REPORT ON COMPUTER NETWORKS LAB(U18CSI5201L)

Submitted by
DHEERAJ VERMA (22BCS025)

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

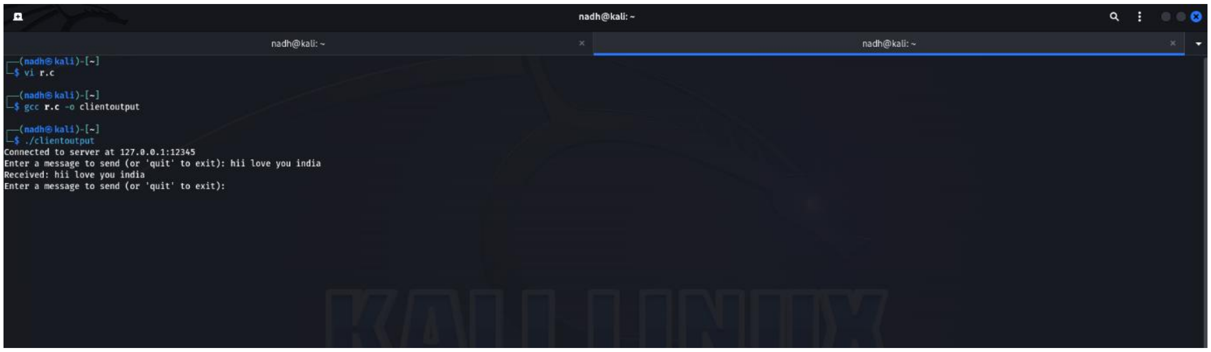
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#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];

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

    // Set up the server address struct
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = INADDR_ANY;

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

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

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

    // Accept a connection from a client
    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");

    // Receive and echo data
    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("Received: %s", buffer);

        // Echo the received data back to the client
        send(client_socket, buffer, strlen(buffer), 0);
    }

    // Close sockets
    close(client_socket);
    close(server_socket);

    return 0;
}
```

CLIENT

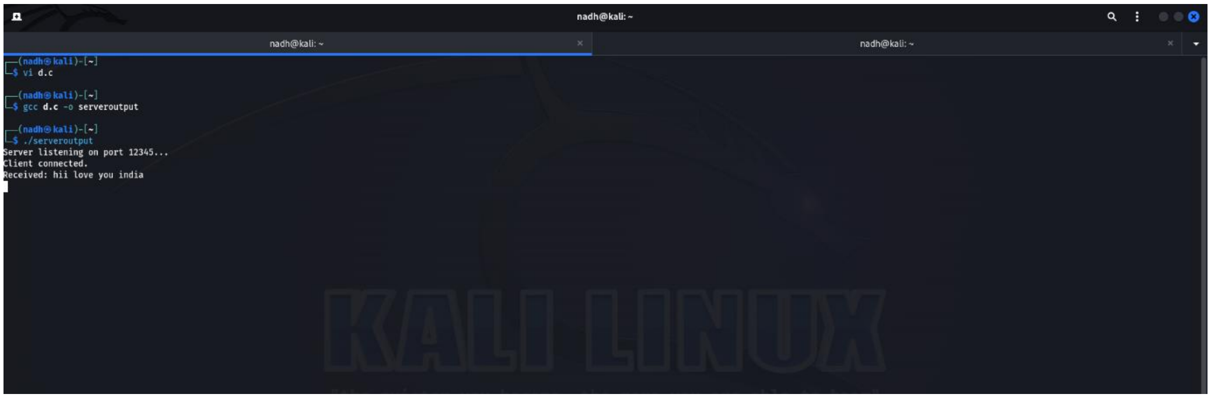
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
```

```
#define SERVER_IP "127.0.0.1" // 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];

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

    // Set up the server address struct
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);



    // Connect to the server
    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);



    // Send and receive data
    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 the message to the server
        send(client_socket, buffer, strlen(buffer), 0);
        // Receive and print the echo from the server
        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("Received: %s", buffer);
    }

    // Close the socket
    close(client_socket);

    return 0;
}
```

OUTPUT



```
┌──(nadh㉿kali)-[~]
└─$ vi r.c

┌──(nadh㉿kali)-[~]
└─$ gcc r.c -o clientoutput

┌──(nadh㉿kali)-[~]
└─$ ./clientoutput
Connected to server at 127.0.0.1:12345
Enter a message to send (or 'quit' to exit): hii love you india
Received: hii love you india
Enter a message to send (or 'quit' to exit):
```



```
┌──(nadh㉿kali)-[~]
└─$ vi d.c

┌──(nadh㉿kali)-[~]
└─$ gcc d.c -o serveroutput

┌──(nadh㉿kali)-[~]
└─$ ./serveroutput
Server listening on port 12345...
Client connected.
Received: hii love you india
```

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

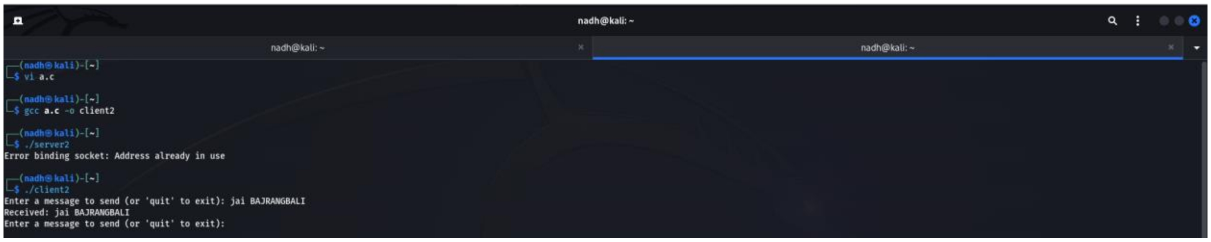
SERVER:

```
#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];

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

    // Set up the server address struct
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = INADDR_ANY;

    // Bind the socket to the server address
    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);

    // Receive and echo data
    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("Received: %s", buffer);

        // Echo the received data back to the client
        sendto(server_socket, buffer, strlen(buffer), 0, (struct sockaddr *)&client_addr, client_addr_len);
    }

    // Close the socket
    close(server_socket);

    return 0;
}
```

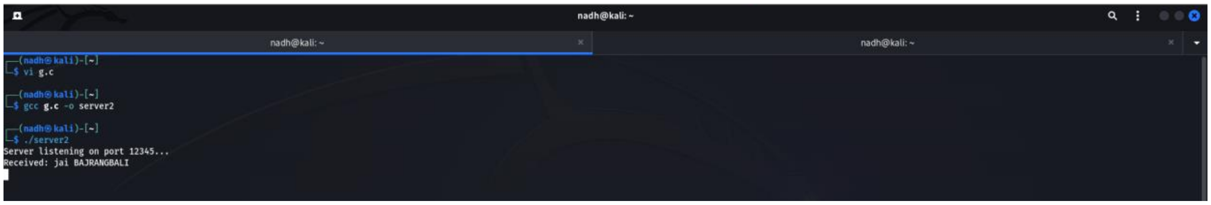
CLIENT:

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



#define SERVER_IP "127.0.0.1" // 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];

    // Create a socket
    client_socket = socket(AF_INET, SOCK_DGRAM, 0);
    if (client_socket == -1) {
        perror("Error creating socket");
        exit(1);
    }
    // Set up the server address struct
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);

    // Send and receive data
    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 the message to the server
        sendto(client_socket, buffer, strlen(buffer), 0, (struct sockaddr *)&server_addr, sizeof(server_addr));

        // Receive and print the echo from the server
        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("Received: %s", buffer);
    }

    // Close the socket
    close(client_socket);

    return 0;
}
```

OUTPUT



```
┌──(nadh㉿kali)-[~]
└─$ vi a.c

┌──(nadh㉿kali)-[~]
└─$ gcc a.c -o client2

┌──(nadh㉿kali)-[~]
└─$ ./server2
Error binding socket: Address already in use

┌──(nadh㉿kali)-[~]
└─$ ./client2
Enter a message to send (or 'quit' to exit): jai BAJRANGBALI
Received: jai BAJRANGBALI
Enter a message to send (or 'quit' to exit):
```



```
┌──(nadh㉿kali)-[~]
└─$ vi g.c

┌──(nadh㉿kali)-[~]
└─$ gcc g.c -o server2

┌──(nadh㉿kali)-[~]
└─$ ./server2
Server listening on port 12345...
Received: jai BAJRANGBALI
```

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.	PC with Linux Platform	1/Student
2.	LAN connection	

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:

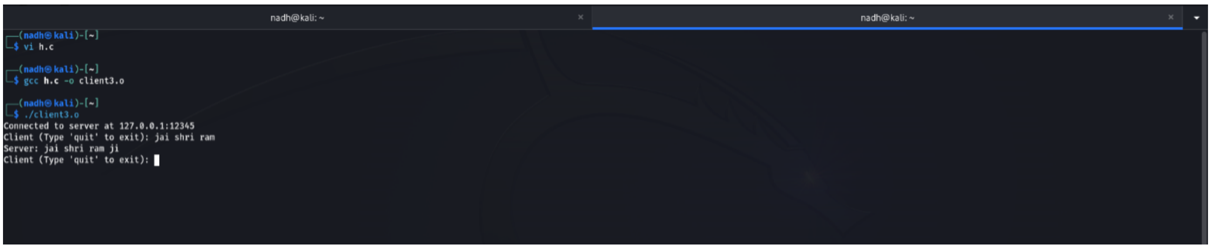
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>



#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];



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



    // Set up the server address struct
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = INADDR_ANY;



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



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



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



    // Accept a connection from a client
    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");



    // Chat loop
    while (1) {
        // Receive a message from the client
        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);



        // Prompt for a reply
        printf("Server (Type 'quit' to exit): ");
        fgets(buffer, sizeof(buffer), stdin);



        // Send the reply to the client
        send(client_socket, buffer, strlen(buffer), 0);



        // Check if the server wants to quit
        if (strcmp(buffer, "quit\n") == 0) {
            break;
        }
    }

    // Close sockets
    close(client_socket);
    close(server_socket);



    return 0;
}
```

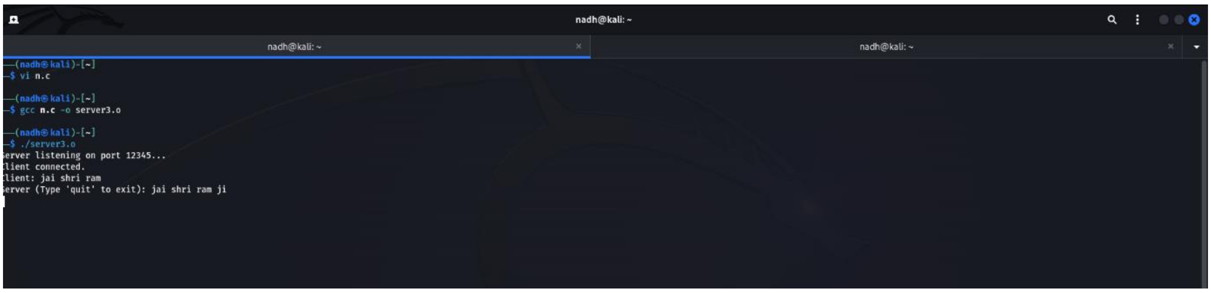
CLIENT:

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



#define SERVER_IP "127.0.0.1" // 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];



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



    // Set up the server address struct
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);



    // Connect to the server
    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);
    // Chat loop
    while (1) {
        // Prompt for a message to send
        printf("Client (Type 'quit' to exit): ");
        fgets(buffer, sizeof(buffer), stdin);



        // Send the message to the server
        send(client_socket, buffer, strlen(buffer), 0);



        // Check if the client wants to quit
        if (strcmp(buffer, "quit\n") == 0) {
            break;
        }



        // Receive a message from the server
        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 the socket
    close(client_socket);



    return 0;
}
```

OUTPUT



```
┌──(nadh㉿kali)-[~]
└─$ vi h.c

┌──(nadh㉿kali)-[~]
└─$ gcc h.c -o client3.o

┌──(nadh㉿kali)-[~]
└─$ ./client3.o
Connected to server at 127.0.0.1:12345
Client (Type 'quit' to exit): jai shri ram
Server: jai shri ram ji
Client (Type 'quit' to exit): 
```



```
──(nadh㉿kali)-[~]
─$ vi n.c

──(nadh㉿kali)-[~]
─$ gcc n.c -o server3.o

──(nadh㉿kali)-[~]
─$ ./server3.o
Server listening on port 12345...
Client connected.
Client: jai shri ram
Server (Type 'quit' to exit): jai shri ram ji
```

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

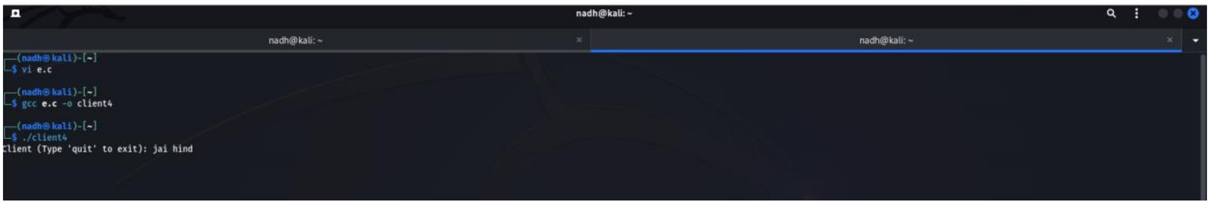
SERVER

```
#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];



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


    // Set up the server address struct
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = INADDR_ANY;


    // Bind the socket to the server address
    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);


    // Receive and send data
    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);


        // Prompt for a reply
        printf("Server (Type 'quit' to exit): ");
        fgets(buffer, sizeof(buffer), stdin);


        // Send the reply to the client
        sendto(server_socket, buffer, strlen(buffer), 0, (struct sockaddr *)&client_addr, client_addr_len);
        // Check if the server wants to quit
        if (strcmp(buffer, "quit\n") == 0) {
            break;
        }
    }



    // Close the socket
    close(server_socket);



    return 0;
}
```

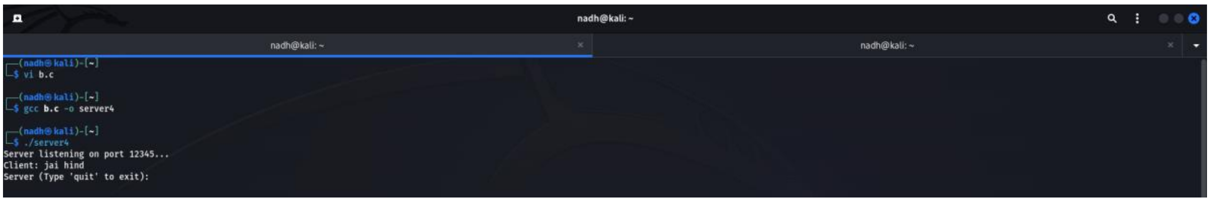
CLIENT

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



#define SERVER_IP "127.0.0.1" // 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];



    // Create a socket
    client_socket = socket(AF_INET, SOCK_DGRAM, 0);
    if (client_socket == -1) {
        perror("Error creating socket");
        exit(1);
    }
    // Set up the server address struct
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);



    // Send and receive data
    while (1) {
        // Prompt for a message to send
        printf("Client (Type 'quit' to exit): ");
        fgets(buffer, sizeof(buffer), stdin);



        // Send the message to the server
        sendto(client_socket, buffer, strlen(buffer), 0, (struct sockaddr *)&server_addr, sizeof(server_addr));



        // Check if the client wants to quit
        if (strcmp(buffer, "quit\n") == 0) {
            break;
        }



        // Receive and print the server's reply
        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 the socket
    close(client_socket);



    return 0;
}
```

OUTPUT



```
┌──(nadh㉿kali)-[~]
└─$ vi e.c

┌──(nadh㉿kali)-[~]
└─$ gcc e.c -o client4

┌──(nadh㉿kali)-[~]
└─$ ./client4
Client (Type 'quit' to exit): jai hind
```



```
┌──(nadh㉿kali)-[~]
└─$ vi b.c

┌──(nadh㉿kali)-[~]
└─$ gcc b.c -o server4

┌──(nadh㉿kali)-[~]
└─$ ./server4
Server listening on port 12345...
Client: jai hind
Server (Type 'quit' to exit):
```

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:

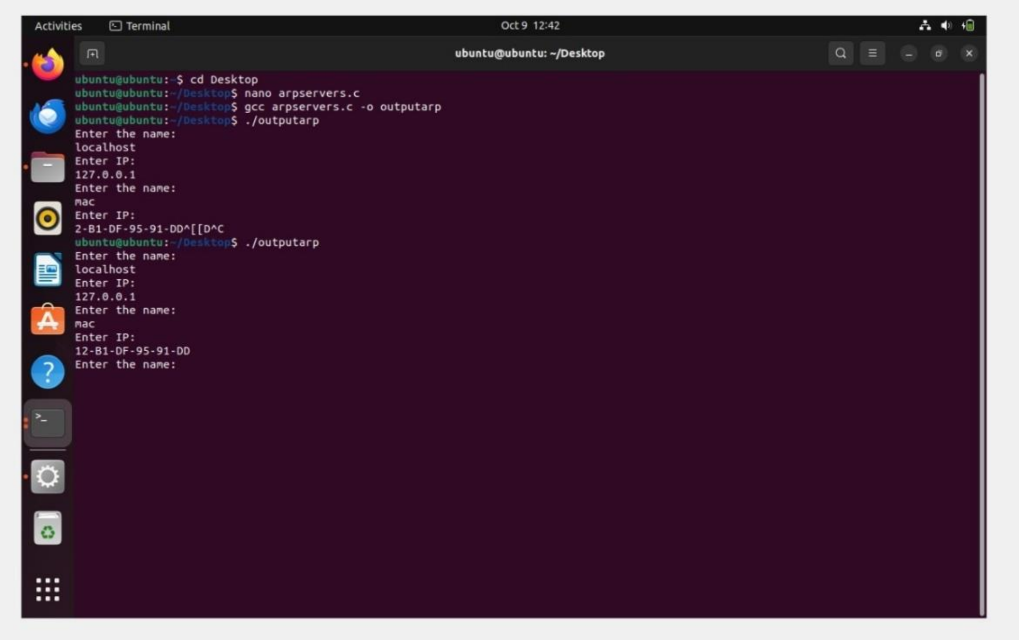
ARP/RARP CLIENT :

```
#include<stdio.h> #include<string.h> #include<sys/types.h> #include<sys/shm.h> main()
{

int shmid,a;

char *ptr,*shmptr;

char ptr2[51],ip[12],mac[26]; shmid=shmget(3000,10,0666); shmptr=shmat(shmid,NULL,0);
puts("The ARPtable is:"); printf("%s",shmptr); printf("\n1.ARP\n2.RARP\n3.EXIT\n");
scanf("%d",&a);
switch(a)

{

case 1:

puts("Enter ip address:"); scanf("%s",ip); ptr=strstr(shmptr,ip);
ptr-=8; sscanf(ptr,"%s%*s",ptr2); printf("mac addr is:%s",ptr2); break;
case 2:

puts("Enter mac addr"); scanf("%s",mac); ptr=strstr(shmptr,mac); sscanf(ptr,"%*s%s",ptr2);
printf("%s",ptr2).break; case 3:

exit(1);

}

}
```

ARP/RARP SERVER :

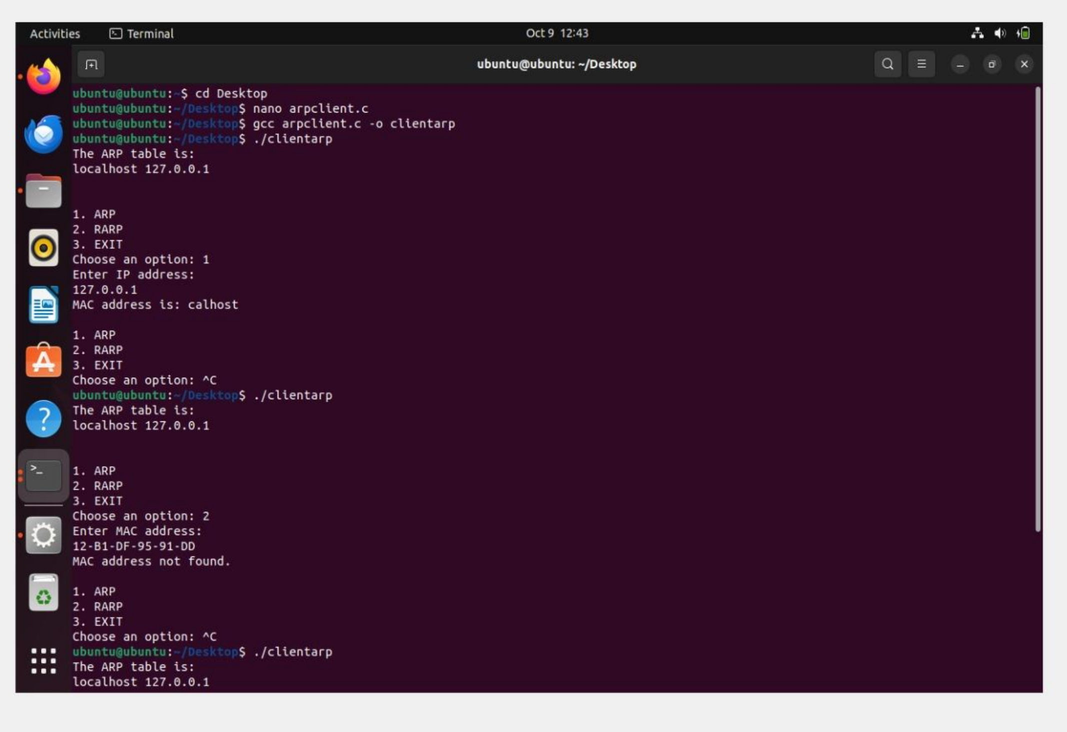
```
#include<stdio.h> #include<sys/types.h> #include<sys/shm.h> #include<string.h> main()
{
int shmid,a,i;
char *ptr,*shmptr; shmid=shmget(3000,10,IPC_CREAT|0666); shmptr=shmat(shmid,NULL,0);
ptr=shmptr;
for(i=0;i<3;i++)
{
puts("Enter the name:"); scanf("%s",ptr); a=strlen(ptr);
printf("String length:%d",a); ptr[a]=' ';
puts("Enter ip:"); ptr=ptr+a+1; scanf("%s",ptr);
ptr[a]='\n'; ptr=ptr+a+1;
}
ptr[strlen(ptr)]='\0';
printf("\nARP table at serverside is=\n%s",shmptr); shmdt(shmptr);
}
```

OUTPUT : ARP/RARP SERVER



```
ubuntu@ubuntu:~$ cd Desktop
ubuntu@ubuntu:~/Desktop$ nano arpservers.c
ubuntu@ubuntu:~/Desktop$ gcc arpservers.c -o outputarp
ubuntu@ubuntu:~/Desktop$ ./outputarp
Enter the name:
localhost
Enter IP:
127.0.0.1
Enter the name:
mac
Enter IP:
2-B1-DF-95-91-DD^[[D^C
ubuntu@ubuntu:~/Desktop$ ./outputarp
Enter the name:
localhost
Enter IP:
127.0.0.1
Enter the name:
mac
Enter IP:
12-B1-DF-95-91-DD
Enter the name:
```

ARP/RARP CLIENT



```
ubuntu@ubuntu:~$ cd Desktop
ubuntu@ubuntu:~/Desktop$ nano arpclient.c
ubuntu@ubuntu:~/Desktop$ gcc arpclient.c -o clientarp
ubuntu@ubuntu:~/Desktop$ ./clientarp
The ARP table is:
localhost 127.0.0.1


1. ARP
2. RARP
3. EXIT
Choose an option: 1
Enter IP address:
127.0.0.1
MAC address is: calhost

1. ARP
2. RARP
3. EXIT
Choose an option: ^C
ubuntu@ubuntu:~/Desktop$ ./clientarp
The ARP table is:
localhost 127.0.0.1


1. ARP
2. RARP
3. EXIT
Choose an option: 2
Enter MAC address:
12-B1-DF-95-91-DD
MAC address not found.

1. ARP
2. RARP
3. EXIT
Choose an option: ^C
ubuntu@ubuntu:~/Desktop$ ./clientarp
The ARP table is:
localhost 127.0.0.1
```

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.

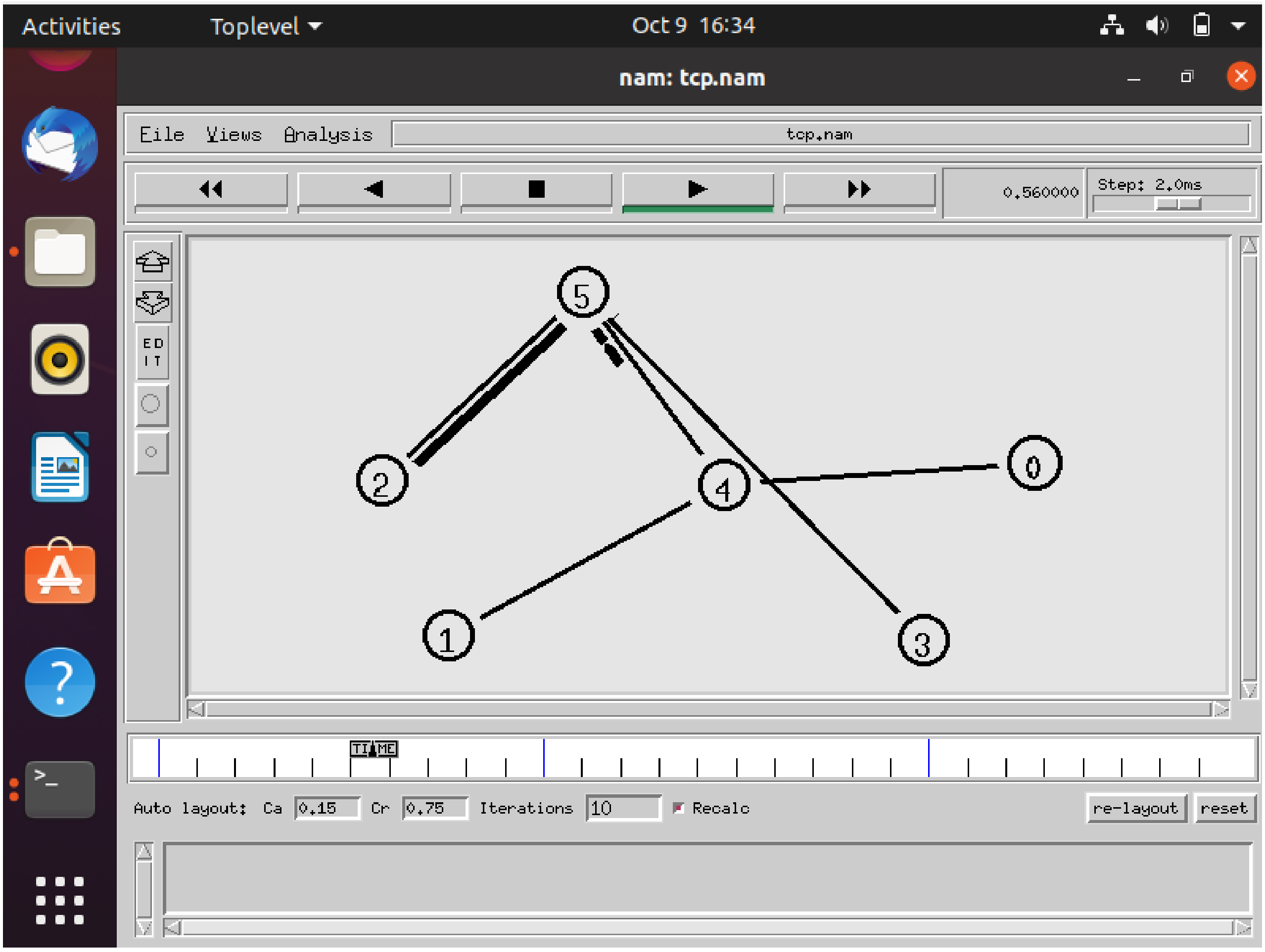
PROGRAM:

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
```

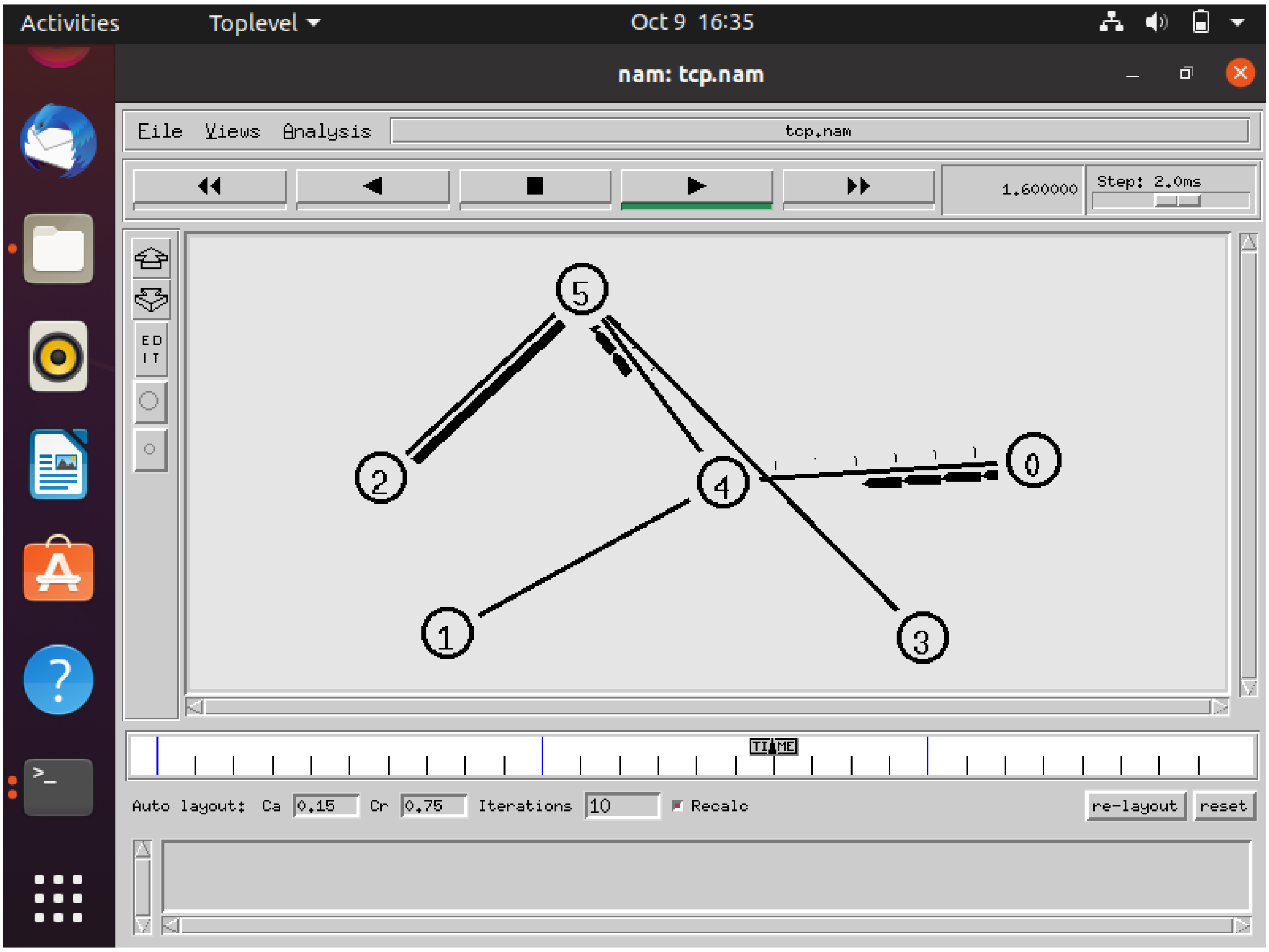
```
$ns duplex-link $n3 $n5 1Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 50ms DropTail
$ns duplex-link-op $n4 $n5 queuePos 0.5 set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp set sink [new Agent/TCPSink]
$ns attach-agent $n2 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent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

OUTPUT :



0.560000 Step: 2.0ms

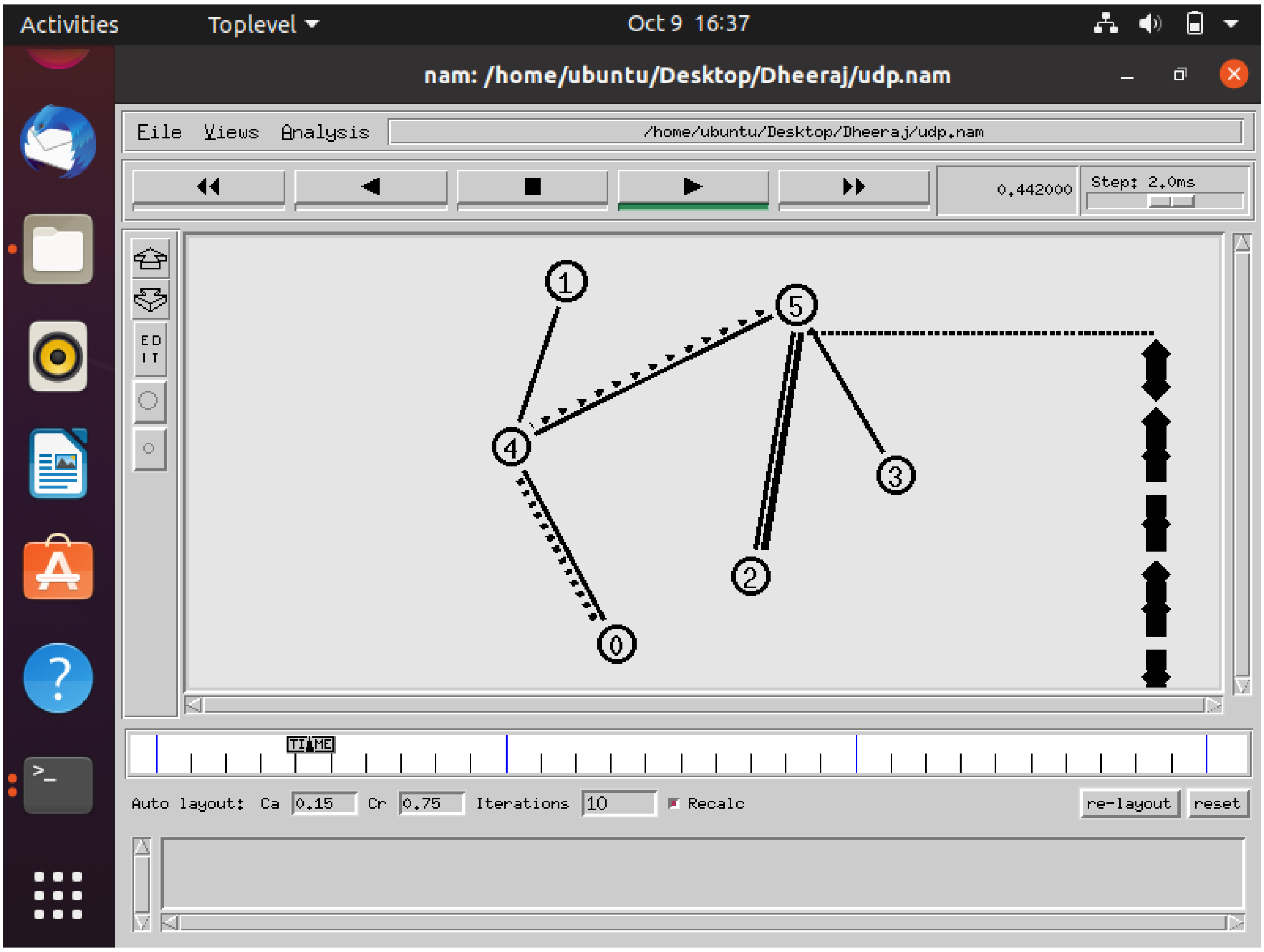
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

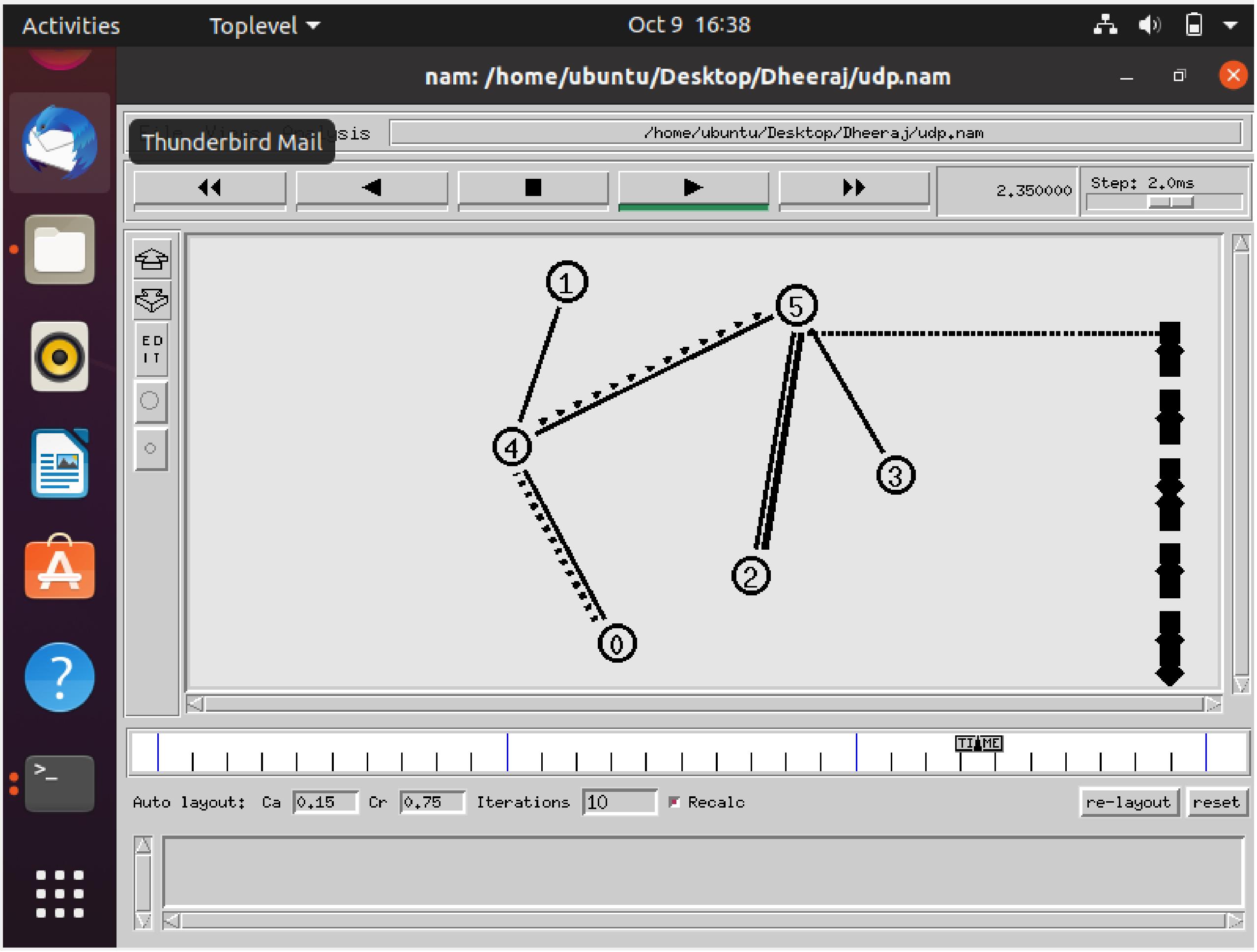

Activities
Toplevel
Oct 9 16:35
nam: tcp.nam
File Views Analysis
tcp.nam
1.600000
Step: 2.0ms
EDIT
5
2
4
0
1
3
TIME
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset

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:


Activities
Toplevel
Oct 9 16:37
nam: /home/ubuntu/Desktop/Dheeraj/udp.nam
File Views Analysis
/home/ubuntu/Desktop/Dheeraj/udp.nam
0.442000
Step: 2.0ms
EDIT
1
5
4
3
2
0
TIME
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset


Activities
Toplevel
Oct 9 16:38
nam: /home/ubuntu/Desktop/Dheeraj/udp.nam
Thunderbird Mail
sis
/home/ubuntu/Desktop/Dheeraj/udp.nam
2.350000
Step: 2.0ms
ED
IT
1
5
4
3
2
0
TIME
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset

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

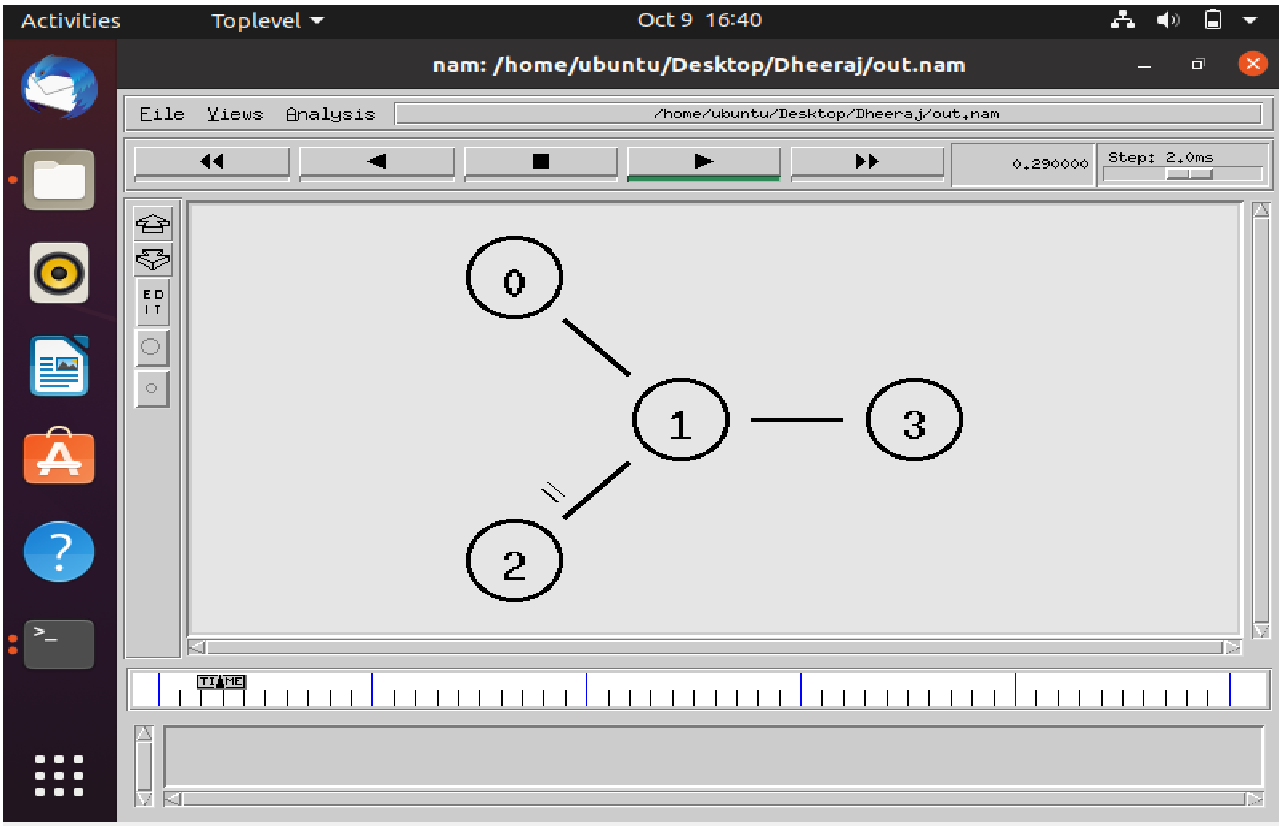
17. View the simulated events and trace file analyze it

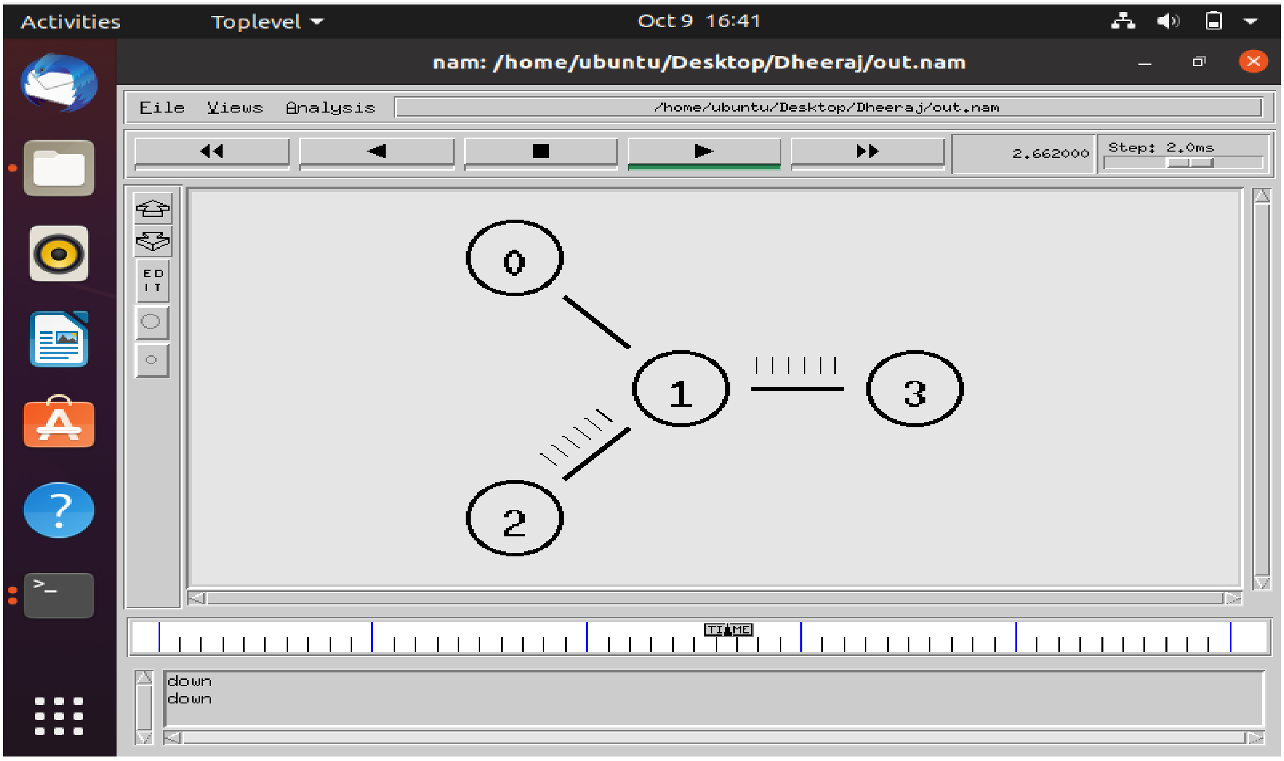
18. Stop

PROGRAM :

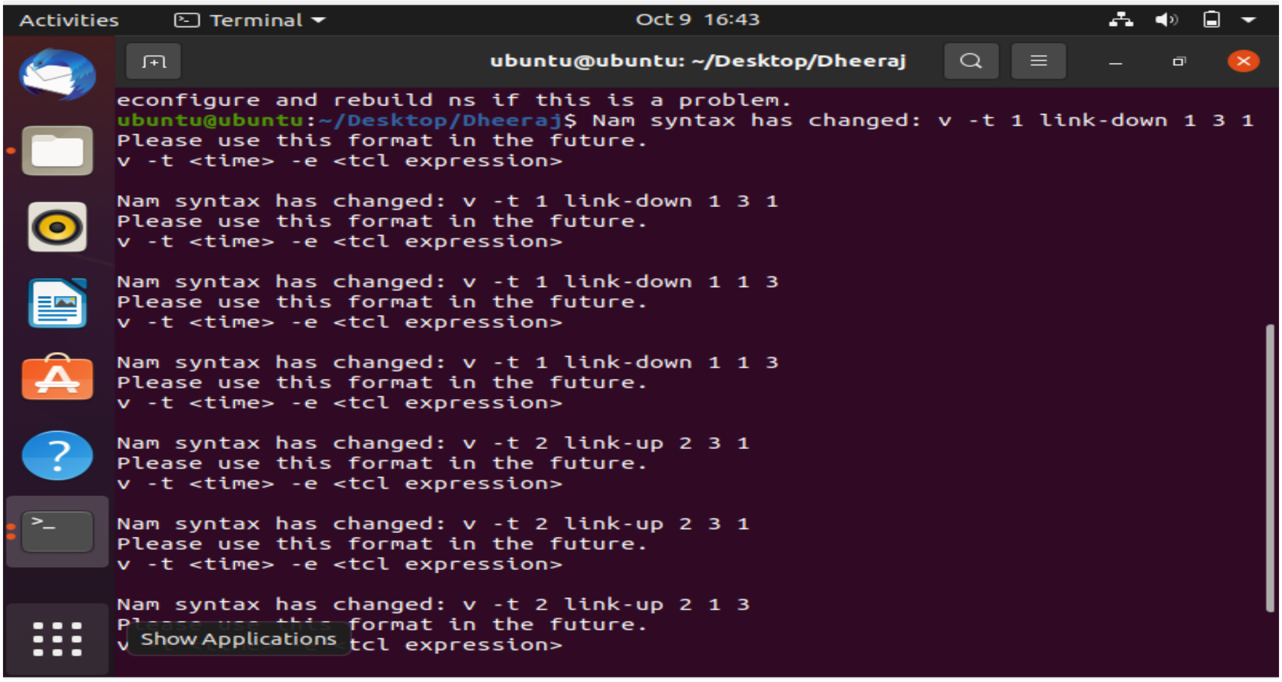
```
set ns [new Simulator] set nf [open out.nam w]
$ns namtrace-all $nfset tr [open out.tr w]
$ns trace-all $trproc finish {} { global nf ns tr
$ns flush-traceclose $tr exec nam out.nam &exit 0
}
set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node]
$ns duplex-link $n0 $n1 10Mb 10ms 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:


Activities
Toplevel
Oct 9 16:40
nam: /home/ubuntu/Desktop/Dheeraj/out.nam
File Views Analysis
/home/ubuntu/Desktop/Dheeraj/out.nam
0.290000
Step: 2.0ms
0
1
2
3
TIME


Activities
Toplevel
Oct 9 16:41
nam: /home/ubuntu/Desktop/Dheeraj/out.nam
File Views Analysis
/home/ubuntu/Desktop/Dheeraj/out.nam
2.662000
Step: 2.0ms
0
1
2
3
TIME

```
down
down
```



```
econfigure and rebuild ns if this is a problem.
ubuntu@ubuntu:~/Desktop/Dheeraj$ Nam syntax has changed: v -t 1 link-down 1 3 1
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 1 link-down 1 3 1
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 1 link-down 1 1 3
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 1 link-down 1 1 3
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 3 1
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 3 1
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 1 3
Please use this format in the future.
v -t <time> -e <tcl expression>
```

DISTANCE VECTOR ROUTINGPROTOCOL AIM:

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

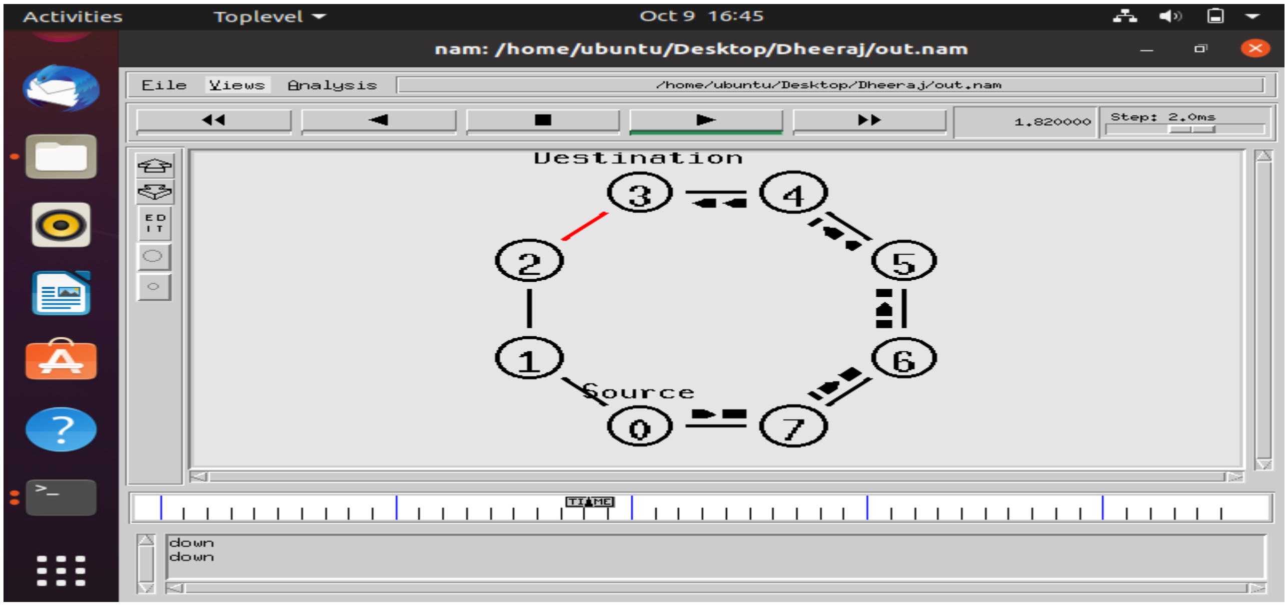
ALGORITHM:

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.

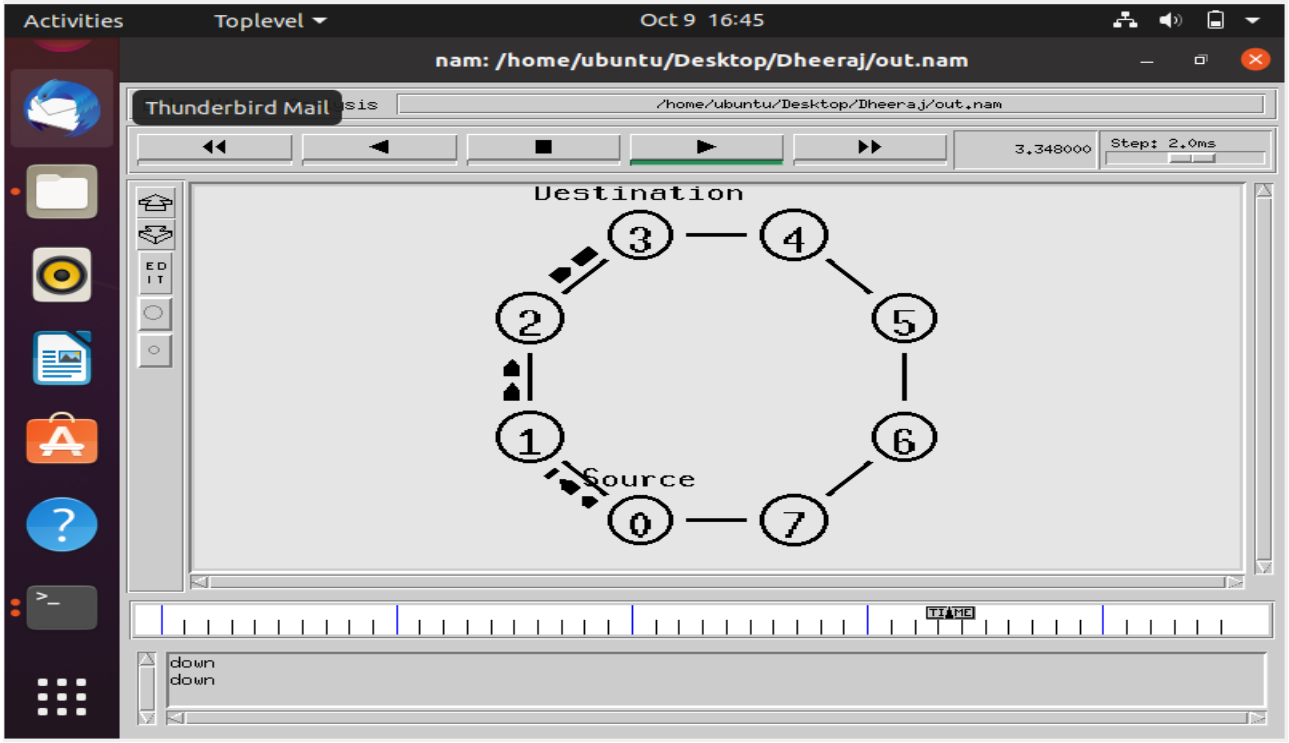
PROGRAM:

```
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
```

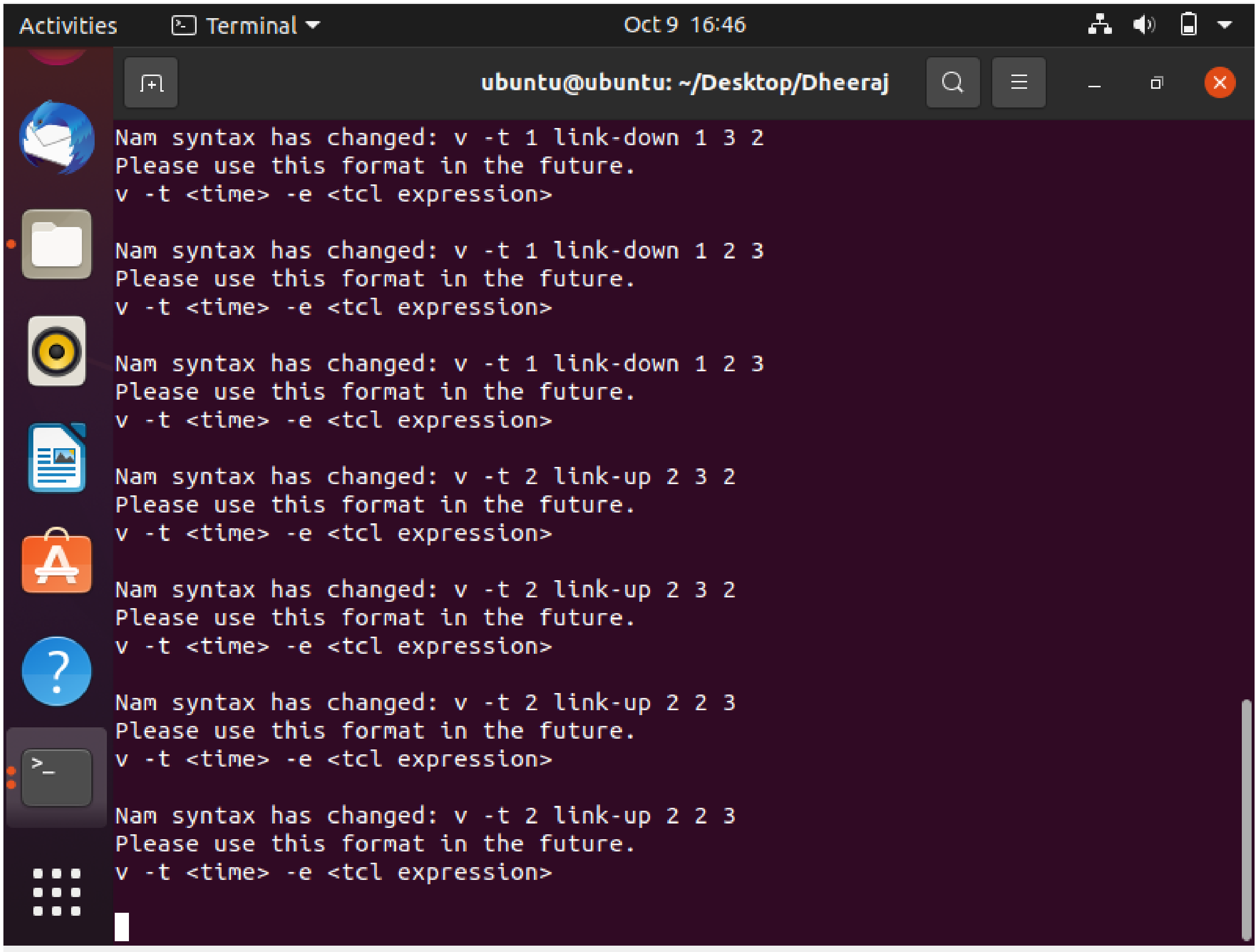
OUTPUT:



```
down
down
```



```
down
down
```



```
Nam syntax has changed: v -t 1 link-down 1 3 2
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 1 link-down 1 2 3
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 1 link-down 1 2 3
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 3 2
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 3 2
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 2 3
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 2 3
Please use this format in the future.
v -t <time> -e <tcl expression>
```

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 6

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

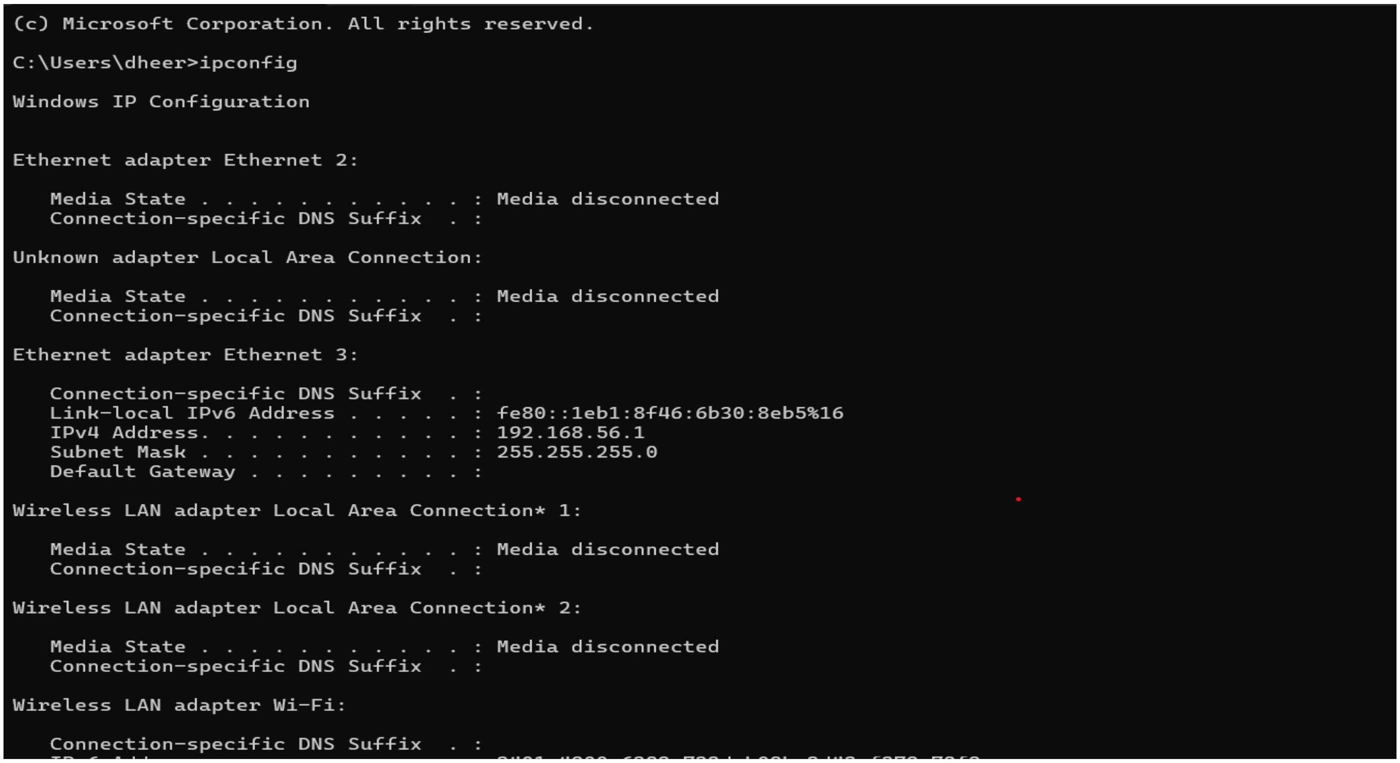
Course / Branch : **III BE CSE**

Title of the exercise : **Demonstrate the working of network tools such as Ping, TCP Dump, Traceroute, Netstat, Ipconfig.**

Networks Commands:

1) ipconfig

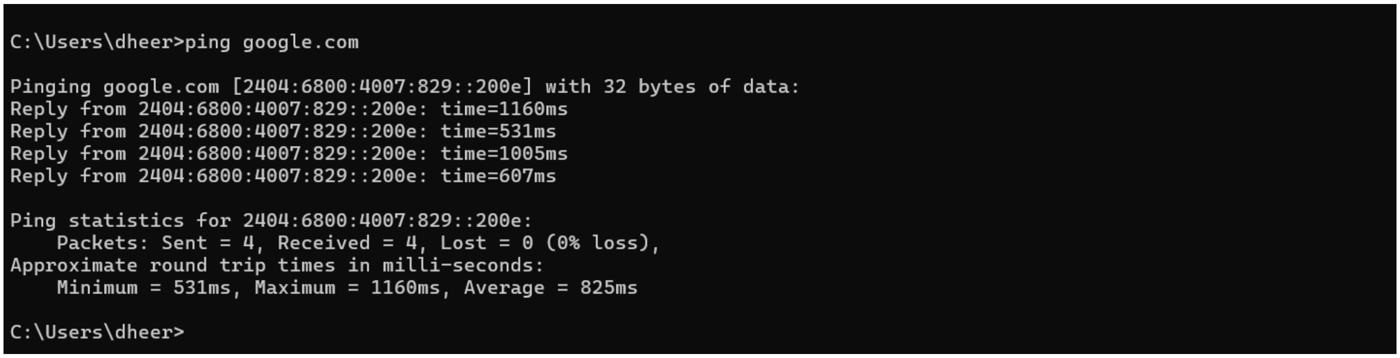
ipconfig (standing for "Internet Protocol configuration") is a console application program of some computer operating systems that displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings.[1] IPCONFIG



```
(c) Microsoft Corporation. All rights reserved.

C:\Users\dheer>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet 2:

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

Unknown adapter Local Area Connection:

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

Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::1eb1:8f46:6b30:8eb5%16
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 1:

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

Wireless LAN adapter Local Area Connection* 2:

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

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
```

2) Ping

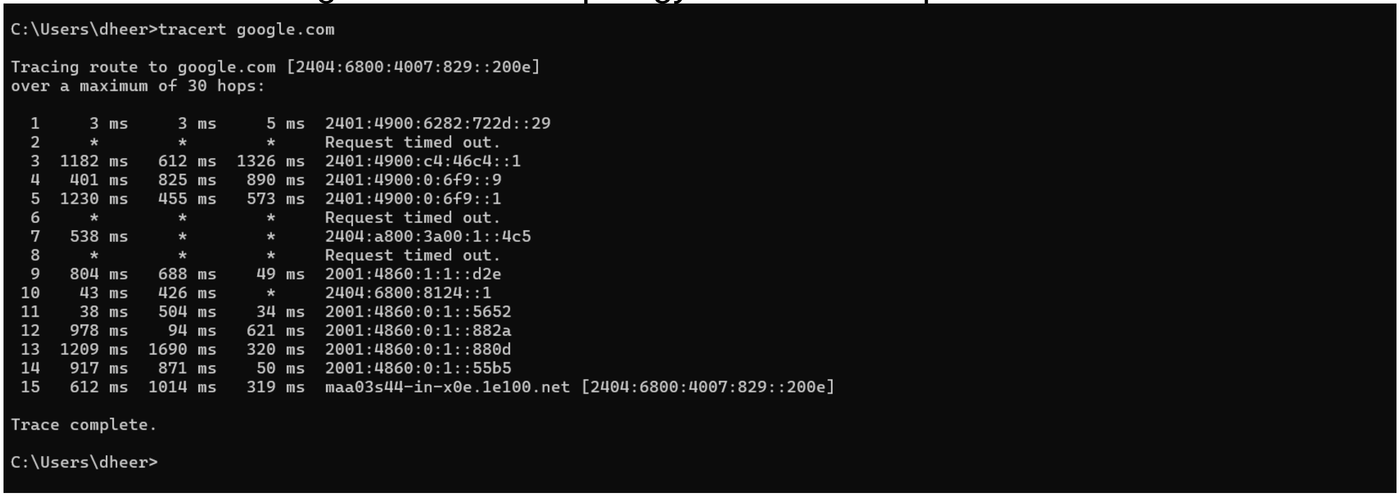
ping is a computer network administration software utility used to test the reachability of a host on an Internet Protocol (IP) network. It is available for virtually all operating systems that have networking capability, including most embedded network administration software.



```
C:\Users\dheer>ping google.com

Pinging google.com [2404:6800:4007:829::200e] with 32 bytes of data:
Reply from 2404:6800:4007:829::200e: time=1160ms
Reply from 2404:6800:4007:829::200e: time=531ms
Reply from 2404:6800:4007:829::200e: time=1005ms
Reply from 2404:6800:4007:829::200e: time=607ms

Ping statistics for 2404:6800:4007:829::200e:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 531ms, Maximum = 1160ms, Average = 825ms

C:\Users\dheer>
```

3) Tracert google.com

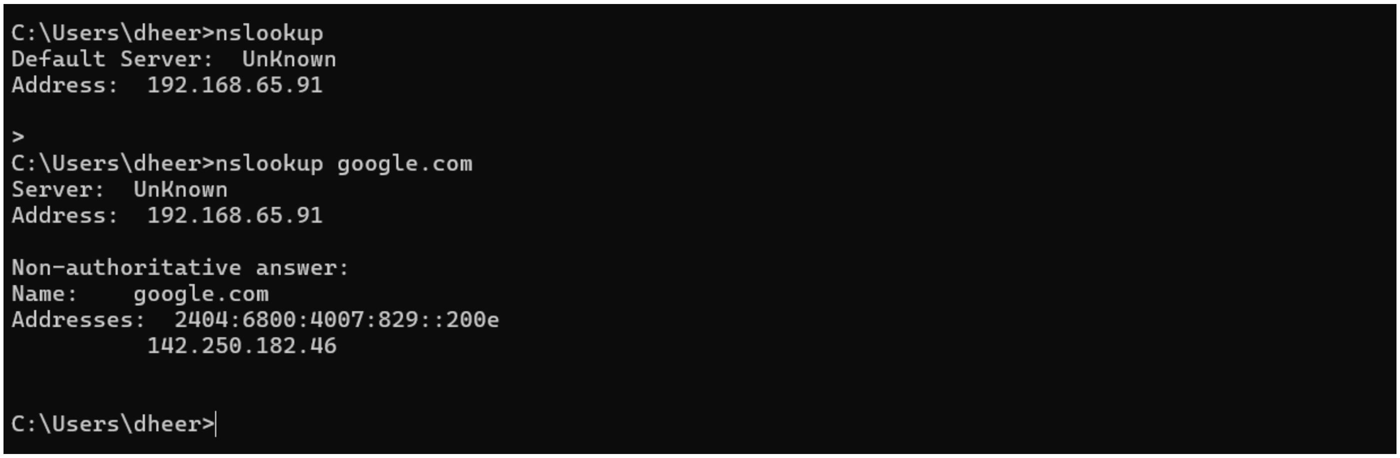
The tracert (short for "trace route") command is a network diagnostic tool used in Windows operating systems to track the path that packets take from the source computer to a specified destination (usually another computer or server). This command is useful for identifying routing issues and understanding the network topology between two points.



```
C:\Users\dheer>tracert google.com

Tracing route to google.com [2404:6800:4007:829::200e]
over a maximum of 30 hops:

  1     3 ms     3 ms     5 ms  2401:4900:6282:722d::29
  2     *        *        *     Request timed out.
  3  1182 ms   612 ms  1326 ms  2401:4900:c4:46c4::1
  4   401 ms   825 ms   890 ms  2401:4900:0:6f9::9
  5  1230 ms   455 ms   573 ms  2401:4900:0:6f9::1
  6     *        *        *     Request timed out.
  7   538 ms     *        *     2404:a800:3a00:1::4c5
  8     *        *        *     Request timed out.
  9   804 ms   688 ms    49 ms  2001:4860:1:1::d2e
 10    43 ms   426 ms     *     2404:6800:8124::1
 11    38 ms   504 ms    34 ms  2001:4860:0:1::5652
 12   978 ms    94 ms   621 ms  2001:4860:0:1::882a
 13  1209 ms  1690 ms   320 ms  2001:4860:0:1::880d
 14   917 ms   871 ms    50 ms  2001:4860:0:1::55b5
 15   612 ms  1014 ms   319 ms  maa03s44-in-x0e.1e100.net [2404:6800:4007:829::200e]

Trace complete.

C:\Users\dheer>
```

4) nslookup

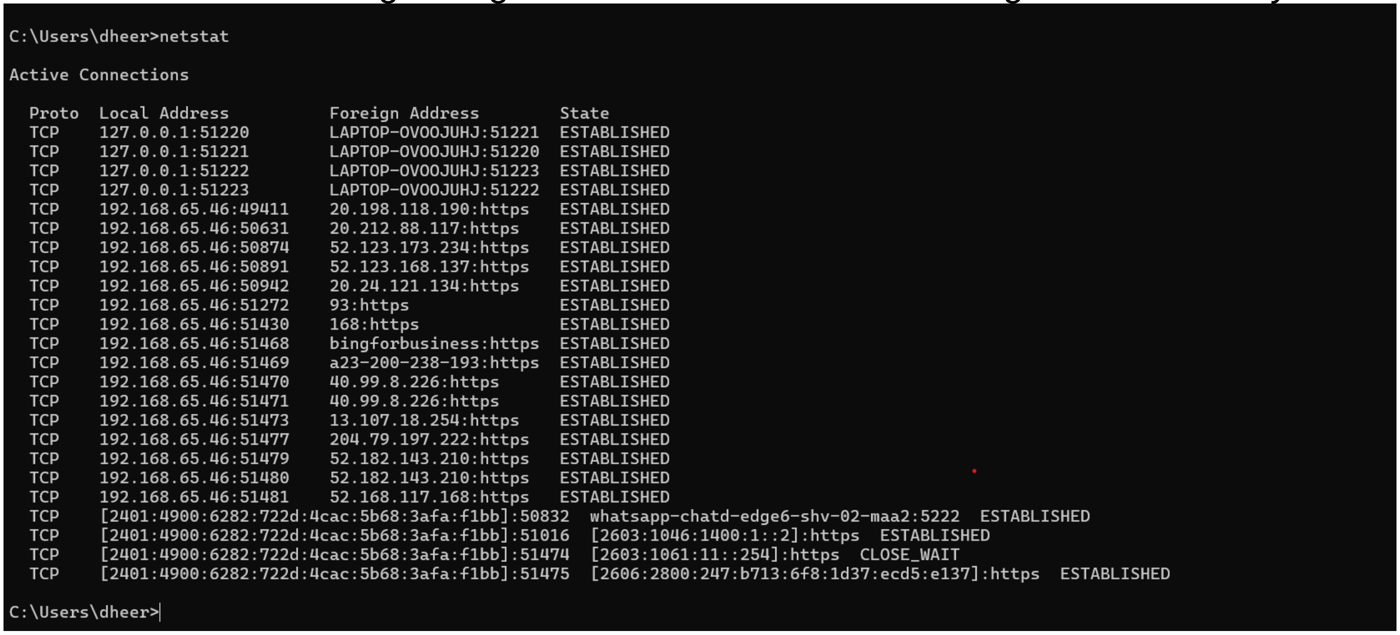
The nslookup (short for "name server lookup") command is a network utility tool used in Windows (and other operating systems) to query the Domain Name System (DNS) to obtain domain name or IP address mapping. It is useful for diagnosing DNS-related issues and for obtaining information about domain names and IP addresses.



```
C:\Users\dheer>nslookup
Default Server:  UnKnown
Address:  192.168.65.91

>
C:\Users\dheer>nslookup google.com
Server:  UnKnown
Address:  192.168.65.91

Non-authoritative answer:
Name:    google.com
Addresses:  2404:6800:4007:829::200e
          142.250.182.46


C:\Users\dheer>
```

5) netstat

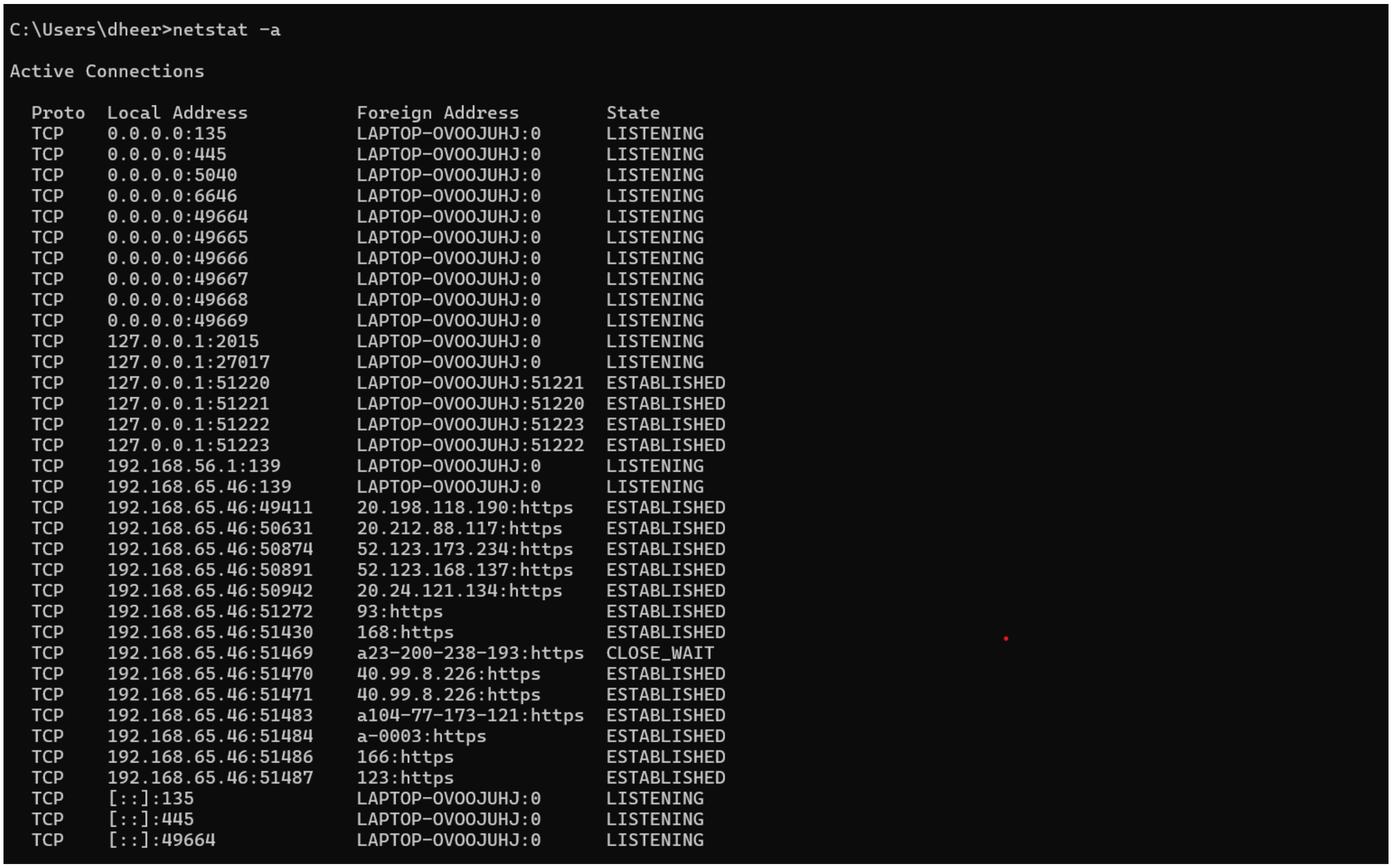
The netstat (short for "network statistics") command is a powerful network utility in Windows (and other operating systems) that displays various network-related information, including active connections, routing tables, and network interface statistics. It is a valuable tool for network administrators and users for diagnosing network issues and monitoring network activity.



```
C:\Users\dheer>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:51220        LAPTOP-OVOOJUHJ:51221  ESTABLISHED
  TCP    127.0.0.1:51221        LAPTOP-OVOOJUHJ:51220  ESTABLISHED
  TCP    127.0.0.1:51222        LAPTOP-OVOOJUHJ:51223  ESTABLISHED
  TCP    127.0.0.1:51223        LAPTOP-OVOOJUHJ:51222  ESTABLISHED
  TCP    192.168.65.46:49411    20.198.118.190:https   ESTABLISHED
  TCP    192.168.65.46:50631    20.212.88.117:https    ESTABLISHED
  TCP    192.168.65.46:50874    52.123.173.234:https   ESTABLISHED
  TCP    192.168.65.46:50891    52.123.168.137:https   ESTABLISHED
  TCP    192.168.65.46:50942    20.24.121.134:https    ESTABLISHED
  TCP    192.168.65.46:51272    93:https               ESTABLISHED
  TCP    192.168.65.46:51430    168:https              ESTABLISHED
  TCP    192.168.65.46:51468    bingforbusiness:https  ESTABLISHED
  TCP    192.168.65.46:51469    a23-200-238-193:https  ESTABLISHED
  TCP    192.168.65.46:51470    40.99.8.226:https      ESTABLISHED
  TCP    192.168.65.46:51471    40.99.8.226:https      ESTABLISHED
  TCP    192.168.65.46:51473    13.107.18.254:https    ESTABLISHED
  TCP    192.168.65.46:51477    204.79.197.222:https   ESTABLISHED
  TCP    192.168.65.46:51479    52.182.143.210:https   ESTABLISHED
  TCP    192.168.65.46:51480    52.182.143.210:https   ESTABLISHED
  TCP    192.168.65.46:51481    52.168.117.168:https   ESTABLISHED
  TCP    [2401:4900:6282:722d:4cac:5b68:3afa:f1bb]:50832  whatsapp-chatd-edge6-shv-02-maa2:5222  ESTABLISHED
  TCP    [2401:4900:6282:722d:4cac:5b68:3afa:f1bb]:51016  [2603:1046:1400:1::2]:https  ESTABLISHED
  TCP    [2401:4900:6282:722d:4cac:5b68:3afa:f1bb]:51474  [2603:1061:11::254]:https  CLOSE_WAIT
  TCP    [2401:4900:6282:722d:4cac:5b68:3afa:f1bb]:51475  [2606:2800:247:b713:6f8:1d37:ecd5:e137]:https  ESTABLISHED

C:\Users\dheer>
```

6) netstat -a

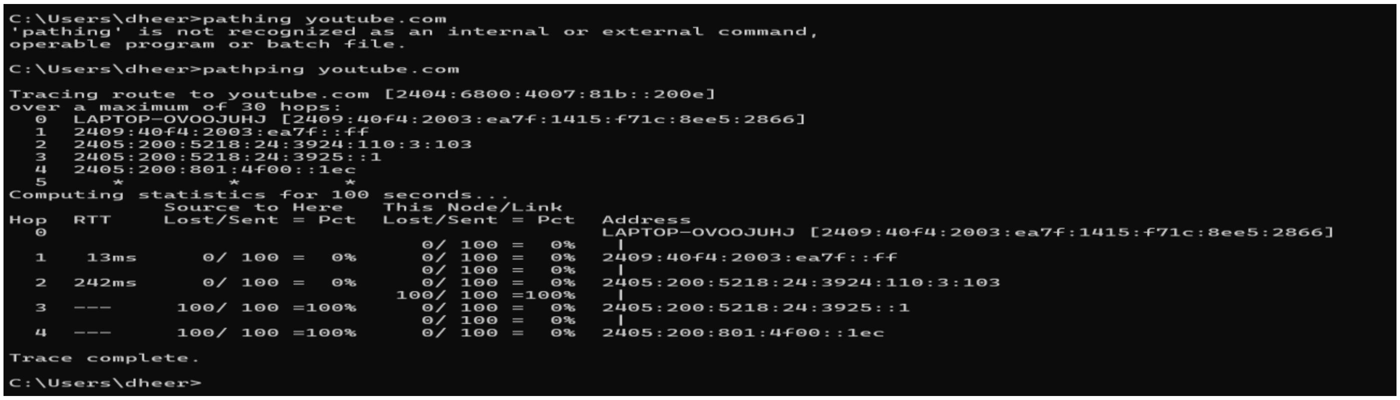
The netstat -a command is a powerful tool used in Windows (and other operating systems) to display all active network connections and listening ports on the local computer. This command provides detailed information about both TCP and UDP connections, including the local and foreign addresses and the state of each connection.



```
C:\Users\dheer>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    0.0.0.0:445            LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    0.0.0.0:5040           LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    0.0.0.0:6646           LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    0.0.0.0:49664          LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    0.0.0.0:49665          LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    0.0.0.0:49666          LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    0.0.0.0:49667          LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    0.0.0.0:49668          LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    0.0.0.0:49669          LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    127.0.0.1:2015         LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    127.0.0.1:27017        LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    127.0.0.1:51220        LAPTOP-OVOOJUHJ:51221  ESTABLISHED
  TCP    127.0.0.1:51221        LAPTOP-OVOOJUHJ:51220  ESTABLISHED
  TCP    127.0.0.1:51222        LAPTOP-OVOOJUHJ:51223  ESTABLISHED
  TCP    127.0.0.1:51223        LAPTOP-OVOOJUHJ:51222  ESTABLISHED
  TCP    192.168.56.1:139       LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    192.168.65.46:139      LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    192.168.65.46:49411    20.198.118.190:https   ESTABLISHED
  TCP    192.168.65.46:50631    20.212.88.117:https    ESTABLISHED
  TCP    192.168.65.46:50874    52.123.173.234:https   ESTABLISHED
  TCP    192.168.65.46:50891    52.123.168.137:https   ESTABLISHED
  TCP    192.168.65.46:50942    20.24.121.134:https    ESTABLISHED
  TCP    192.168.65.46:51272    93:https               ESTABLISHED
  TCP    192.168.65.46:51430    168:https              ESTABLISHED
  TCP    192.168.65.46:51469    a23-200-238-193:https  CLOSE_WAIT
  TCP    192.168.65.46:51470    40.99.8.226:https      ESTABLISHED
  TCP    192.168.65.46:51471    40.99.8.226:https      ESTABLISHED
  TCP    192.168.65.46:51483    a104-77-173-121:https  ESTABLISHED
  TCP    192.168.65.46:51484    a-0003:https           ESTABLISHED
  TCP    192.168.65.46:51486    166:https              ESTABLISHED
  TCP    192.168.65.46:51487    123:https              ESTABLISHED
  TCP    [::]:135               LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    [::]:445               LAPTOP-OVOOJUHJ:0      LISTENING
  TCP    [::]:49664             LAPTOP-OVOOJUHJ:0      LISTENING
```

7) pathing

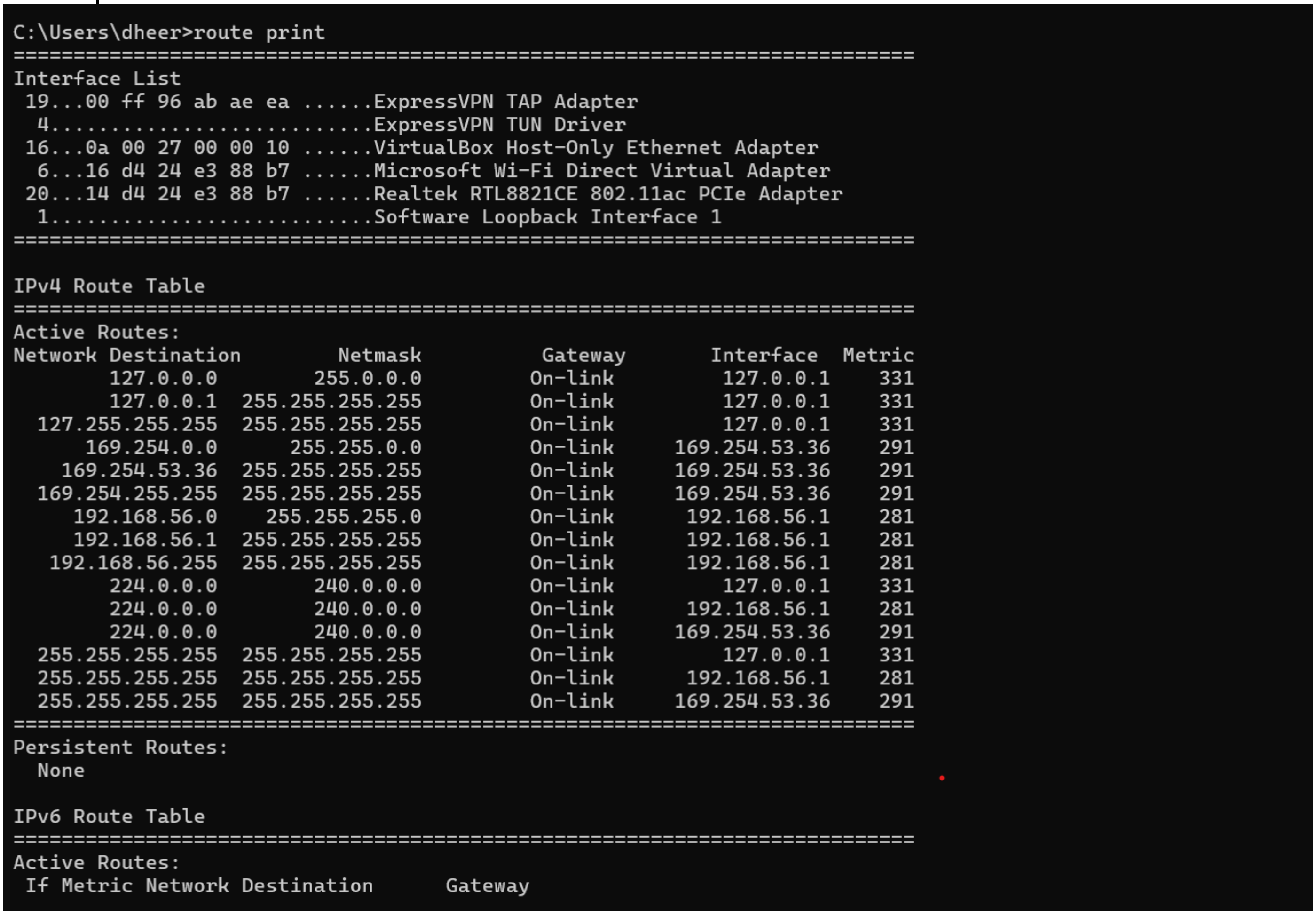
Pathping is a network diagnostic utility that combines the functionality of ping and tracert commands. It provides detailed information about the route packets take to a destination and analyzes the performance and reliability of each hop along the route. Pathping helps identify the path and pinpoint specific routers or network segments that may be causing network issues.



```
C:\Users\dheer>pathing youtube.com
'pathing' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\dheer>pathping youtube.com

Tracing route to youtube.com [2404:6800:4007:81b::200e]
over a maximum of 30 hops:
  0  LAPTOP-OVOOJUHJ [2409:40f4:2003:ea7f:1415:f71c:8ee5:2866]
  1  2409:40f4:2003:ea7f::ff
  2  2405:200:5218:24:3924:110:3:103
  3  2405:200:5218:24:3925::1
  4  2405:200:801:4f00::1ec
  5     *        *        *
Computing statistics for 100 seconds...
            Source to Here   This Node/Link
Hop  RTT    Lost/Sent = Pct  Lost/Sent = Pct  Address
  0                                           LAPTOP-OVOOJUHJ [2409:40f4:2003:ea7f:1415:f71c:8ee5:2866]
                                0/ 100 =  0%   |
  1   13ms     0/ 100 =  0%     0/ 100 =  0%  2409:40f4:2003:ea7f::ff
                                0/ 100 =  0%   |
  2  242ms     0/ 100 =  0%     0/ 100 =  0%  2405:200:5218:24:3924:110:3:103
                              100/ 100 =100%   |
  3  ---     100/ 100 =100%     0/ 100 =  0%  2405:200:5218:24:3925::1
                                0/ 100 =  0%   |
  4  ---     100/ 100 =100%     0/ 100 =  0%  2405:200:801:4f00::1ec

Trace complete.

C:\Users\dheer>
```

9) Route

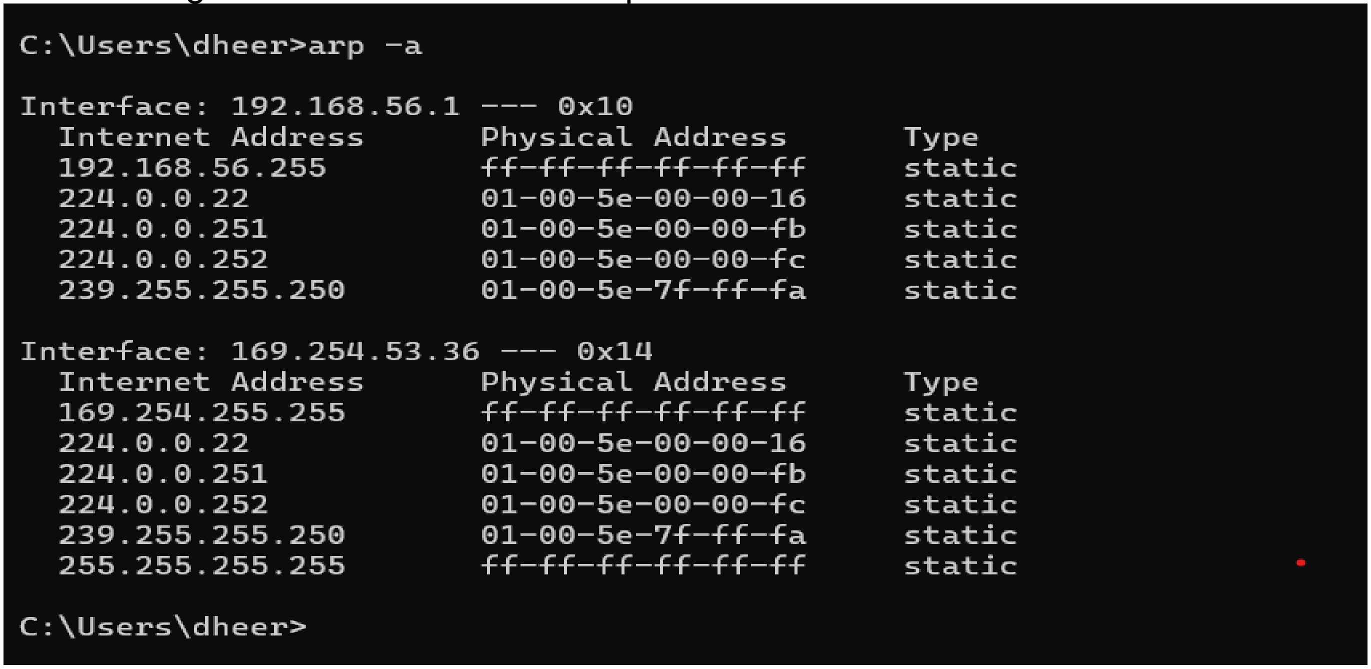
The route command in Windows is used to display and modify the routing table. The routing table determines the path that network traffic takes from your computer to its destination. It is an essential tool for network configuration and troubleshooting, allowing administrators to manage how packets are routed across a network.



```
C:\Users\dheer>route print
===========================================================================
Interface List
 19...00 ff 96 ab ae ea ......ExpressVPN TAP Adapter
  4...........................ExpressVPN TUN Driver
 16...0a 00 27 00 00 10 ......VirtualBox Host-Only Ethernet Adapter
  6...16 d4 24 e3 88 b7 ......Microsoft Wi-Fi Direct Virtual Adapter
 20...14 d4 24 e3 88 b7 ......Realtek RTL8821CE 802.11ac PCIe Adapter
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
      169.254.0.0      255.255.0.0         On-link     169.254.53.36    291
    169.254.53.36  255.255.255.255         On-link     169.254.53.36    291
  169.254.255.255  255.255.255.255         On-link     169.254.53.36    291
     192.168.56.0    255.255.255.0         On-link      192.168.56.1    281
     192.168.56.1  255.255.255.255         On-link      192.168.56.1    281
   192.168.56.255  255.255.255.255         On-link      192.168.56.1    281
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link      192.168.56.1    281
        224.0.0.0        240.0.0.0         On-link     169.254.53.36    291
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link      192.168.56.1    281
  255.255.255.255  255.255.255.255         On-link     169.254.53.36    291
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
```

10) **arp -a**

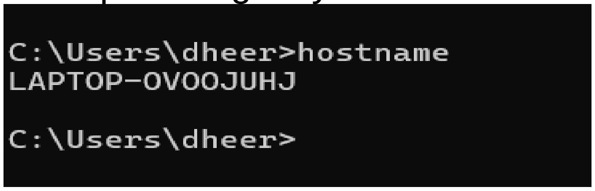
The arp -a command in Windows is used to display the Address Resolution Protocol (ARP) cache, which contains mappings between IP addresses and their corresponding MAC (Media Access Control) addresses. The ARP cache is used to store IP-to-MAC address mappings that the system has discovered, making it quicker to find the MAC address for a given IP address in subsequent communications.



```
C:\Users\dheer>arp -a

Interface: 192.168.56.1 --- 0x10
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static

Interface: 169.254.53.36 --- 0x14
  Internet Address      Physical Address      Type
  169.254.255.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\dheer>
```

11) **hostname**

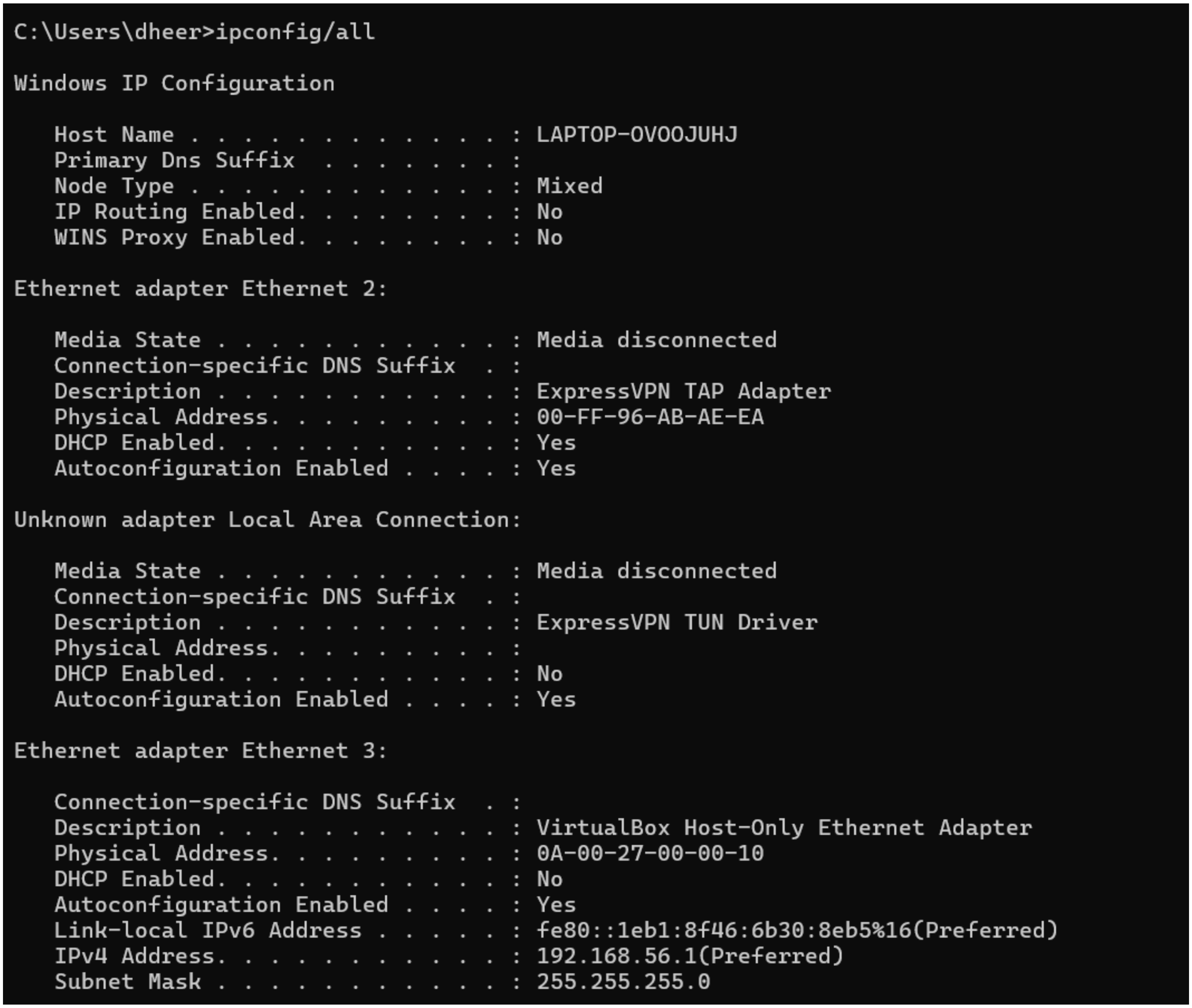
The hostname command in Windows is used to display the name of the current machine or host. This command is simple and straightforward, providing only the hostname of the computer on which it is run.



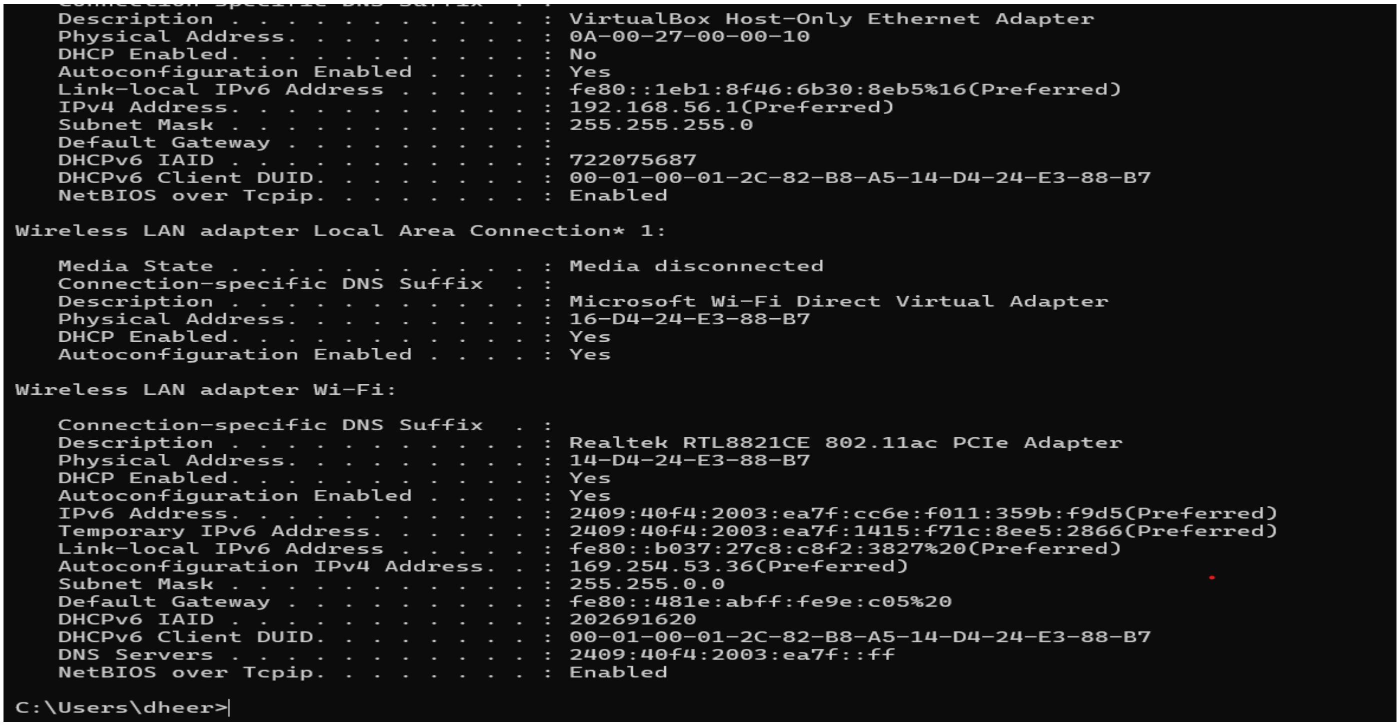
```
C:\Users\dheer>hostname
LAPTOP-OVOOJUHJ

C:\Users\dheer>
```

12) ipconfig / all

The ipconfig /all command in Windows displays detailed information about the network configuration of all network interfaces on the computer. This includes IP addresses, subnet masks, default gateways, DNS servers, and much more. It provides a comprehensive view of the network settings, making it a valuable tool for troubleshooting and configuring network connections.



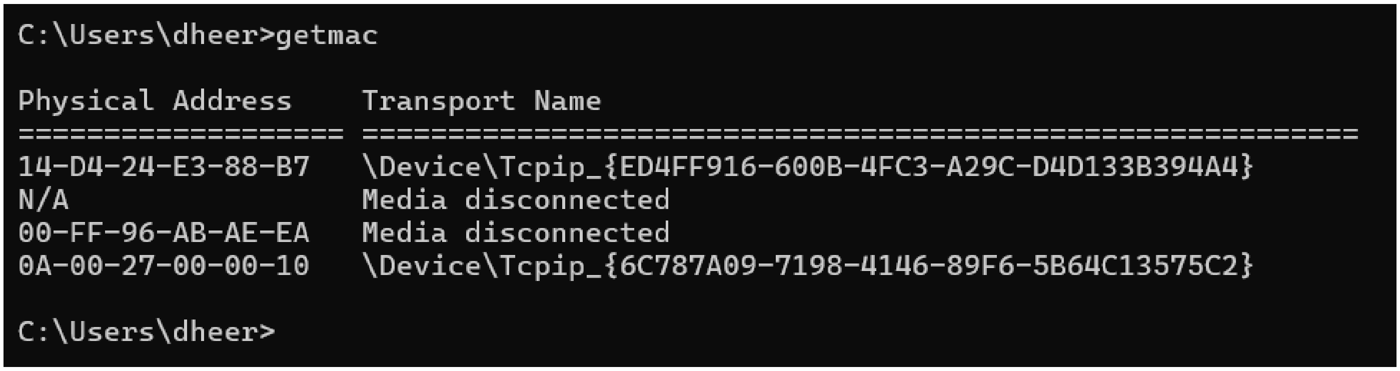
```
C:\Users\dheer>ipconfig/all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : LAPTOP-OVOOJUHJ
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Mixed
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : ExpressVPN TAP Adapter
   Physical Address. . . . . . . . . : 00-FF-96-AB-AE-EA
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

Unknown adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : ExpressVPN TUN Driver
   Physical Address. . . . . . . . . :
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes

Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : VirtualBox Host-Only Ethernet Adapter
   Physical Address. . . . . . . . . : 0A-00-27-00-00-10
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::1eb1:8f46:6b30:8eb5%16(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.56.1(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
```



```
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : VirtualBox Host-Only Ethernet Adapter
   Physical Address. . . . . . . . . : 0A-00-27-00-00-10
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::1eb1:8f46:6b30:8eb5%16(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.56.1(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
   DHCPv6 IAID . . . . . . . . . . . : 722075687
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2C-82-B8-A5-14-D4-24-E3-88-B7
   NetBIOS over Tcpip. . . . . . . . : Enabled

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter
   Physical Address. . . . . . . . . : 16-D4-24-E3-88-B7
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Realtek RTL8821CE 802.11ac PCIe Adapter
   Physical Address. . . . . . . . . : 14-D4-24-E3-88-B7
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   IPv6 Address. . . . . . . . . . . : 2409:40f4:2003:ea7f:cc6e:f011:359b:f9d5(Preferred)
   Temporary IPv6 Address. . . . . . : 2409:40f4:2003:ea7f:1415:f71c:8ee5:2866(Preferred)
   Link-local IPv6 Address . . . . . : fe80::b037:27c8:c8f2:3827%20(Preferred)
   Autoconfiguration IPv4 Address. . : 169.254.53.36(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : fe80::481e:abff:fe9e:c05%20
   DHCPv6 IAID . . . . . . . . . . . : 202691620
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2C-82-B8-A5-14-D4-24-E3-88-B7
   DNS Servers . . . . . . . . . . . : 2409:40f4:2003:ea7f::ff
   NetBIOS over Tcpip. . . . . . . . : Enabled

C:\Users\dheer>
```

13) getmac

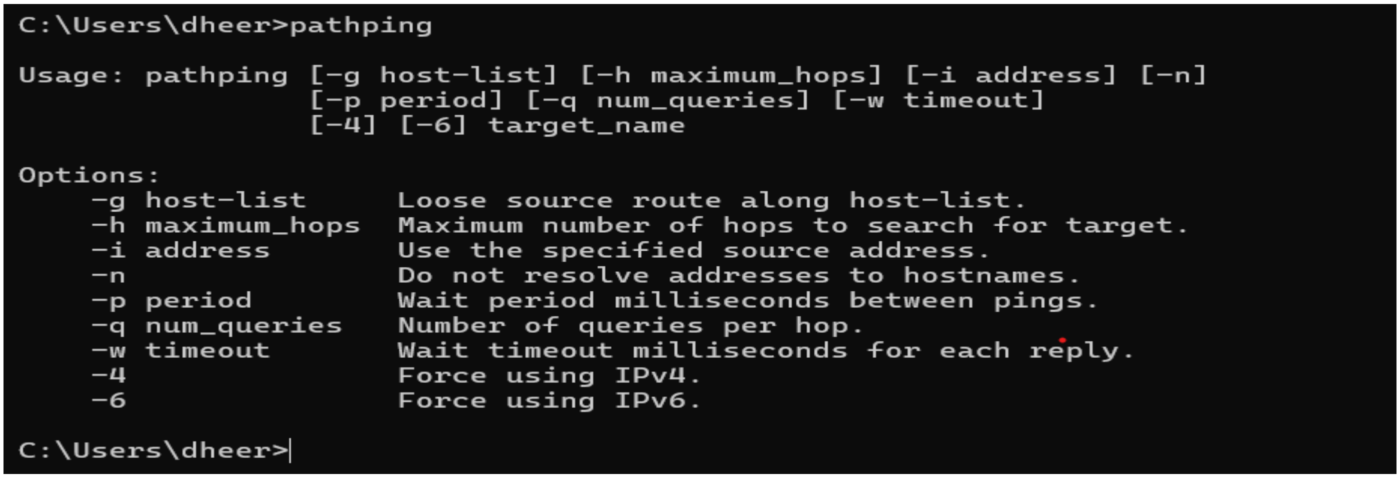
The getmac command in Windows is used to display the MAC (Media Access Control) addresses for the network adapters on the system. The MAC address is a unique identifier assigned to network interfaces for communications on the physical network segment. This command can be useful for network management, troubleshooting, and inventory purposes.



```
C:\Users\dheer>getmac

Physical Address    Transport Name
=================== ==========================================================
14-D4-24-E3-88-B7   \Device\Tcpip_{ED4FF916-600B-4FC3-A29C-D4D133B394A4}
N/A                 Media disconnected
00-FF-96-AB-AE-EA   Media disconnected
0A-00-27-00-00-10   \Device\Tcpip_{6C787A09-7198-4146-89F6-5B64C13575C2}

C:\Users\dheer>
```

14) pathping

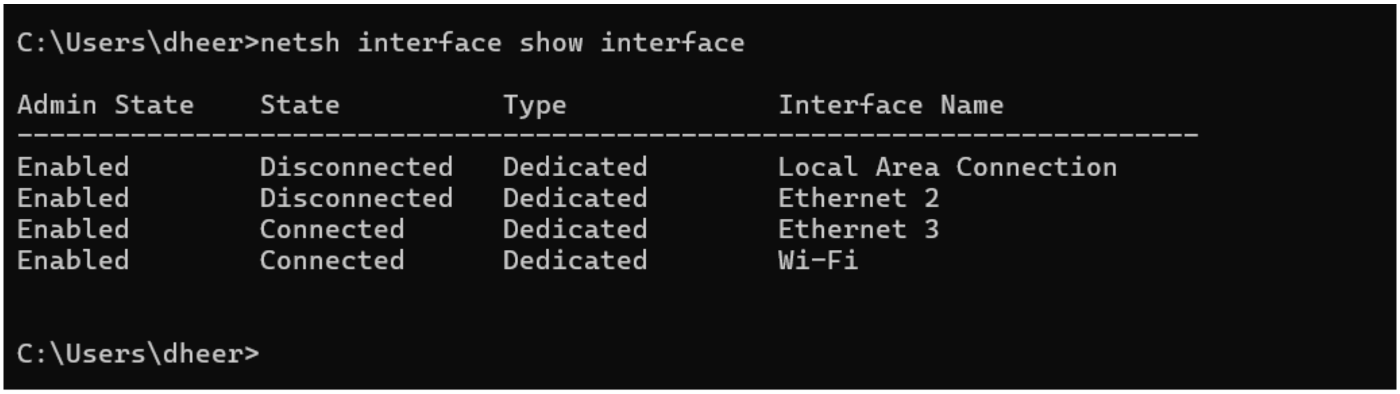
The pathping command in Windows combines the functionality of ping and tracert to provide detailed information about network latency and packet loss at each hop between a source and destination. It helps diagnose network issues by identifying problematic nodes along the route to a target host.



```
C:\Users\dheer>pathping

Usage: pathping [-g host-list] [-h maximum_hops] [-i address] [-n]
                [-p period] [-q num_queries] [-w timeout]
                [-4] [-6] target_name

Options:
    -g host-list     Loose source route along host-list.
    -h maximum_hops  Maximum number of hops to search for target.
    -i address       Use the specified source address.
    -n               Do not resolve addresses to hostnames.
    -p period        Wait period milliseconds between pings.
    -q num_queries   Number of queries per hop.
    -w timeout       Wait timeout milliseconds for each reply.
    -4               Force using IPv4.
    -6               Force using IPv6.

C:\Users\dheer>
```

15) netsh interface show interface

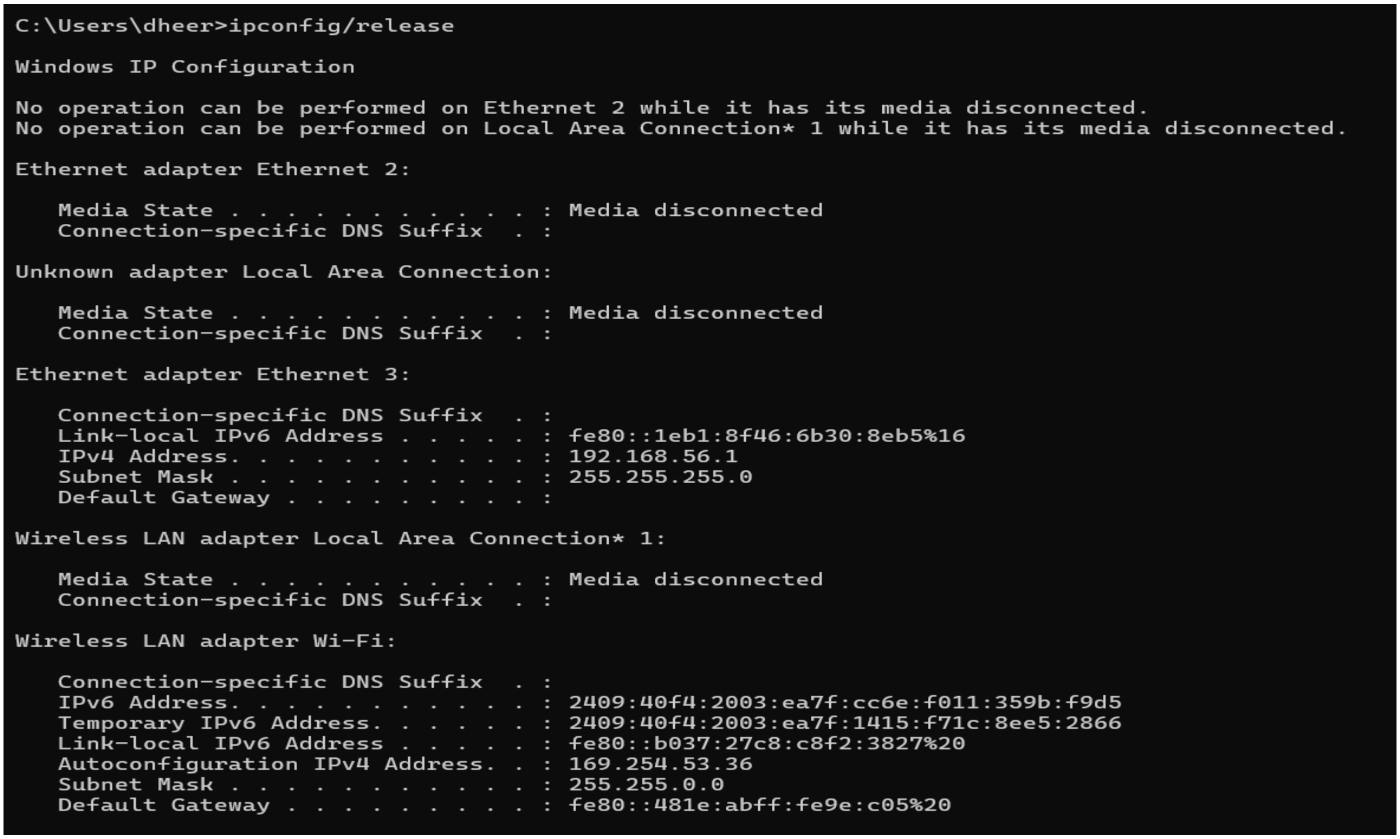
In Windows, the equivalent to nmcli connection show to display network connections and their details can be achieved using several commands and tools. Here are a few ways to get detailed information about network connections in Windows:



```
C:\Users\dheer>netsh interface show interface

Admin State    State          Type             Interface Name
-------------------------------------------------------------------------
Enabled        Disconnected   Dedicated        Local Area Connection
Enabled        Disconnected   Dedicated        Ethernet 2
Enabled        Connected      Dedicated        Ethernet 3
Enabled        Connected      Dedicated        Wi-Fi


C:\Users\dheer>
```

16) ipconfig / release

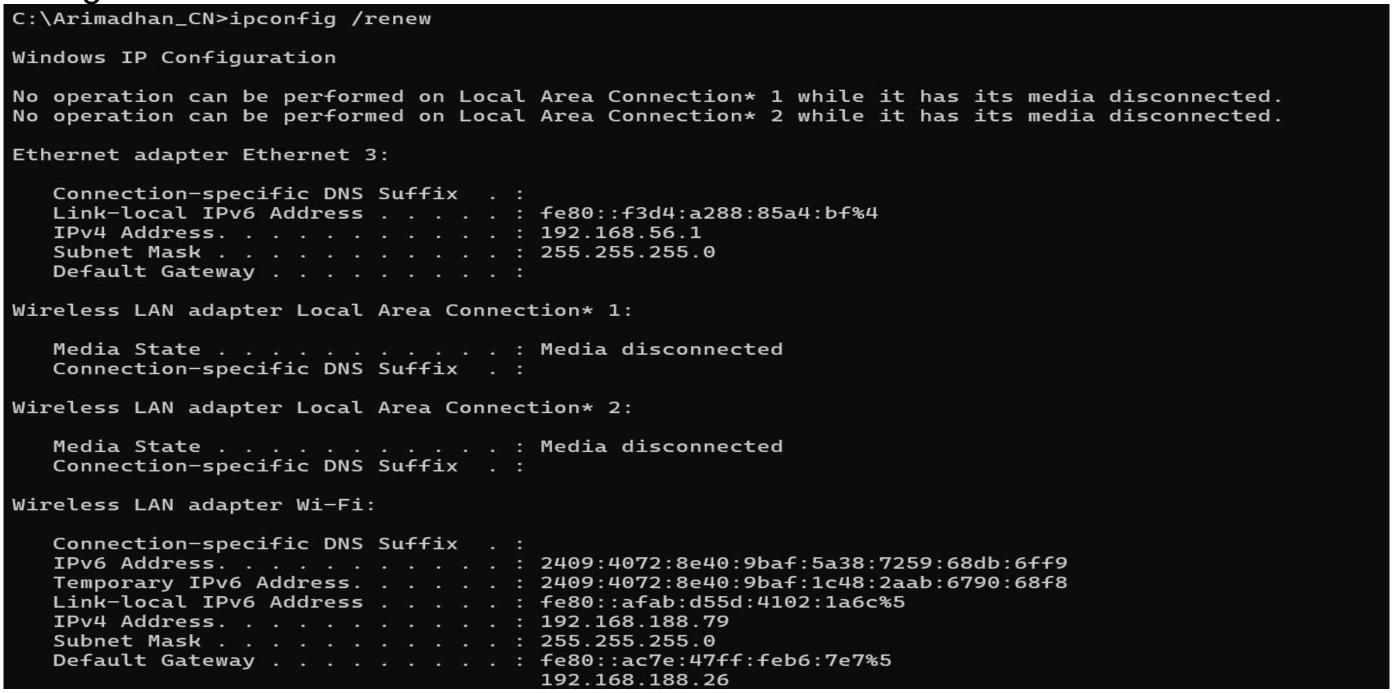
The ipconfig /release command in Windows is used to release the current IP address configuration for all network adapters. This means it will release the DHCP lease, effectively removing the current IP address assigned to the network interfaces.



```
C:\Users\dheer>ipconfig/release

Windows IP Configuration

No operation can be performed on Ethernet 2 while it has its media disconnected.
No operation can be performed on Local Area Connection* 1 while it has its media disconnected.

Ethernet adapter Ethernet 2:

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

Unknown adapter Local Area Connection:

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

Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::1eb1:8f46:6b30:8eb5%16
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 1:

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

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2409:40f4:2003:ea7f:cc6e:f011:359b:f9d5
   Temporary IPv6 Address. . . . . . : 2409:40f4:2003:ea7f:1415:f71c:8ee5:2866
   Link-local IPv6 Address . . . . . : fe80::b037:27c8:c8f2:3827%20
   Autoconfiguration IPv4 Address. . : 169.254.53.36
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : fe80::481e:abff:fe9e:c05%20
```

17) ipconfig /renew

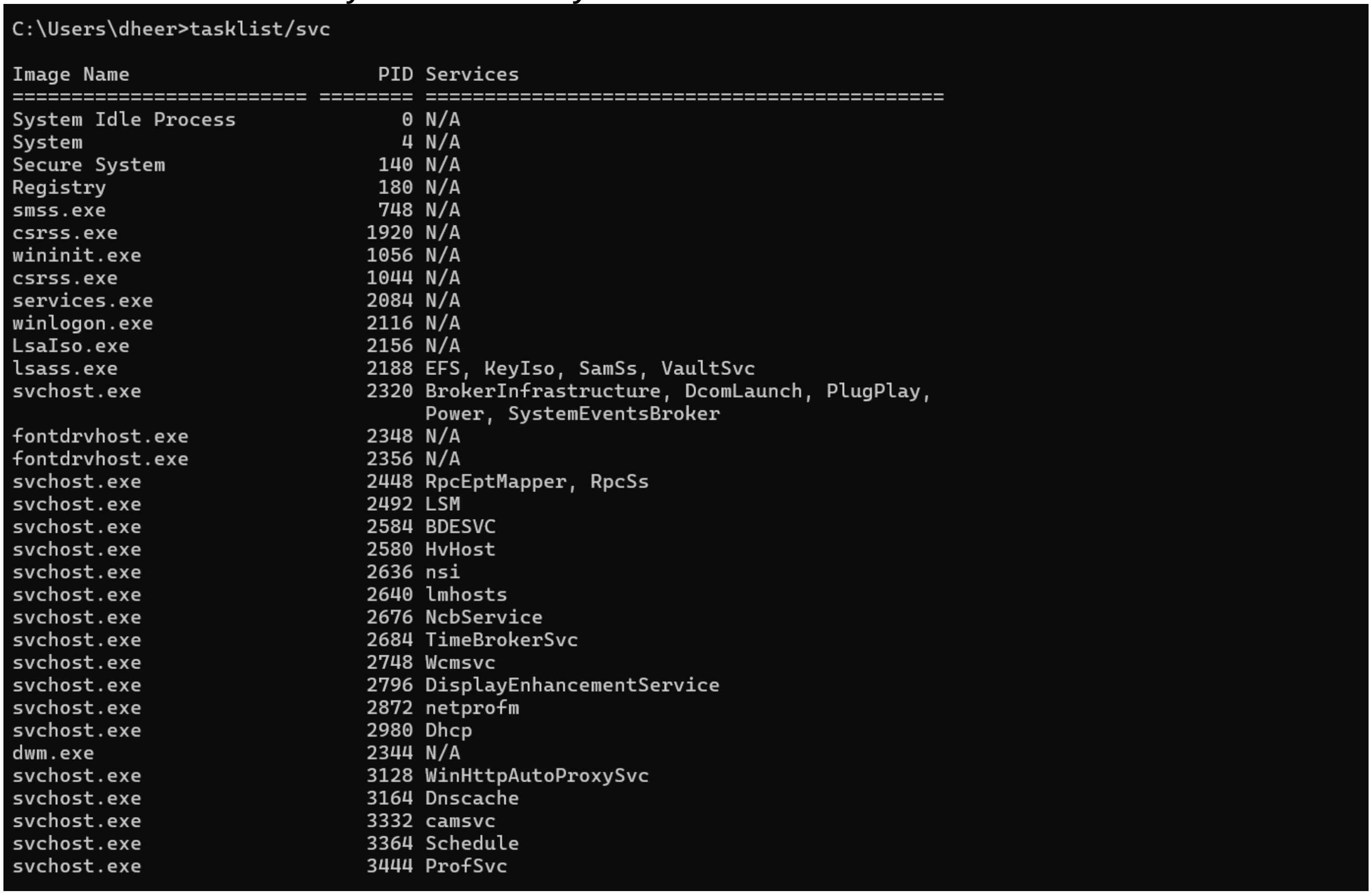
The ipconfig /renew command in Windows is used to renew the DHCP lease for all network adapters. This means it will request a new IP address from the DHCP server for the network interfaces, effectively updating the current IP address configuration.



```
C:\Arimadhan_CN>ipconfig /renew

Windows IP Configuration

No operation can be performed on Local Area Connection* 1 while it has its media disconnected.
No operation can be performed on Local Area Connection* 2 while it has its media disconnected.

Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f3d4:a288:85a4:bf%4
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 1:

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

Wireless LAN adapter Local Area Connection* 2:

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

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2409:4072:8e40:9baf:5a38:7259:68db:6ff9
   Temporary IPv6 Address. . . . . . : 2409:4072:8e40:9baf:1c48:2aab:6790:68f8
   Link-local IPv6 Address . . . . . : fe80::afab:d55d:4102:1a6c%5
   IPv4 Address. . . . . . . . . . . : 192.168.188.79
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::ac7e:47ff:feb6:7e7%5
                                       192.168.188.26
```

18) tasklist / svc

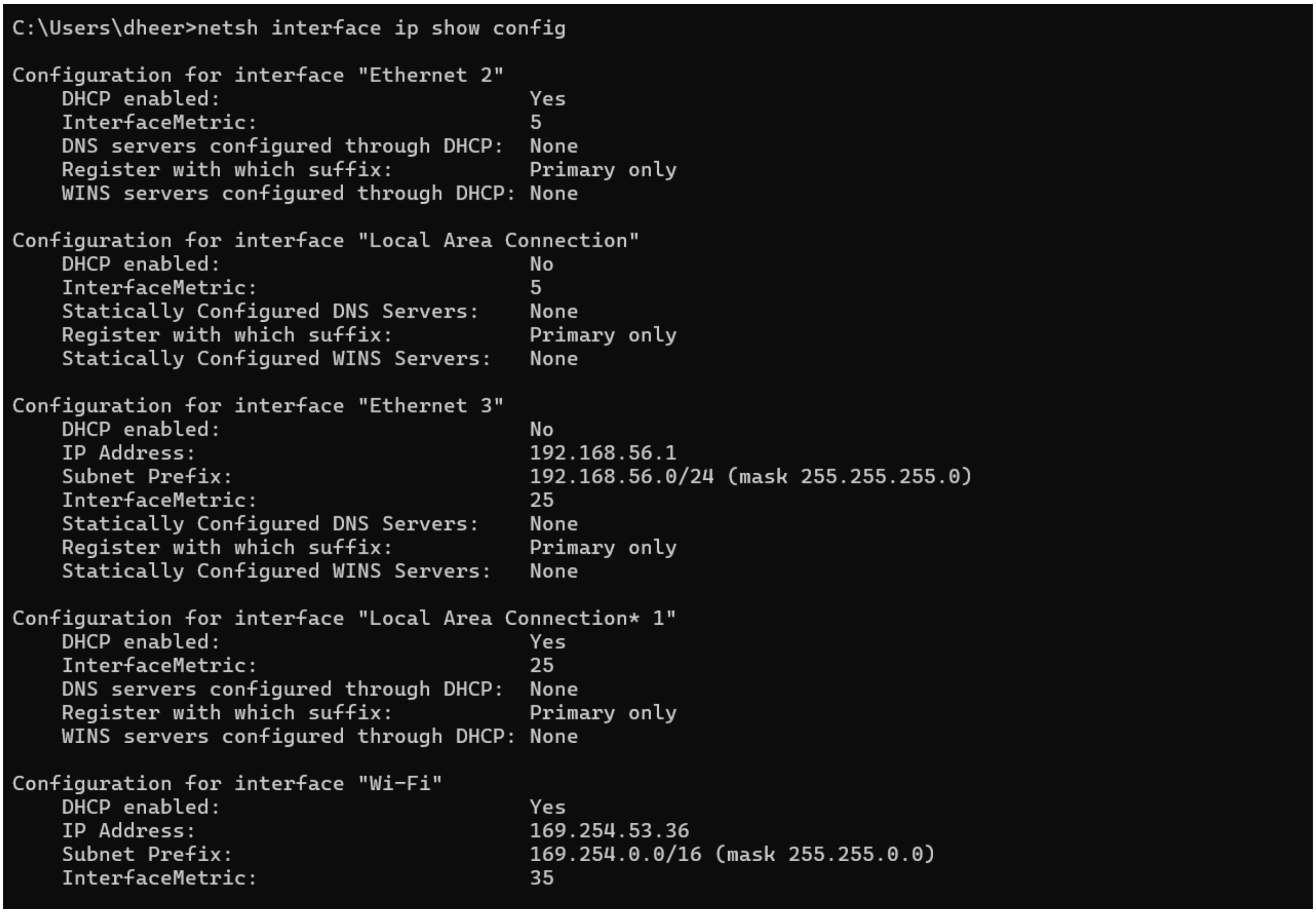
The tasklist /svc command in Windows is used to display a list of active processes and the services that are running within each process. This is useful for identifying which services are associated with which processes, providing a detailed view of the system's activity.



```
C:\Users\dheer>tasklist/svc

Image Name                     PID Services
========================= ======== ============================================
System Idle Process              0 N/A
System                           4 N/A
Secure System                  140 N/A
Registry                       180 N/A
smss.exe                       748 N/A
csrss.exe                     1920 N/A
wininit.exe                   1056 N/A
csrss.exe                     1044 N/A
services.exe                  2084 N/A
winlogon.exe                  2116 N/A
LsaIso.exe                    2156 N/A
lsass.exe                     2188 EFS, KeyIso, SamSs, VaultSvc
svchost.exe                   2320 BrokerInfrastructure, DcomLaunch, PlugPlay,
                                   Power, SystemEventsBroker
fontdrvhost.exe               2348 N/A
fontdrvhost.exe               2356 N/A
svchost.exe                   2448 RpcEptMapper, RpcSs
svchost.exe                   2492 LSM
svchost.exe                   2584 BDESVC
svchost.exe                   2580 HvHost
svchost.exe                   2636 nsi
svchost.exe                   2640 lmhosts
svchost.exe                   2676 NcbService
svchost.exe                   2684 TimeBrokerSvc
svchost.exe                   2748 Wcmsvc
svchost.exe                   2796 DisplayEnhancementService
svchost.exe                   2872 netprofm
svchost.exe                   2980 Dhcp
dwm.exe                       2344 N/A
svchost.exe                   3128 WinHttpAutoProxySvc
svchost.exe                   3164 Dnscache
svchost.exe                   3332 camsvc
svchost.exe                   3364 Schedule
svchost.exe                   3444 ProfSvc
```

19) netsh interface ip show config

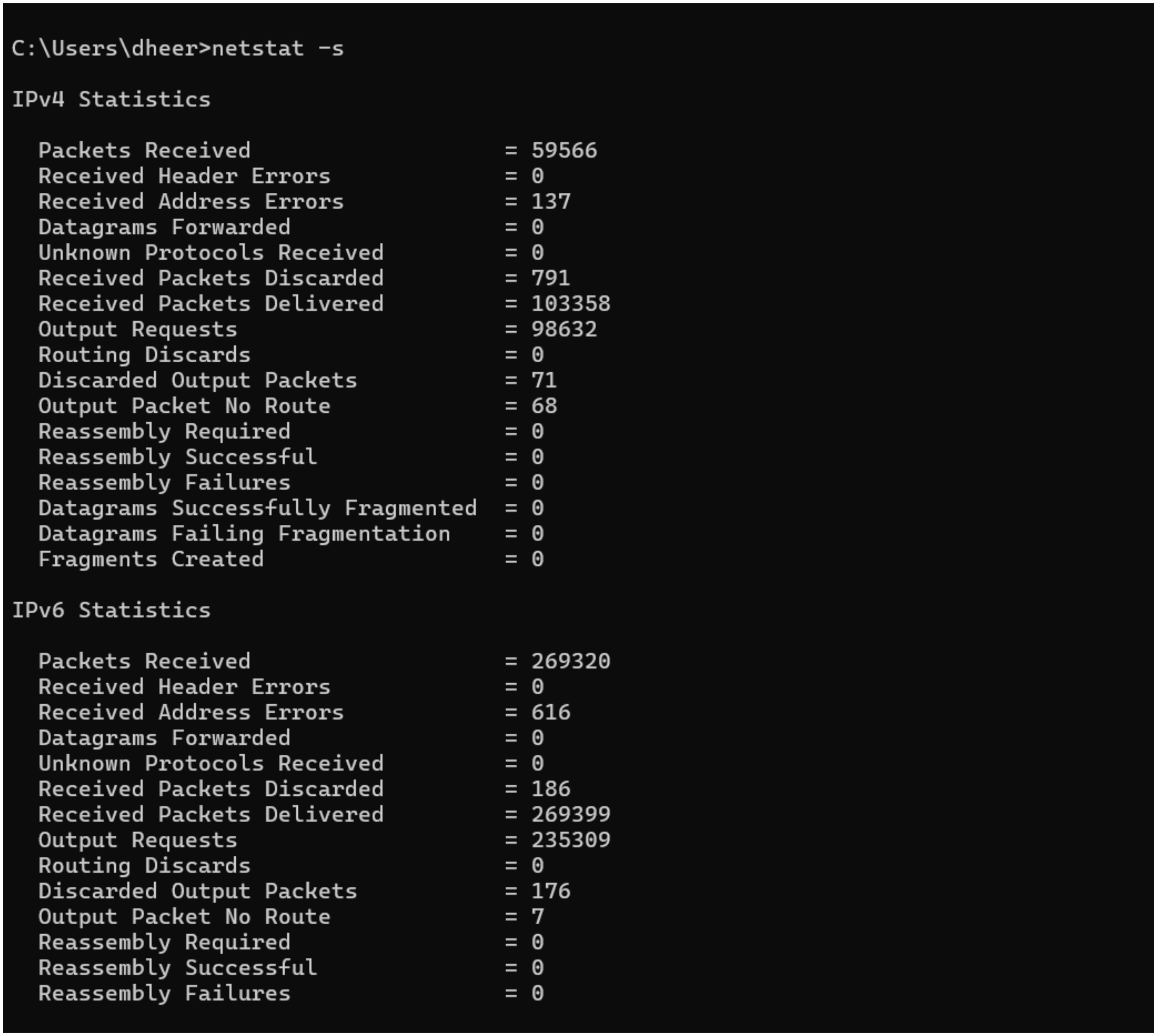
The netsh interface ip show config command in Windows is used to display detailed configuration information for all network interfaces (both IPv4 and IPv6) on the system. This includes IP addresses, subnet masks, default gateways, DNS servers, and more.



```
C:\Users\dheer>netsh interface ip show config

Configuration for interface "Ethernet 2"
    DHCP enabled:                         Yes
    InterfaceMetric:                      5
    DNS servers configured through DHCP:  None
    Register with which suffix:           Primary only
    WINS servers configured through DHCP: None

Configuration for interface "Local Area Connection"
    DHCP enabled:                         No
    InterfaceMetric:                      5
    Statically Configured DNS Servers:    None
    Register with which suffix:           Primary only
    Statically Configured WINS Servers:   None

Configuration for interface "Ethernet 3"
    DHCP enabled:                         No
    IP Address:                           192.168.56.1
    Subnet Prefix:                        192.168.56.0/24 (mask 255.255.255.0)
    InterfaceMetric:                      25
    Statically Configured DNS Servers:    None
    Register with which suffix:           Primary only
    Statically Configured WINS Servers:   None

Configuration for interface "Local Area Connection* 1"
    DHCP enabled:                         Yes
    InterfaceMetric:                      25
    DNS servers configured through DHCP:  None
    Register with which suffix:           Primary only
    WINS servers configured through DHCP: None

Configuration for interface "Wi-Fi"
    DHCP enabled:                         Yes
    IP Address:                           169.254.53.36
    Subnet Prefix:                        169.254.0.0/16 (mask 255.255.0.0)
    InterfaceMetric:                      35
```

20) netstat -s

The netstat -s command in Windows displays statistics for a variety of network protocols and services. It provides a comprehensive summary of network activity and performance metrics, which can be useful for diagnosing network issues and monitoring network usage.



```
C:\Users\dheer>netstat -s

IPv4 Statistics

  Packets Received                   = 59566
  Received Header Errors             = 0
  Received Address Errors            = 137
  Datagrams Forwarded                = 0
  Unknown Protocols Received         = 0
  Received Packets Discarded         = 791
  Received Packets Delivered         = 103358
  Output Requests                    = 98632
  Routing Discards                   = 0
  Discarded Output Packets           = 71
  Output Packet No Route             = 68
  Reassembly Required                = 0
  Reassembly Successful              = 0
  Reassembly Failures                = 0
  Datagrams Successfully Fragmented  = 0
  Datagrams Failing Fragmentation    = 0
  Fragments Created                  = 0

IPv6 Statistics

  Packets Received                   = 269320
  Received Header Errors             = 0
  Received Address Errors            = 616
  Datagrams Forwarded                = 0
  Unknown Protocols Received         = 0
  Received Packets Discarded         = 186
  Received Packets Delivered         = 269399
  Output Requests                    = 235309
  Routing Discards                   = 0
  Discarded Output Packets           = 176
  Output Packet No Route             = 7
  Reassembly Required                = 0
  Reassembly Successful              = 0
  Reassembly Failures                = 0
```

KUMARAGURU COLLEGE OF TECHNOLOGY

Exercise/Experiment Number: 7

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

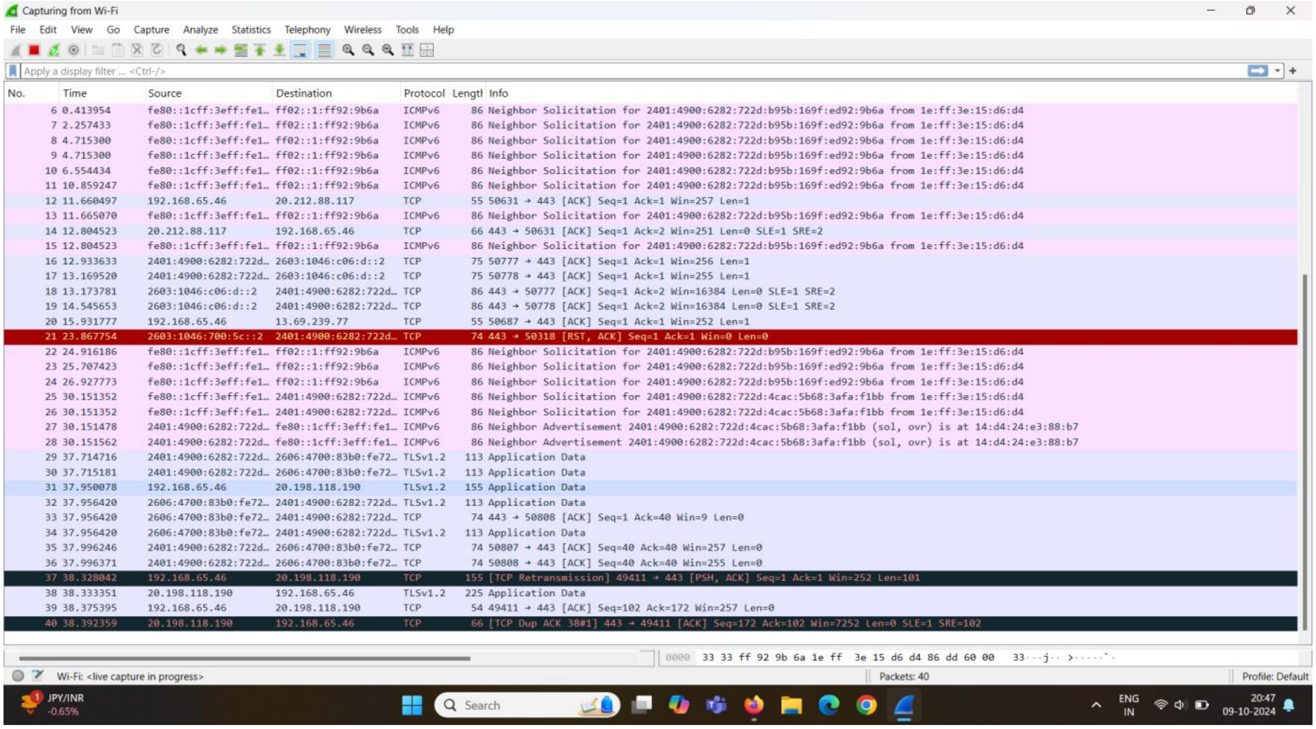
AIM : To know how to capture packets in wireshark

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

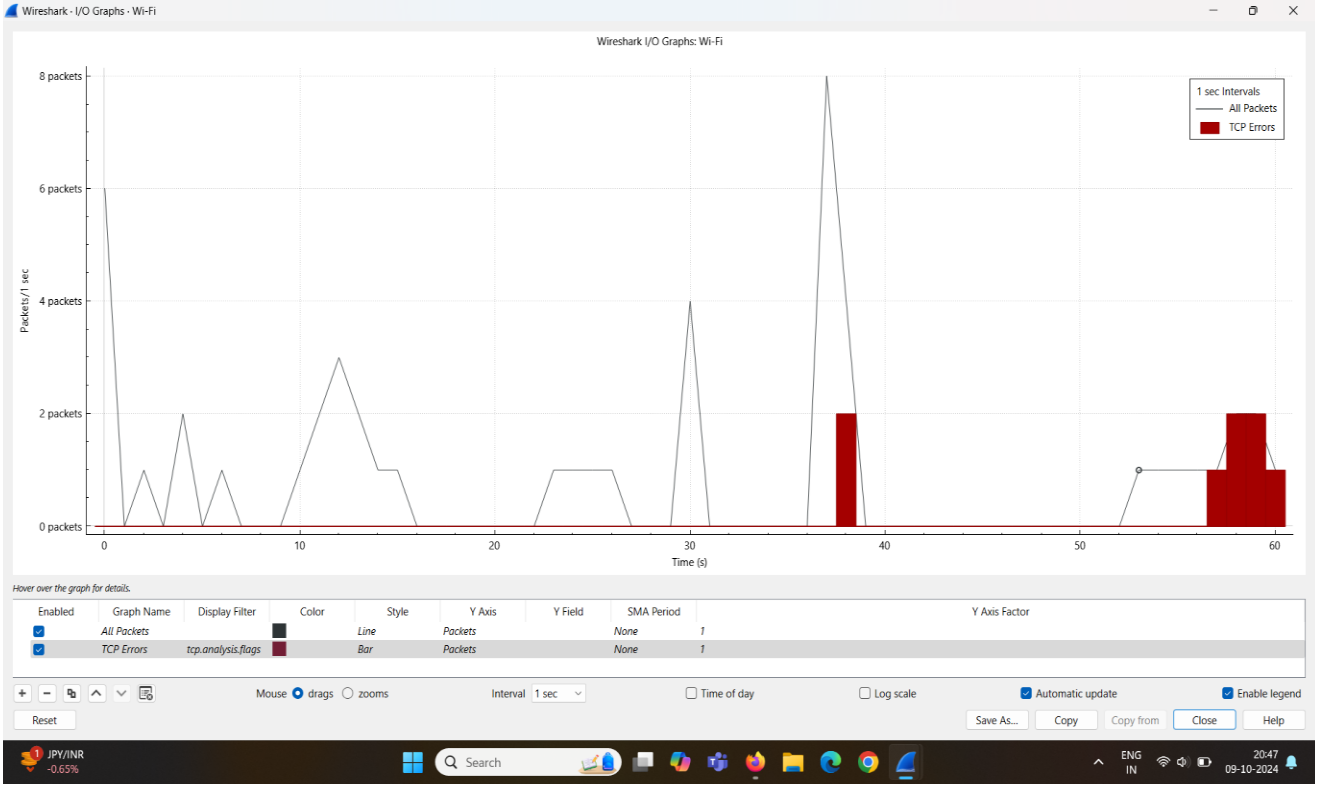
OUTPUT :



Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

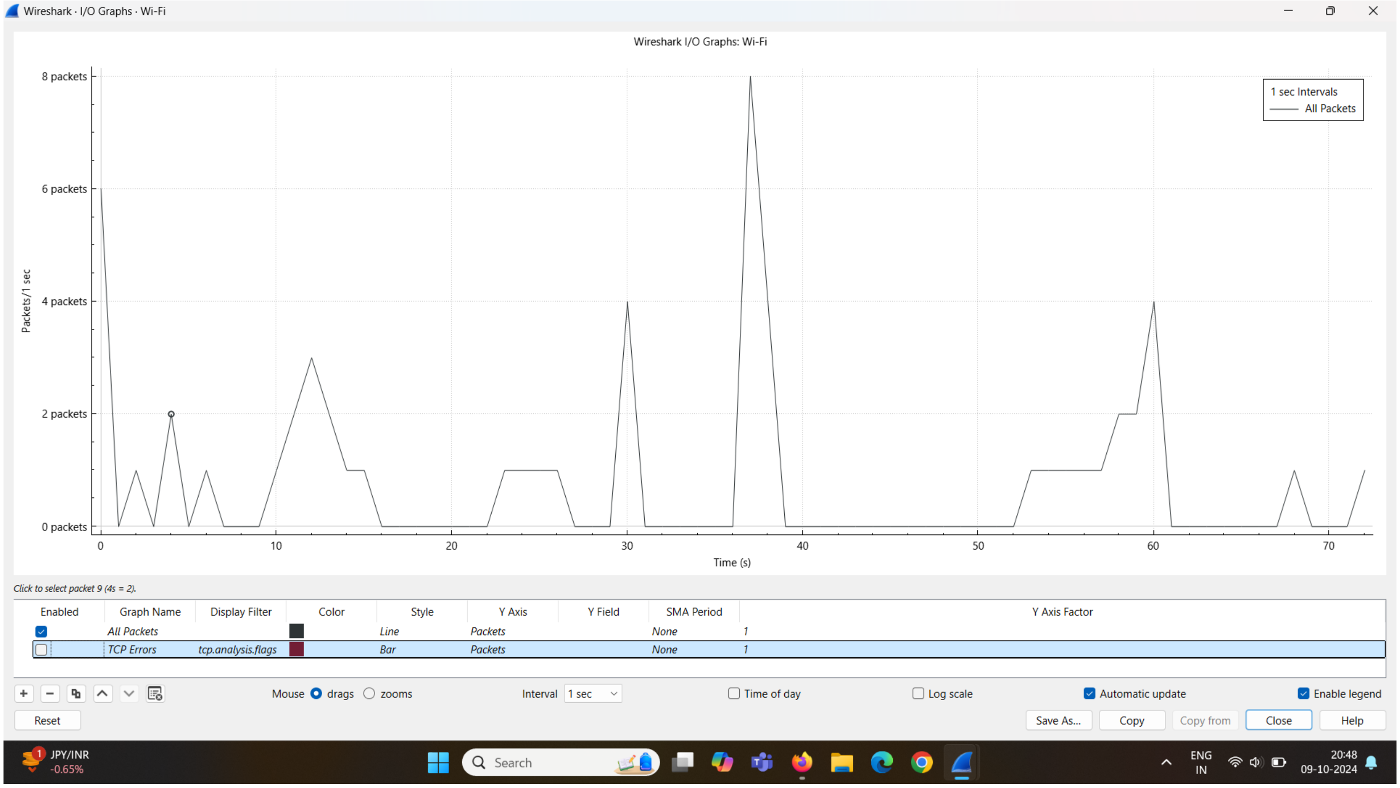
```
No.   Time       Source                 Destination            Protocol Length Info
    6 0.413954   fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
    7 2.257433   fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
    8 4.715300   fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
    9 4.715300   fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
   10 6.554434   fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
   11 10.859247  fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
   12 11.660497  192.168.65.46          20.212.88.117          TCP       55 50631 → 443 [ACK] Seq=1 Ack=1 Win=257 Len=1
   13 11.665070  fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
   14 12.804523  20.212.88.117          192.168.65.46          TCP       66 443 → 50631 [ACK] Seq=1 Ack=2 Win=251 Len=0 SLE=1 SRE=2
   15 12.804523  fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
   16 12.933633  2401:4900:6282:722d…   2603:1046:c06:d::2     TCP       75 50777 → 443 [ACK] Seq=1 Ack=1 Win=256 Len=1
   17 13.169520  2401:4900:6282:722d…   2603:1046:c06:d::2     TCP       75 50778 → 443 [ACK] Seq=1 Ack=1 Win=255 Len=1
   18 13.173781  2603:1046:c06:d::2     2401:4900:6282:722d…   TCP       86 443 → 50777 [ACK] Seq=1 Ack=2 Win=16384 Len=0 SLE=1 SRE=2
   19 14.545653  2603:1046:c06:d::2     2401:4900:6282:722d…   TCP       86 443 → 50778 [ACK] Seq=1 Ack=2 Win=16384 Len=0 SLE=1 SRE=2
   20 15.931777  192.168.65.46          13.69.239.77           TCP       55 50687 → 443 [ACK] Seq=1 Ack=1 Win=252 Len=1
   21 23.867754  2603:1046:700:5c::2    2401:4900:6282:722d…   TCP       74 443 → 50318 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   22 24.916186  fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
   23 25.707423  fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
   24 26.927773  fe80::1cff:3eff:fe1…   ff02::1:ff92:9b6a      ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:b95b:169f:ed92:9b6a from 1e:ff:3e:15:d6:d4
   25 30.151352  fe80::1cff:3eff:fe1…   2401:4900:6282:722d…   ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:4cac:5b68:3afa:f1bb from 1e:ff:3e:15:d6:d4
   26 30.151352  fe80::1cff:3eff:fe1…   2401:4900:6282:722d…   ICMPv6    86 Neighbor Solicitation for 2401:4900:6282:722d:4cac:5b68:3afa:f1bb from 1e:ff:3e:15:d6:d4
   27 30.151478  2401:4900:6282:722d…   fe80::1cff:3eff:fe1…   ICMPv6    86 Neighbor Advertisement 2401:4900:6282:722d:4cac:5b68:3afa:f1bb (sol, ovr) is at 14:d4:24:e3:88:b7
   28 30.151562  2401:4900:6282:722d…   fe80::1cff:3eff:fe1…   ICMPv6    86 Neighbor Advertisement 2401:4900:6282:722d:4cac:5b68:3afa:f1bb (sol, ovr) is at 14:d4:24:e3:88:b7
   29 37.714716  2401:4900:6282:722d…   2606:4700:83b0:fe72…   TLSv1.2  113 Application Data
   30 37.715181  2401:4900:6282:722d…   2606:4700:83b0:fe72…   TLSv1.2  113 Application Data
   31 37.950078  192.168.65.46          20.198.118.190         TLSv1.2  155 Application Data
   32 37.956420  2606:4700:83b0:fe72…   2401:4900:6282:722d…   TLSv1.2  113 Application Data
   33 37.956420  2606:4700:83b0:fe72…   2401:4900:6282:722d…   TCP       74 443 → 50808 [ACK] Seq=1 Ack=40 Win=9 Len=0
   34 37.956420  2606:4700:83b0:fe72…   2401:4900:6282:722d…   TLSv1.2  113 Application Data
   35 37.996246  2401:4900:6282:722d…   2606:4700:83b0:fe72…   TCP       74 50807 → 443 [ACK] Seq=40 Ack=40 Win=257 Len=0
   36 37.996371  2401:4900:6282:722d…   2606:4700:83b0:fe72…   TCP       74 50808 → 443 [ACK] Seq=40 Ack=40 Win=255 Len=0
   37 38.328042  192.168.65.46          20.198.118.190         TCP      155 [TCP Retransmission] 49411 → 443 [PSH, ACK] Seq=1 Ack=1 Win=252 Len=101
   38 38.333351  20.198.118.190         192.168.65.46          TLSv1.2  225 Application Data
   39 38.375395  192.168.65.46          20.198.118.190         TCP       54 49411 → 443 [ACK] Seq=102 Ack=172 Win=257 Len=0
   40 38.392359  20.198.118.190         192.168.65.46          TCP       66 [TCP Dup ACK 38#1] 443 → 49411 [ACK] Seq=172 Ack=102 Win=7252 Len=0 SLE=1 SRE=102

0000  33 33 ff 92 9b 6a 1e ff  3e 15 d6 d4 86 dd 60 00   33···j·· >·····`·
```

Wi-Fi: <live capture in progress>   Packets: 40   Profile: Default



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	All Packets		■	Line	Packets		None	1
☑	TCP Errors	tcp.analysis.flags	■	Bar	Packets		None	1

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



Click to select packet 9 (4s = 2).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	*All Packets*		⬛	*Line*	*Packets*		*None*	*1*
☐	*TCP Errors*	*tcp.analysis.flags*	🟥	*Bar*	*Packets*		*None*	*1*

Mouse ● drags ○ zooms

Interval 1 sec

☐ Time of day

☐ Log scale

☑ Automatic update

☑ Enable legend