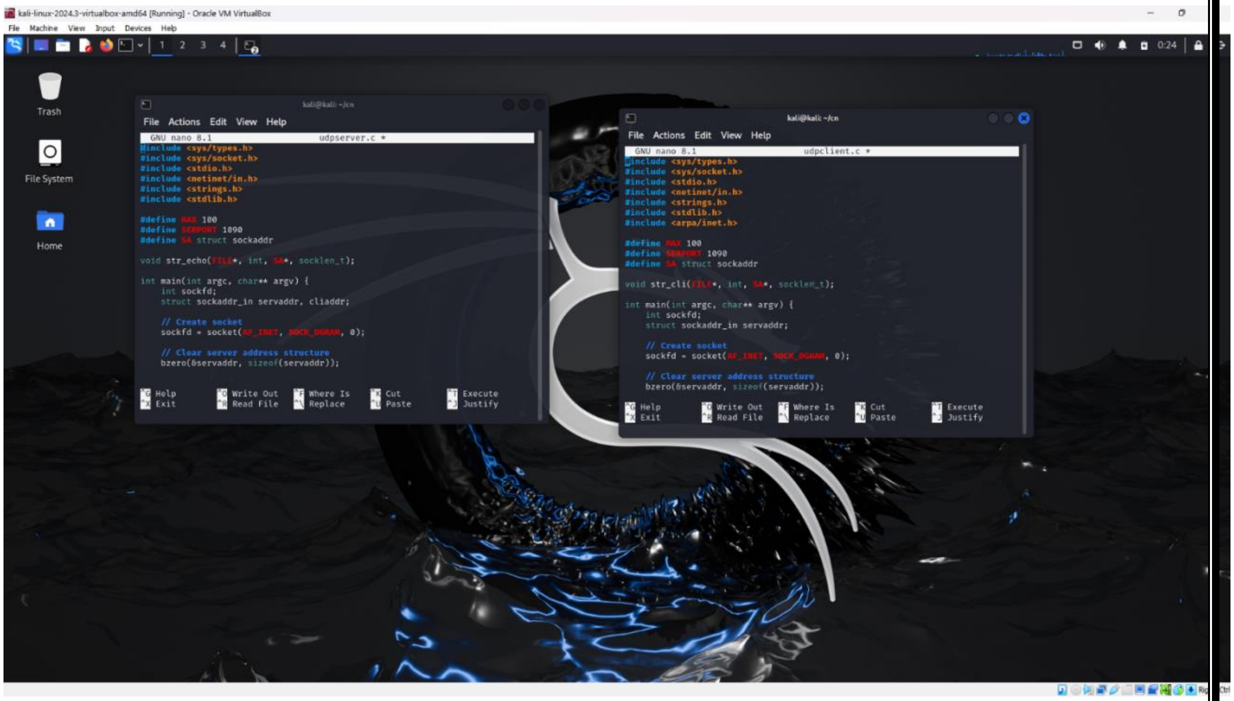
```
close(client_socket);
close(server_socket);

return 0;
```

CLIENT:

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

#define SERVER_IP "127.0.0.1"
#define PORT 12345
#define BUFFER_SIZE 1024

int main(){
int client_socket;
struct sockaddr_in server_addr;
char buffer[BUFFER_SIZE];

client_socket=socket(AF_INET,SOCK_STREAM,0);
if(client_socket==-1){
perror("Error creating socket");
exit(1);
}

server_addr.sin_family=AF_INET;
server_addr.sin_port=htons(PORT);
server_addr.sin_addr.s_addr=inet_addr(SERVER_IP);

if(connect(client_socket,(struct sockaddr *)&server_addr,sizeof(server_addr))==-1){
perror("Error connecting to server");
close(client_socket);
exit(1);
}

printf("Connected to server at %s:%d\n",SERVER_IP,PORT);

while(1){
printf("Client (Type 'quit' to exit): ");
fgets(buffer,sizeof(buffer),stdin);

send(client_socket,buffer,strlen(buffer),0);

if(strcmp(buffer,"quit\n")==0){
```

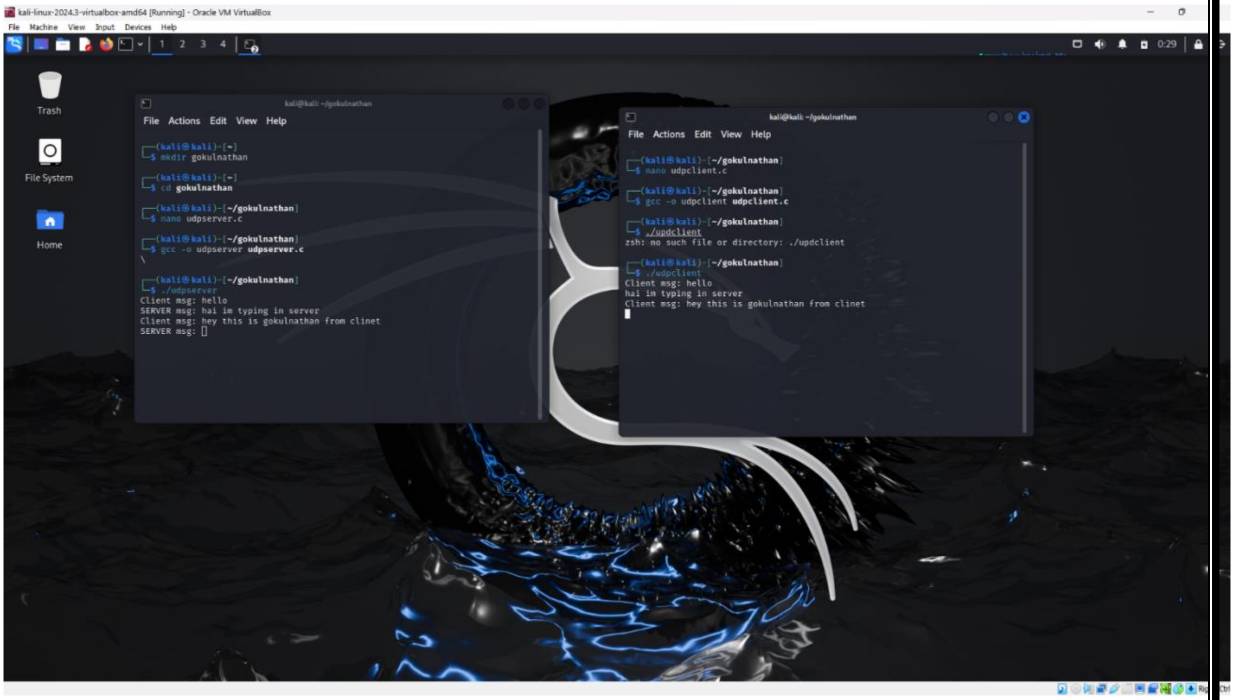
```
break;
}

int bytes_received=recv(client_socket,buffer,sizeof(buffer),0);
if(bytes_received<=0){
printf("Connection closed by server.\n");
break;
}
buffer[bytes_received]='\0';
printf("Server: %s",buffer);
}

close(client_socket);

return 0;
}
```

OUTPUT


kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
File System
Home
kali@kali: ~/cn
File Actions Edit View Help
0:24

```
  GNU nano 8.1                     udpserver.c *
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <strings.h>
#include <stdlib.h>

#define MAX 100
#define SERPORT 1090
#define SA struct sockaddr

void str_echo(FILE*, int, SA*, socklen_t);

int main(int argc, char** argv) {
    int sockfd;
    struct sockaddr_in servaddr, cliaddr;

    // Create socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    // Clear server address structure
    bzero(&servaddr, sizeof(servaddr));

^G Help      ^O Write Out  ^F Where Is  ^K Cut     ^T Execute
^X Exit      ^R Read File  ^\ Replace   ^U Paste   ^J Justify
```

```
  GNU nano 8.1                     udpclient.c *
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <strings.h>
#include <stdlib.h>
#include <arpa/inet.h>

#define MAX 100
#define SERPORT 1090
#define SA struct sockaddr

void str_cli(FILE*, int, SA*, socklen_t);

int main(int argc, char** argv) {
    int sockfd;
    struct sockaddr_in servaddr;

    // Create socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    // Clear server address structure
    bzero(&servaddr, sizeof(servaddr));

^G Help      ^O Write Out  ^F Where Is  ^K Cut     ^T Execute
^X Exit      ^R Read File  ^\ Replace   ^U Paste   ^J Justify
```


kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
File System
Home
0:29
kali@kali: ~/gokulnathan
File Actions Edit View Help
kali@kali: ~/gokulnathan
File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ mkdir gokulnathan

┌──(kali㉿kali)-[~]
└─$ cd gokulnathan

┌──(kali㉿kali)-[~/gokulnathan]
└─$ nano udpserver.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ gcc -o udpserver udpserver.c
\

┌──(kali㉿kali)-[~/gokulnathan]
└─$ ./udpserver
Client msg: hello
SERVER msg: hai im typing in server
Client msg: hey this is gokulnathan from clinet
SERVER msg: 
```

```
┌──(kali㉿kali)-[~/gokulnathan]
└─$ nano udpclient.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ gcc -o udpclient udpclient.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ ./updclient
zsh: no such file or directory: ./updclient

┌──(kali㉿kali)-[~/gokulnathan]
└─$ ./udpclient
Client msg: hello
hai im typing in server
Client msg: hey this is gokulnathan from clinet
```

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 2.b

Lab Code / Lab : **U18CSI5201L/ COMPUTER NETWORKS LABORATORY**

Course / Branch : **III BE CSE**

Title of the exercise **:** Develop a UDP Chat Client and Server Program.

STEP 1: INTRODUCTION

d) OBJECTIVE OF THE EXERCISE/EXPERIMENT

To develop CHAT client server application using UDP

STEP 2: ACQUISITION

e) Facilities/material required to do the exercise/experiment:

Sl.No.	Facilities/material required	Quantity
1. **2.**	**PC with Linux Platform** **LAN connection**	**1/Student**

f) Procedure for doing the exercise/experiment:

- Start the program, declare the variables
- Create a socket using the socket structure socket(AF_INET, SOCK_STREAM,0)
- Set the socket family, IP address and the port using the server address
- Set the socket address of 8 bytes to zero using the memset() function
- Establish the connection to the server, and then create a child process

- The child process send a message to the server using send function and receive themessage from the server
- The client terminate the connection whenever it receive the bye message from theserver
- Compile and execute the program

SERVER

- Start the program, declare the variables
- Create a socket using the socket structure socket(AF_INET, SOCK_STREAM,0)
- Set the socket family, IP address and the port using the server address
- Set the socket address of 8 bytes to zero using the memset() function
- Bind and listen the socket structure
- Accept the client connection using the socket descriptor and the server address
- The child process receive the message from the client using the socket descriptor
- The child process send the response to the client, and terminate the connectionwhenever it receive the bye message from the client
- Compile and execute the program
- Start the program, declare the variables

Program

SERVER

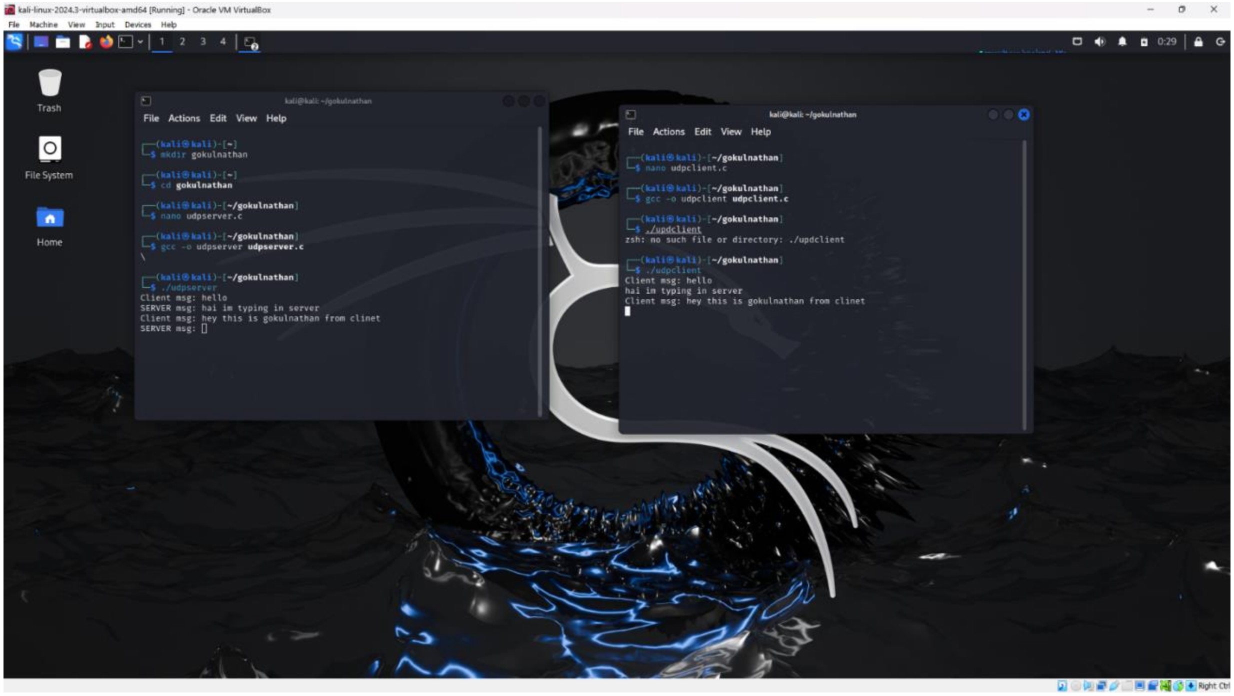
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>

#define PORT 12345
#define BUFFER_SIZE 1024

int main(){
int server_socket;
struct sockaddr_in server_addr,client_addr;
socklen_t client_addr_len=sizeof(client_addr);
char buffer[BUFFER_SIZE];

server_socket=socket(AF_INET,SOCK_DGRAM,0);
if(server_socket==-1){
perror("Error creating socket");
exit(1);
}

server_addr.sin_family=AF_INET;
server_addr.sin_port=htons(PORT);
server_addr.sin_addr.s_addr=INADDR_ANY;
if(bind(server_socket,(struct sockaddr *)&server_addr,sizeof(server_addr))==-1){
perror("Error binding socket");
close(server_socket);
exit(1);
}

printf("Server listening on port %d...\n",PORT);

while(1){
int bytes_received=recvfrom(server_socket,buffer,sizeof(buffer),0,(struct sockaddr
*)&client_addr,&client_addr_len);
if(bytes_received<=0){
perror("Error receiving data");
break;
}
buffer[bytes_received]='\0';
printf("Client: %s",buffer);

printf("Server (Type 'quit' to exit): ");
fgets(buffer,sizeof(buffer),stdin);

sendto(server_socket,buffer,strlen(buffer),0,(struct sockaddr *)&client_addr,client_addr_len);

if(strcmp(buffer,"quit\n")==0){
break;
}
}

close(server_socket);

return 0;
}
```

CLIENT

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

#define SERVER_IP "127.0.0.1" // Change this to the server's IP address
#define PORT 12345
#define BUFFER_SIZE 1024

int main(){
```

```
int client_socket;
struct sockaddr_in server_addr;
char buffer[BUFFER_SIZE];

client_socket=socket(AF_INET,SOCK_DGRAM,0);
if(client_socket==-1){
perror("Error creating socket");
exit(1);
}

server_addr.sin_family=AF_INET;
server_addr.sin_port=htons(PORT);
server_addr.sin_addr.s_addr=inet_addr(SERVER_IP);

while(1){
printf("Client (Type 'quit' to exit): ");
fgets(buffer,sizeof(buffer),stdin);

sendto(client_socket,buffer,strlen(buffer),0,(struct sockaddr *)&server_addr,sizeof(server_addr));

if(strcmp(buffer,"quit\n")==0){
break;
}

int bytes_received=recvfrom(client_socket,buffer,sizeof(buffer),0,NULL,NULL);
if(bytes_received<=0){
perror("Error receiving data");
break;
}
buffer[bytes_received]='\0';
printf("Server: %s",buffer);
}

close(client_socket);

return 0;
}
```

OUTPUT:


kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
File System
Home
kali@kali: ~/gokulnathan
File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ mkdir gokulnathan

┌──(kali㉿kali)-[~]
└─$ cd gokulnathan

┌──(kali㉿kali)-[~/gokulnathan]
└─$ nano udpserver.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ gcc -o udpserver udpserver.c
\

┌──(kali㉿kali)-[~/gokulnathan]
└─$ ./udpserver
Client msg: hello
SERVER msg: hai im typing in server
Client msg: hey this is gokulnathan from clinet
SERVER msg: []
```

```
┌──(kali㉿kali)-[~/gokulnathan]
└─$ nano udpclient.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ gcc -o udpclient udpclient.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ ./updclient
zsh: no such file or directory: ./updclient

┌──(kali㉿kali)-[~/gokulnathan]
└─$ ./udpclient
Client msg: hello
hai im typing in server
Client msg: hey this is gokulnathan from clinet
```

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 3

Lab Code / Lab : U18CSI5201L/ COMPUTER NETWORKS LABORATORY

Course / Branch : III BE CSE

Title of the exercise : Simulation of datalink and network layer protocols

AIM :

To write a C program to implement simulation of ARP and RARP network protocols.
THEORY :

The term ARP is an abbreviation for Address resolution protocol. The ARP retrieves the receiver's physical address in a network.
The term RARP is an abbreviation for Reverse Address Resolution Protocol. The RARP retrieves a computer's logical address from its available server.

PROGRAM:

ARP/RARP SERVER :

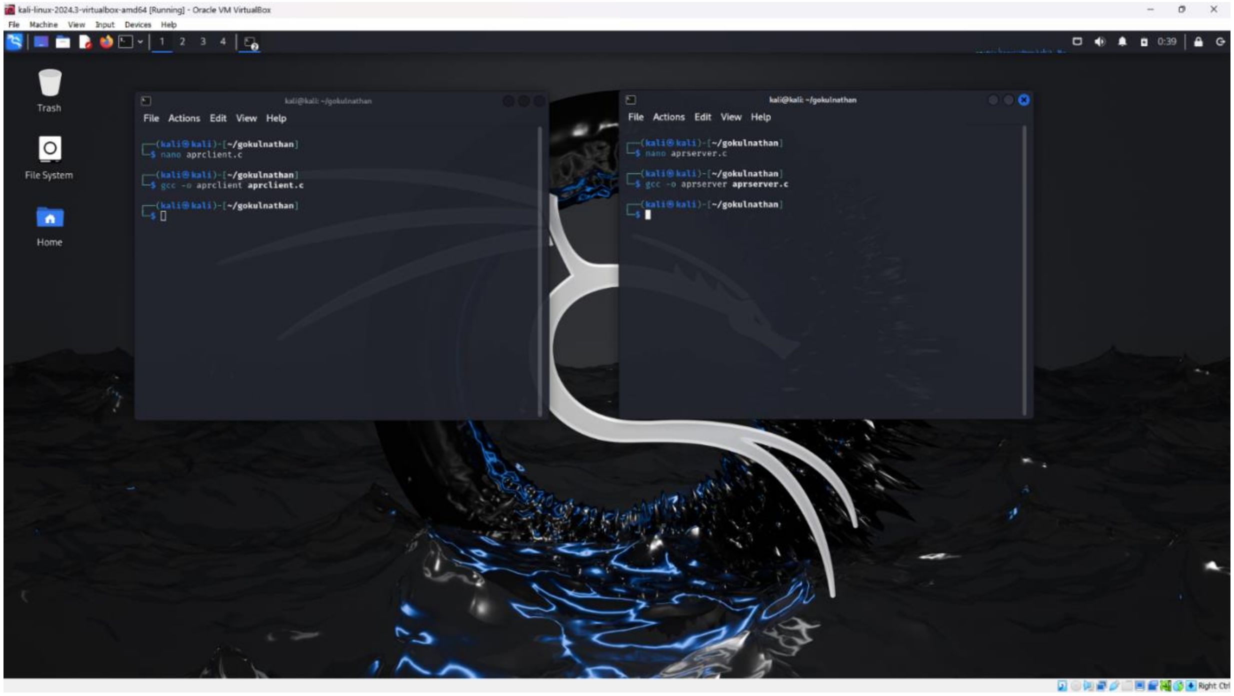
```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <string.h>
#include <unistd.h>

#define SHM_KEY 3000
#define MAX_ENTRIES 3
#define NAME_LENGTH 50
#define IP_LENGTH 16
#define SHM_SIZE (MAX_ENTRIES * (NAME_LENGTH + IP_LENGTH + 2)) // Name + IP + newline
+ space

int main() {
    int shmid, a, i;
    char *ptr, *shmptr;
    // Create a shared memory segment
    shmid = shmget(SHM_KEY, SHM_SIZE, IPC_CREAT | 0666);
    if (shmid < 0) {
        perror("shmget failed");
        exit(1);
    }

    // Attach the shared memory segment
    shmptr = shmat(shmid, NULL, 0);
    if (shmptr == (char *)(-1)) {
        perror("shmat failed");
        exit(1);
    }

    ptr = shmptr;

    // Collect entries
    for (i = 0; i < MAX_ENTRIES; i++) {
        puts("Enter the name:");
        scanf("%s", ptr);
        a = strlen(ptr);
        ptr[a] = ' '; // Add space after the name
        puts("Enter IP:");
        ptr = ptr + a + 1; // Move pointer to the next position
        scanf("%s", ptr);
        ptr[strlen(ptr)] = '\n'; // Add newline after the IP
        ptr += strlen(ptr) + 1; // Move pointer for the next entry
    }

    // Null terminate the string
    *ptr = '\0';

    // Print ARP table
    printf("\nARP table at server side is:\n%s", shmptr);

    // Detach the shared memory
    shmdt(shmptr);

    return 0;
}
```

Client:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <unistd.h>

#define SHM_KEY 3000
#define MAX_ENTRIES 3
#define NAME_LENGTH 50
```

```
#define IP_LENGTH 16
#define MAC_LENGTH 26
#define SHM_SIZE (MAX_ENTRIES * (NAME_LENGTH + IP_LENGTH + 2)) // Name + IP + newline
+ space

int main() {
    int shmid, a;
    char *ptr, *shmptr;
    char ptr2[NAME_LENGTH], ip[IP_LENGTH], mac[MAC_LENGTH];

    // Access the shared memory segment
    shmid = shmget(SHM_KEY, SHM_SIZE, 0666);
    if (shmid < 0) {
        perror("shmget failed");
        exit(1);
    }

    // Attach the shared memory segment
    shmptr = shmat(shmid, NULL, 0);
    if (shmptr == (char *)(-1)) {
        perror("shmat failed");
        exit(1);
    }

    // Print the ARP table
    puts("The ARP table is:");
    printf("%s\n", shmptr);

    while (1) {
        printf("\n1. ARP\n2. RARP\n3. EXIT\n");
        printf("Choose an option: ");
        scanf("%d", &a);

        switch (a) {
            case 1: // ARP
                puts("Enter IP address:");
                scanf("%s", ip);
                ptr = strstr(shmptr, ip);
                if (ptr != NULL) {
                    ptr -= 8; // Adjust pointer to MAC address
                    sscanf(ptr, "%s%*s", ptr2); // Read MAC address
                    printf("MAC address is: %s\n", ptr2);
                } else {
                    printf("IP address not found.\n");
                }
                break;

            case 2: // RARP
                puts("Enter MAC address:");
                scanf("%s", mac);
                ptr = strstr(shmptr, mac);
                if (ptr != NULL) {
                    sscanf(ptr, "%*s%s", ptr2); // Read IP address
                    printf("IP address is: %s\n", ptr2);
                } else {
                    printf("MAC address not found.\n");
                }
                break;

            case 3: // EXIT
                shmdt(shmptr); // Detach shared memory before exit
                exit(0);

            default:
                printf("Invalid option. Please try again.\n");
        }
    }

    // Detach the shared memory (in case of exiting from loop)
    shmdt(shmptr);

    return 0;
}
```

OUTPUT : ARP/RARP SERVER

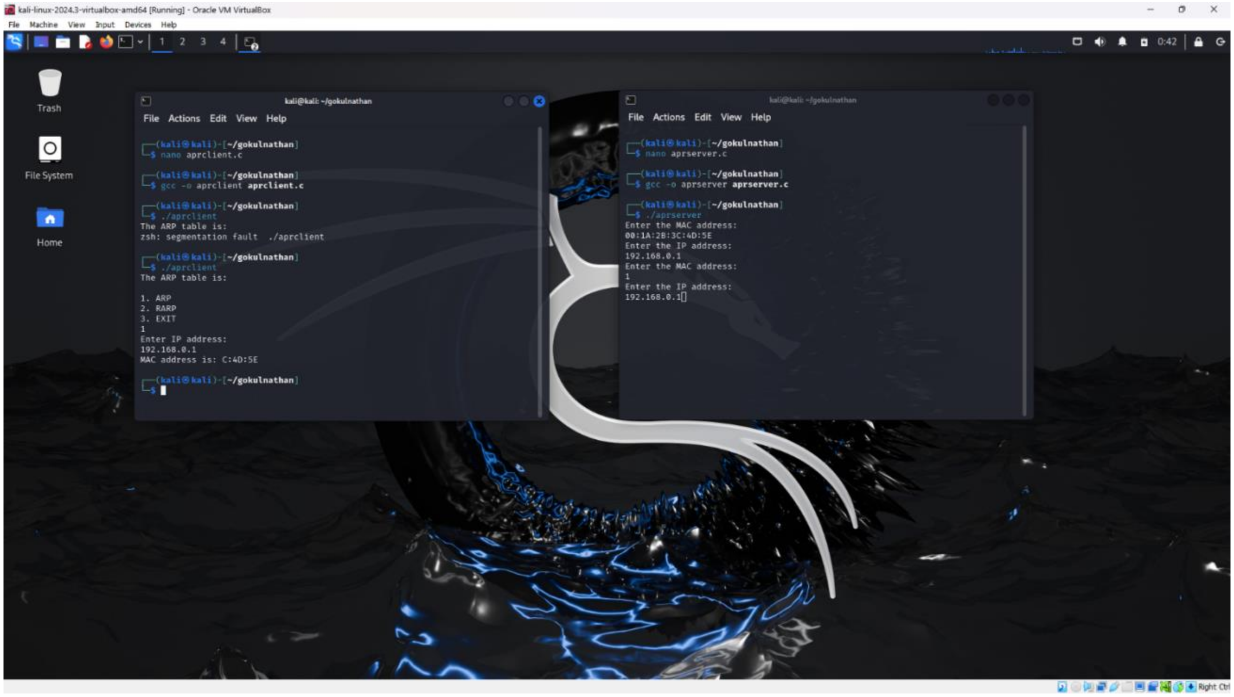


kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
File System
Home
kali@kali: ~/gokulnathan
File Actions Edit View Help

```
┌──(kali㉿kali)-[~/gokulnathan]
└─$ nano aprclient.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ gcc -o aprclient aprclient.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ 
```

```
┌──(kali㉿kali)-[~/gokulnathan]
└─$ nano aprserver.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ gcc -o aprserver aprserver.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ 
```

ARP/RARP CLIENT



kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
File System
Home
kali@kali: ~/gokulnathan
File Actions Edit View Help

```
┌──(kali㉿kali)-[~/gokulnathan]
└─$ nano aprclient.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ gcc -o aprclient aprclient.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ ./aprclient
The ARP table is:
zsh: segmentation fault  ./aprclient

┌──(kali㉿kali)-[~/gokulnathan]
└─$ ./aprclient
The ARP table is:

1. ARP
2. RARP
3. EXIT
1
Enter IP address:
192.168.0.1
MAC address is: C:4D:5E

┌──(kali㉿kali)-[~/gokulnathan]
└─$ 
```

```
┌──(kali㉿kali)-[~/gokulnathan]
└─$ nano aprserver.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ gcc -o aprserver aprserver.c

┌──(kali㉿kali)-[~/gokulnathan]
└─$ ./aprserver
Enter the MAC address:
00:1A:2B:3C:4D:5E
Enter the IP address:
192.168.0.1
Enter the MAC address:
1
Enter the IP address:
192.168.0.1
```

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 4

Lab Code / Lab : **U18CSI5201L/ COMPUTER NETWORKS LABORATORY**

Course / Branch : **III BE CSE**

Title of the exercise : **Performance analysis of TCP and UDP using simulation tool**

AIM :

To write a C program to perform analysis of TCP and UDP using simulation tool- ns2.

THEORY :

Ns is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.

NS2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks. It simulates wired and wireless network.

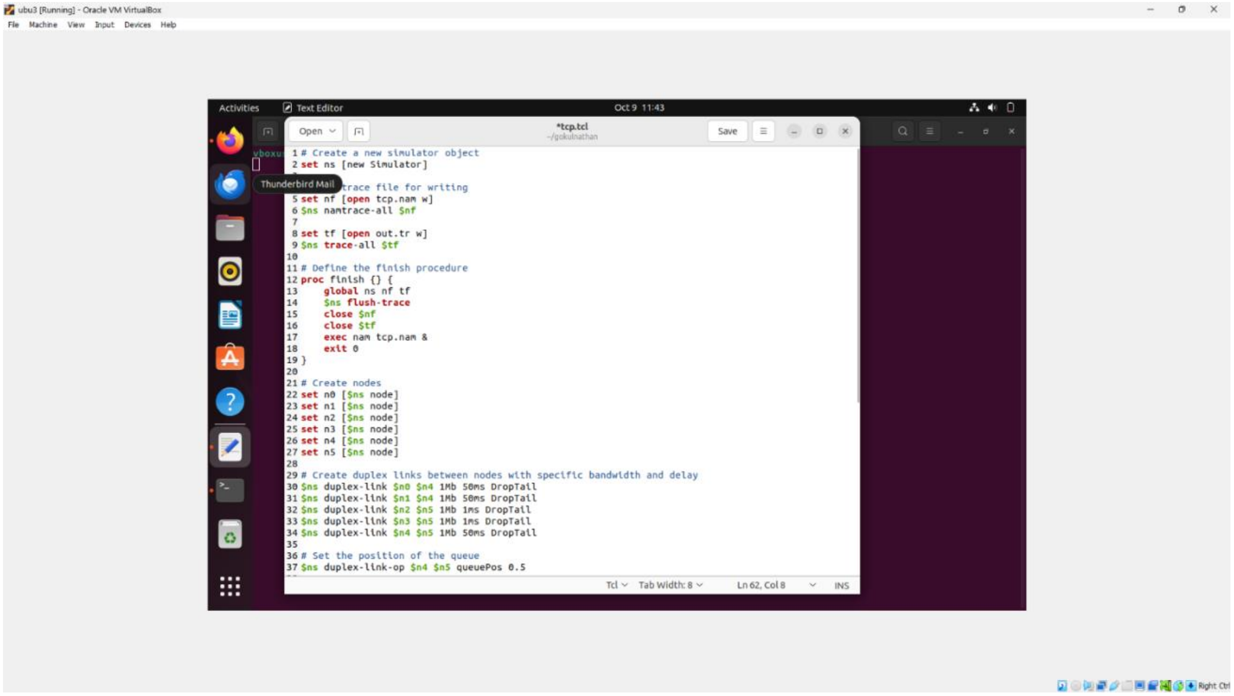
PROGRAM:

TCP

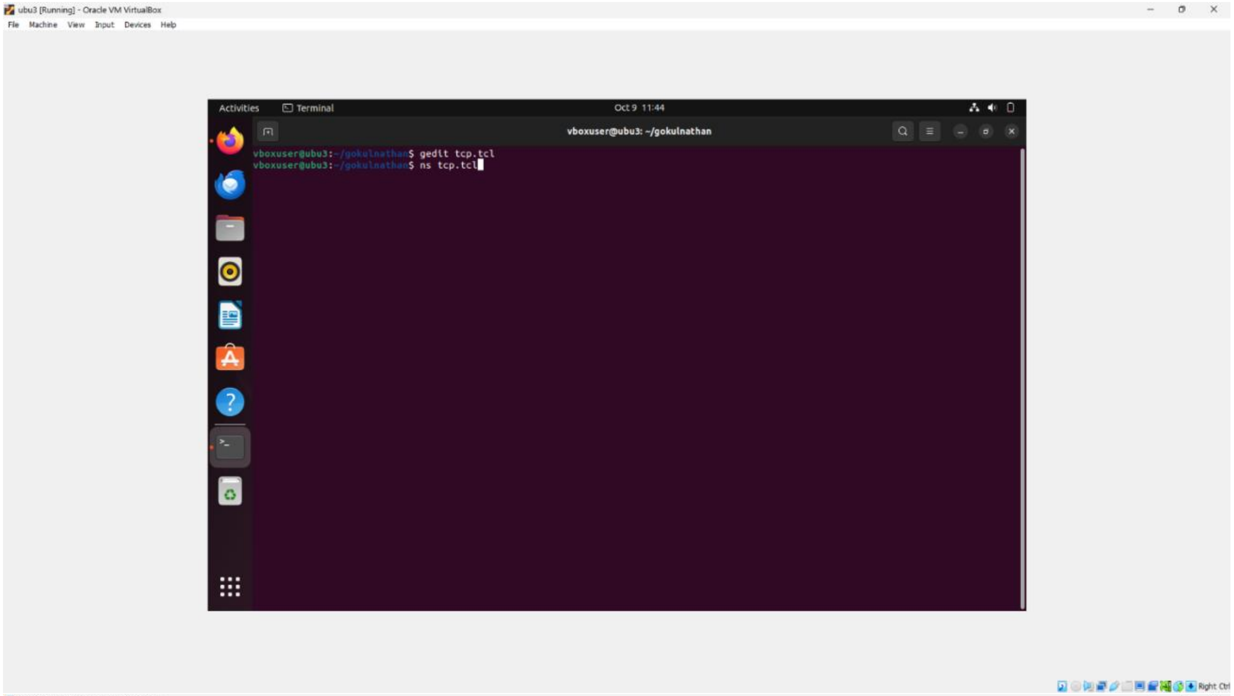
```
set ns [new Simulator] set nf [open tcp.nam w]
$ns namtrace-all $nf set tf [open out.tr w]
$ns trace-all $tf proc finish {} { global ns nf tf
$ns flush-trace close $nf
close $tf
exec nam tcp.nam & exit 0
}
set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node]
set n5 [$ns node]
$ns duplex-link $n0 $n4 1Mb 50ms DropTail
$ns duplex-link $n1 $n4 1Mb 50ms DropTail
$ns duplex-link $n2 $n5 1Mb 1ms DropTail
```

```
$ns duplex-link $n3 $n5 1Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 50ms DropTail
$ns duplex-link-op $n4 $n5 queuePos 0.5 set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp set sink [new Agent/TCPSink]
$ns attach-agent $n2 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.0 "$ftp start"
$ns at 2.5 "$ftp stop"
$ns at 3 "finish"
$ns run
```

OUTPUT :

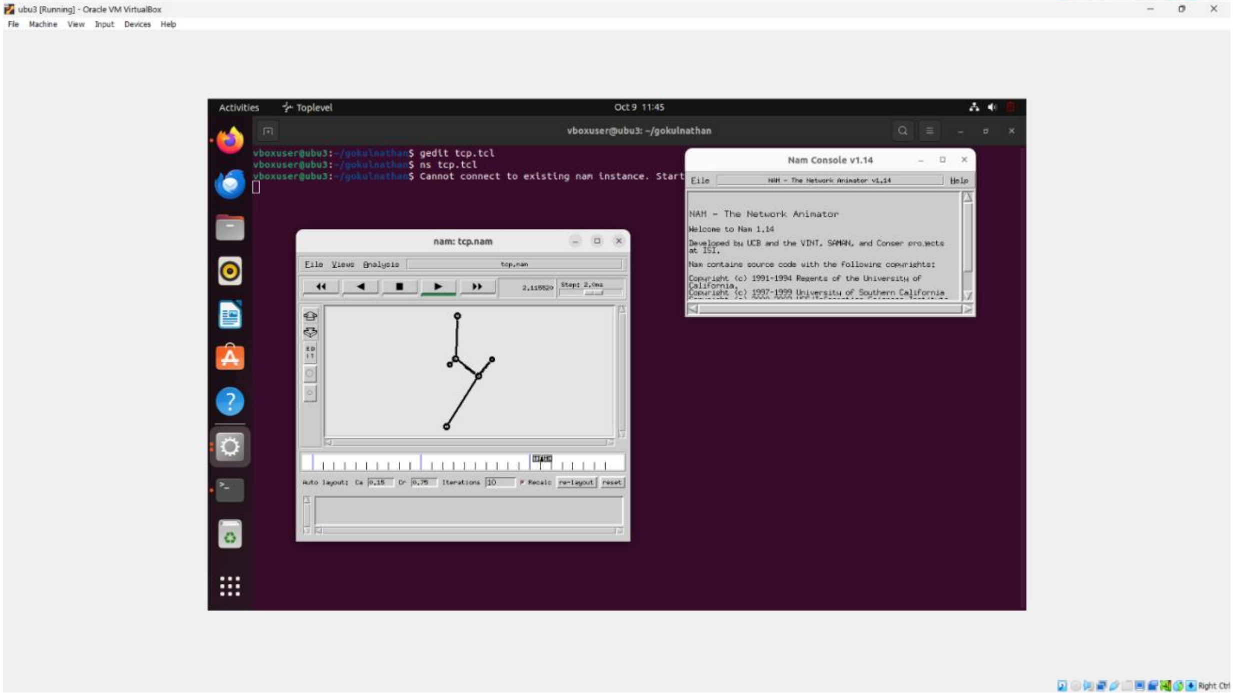


```
# Create a new simulator object
set ns [new Simulator]

trace file for writing
set nf [open tcp.nam w]
$ns namtrace-all $nf

set tf [open out.tr w]
$ns trace-all $tf

# Define the finish procedure
proc finish {} {
    global ns nf tf
    $ns flush-trace
    close $nf
    close $tf
    exec nam tcp.nam &
    exit 0
}

# Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

# Create duplex links between nodes with specific bandwidth and delay
$ns duplex-link $n0 $n4 1Mb 50ms DropTail
$ns duplex-link $n1 $n4 1Mb 50ms DropTail
$ns duplex-link $n2 $n5 1Mb 1ms DropTail
$ns duplex-link $n3 $n5 1Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 50ms DropTail

# Set the position of the queue
$ns duplex-link-op $n4 $n5 queuePos 0.5
```



ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Oct 9 11:44
vboxuser@ubu3: ~/gokulnathan
Right Ctrl

```
vboxuser@ubu3:~/gokulnathan$ gedit tcp.tcl
vboxuser@ubu3:~/gokulnathan$ ns tcp.tcl
```



ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Toplevel
Oct 9 11:45
vboxuser@ubu3: ~/gokulnathan
nam: tcp.nam
File Views Analysis
tcp.nam
2.118820
Step: 2.0ms
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc re-layout reset
Nam Console v1.14
File
NAM - The Network Animator v1.14
Help
Right Ctrl

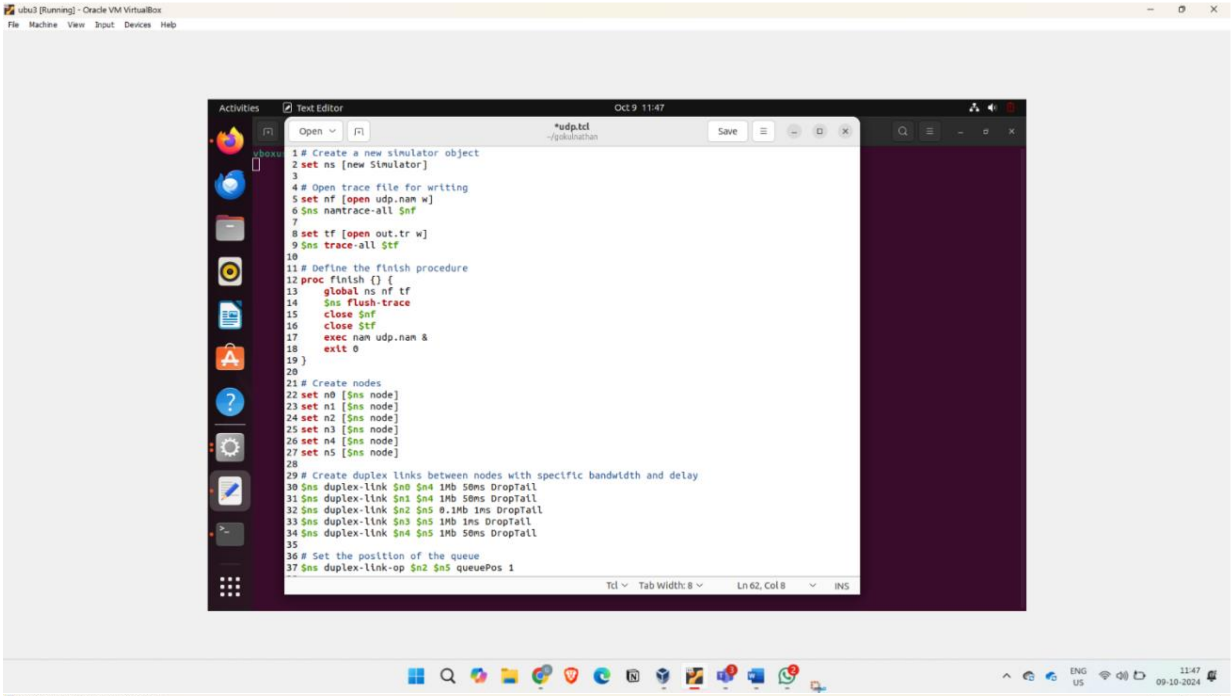
```
vboxuser@ubu3:~/gokulnathan$ gedit tcp.tcl
vboxuser@ubu3:~/gokulnathan$ ns tcp.tcl
vboxuser@ubu3:~/gokulnathan$ Cannot connect to existing nam instance. Start
```

```
NAM - The Network Animator
Welcome to Nam 1.14
Developed by UCB and the VINT, SAMAN, and Conser projects
at ISI.
Nam contains source code with the following copyrights:
Copyright (c) 1991-1994 Regents of the University of
California.
Copyright (c) 1997-1999 University of Southern California
```

UDP

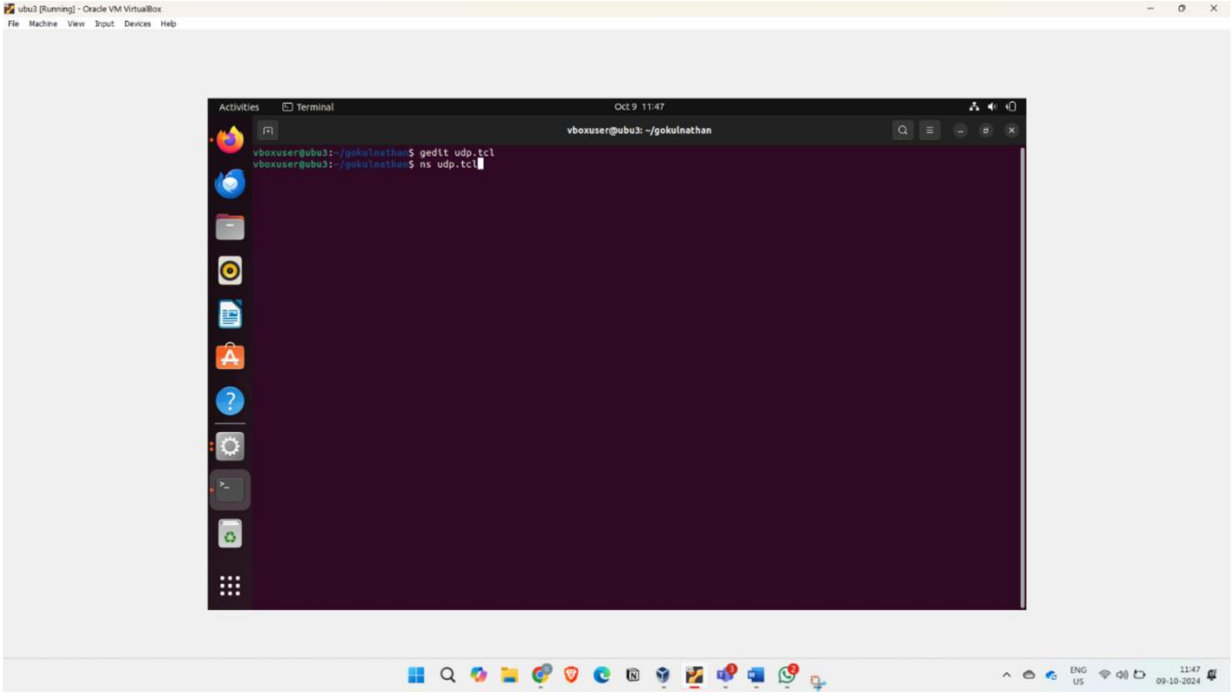
```
set ns [new Simulator] set nf [open udp.nam w]
$ns namtrace-all $nf set tf [open out.tr w]
$ns trace-all $tf proc finish {} { global ns nf tf
$ns flush-trace close $nf
close $tf
exec nam udp.nam & exit 0
}
set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node]
set n5 [$ns node]
$ns duplex-link $n0 $n4 1Mb 50ms DropTail
$ns duplex-link $n1 $n4 1Mb 50ms DropTail
$ns duplex-link $n2 $n5 0.1Mb 1ms DropTail
$ns duplex-link $n3 $n5 1Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 50ms DropTail
$ns duplex-link-op $n2 $n5 queuePos 1
set tcp [new Agent/UDP]
$ns attach-agent $n0 $tcp set sink [new Agent/Null]
$ns attach-agent $n2 $sink
$ns connect $tcp $sink
set ftp [new Application/Traffic/CBR]
$ftp attach-agent $tcp
$ns at 0.0 "$ftp start"
$ns at 2.5 "$ftp stop"
$ns at 3 "finish"
$ns run
```

OUTPUT:



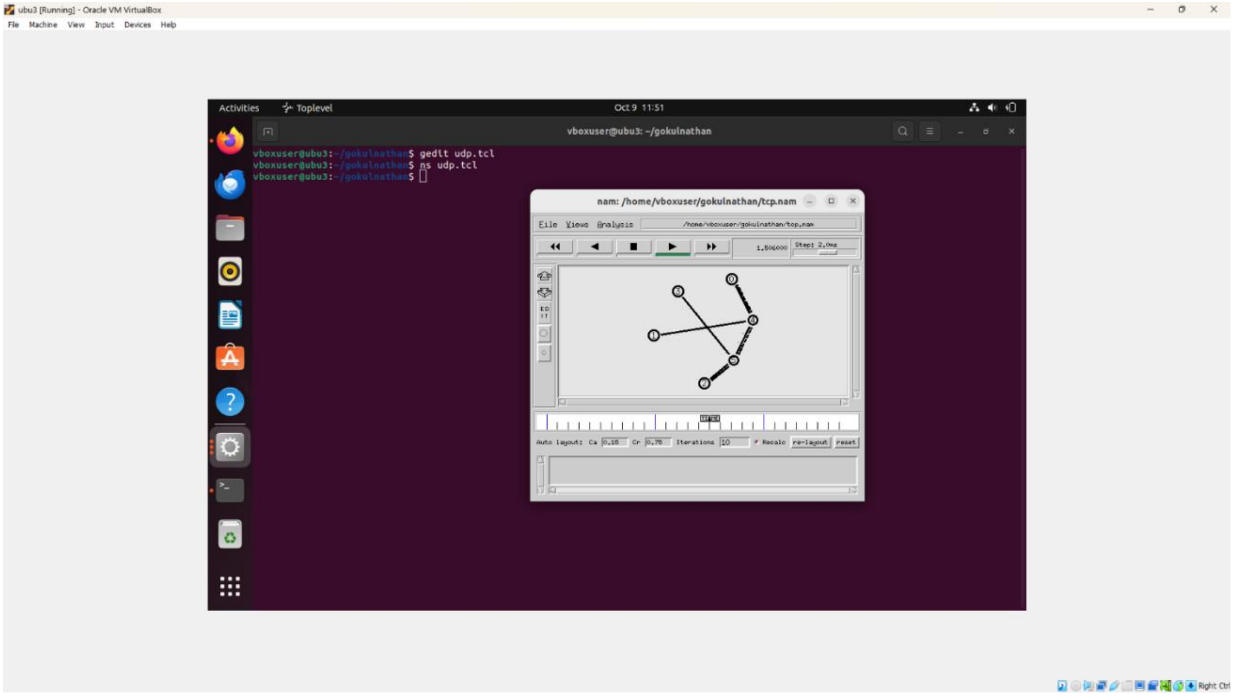
```
# Create a new simulator object
set ns [new Simulator]

# Open trace file for writing
set nf [open udp.nam w]
$ns namtrace-all $nf

set tf [open out.tr w]
$ns trace-all $tf

# Define the finish procedure
proc finish {} {
    global ns nf tf
    $ns flush-trace
    close $nf
    close $tf
    exec nam udp.nam &
    exit 0
}

# Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

# Create duplex links between nodes with specific bandwidth and delay
$ns duplex-link $n0 $n4 1Mb 50ms DropTail
$ns duplex-link $n1 $n4 1Mb 50ms DropTail
$ns duplex-link $n2 $n5 0.1Mb 1ms DropTail
$ns duplex-link $n3 $n5 1Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 50ms DropTail

# Set the position of the queue
$ns duplex-link-op $n2 $n5 queuePos 1
```

ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Text Editor
Oct 9 11:47
Open
*udp.tcl
~/gokulnathan
Save
Tcl
Tab Width: 8
Ln 62, Col 8
INS
11:47
09-10-2024



```
vboxuser@ubu3:~/gokulnathan$ gedit udp.tcl
vboxuser@ubu3:~/gokulnathan$ ns udp.tcl
```

ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Oct 9 11:47
vboxuser@ubu3: ~/gokulnathan
11:47
09-10-2024


ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Toplevel
Oct 9 11:51
vboxuser@ubu3: ~/gokulnathan
nam: /home/vboxuser/gokulnathan/tcp.nam
File Views Analysis
/home/vboxuser/gokulnathan/tcp.nam
1.806000
Step: 2.0ms
0
1
2
3
4
5
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc re-layout reset
Right Ctrl

```
vboxuser@ubu3:~/gokulnathan$ gedit udp.tcl
vboxuser@ubu3:~/gokulnathan$ ns udp.tcl
vboxuser@ubu3:~/gokulnathan$ 
```

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 5

Lab Code / Lab	**: U18CSI5201L/ COMPUTER NETWORKS LABORATORY**
Course / Branch	**: III BE CSE**
Title of the exercise	**: Performance analysis of routing protocols using simulation tool.**

LINK STATE ROUTING PROTOCOL AIM:

To simulate a link failure and to observe link state routing protocol in action.

ALGORITHM:

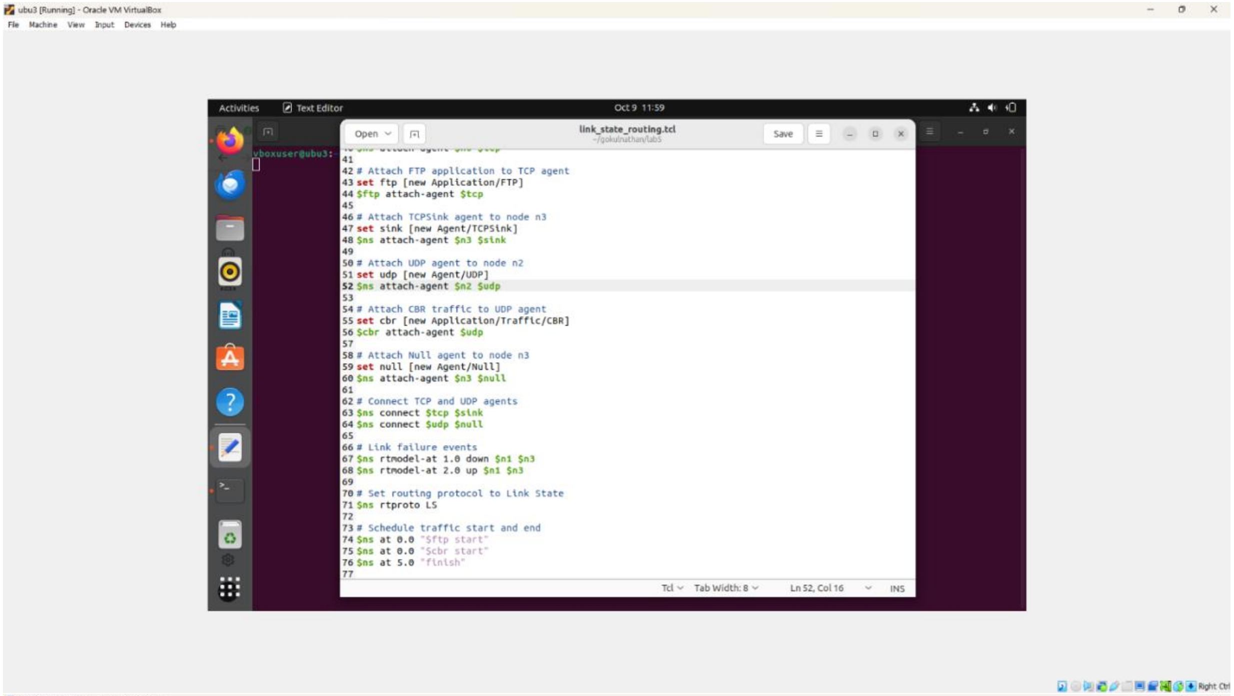
1. Create a simulator object
2. Set routing protocol to link state routing
3. Trace packets on all links onto NAM trace and text trace file
4. Define finish procedure to close files, flush tracing and run NAM
5. Create four nodes
6. Specify the link characteristics between nodes
7. Describe their layout topology as a quad node.
8. Add TCP agent for node n0
9. Create FTP traffic on top of TCP and set traffic parameters.
10. Add a sink agent to node n3
11. Add UDP agent for node n2
12. Create CBR traffic on top of UDP and set traffic parameters.
13. Connect source and the sink
14. Schedule events as follows:
    a. Start traffic flow at 0.0
    b. Down the link n1-n3 at 1.0
    c. Up the link n1-n3 at 2.0
    d. Call finish procedure at 5.0
15. Start the scheduler
16. Observe the traffic route when link is up and down

17. View the simulated events and trace file analyze it
18. Stop

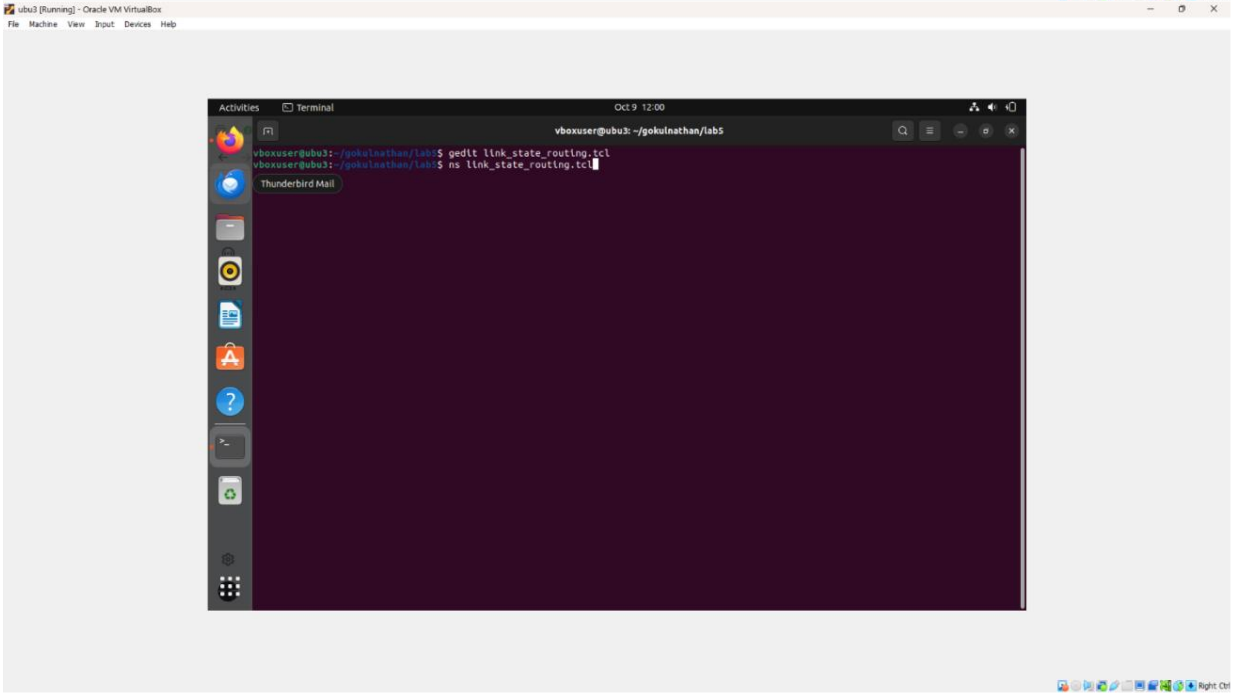
PROGRAM :

```
set ns [new Simulator] set nf [open out.nam w]
$ns namtrace-all $nfset tr [open out.tr w]
$ns trace-all $trproc finish {} { global nf ns tr
$ns flush-traceclose $tr exec nam out.nam &exit 0
}
set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node]
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n1 $n3 10Mb 10ms DropTail
$ns duplex-link $n2 $n1 10Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right-down
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n2 $n1 orient right-upset tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp set ftp [new Application/FTP]
$ftp attach-agent $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sinkset udp [new Agent/UDP]
$ns attach-agent $n2 $udp
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp set null [new Agent/Null]$ns attach-agent $n3 $null
$ns connect $tcp $sink
$ns connect $udp $null
$ns rtmodel-at 1.0 down $n1 $n3
$ns rtmodel-at 2.0 up $n1 $n3
$ns rtproto LS
$ns at 0.0 "$ftp start"
$ns at 0.0 "$cbr start"
$ns at 5.0 "finish"
$ns run
```

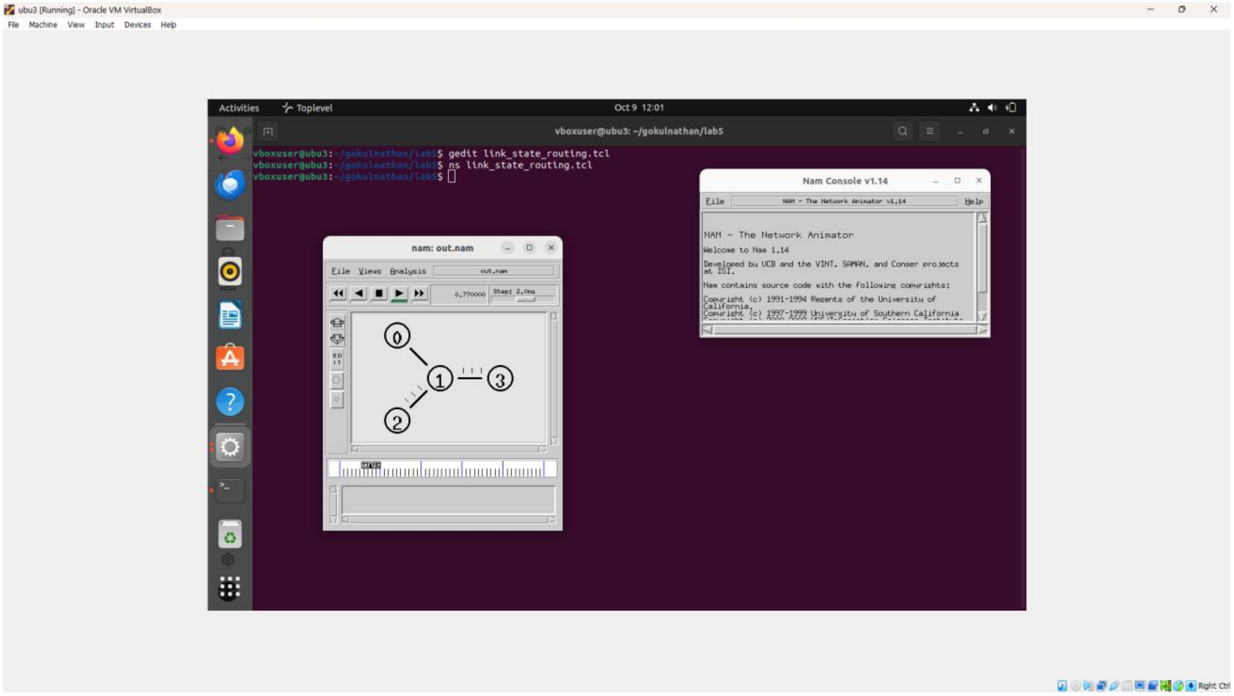
OUTPUT:



```

# Attach FTP application to TCP agent
set ftp [new Application/FTP]
$ftp attach-agent $tcp

# Attach TCPSink agent to node n3
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink

# Attach UDP agent to node n2
set udp [new Agent/UDP]
$ns attach-agent $n2 $udp

# Attach CBR traffic to UDP agent
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp

# Attach Null agent to node n3
set null [new Agent/Null]
$ns attach-agent $n3 $null

# Connect TCP and UDP agents
$ns connect $tcp $sink
$ns connect $udp $null

# Link failure events
$ns rtmodel-at 1.0 down $n1 $n3
$ns rtmodel-at 2.0 up $n1 $n3

# Set routing protocol to Link State
$ns rtproto LS

# Schedule traffic start and end
$ns at 0.0 "$ftp start"
$ns at 0.0 "$cbr start"
$ns at 5.0 "finish"

```


ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Oct 9 12:00
vboxuser@ubu3: ~/gokulnathan/lab5
Thunderbird Mail
Right Ctrl

```
vboxuser@ubu3:~/gokulnathan/lab5$ gedit link_state_routing.tcl
vboxuser@ubu3:~/gokulnathan/lab5$ ns link_state_routing.tcl
```



ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Toplevel
Oct 9 12:01
vboxuser@ubu3: ~/gokulnathan/lab5
nam: out.nam
File Views Analysis
out.nam
0.770000
Step: 2.0ms
0
1
2
3
Nam Console v1.14
File
NAM - The Network Animator v1.14
Help
Right Ctrl

```
vboxuser@ubu3:~/gokulnathan/lab5$ gedit link_state_routing.tcl
vboxuser@ubu3:~/gokulnathan/lab5$ ns link_state_routing.tcl
vboxuser@ubu3:~/gokulnathan/lab5$ 
```

```
NAM - The Network Animator

Welcome to Nam 1.14

Developed by UCB and the VINT, SAMAN, and Conser projects
at ISI.

Nam contains source code with the following copyrights:

Copyright (c) 1991-1994 Regents of the University of
California.
Copyright (c) 1997-1999 University of Southern California
```

DISTANCE VECTOR ROUTINGPROTOCOL AIM:

To simulate a link failure and to observe distance vector routing protocol in action.

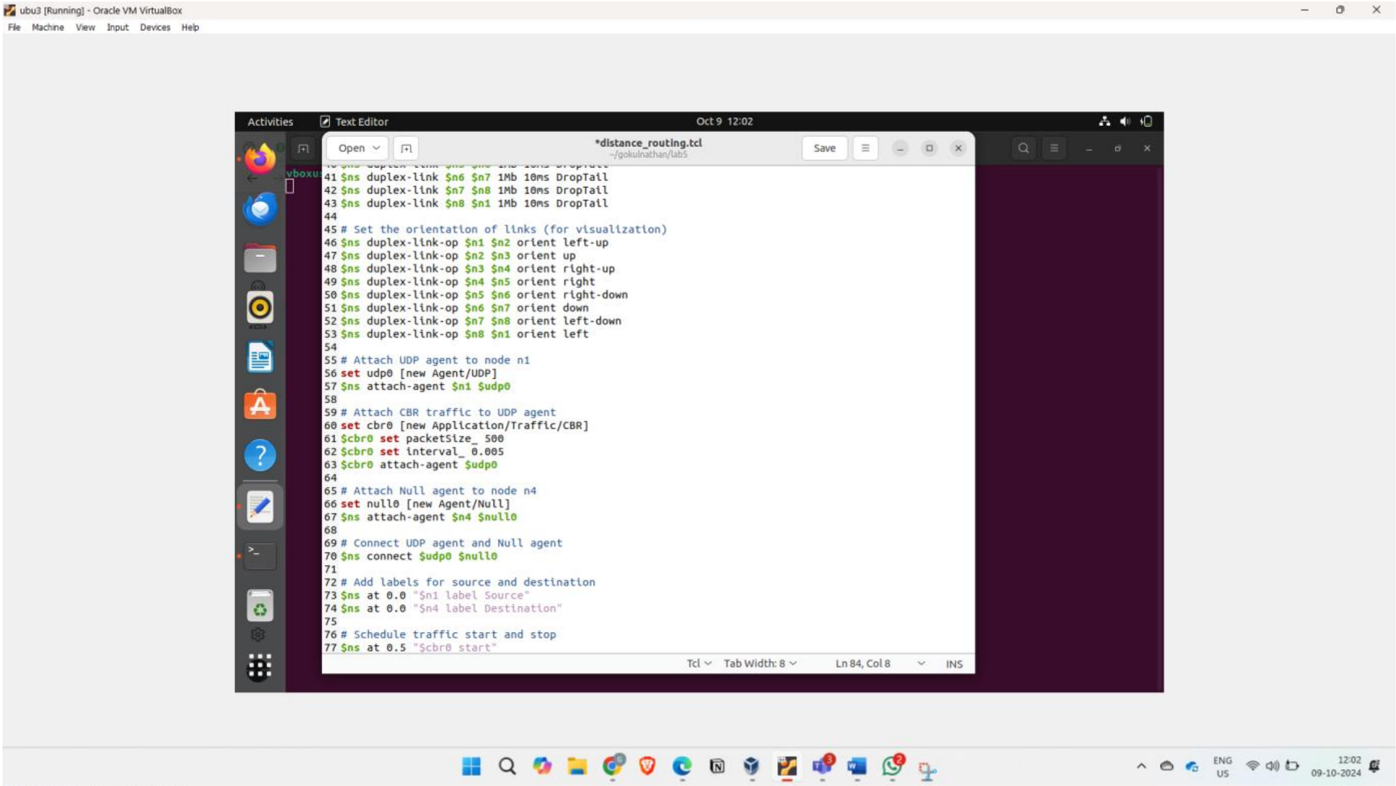
ALGORITHM:

1. Create a simulator object
2. Set routing protocol to Distance Vector routing
1. Trace packets on all links onto NAM trace and text trace file
2. Define finish procedure to close files, flush tracing and run NAM
3. Create eight nodes
4. Specify the link characteristics between nodes
5. Describe their layout topology as a octagon
6. Add UDP agent for node n1
7. Create CBR traffic on top of UDP and set traffic parameters.
8. Add a sink agent to node n4
9. Connect source and the sink
10. Schedule events as follows:
    a. Start traffic flow at 0.5
    b. Down the link n3-n4 at 1.0
    c. Up the link n3-n4 at 2.0
    d. Stop traffic at 3.0
    e. Call finish procedure at 5.0
11. Start the scheduler
12. Observe the traffic route when link is up and down
13. View the simulated events and trace file analyze it
14. Stop the program.

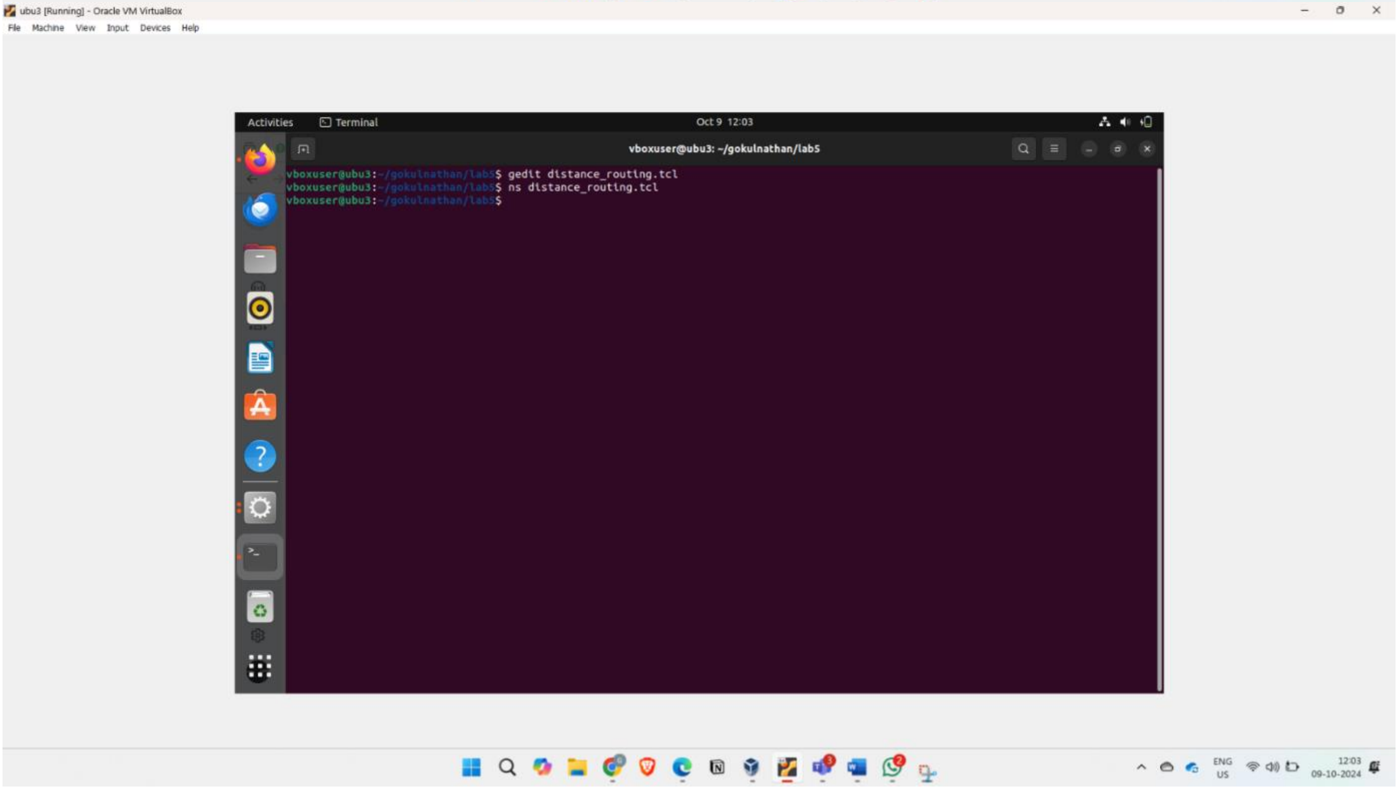
PROGRAM:

```
set ns [new Simulator]
$ns rtproto DV
set nf [open out.nam w]
$ns namtrace-all $nf set nt [open trace.tr w]
$ns trace-all $ntproc finish {} { global ns nf
$ns flush-traceclose $nf
exec nam -a out.nam &exit 0
}
set n1 [$ns node]set n2 [$ns node]set n3 [$ns node]set n4 [$ns node]set n5 [$ns
node]set n6 [$ns node]set n7 [$ns node]set n8 [$ns node]
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n6 1Mb 10ms DropTail
$ns duplex-link $n6 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n8 1Mb 10ms DropTail
$ns duplex-link $n8 $n1 1Mb 10ms DropTail
$ns duplex-link-op $n1 $n2 orient left-up
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right-down
$ns duplex-link-op $n6 $n7 orient down
$ns duplex-link-op $n7 $n8 orient left-down
$ns duplex-link-op $n8 $n1 orient leftset udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0set null0 [new Agent/Null]
$ns attach-agent $n4 $null0
$ns connect $udp0 $null0
$ns at 0.0 "$n1 label Source"
$ns at 0.0 "$n4 label Destination"
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n3 $n4
$ns rtmodel-at 2.0 up $n3 $n4
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

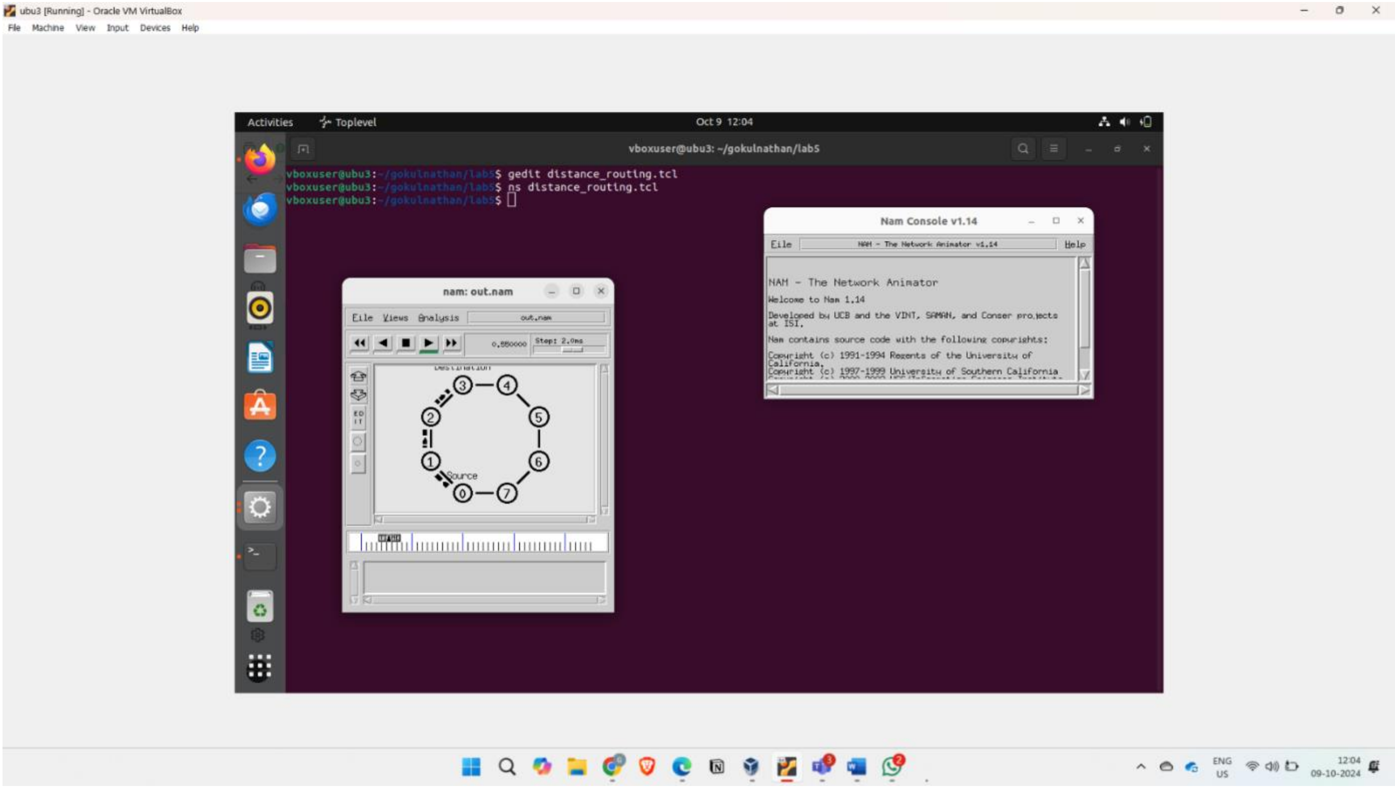
OUTPUT:



```
$ns duplex-link $n6 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n8 1Mb 10ms DropTail
$ns duplex-link $n8 $n1 1Mb 10ms DropTail

# Set the orientation of links (for visualization)
$ns duplex-link-op $n1 $n2 orient left-up
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right-down
$ns duplex-link-op $n6 $n7 orient down
$ns duplex-link-op $n7 $n8 orient left-down
$ns duplex-link-op $n8 $n1 orient left

# Attach UDP agent to node n1
set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0

# Attach CBR traffic to UDP agent
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

# Attach Null agent to node n4
set null0 [new Agent/Null]
$ns attach-agent $n4 $null0

# Connect UDP agent and Null agent
$ns connect $udp0 $null0

# Add labels for source and destination
$ns at 0.0 "$n1 label Source"
$ns at 0.0 "$n4 label Destination"

# Schedule traffic start and stop
$ns at 0.5 "$cbr0 start"
```


ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Oct 9 12:03
vboxuser@ubu3: ~/gokulnathan/lab5
ENG
US
12:03
09-10-2024

```
vboxuser@ubu3:~/gokulnathan/lab5$ gedit distance_routing.tcl
vboxuser@ubu3:~/gokulnathan/lab5$ ns distance_routing.tcl
vboxuser@ubu3:~/gokulnathan/lab5$
```



```
vboxuser@ubu3:~/gokulnathan/lab5$ gedit distance_routing.tcl
vboxuser@ubu3:~/gokulnathan/lab5$ ns distance_routing.tcl
vboxuser@ubu3:~/gokulnathan/lab5$ 
```

```
NAM - The Network Animator
Welcome to Nam 1.14
Developed by UCB and the VINT, SAMAN, and Conser projects
at ISI.
Nam contains source code with the following copyrights:
Copyright (c) 1991-1994 Regents of the University of
California.
Copyright (c) 1997-1999 University of Southern California
```

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 6

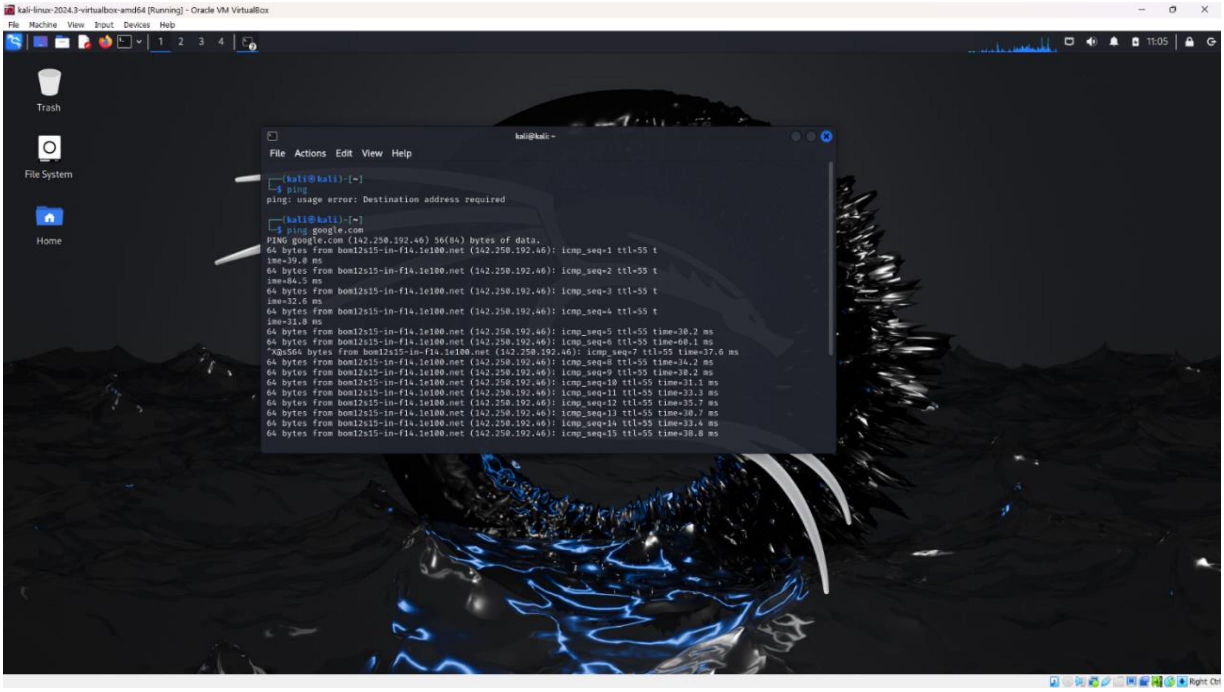
Lab Code / Lab	**: U18CSI5201L/ COMPUTER NETWORKS LABORATORY**
Course / Branch	**: III BE CSE**
Title of the exercise TCP	**: Demonstrate the working of network tools such as Ping, Dump, Traceroute, Netstat, Ipconfig.**

NETWORKING COMMANDS:

EXPERIMENT 6 : Demonstrate the working of network tools such as Ping,TCP Dump, Traceroute, Netstat, Ipconfig.

Ping:

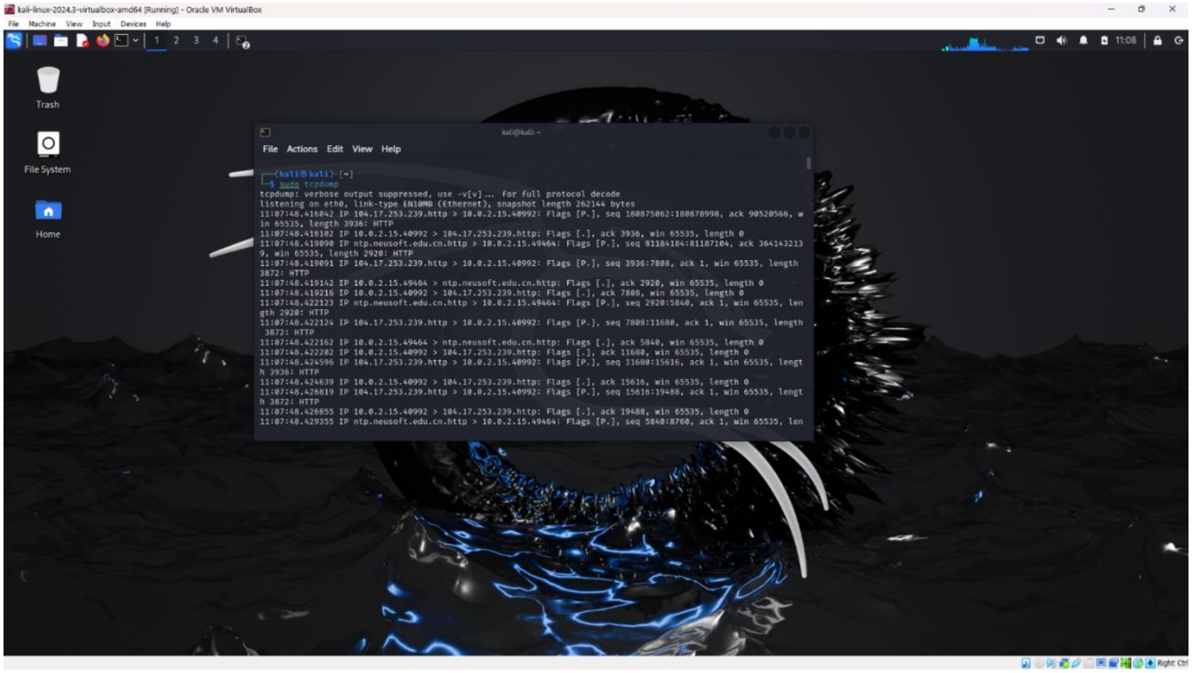
The ping command is a general utility which is used for checking whether any network ispresent and if a host is attainable. We can test if the server is up and executing using this command. Also, it helps several connectivity issues with troubleshooting.


kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
11:05
Trash
File System
Home
kali@kali: ~
File Actions Edit View Help
Right Ctrl

```
┌──(kali㉿kali)-[~]
└─$ ping
ping: usage error: Destination address required

┌──(kali㉿kali)-[~]
└─$ ping google.com
PING google.com (142.250.192.46) 56(84) bytes of data.
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=1 ttl=55 t
ime=39.0 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=2 ttl=55 t
ime=84.5 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=3 ttl=55 t
ime=32.6 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=4 ttl=55 t
ime=31.8 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=5 ttl=55 time=30.2 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=6 ttl=55 time=60.1 ms
^X@sS64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=7 ttl=55 time=37.6 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=8 ttl=55 time=34.2 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=9 ttl=55 time=30.2 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=10 ttl=55 time=31.1 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=11 ttl=55 time=33.3 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=12 ttl=55 time=35.7 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=13 ttl=55 time=30.7 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=14 ttl=55 time=33.4 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=15 ttl=55 time=38.8 ms
```

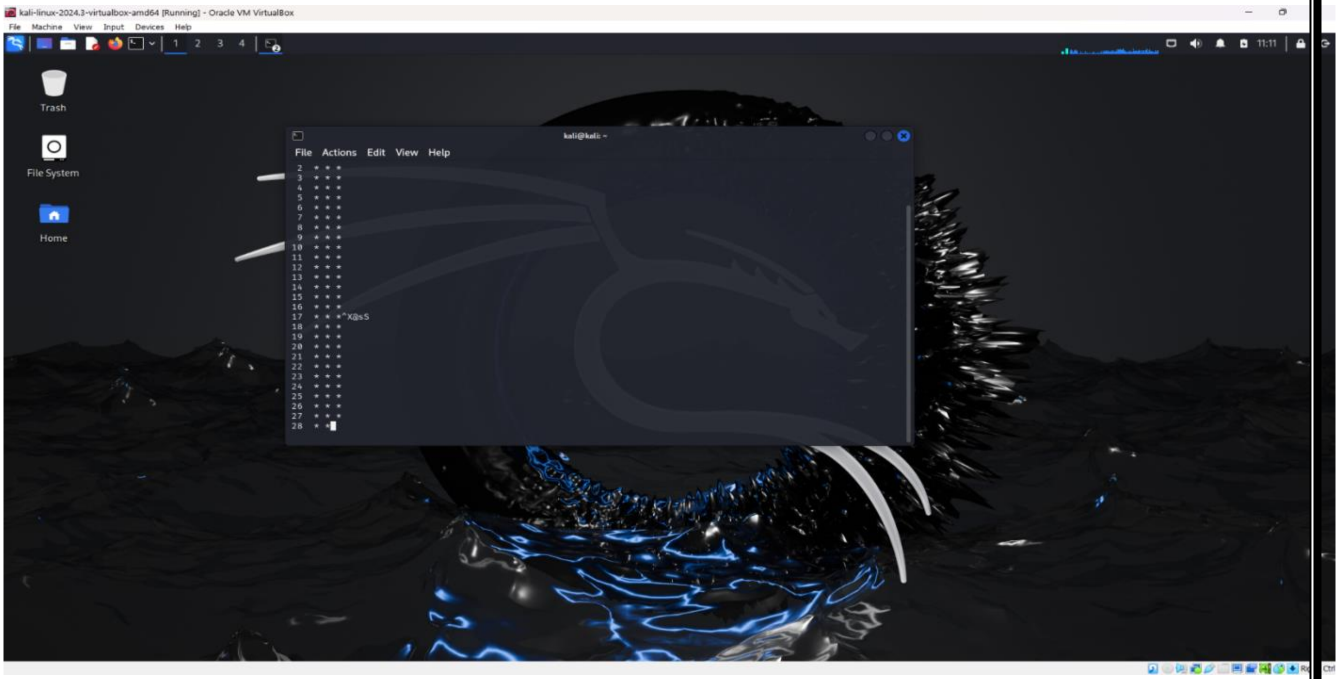
Tcpdump:

Tcpdump is a packet analyzer that is launched from the command line. It can be used to analyze network traffic by intercepting and displaying packets that are being created or received by the computer it's running on. It runs on Linux and most UNIX-type operating systems.


kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
File System
Home
kali@kali: ~
File Actions Edit View Help
11:08
Right Ctrl

```
┌──(kali㉿kali)-[~]
└─$ sudo tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
11:07:48.416042 IP 104.17.253.239.http > 10.0.2.15.40992: Flags [P.], seq 180875062:180878998, ack 90520566, w
in 65535, length 3936: HTTP
11:07:48.416102 IP 10.0.2.15.40992 > 104.17.253.239.http: Flags [.], ack 3936, win 65535, length 0
11:07:48.419090 IP ntp.neusoft.edu.cn.http > 10.0.2.15.49464: Flags [P.], seq 81184184:81187104, ack 364143213
9, win 65535, length 2920: HTTP
11:07:48.419091 IP 104.17.253.239.http > 10.0.2.15.40992: Flags [P.], seq 3936:7808, ack 1, win 65535, length
3872: HTTP
11:07:48.419142 IP 10.0.2.15.49464 > ntp.neusoft.edu.cn.http: Flags [.], ack 2920, win 65535, length 0
11:07:48.419216 IP 10.0.2.15.40992 > 104.17.253.239.http: Flags [.], ack 7808, win 65535, length 0
11:07:48.422123 IP ntp.neusoft.edu.cn.http > 10.0.2.15.49464: Flags [P.], seq 2920:5840, ack 1, win 65535, len
gth 2920: HTTP
11:07:48.422124 IP 104.17.253.239.http > 10.0.2.15.40992: Flags [P.], seq 7808:11680, ack 1, win 65535, length
 3872: HTTP
11:07:48.422162 IP 10.0.2.15.49464 > ntp.neusoft.edu.cn.http: Flags [.], ack 5840, win 65535, length 0
11:07:48.422202 IP 10.0.2.15.40992 > 104.17.253.239.http: Flags [.], ack 11680, win 65535, length 0
11:07:48.424596 IP 104.17.253.239.http > 10.0.2.15.40992: Flags [P.], seq 11680:15616, ack 1, win 65535, lengt
h 3936: HTTP
11:07:48.424639 IP 10.0.2.15.40992 > 104.17.253.239.http: Flags [.], ack 15616, win 65535, length 0
11:07:48.426819 IP 104.17.253.239.http > 10.0.2.15.40992: Flags [P.], seq 15616:19488, ack 1, win 65535, lengt
h 3872: HTTP
11:07:48.426855 IP 10.0.2.15.40992 > 104.17.253.239.http: Flags [.], ack 19488, win 65535, length 0
11:07:48.429355 IP ntp.neusoft.edu.cn.http > 10.0.2.15.49464: Flags [P.], seq 5840:8760, ack 1, win 65535, len
```

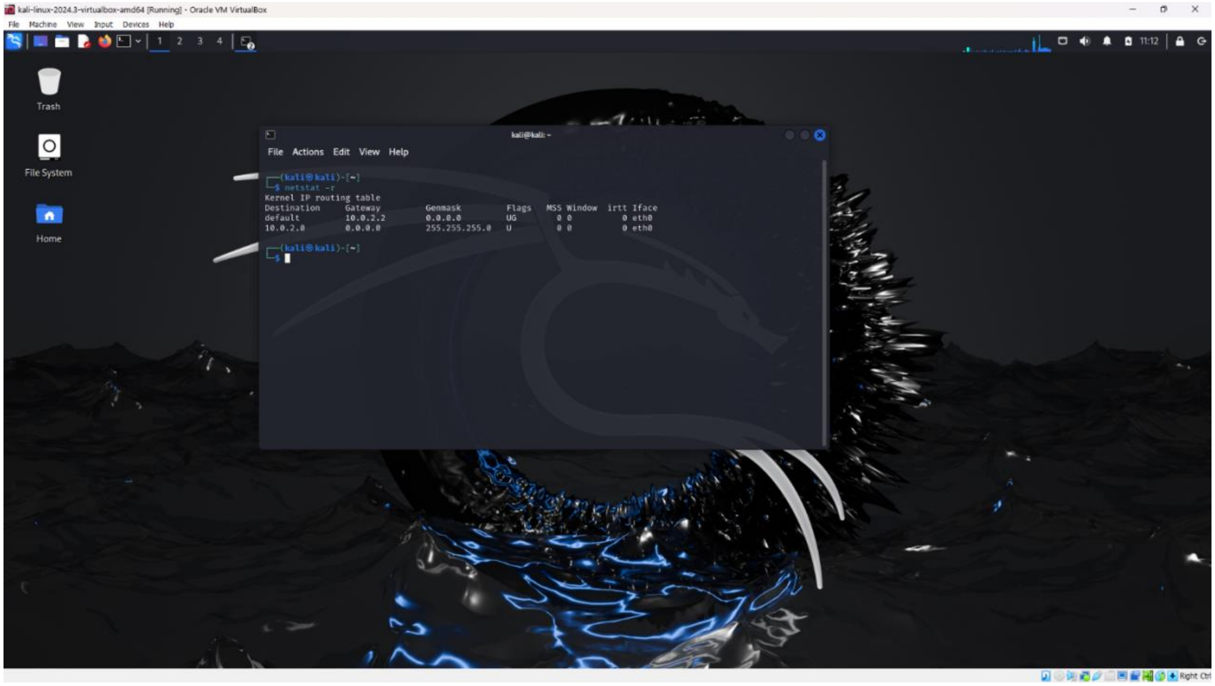
Traceroute:
The traceroute command attempts to trace the route an IP packet follows to an Internet host by launching UDP probe packets with a small maximum time-to-live(Max_ttl variable), then listening for an ICMP TIME_EXCEEDED response from gateways along the way.


kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
File System
Home
kali@kali: ~
File Actions Edit View Help

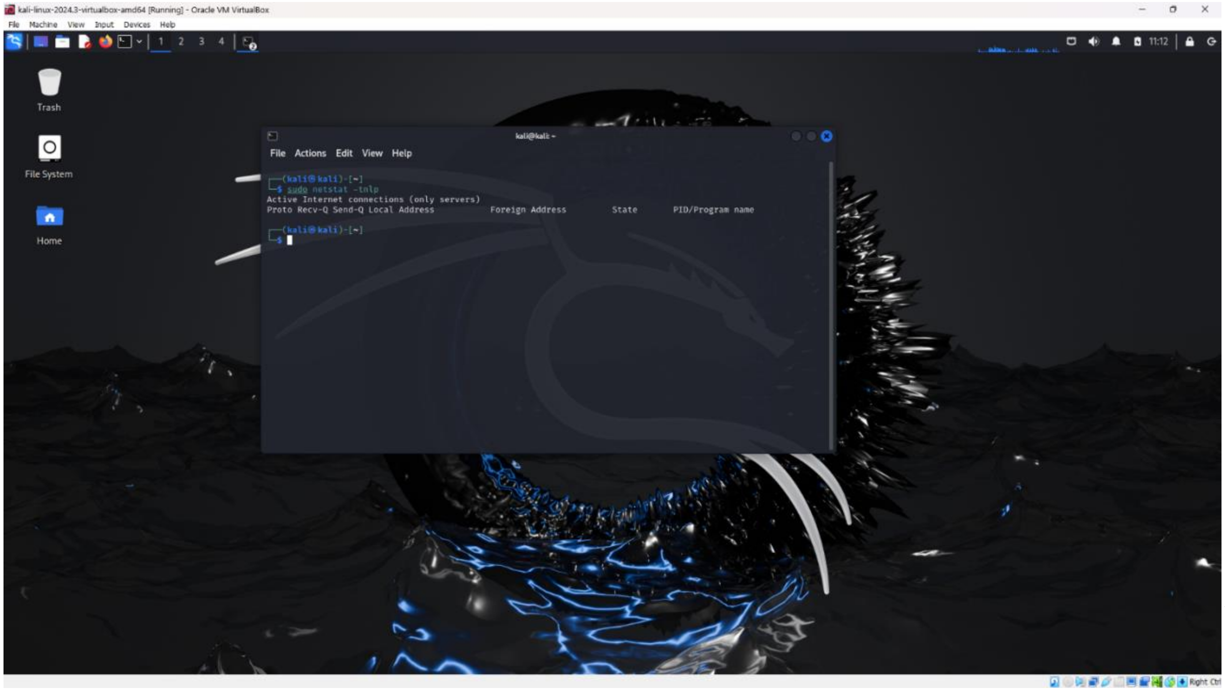
```
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *^X@sS
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * *
```

Netstat:

The network statistics (netstat) command is a networking tool used for troubleshooting and configuration, that can also serve as a monitoring tool for connections over the network. Both incoming and outgoing connections, routingtables, port listening, and usage statistics are common uses for this command.



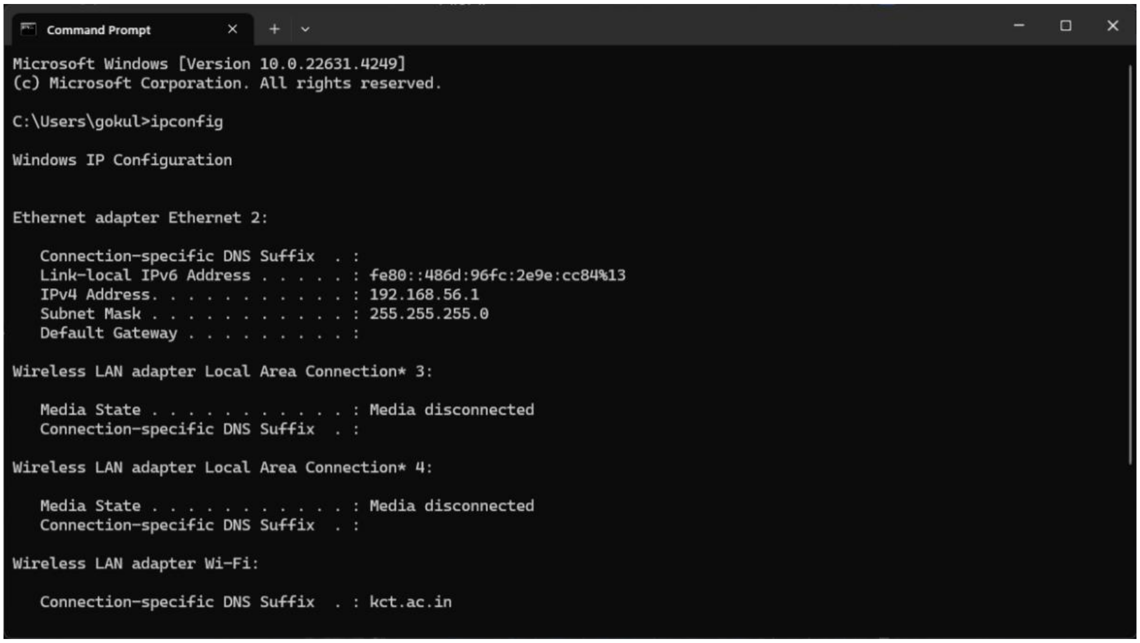
```
┌──(kali㉿kali)-[~]
└─$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.2.2        0.0.0.0         UG        0 0          0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0

┌──(kali㉿kali)-[~]
└─$ 
```



```
┌──(kali㉿kali)-[~]
└─$ sudo netstat -tnlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name

┌──(kali㉿kali)-[~]
└─$ 
```

Ipconfig:

Ipconfig displays useful information such as IP address, subnet mask and thedefault gateway for all of the different network connections in the computer.



```
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\gokul>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::486d:96fc:2e9e:cc84%13
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 4:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : kct.ac.in
```

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 7

Lab Code / Lab	**: U18CSI5201L/ COMPUTER NETWORKS LABORATORY**
Course / Branch	**: III BE CSE**
Title of the exercise	**: Analyze the network traffic using Wireshark tool/Packet tracer tool.**

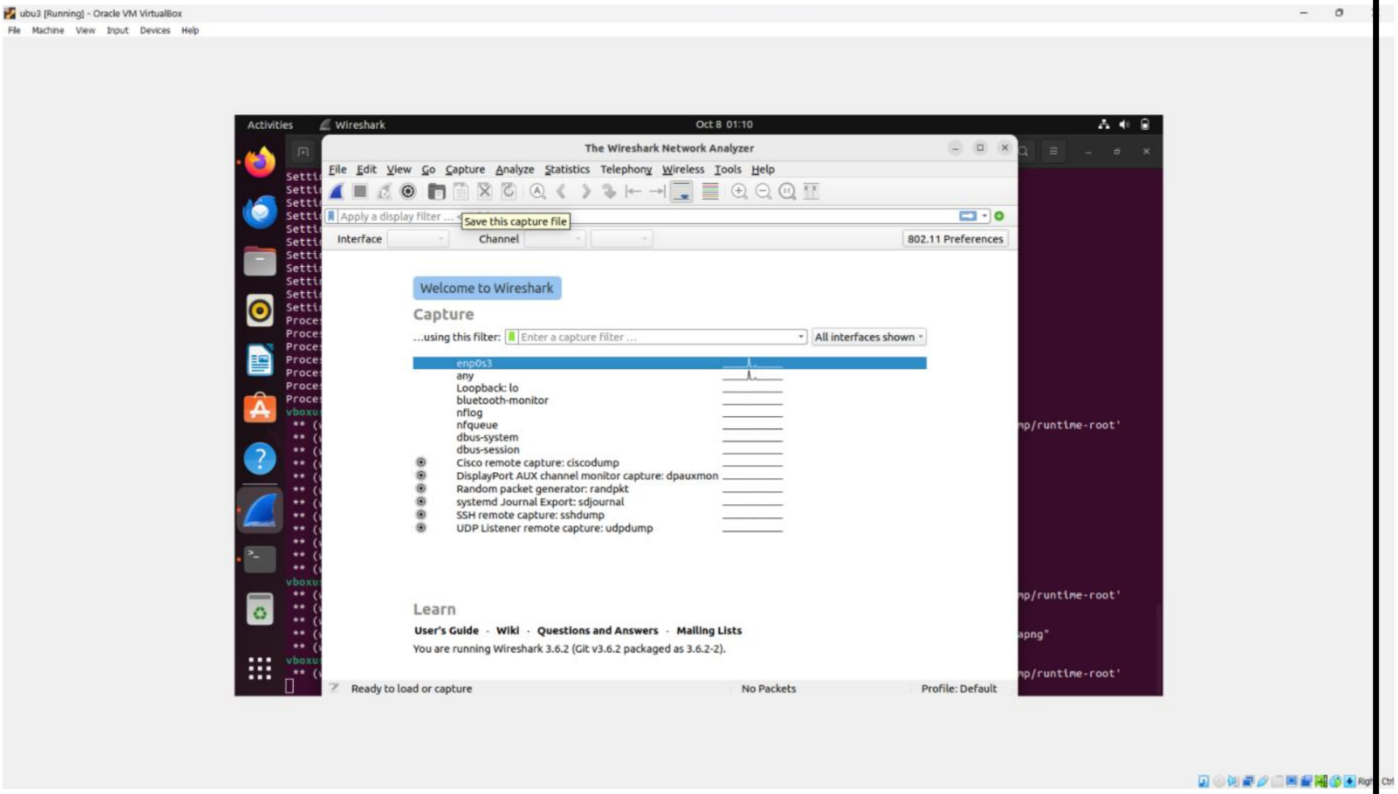
AIM : To know how to capture packets in wireshark

THEORY:

Wireshark is the world's foremost and widely used network protocol analyser. It lets you see what is happening on your network at a microscopic level.
Wireshark has a rich feature set which includes the following:

- Deep inspection of hundreds of protocols, with more being added all the time
- Live capture and offline analysis
- Capture files compressed with gzip can be decompressed on the fly
- Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others

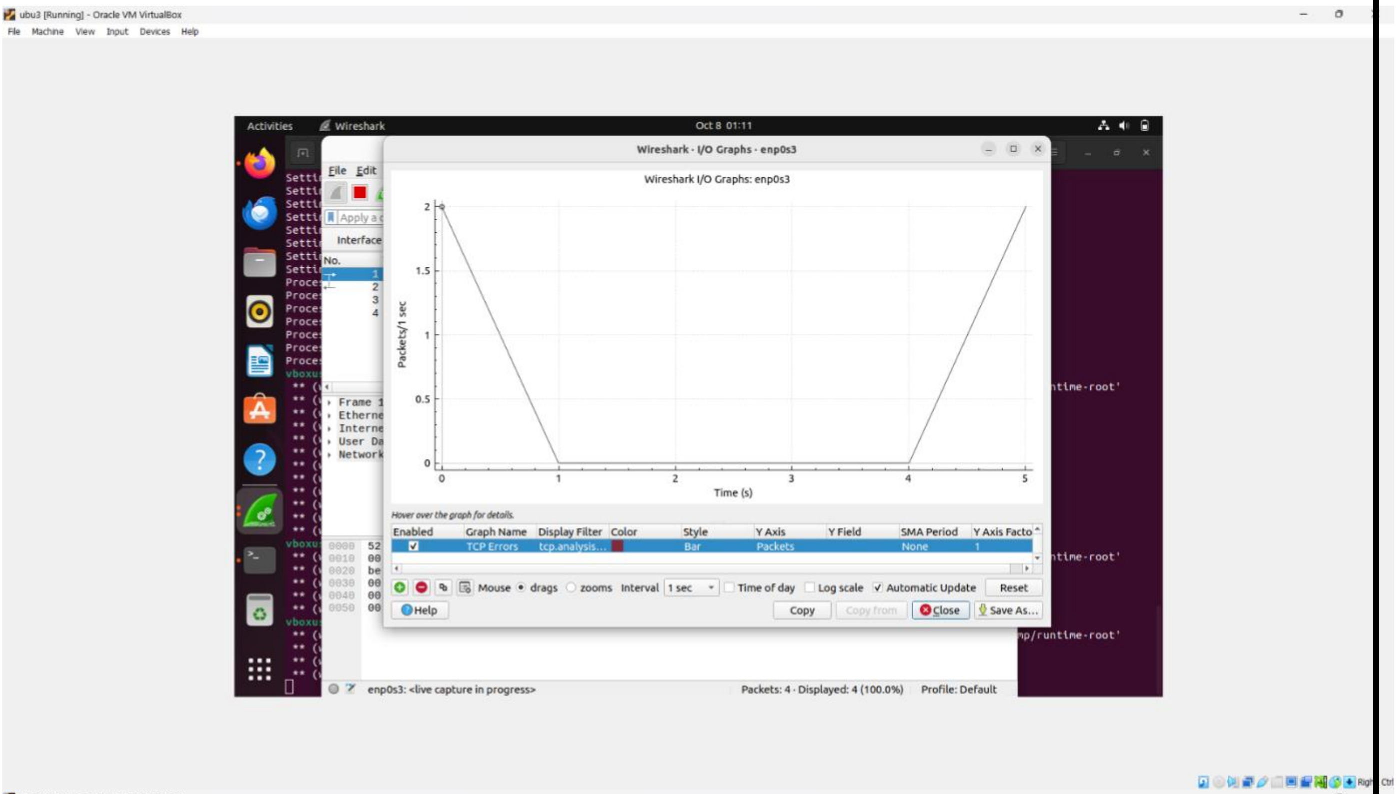
OUTPUT :


ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Wireshark
Oct 8 01:10
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ...
Save this capture file
Interface
Channel
802.11 Preferences
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists

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

You are running Wireshark 3.6.2 (Git v3.6.2 packaged as 3.6.2-2).

Ready to load or capture   No Packets   Profile: Default



ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Wireshark
Oct 8 01:11
Wireshark · I/O Graphs · enp0s3
Wireshark I/O Graphs: enp0s3
Packets/1 sec
Time (s)
0
0.5
1
1.5
2
0
1
2
3
4
5

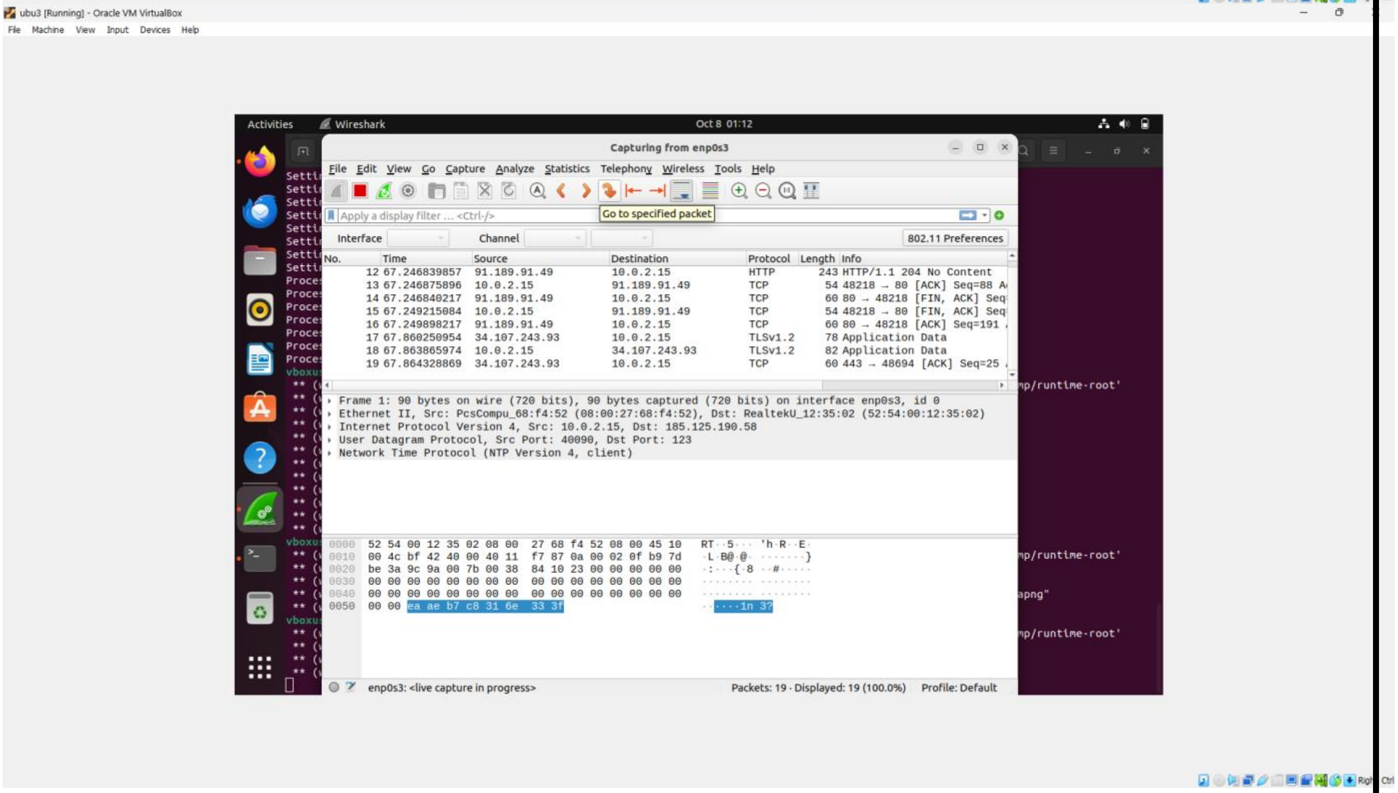
Hover over the graph for details.

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Facto
☑	TCP Errors	tcp.analysis…	🟥	Bar	Packets		None	1

Mouse ● drags ○ zooms Interval 1 sec □ Time of day □ Log scale ☑ Automatic Update

Reset
Help
Copy
Copy from
Close
Save As…

enp0s3: <live capture in progress>   Packets: 4 · Displayed: 4 (100.0%)   Profile: Default



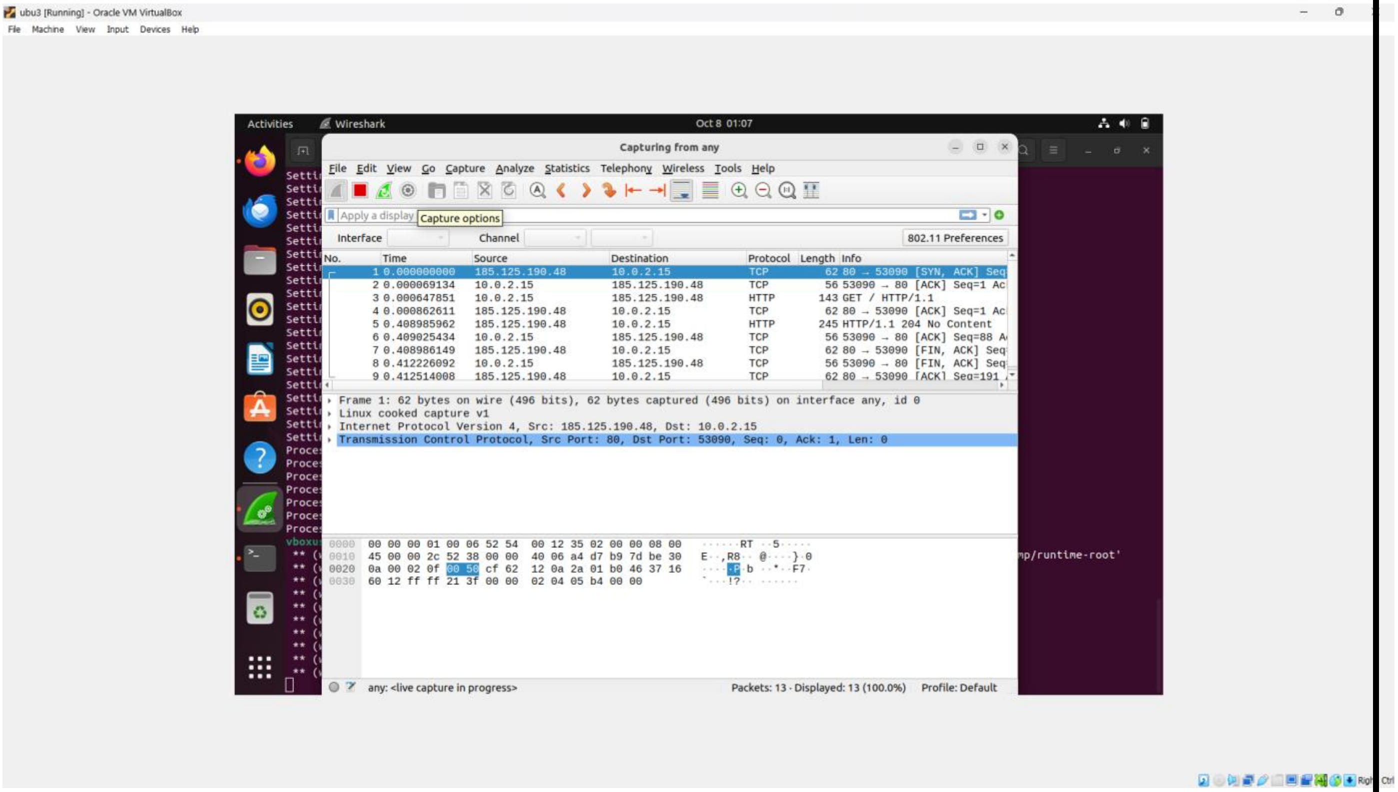
ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Wireshark
Oct 8 01:12
Capturing from enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Go to specified packet
Apply a display filter … <Ctrl-/>
Interface
Channel
802.11 Preferences

```
No.   Time          Source          Destination     Protocol Length Info
   12 67.246839857  91.189.91.49    10.0.2.15       HTTP      243 HTTP/1.1 204 No Content
   13 67.246875896  10.0.2.15       91.189.91.49    TCP        54 48218 → 80 [ACK] Seq=88 A
   14 67.246840217  91.189.91.49    10.0.2.15       TCP        60 80 → 48218 [FIN, ACK] Seq
   15 67.249215084  10.0.2.15       91.189.91.49    TCP        54 48218 → 80 [FIN, ACK] Seq
   16 67.249898217  91.189.91.49    10.0.2.15       TCP        60 80 → 48218 [ACK] Seq=191
   17 67.860250954  34.107.243.93   10.0.2.15       TLSv1.2    78 Application Data
   18 67.863865974  10.0.2.15       34.107.243.93   TLSv1.2    82 Application Data
   19 67.864328869  34.107.243.93   10.0.2.15       TCP        60 443 → 48694 [ACK] Seq=25
```

```
› Frame 1: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface enp0s3, id 0
› Ethernet II, Src: PcsCompu_68:f4:52 (08:00:27:68:f4:52), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
› Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.125.190.58
› User Datagram Protocol, Src Port: 40090, Dst Port: 123
› Network Time Protocol (NTP Version 4, client)
```

```
0000  52 54 00 12 35 02 08 00  27 68 f4 52 08 00 45 10   RT··5···'h·R··E·
0010  00 4c bf 42 40 00 40 11  f7 87 0a 00 02 0f b9 7d   ·L·B@·@·········}
0020  be 3a 9c 9a 00 7b 00 38  84 10 23 00 00 00 00 00   ·:···{·8··#·····
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ················
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ················
0050  00 00 ea ae b7 c8 31 6e  33 3f                     ······1n 3?
```

enp0s3: <live capture in progress>   Packets: 19 · Displayed: 19 (100.0%)   Profile: Default


ubu3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Wireshark
Oct 8 01:07
Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display
Capture options
Interface
Channel
802.11 Preferences

```
No.  Time          Source           Destination      Protocol Length Info
   1 0.000000000   185.125.190.48   10.0.2.15        TCP        62 80 → 53090 [SYN, ACK] Seq
   2 0.000069134   10.0.2.15        185.125.190.48   TCP        56 53090 → 80 [ACK] Seq=1 Ac
   3 0.000647851   10.0.2.15        185.125.190.48   HTTP      143 GET / HTTP/1.1
   4 0.000862611   185.125.190.48   10.0.2.15        TCP        62 80 → 53090 [ACK] Seq=1 Ac
   5 0.408985962   185.125.190.48   10.0.2.15        HTTP      245 HTTP/1.1 204 No Content
   6 0.409025434   10.0.2.15        185.125.190.48   TCP        56 53090 → 80 [ACK] Seq=88 A
   7 0.408986149   185.125.190.48   10.0.2.15        TCP        62 80 → 53090 [FIN, ACK] Seq
   8 0.412226092   10.0.2.15        185.125.190.48   TCP        56 53090 → 80 [FIN, ACK] Seq
   9 0.412514008   185.125.190.48   10.0.2.15        TCP        62 80 → 53090 [ACK] Seq=191
```

```
› Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface any, id 0
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 185.125.190.48, Dst: 10.0.2.15
› Transmission Control Protocol, Src Port: 80, Dst Port: 53090, Seq: 0, Ack: 1, Len: 0
```

```
0000  00 00 00 01 00 06 52 54  00 12 35 02 00 00 08 00   ······RT ··5·····
0010  45 00 00 2c 52 38 00 00  40 06 a4 d7 b9 7d be 30   E··,R8·· @····}·0
0020  0a 00 02 0f 00 50 cf 62  12 0a 2a 01 b0 46 37 16   ·····P·b ··*··F7·
0030  60 12 ff ff 21 3f 00 00  02 04 05 b4 00 00         `···!?·· ······
```

any: <live capture in progress>   Packets: 13 · Displayed: 13 (100.0%)   Profile: Default