Experiment no: 1

Aim: To learn handling and configuration of networking hardware like RJ-45 connector, CAT-6 cable, crimping tool, etc.

Requirement:
- Ethernet cable - category 5e or CAT5e or CAT6.
- RJ-45 crimping tool
- RJ45 crimpable connectors

Theory:

RJ45 connector

RJ45 is a type of connector commonly used for Ethernet networking. It looks similar to a telephone Jack, but is slightly wider.
- The 'RJ' in RJ45 stands for "registered jack", since it is a standardized networking interface.
- The "45" simply refers to the number of the interface standard.
- Each RJ45 connector has eight pins, which means an RJ45 cable contain eight separate wires.

T-568A
- White / Green (Receive +)
- White / orange (Transmit +)

T-568B
- Blue
- White / orange (Transmit +)

Cat 6 Cable

Category 6 is an Ethernet cable standard defined by the Electronic Industries Association (EIA) and Telecommunication Industry Association (TIA).
- Cat 6 is the sixth generation of twisted pair Ethernet cabling that is used in home and business network.
- Cat 6 cabling is backward compatible with the cat 5 and cat 5e standards that preceded it.
- Compared with 5 and 5e, cat 6 features more stringent specification for crosstalk and system noise.

Working:

Cat6 cables support Gigabit Ethernet data rates of 1 gigabit per second. They can accomodate 10 Gigabit Ethernet connections over a limited distance 164 feet for a single cable. Cat 6 cable contains four pairs of copper wire and uses all pairs for signaling in order to obtain its highly level of performance.

Crimping tool

A crimping tool is a device used to conjoin two pieces of metal by deforming one or both of them in a way that causes them to hold each other. The result of the tool's work is called crimp. A good example of crimping is the process of affixing a connector to a end of a cable.

Working:

To use this Crimping tool, each wire is first placed into the connector. Once all the wires are in the Jack, the connectors with wires are placed into the crimping tool, and the handles are squeezed together. Crimping punctures the plastic connector and holds each of the wires, allowing for data to be transmitted through the connector.

Experiment No:2

Aim: Configuration of router, hub, switch etc. (using real devices or simulators).

Apparatus (Software): No software or hardware needed

Theory:

Configuration of Router

A router is a networking device that forwards data packets b/w computer networks. Routers perform the traffic directing functions on the Internet. Data sent through the internet, such as web page or email, is in the form of data packages.

- A router is connected to two or more data lines from different networks. When a data packet comes in one of the lines, the router reads the network address information in the packet to determine the ultimate destination.
- Capabilities of router:
  A router has a lot more capabilities than other network devices, such as a hub or a switch that are only able to perform basic network functions.
- Router types:
  - → Wireless (Wi-Fi) router
  - → Brouter
  - → Core router
  - → Virtual router

Configuration of Hub

A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations.

Hub cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one. Also, they do not have intelligence to find out best path for data packets which leads to inefficiencies and wastage.

Types of Hub:

- Active Hub
- Passive Hub
- Fast Ethernet classes
- Dual-speed hub
- Gigabit Ethernet hub

Configuration of Switch

Switching is the most valuable asset of computer networking. Every time in computer network you access the internet or another computer outside your immediate location, or your messages are sent through a maze of transmission media and connection devices.

Hardware devices that can be used for switching or transfeering data from one location to another that can use multiple layers of the Open Systems Interconnection (OSI)

Types of Switching Techniques:

- Circuit Switching: It is a real time connection Oriented System. In this, dedicated channel is set up for a single connection between the sender and recipient during the communication session.
- Packet Switching: The best example of packet switching is the Internet. In this, data can be fragment into suitably-sized pieces in variable length are called Packet.
- Message Switchiy: Message switching does not set up a dedicated channel (or circuit) between the sender and recipient during the communication session.

Experiment No: 3

Aim: Running and Using Services/ commands like Ping, tracert, route, nslookup, arp, etc.

Theory:

1. Ping:

Ping is a basic Internet program that allows a user to verify that a particular IP address exist and can be accept requests.

Ping is used diagnostically to ensure that a host computer the user is trying to reach is actually operating. Ping works by sending an Internet control Message Protocol (ICMP).

Ping Command Syntax:

Ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS] [-r count] [-s count] [-w timeout] [-R] [-S srcaddr] [-P] [-4] [-6] target [/?]

2. Traceroute:

A traceroute is a function which traces the path from one network to another. It allows us to diagnose the source of many problems.

The tracert command is a command prompt command that's used to show several details about the path that a packet takes from the computer or device you're on to whatever destination your specify.

Tracert Command Syntax:

tracert [-d] [-h maxHops] [-w Timeout] [-4] [-6] [target [/?]

3. Command nslookup:

The nslookup (which stands for name server lookup) command is a network utility program used to obtain information about internet servers. It finds name server information for domains by querying the Domain Name System.

command nslookup sends a domain name query packet to a destinated (or defaulted) domain name system (DNS) server.

nslookup command syntax:

nslookup [-SubCommand ...] [{computerToFind | [-server]}]

4. ARP (Address Resolution Protocol)

ARP is used with the IP for mapping a 32-bit Internet Protocol address to a MAC address that is recognized in the local method in RFC 826. Once recognized, the server or networking device returns a response containing the required address

ARP command Syntax:

ARP -s inet_addr eth_addr [if_addr]
ARP -d inet_addr [if_addr]

5. TelNet:

Telnet is a user command and an underlying TCP/I protocol for accessing remote computers. Through TelNet, an administrator or another user can access someone else's computer remotely.

TelNet command Syntax:

telnet [/a] [/e <Escape char>] [/if <File Name>] [/L <UserName>
[/t {vt100 | vt52 | ansi | vtnt}] [<Host> [<port>]] [/?]

Experiment No: 4

Aim: Implementation of stop and wait Protocol and Sliding window protocol.

Algorithm:

- Start the program.
- Get the frame size from the user.
- To create the frame based on the upper request
- To send frames to server from the client side.
- If your frames reach the server it will send Ack signal to client otherwise it will send Nack signal to client.
- Stop the program.

Program:

```
import java.net.*;
import java.io.*;
import java.rmi.*;
Public class SlidSender
{
Public static void main (String a[])throws Exception
{
ServerSocket ser = new ServerSocket(10);
 Socket s = ser.accept();
DataInputStream in = new DataInputStream(System.in);
DataInputStream in1 = new DataInputStream(s.getInput
                                          Stream());
String sbuff[] = new String[8];
 PrintStream p;
int sptr=0, sws=8, nf, ano, i;
String ch;
do{
  p= new PrintStream(s.getOutputStream());
  System.out.print("Enter the no. of frames: ");
nf = Integer.parseInt(in.readLine());
  p.println(nf);
  if (nf<=sws-1)
{
  System.out.println("Enter "+nf+" Messages to
    be send in ");
  for(i=1; i<=nf; i++)
```

```
{
Sbuff[sptr] = in.readLine();
p.println(sbuff[sptr]);
sptr = ++sptr % x;
 }
  sws = nf;
System.out.println("Acknowledgment received");
 ano = Integer.parseInt(in1.readLine());
System.out.println("for "+ano+" frames");
 sws+ = nf;
 }
 else
    { System.out.println("The no. of frames exceeds the
                          window size");
    break;
    }
   System.out.println("\n Do you wants to send some more
                       frames:");
   ch = in.readLine(); p.println(ch);
  }while(ch.equals("yes"));
  s.close();
  }}
```

Output:

```
Enter the no. of frames: 4
 Enter 4 message to be send:
hi
how r u
i am fine
how is everyone
Acknowledgment received for 4 frames.
Do you wants to send some more frames: no
```

Experiment No: 5

Aim: Implementation of Dijkstra's shortest path algorithm

Algorithm: → Create a set sptSet (shortest path tree set) that keeps track of vertices included in shortest path tree.
→ Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE.
→ Assign distance values as 0 for the source vertex so that it is picked first.
→ While sptSet doesn't include all vertices
→ Pick a vertex u which is not there in sptset and has minimum distance value.
• Include u to sptset.
• Update distance value of all adjacent vertices of u
• To update the distance value, iterate through all adjacent vertices.

Program:

```
import java.util.*;
import java.lang.*;
import java.io.*;
class shortestpath {
    static final int V=9;
int minDistance (int dist[], Boolean sptset[])
{
  int min = Integer.MAX_VALUE, min_index = -1;
for (int v=0; v<V; v++)
  if (sptset[v] == false && dist[v]<= min) {
    min = dist[v];
     min_index = v;
  }
  return min_index;
  }
void printSolution(int dist[],int n)
{ System.out.println("vertex Distance from Source");
  for (int i=0; i<V; i++)
   System.out.println(i+ "tt" + dist[i]);
  }
```

```
void dijkstra (int graph[][], int src)
{
  int dist[] = new int[V];
   Boolean sptset[] = new Boolean[V];
  for (int i=0; i<V; i++) {
     dist[i] = Integer.max_value;
      sptset[i] = false;
    }
dist[src]=0;
for (int count=0; count<V-1; count++) {
 int u = minDistance (dist, sptset);
  sptset[u]= true;
for ( int v=0; v<V; v++)
  if (!sptset[v] && graph[u][v]!=0 &&
 dist[u] != Integer.MAX_value && dist[u]+ graph[u][v]<
                                          dist[v]
   dist[v]=dist[u]+ graph[u][v];
    }
printSolution (dist,V);
 }
   Public static void main (String[] args)
  {int graph[][]=new Int[][]{
        { 0, 4, 0, 0, 0,0,0, 8,0},
        { 4,0,8,0,0,0,0,11,0},
        { 0, 8,0,7,0,4,0,0,2},
        { 0,0,7,0,9,14,0,0,0},
        { 0,0,0,9,0,10,0,0,0},
        { 0,0,4,14,10,0,2,0,0},
        { 0,0,0,0,0,2,0,1,6},
         { 8,11,0,0,0,0,1,0,7},
        { 0,0,2,0,0,0,6,7,0};
  ShortestPath t = new ShortestPath();
    t.dijkstra(graph,0);
 }}
```

Output:

```
Vertex Distance from Source
0 tt 0
1 tt 4
2 tt 12
3 tt 19
4 tt 21
5 tt 11
6 tt 9
7 tt 8
```

Experiment No: 6

Aim: Implementation of Distance Vector Routing (DVR) protocol.

Algorithm: $d_x(v) = \min_v \{ c(x,v) + d_v(y) \}$

- For each neighbour v, if the cost c(x,v) is the path cost from x to directly attached neighbour, v.
- The distance vector x i.e., $D_x = [D_x(y) : y \text{ in } N]$, containing its cost to all destinations, y, in N.
- The distance vector of each of its neighbours i.e., $D_v = [D_v(y) : y \text{ in } N]$ for each neighbour v of x.

Program:

```
import java.io.*;
Public Class DVR {
 Static int graph[][];
 Static int via[][];
 Static int rt[][];
  Static int v;
  Static int e;
 Public static void main(String args[]) throws IOException
 {
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
System.out.println("please enter the number of Vertices:");
V= Integer.parseInt(br.readline());
System.out.println("please enter the number of Edges:");
 e= Integer.parseInt(br.readline());
  graph = new int[v][v];
   via = new int[v][v];
    rt = new int[v][v];
    for(int i=0; i<v; i++)
     for(int j=0; j<v; j++)
      {
        if(i==j)
        graph[i][j]=0;
        else
          graph[i][j]=9999;
      }
      for(int i=0; i<e; i++)
      {
  System.out.println("Please enter data for Edge "+(i+1)+":");
  System.out.println("source:");
int s= Integer.parseInt(br.readline());
```

```
s--;
System.out.println("Destination:");
int d = Integer.parseInt(br.readLine());
d--;
System.out.print("cost:");
int c = Integer.parseInt(br.readline());
graph[s][d] = c;
graph[d][s] = c;
}
dvr_calc_disp("The initial Routing Tables are:");
System.out.println("Please enter the source Node for the
                    edge whose cost has changed:");
int s = Integer.parseInt(br.readLine());
s--;
System.out.println("Please enter the Destination Node for the edge
                    whose cost has changed:");
int d = Integer.parseInt(br.readLine());
d--;
System.out.print("Please enter the new cost:");
int c = Integer.parseInt(br.readLine());
graph[s][d] = c;
graph[d][s] = c;
dvr_calc_disp("The new Routing Tables are:");
}
static void dvr_calc_disp(String message)
{
System.out.println();
init_tables();
update_tables();
System.out.println(message);
print_tables();
System.out.println();
}
static void update_table(int source)
{ for(int i=0; i<v; i++)
{ if(graph[source[i]!=9999)
{ int dist=graph[source[i]);
for(int j=0; j<v; j++){
int inter_dist=rt[i][j];
if(via[i][j]==source)
inter_dist=9999;
```

```
if (dist + inter_dist < rt[source][i])
{ rt[source][i] = dist + inter_dist;
  via[source][i] = i;
}}}}
static void update_tables()
{ int k=0;
  for(int i=0; i<4*V; i++)
  { updat_table(k);
    k++;
    if(k==V)
    k=0;
  }}
  static void int_tables()
  { for(int i=0; i<V; i++)
    { for(int j=0; j<V; j++)
      { if(i==j)
        { rt[i][j]=0;
          via[i][j]=i;
        }else{rt[i][j]=9999;
              via[i][j]=100;
        }}}}
static void print_tables()
  { for(int i=0; i<V; i++)
    { for(int j=0; j<V; j++)
      { System.out.print("Dist:"+rt[i][j]+"  ");
      }System.out.println();
    }}}
```

Experiment No: 7

Aim: Network packet analysis using tools like Wireshark, tcpdump etc.

TOOLS:
- Wireshark
- Tcpdump

Theory:

A packet analyzer (also known as packet sniffer) is a computer program or piece of computer hardware that can intercept and log traffic that passes over a digital network or part of a network.

Wireshark:

Wireshark is a free and open-source packet analyzer.
- It is used for network troubleshooting, analysis, software and communications protocol development and education.
- Wireshark is cross platform.
- It runs on Linux, MacOS, BSD, Solaris.

Functionality:
- Wireshark is very similar to tcpdump, but has a graphical front-end, plus some integrated sorting and filtering options.
- Wireshark lets the user put network interface controllers into promiscous mode, so they can see all the traffic visible on that interface.

Features:

Wireshark is a data capturing program that "understands" the structure (encapsulation) of different networking protocols.
- It can parse and display the fields, along with their meanings as specified by different networking protocols.

Tcpdump:

Tcpdump is a common packet analyzer that run under the command line.
- It allows the user to display TCP/IP and other packet being transmitted or recived over a network

to which the computer is attached.

- Distributed under the BSD license, tcpdump is free software
- Tcpdump works on most Unix-like Operating systems: Like Linux, Solaris, FreeBSD.
- In these System, tcpdump uses the libcap library to capture packets.

Functionality:

- Tcpdump prints the concept of network packets. It can be read packets from a network interface card or from a previously created saved packet file.
- Tcpdump can write packets to standard output or file
- It is also possible to use tcpdump for the specific purpose of intercepting and displaying the communicat of another user or computer.
- The User may optionally apply a BSF-based filter to limit the number of packets seen by tcpdump; this renders the output more usable on network with a high volume of traffic.