**Exp 1**

### **RJ45**

**RJ45** is like the plug at the ends of your network cables, similar to how a charger plugs into your phone. It's the connector that lets your network cable hook up to your devices, like computers, routers, and switches. You might have noticed these connectors—they look like slightly bigger versions of the connectors on phone cables, with a little plastic clip that snaps into place.

* **Why it's important**: RJ45 connectors are essential because they help your devices talk to each other over the network, letting you access the internet or share files between computers.

### **CAT6 Cable**

**CAT6** is the actual cable that carries the information, like a road for data to travel on. It's a step up from older types of network cables (like CAT5e) because it's designed to handle more data faster and with less interference.

* **Speed and performance**: CAT6 can support really fast internet speeds—up to 10 Gbps for shorter distances (about 55 meters). It also reduces the chance of interference, which means fewer hiccups in your internet connection.

### **How They Work Together**

Think of CAT6 and RJ45 as a team:

* **CAT6 cable** is the super-fast highway for your data.
* **RJ45 connector** is the entrance ramp that connects your highway (cable) to your devices.

When you plug the RJ45 connector of a CAT6 cable into your computer and your router, you're creating a fast, stable connection that lets your devices communicate smoothly. This combination is perfect for things like streaming HD videos, gaming, or handling large file transfers, ensuring everything runs quickly and reliably.

In short, RJ45 and CAT6 work hand-in-hand to make sure you get a strong, fast, and reliable network connection.

### **Practical Applications**

* **Home Networking**: If you're setting up a home network, using CAT6 cables with RJ45 connectors can future-proof your setup, supporting the latest high-speed internet services and devices.
* **Office Networks**: In offices where multiple users need reliable and fast access to shared resources, CAT6 ensures that the network can handle the load without slowdowns.
* **Data Centers**: In data centers, where massive amounts of data are transmitted constantly, CAT6 cables help maintain high-speed connections with minimal latency.

**Exp 2**

sudo apt install net-tools

1.ifconfig (gives ip address)

2.ping 192.168.3.25 (check connectivity)

3.netstat (shows information about active network connections)

4.route -n (to show or update the Kernel IP routing table)

5.whois (to retrieve information about domain names, IP addresses, and network devices)

6.dig (to show about the Domain Name System (DNS) servers)

7.date (shows day, date and time)

8.nmap [www.google.com](http://www.google.com) (for network exploration and security auditing)

9.traceroute [www.google.com](http://www.google.com) (to trace the route an IP packet follows to an Internet host by launching UDP probe packets)

10.nslookup [www.google.com](http://www.google.com) (find the IP address or DNS record of a domain name)

11. ARP (ARP is used in sender's side to map the receiver's MAC)

12.RARP (RARP is used in receiver's side to map the sender's IP)

13.ip addr show (allows users to interact with various networking components)

**Exp 3**

cisco packet tracer

**Exp 4**

**STEP 1 - Open terminal in your system or press ALT + CTRL + T and run the below command:**

sudo add-apt-repository ppa:wireshark-dev/stable

**STEP 2 - Update the repository:**

sudo apt-get update

**STEP 3 - Install wire shark using the** **below command:**

sudo apt-get install wireshark

**STEP 4 - To run the wire shark use the** **below command**

sudo wireshark

tcp, udp, eth, arp, ip

tcp.port == 443 -

udp.port == 443 -

ip addr == 192.168.3.10 -

Go to capture > capture > go to edit > go to edit > preferences > capture > check the option “Capture packet in promiscuous mode”

**Exp 5**

sudo apt-get install ns2

sudo apt-get install tcl

File name (harsh.tcl)

Command to run (ns harsh.tcl) and exec nam out.nam &

Code :

# Create a simulator object

set ns [new Simulator]

# Define different colors for data flows

$ns color 1 Blue

$ns color 2 Red

# Open the nam trace file

set nf [open out.nam w]

$ns namtrace-all $nf

# Create two nodes

set n0 [$ns node]

set n1 [$ns node]

# Create a duplex link between the nodes

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

# Create a UDP agent and attach it to node n0

set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0

# Create a Null agent (a sink) and attach it to node n1

set null1 [new Agent/Null]

$ns attach-agent $n1 $null1

# Connect the UDP agent to the Null agent

$ns connect $udp0 $null1

# Create a CBR traffic generator and attach it to the UDP agent

set cbr [new Application/Traffic/CBR]

$cbr set packetSize\_ 500

$cbr set interval\_ 0.005

$cbr attach-agent $udp0

# Schedule the traffic to start at time 0.5 seconds and stop at 4.5 seconds

$ns at 0.5 "$cbr start"

$ns at 4.5 "$cbr stop"

# Schedule the finish procedure to be called after 5 seconds

$ns at 5.0 "finish"

# Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

# Close the trace file

close $nf

# Execute nam on the trace file

exec nam out.nam &

exit 0

}

# Run the simulation

$ns run

**Exp 6**

set ns [new Simulator]

$ns color 1 Blue

# set nam output file

set nf [open out.nam w]

$ns namtrace-all $nf

# destructor

proc finish {} {

global ns nf

$ns flush-trace

close $nf

exec nam out.nam &

exit 0

}

# create two new nodes and create labels for them

set n0 [$ns node]

set n1 [$ns node]

$ns at 0.0 "$n0 label \" Sender \" "

$ns at 0.0 "$n1 label \"Receiver\" "

# set up a new duplex link

$ns duplex-link $n0 $n1 1Mb 200ms DropTail

$ns duplex-link-op $n0 $n1 orient right

# create a new TCP agent

set tcp [new Agent/TCP]

# attach the agent to first node

$ns attach-agent $n0 $tcp

$tcp set fid\_ 1

$tcp set window\_ 1

$tcp set maxcwnd\_ 1

$ns add-agent-trace $tcp tcp

$ns monitor-agent-trace $tcp

set tcpsink [new Agent/TCPSink]

$ns attach-agent $n1 $tcpsink

$ns connect $tcp $tcpsink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns at 0.5 "$ftp start"

$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $tcpsink "

$ns at 1.0 "$ns trace-annotate \"send packet 1\""

$ns at 1.4 "$ns trace-annotate \"receive ack 1\""

$ns at 2.0 "$ns trace-annotate \"send packet 2\""

$ns at 2.5 "$ns trace-annotate \"receive ack 2\""

$ns at 3.2 "$ns trace-annotate \"send packet 3\""

$ns at 3.5 "$ns trace-annotate \"receive ack 3\""

$ns at 3.8 "$ns trace-annotate \"send packet 4\""

$ns at 4.0 "finish"

$ns run

**Exp 7**

sudo apt-get install nmap

nmap 192.168.3.45 (scan a System with Hostname and IP address)

nmap www.geeksforgeeks.org

nmap 172.217.27.174

nmap -p 1-80 192.168.3.45 (specify the range of ports to scan on a network)

sudo nmap -sN 192.168.3.45 (to pin scan)

sudo nmap -sA 192.168.3.45 (detect firewalls settings)

sudo nmap -sV 192.168.3.45 (version)

sudo nmap -sS 192.168.3.45 (to TCP syn scan)

sudo -A 192.168.3.45 (give us extra information, like OS detection (-O), version detection, script scanning (-sC) and traceroute (–traceroute))

U:53

-v -A

-p T:80

**EXP 8**

**TCP Client**

import java.io.\*;

import java.net.\*;

public class Client{

private Socket socket=null;

private PrintWriter out=null;

private BufferedReader input=null;

private BufferedReader serverInput=null;

public Client(String address, int port){

try{

socket=new Socket(address, port);

System.out.println("Client connected");

serverInput=new BufferedReader(new InputStreamReader(socket.getInputStream()));

out=new PrintWriter(socket.getOutputStream(), true);

input=new BufferedReader(new InputStreamReader(System.in));

Thread readThread=new Thread(()->{

try{

String line;

while((line=serverInput.readLine())!=null)

{

System.out.println("Server says: "+line);

}

} catch(IOException e){

System.out.println("Server Disconnected");

}

});

readThread.start();

String line;

while(!(line=input.readLine()).equals("Over"))

{

System.out.println("Client Says: "+line);

out.println(line);

}

} catch(IOException i)

{

System.out.println("Client Disconnected");

}

}

public static void main(String args[])

{

new Client("127.0.0.1", 5000);

}

}

**TCP Server**

import java.io.\*;

import java.net.\*;

public class Server {

    private Socket socket = null;

    private ServerSocket server = null;

    private BufferedReader clientInput = null;

    private PrintWriter out = null;

    private BufferedReader input = null;

    public Server(int port) {

        try {

            server = new ServerSocket(port);

            System.out.println("Server started");

            System.out.println("Waiting for a client ...");

            socket = server.accept();

            System.out.println("Client accepted");

            clientInput = new BufferedReader(new InputStreamReader(socket.getInputStream()));

                out = new PrintWriter(socket.getOutputStream(), true);

                input = new BufferedReader(new InputStreamReader(System.in));

            // Thread to handle client messages

            Thread readThread = new Thread(() -> {

                String line;

                try {

                    while ((line = clientInput.readLine()) != null) {

                        System.out.println("Client says: " + line);  // Debug line for server

                    }

                } catch (IOException e) {

                    System.out.println("Connection closed by client.");

                }

            });

            readThread.start();

            // Main thread for sending messages to client

            String line;

            while (!(line = input.readLine()).equals("Over")) {

                System.out.println("Server sending: " + line);  // Debug line for server

                out.println(line);

            }

            // Close everything

            socket.close();

            clientInput.close();

            out.close();

            input.close();

        } catch (IOException i) {

            System.out.println("Server error: " + i);

        }

    }

    public static void main(String args[]) {

        new Server(5000);

    }

}

**UDP Server**

import socket

# Create and configure the UDP server socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_socket.bind(('localhost', 12345))

print("UDP Server is listening on port 12345...")

# Receive data from a client

data, addr = server\_socket.recvfrom(1024)

print("Received from client:", data.decode())

# Send a response back to the client

server\_socket.sendto("Hello from UDP server!".encode(), addr)

**UDP Client**

import socket

# Create a UDP client socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# Send a message to the server

client\_socket.sendto("Hello from UDP client!".encode(), ('localhost', 12345))

# Receive the response from the server

data, server\_addr = client\_socket.recvfrom(1024)

print("Received from server:", data.decode())

# Close the socket

client\_socket.close()

**Exp 9**

sudo apt update

sudo apt install vsftpd

sudo systemctl enable vsftpd

comp@comp-HP:~$ sudo systemctl start vsftpd

comp@comp-HP:~$ sudo nano /etc/vsftpd.conf

comp@comp-HP:~$ sudo systemctl restart vsftpd

comp@comp-HP:~$ sudo ufw allow ftp

sudo ufw allow ftp

sudo useradd -m testuser

sudo passwd testuser

sudo ftp comp@...

ftp > open

(to) 192.168.3.45

**Exp10**

sudo apt-get install telnetd -y

sudo systemctl enable telnet

sudo systemctl start telnet

Sudo systemctl status netd

telnet [www.google.com](http://www.google.com) 80

top (list)

kill Pid(7008)

**tcp\_server.py**

import socket

import time

# Create a TCP socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('localhost', 12345))

server\_socket.listen(1)

print("TCP Server listening on port 12345...")

# Accept a connection from a client

client\_socket, addr = server\_socket.accept()

print(f"Connection from {addr}")

# Receive data

data = client\_socket.recv(1024).decode()

print("Received from client:", data)

# Send response to client

client\_socket.send("Hello from TCP server!".encode())

# Close the connection

time.sleep(1)

client\_socket.close()

server\_socket.close()

**tcp\_client.py**

import socket

# Create a TCP socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(('localhost', 12345))

# Send data to server

client\_socket.send("Hello from TCP client!".encode())

# Receive response from server

data = client\_socket.recv(1024).decode()

print("Received from server:", data)

# Close the connection

client\_socket.close()