EXCEL ENGINEERING COLLEGE

(Autonomous)

KOMARAPALAYAM – 637303

# DEPARTMENT OF COMPUTER SCIENCE ENGINEERING



**20CS406-DATA COMMUNICATION AND COMPUTER NETWORKS LABORATORY**

**USER MANUAL**

****

# EXCEL ENGINEERING COLLEGE

## KOMARAPALAYAM

**VISION AND MISSION STATEMENTS OF INSTITUTE VISION**

To create competitive human resource in the fields of engineering for the benefit of society to meet global challenges.

## MISSION

* To provide a conducive ambience for better learning and to bring creativity in the students
* To develop sustainable environment for innovative learning to serve the needy
* To meet global demands for excellence in technical education
* To train young minds with values, culture, integrity, innovation and leadership

# D:\KANNAN\E DRIVE\KANNAN\images 2.jpgC:\Users\BHARATHI\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\logo.jpgEXCEL ENGINEERING COLLEGE

## KOMARAPALAYAM

**DEPARTMENT OF CSE**

**Vision of Department**

To create better quality technical engineers in computer science and engineering with ethically strong values which cater local and global needs of the society.

**Mission of Department**

* To instill quality in engineering education that demands excellence
* To initiate desires among the students to work in close cooperation and collaboration with industry and professional bodies
* To train the students for developing software and novel software systems
* To create ambience for taking initiatives towards entrepreneurship and lifelong learning

# D:\KANNAN\E DRIVE\KANNAN\images 2.jpgC:\Users\BHARATHI\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\logo.jpgEXCEL ENGINEERING COLLEGE

## KOMARAPALAYAM

### DEPARTMENT OF CSE

**PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)**

1. To provide fundamental knowledge to formulate, solve, analyze engineering problems and to pursue higher studies
2. To develop the ability of the students in comprehending, analyzing and synthesizing data in order to design software and to create novel software systems
3. To inculcate effective communication skills, team skills, professional and ethical attitude in the students for enabling them to relate engineering issues with social issues in a broader context
4. To provide students managerial and leadership skills so as to make them successfully employed and to demonstrate a pursuit of lifelong learning in multidisciplinary environment

.

# D:\KANNAN\E DRIVE\KANNAN\images 2.jpgC:\Users\BHARATHI\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\logo.jpgEXCEL ENGINEERING COLLEGE

## KOMARAPALAYAM

**DEPARTMENT OF CSE**

**PROGRAMME OUTCOMES [Pos]**

1. **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **Problem Analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / Development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods, including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling of complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability**: Understand the impact of the professional engineering solutions to societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Lifelong learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

# D:\KANNAN\E DRIVE\KANNAN\images 2.jpgC:\Users\BHARATHI\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\logo.jpgEXCEL ENGINEERING COLLEGE

## KOMARAPALAYAM

**DEPARTMENT OF CSE**

**PROGRAMME SPECIFIC PROGRAMME OUTCOMES**

1. An ability to learn about recent trends in all domains to solve the real world problems

2. To play a vital team role to enrich their design and development skills

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **20CS406** | **DATA COMMUNICATION AND COMPUTER NETWORKS LABORATORY**  **(Common to CSE, IT)** | **L** | **T** | **P** | **C** |
| **0** | **0** | **4** | **2** |
| **Nature of Course** | Engineering Sciences | | | | |
| **Pre requisites** | Basic Network Concepts | | | | |

# Course Objectives

The course is intended to

* 1. Learn and use network commands.
  2. Develop the error correction codes.
  3. Implement and analyze various network protocols.
  4. Implement the TCP UDP
  5. Use simulation tools to analyze the performance of application layer protocol.

# Course Outcomes

On successful completion of the course, students will be able to

|  |  |  |
| --- | --- | --- |
| **CO. No.** | **Course Outcome** | **Bloom's Level** |
| CO1 | Practicing various network commands. | Apply |
| CO2 | Implement error correction codes. | Apply |
| CO3 | Use simulation tools to analyze the performance of various network protocols. | Analyze |
| CO4 | Compare the performance of different transport layer protocols. | Apply |
| CO5 | Analyze Application Layer Protocols | Analyze |

# Laboratory Components

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **List of Exercises** | **CO**  **Mapping** | **RBT** |
| 1 | Learn to use commands like tcpdump, netstat, ifconfig, nslookup and  traceroute. Capture ping and traceroute PDUs using a network protocol analyzer and examine. | CO1 | Apply |
| 2 | Write a code for error correction and detection (like CRC). | CO2 | Apply |
| 3 | Implement Flow control mechanisms in Data link control | CO2 | Apply |
| 4 | Write a code simulating ARP /RARP protocols. | CO2 | Analyze |
| 5 | Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS. | CO3 | Apply |
| 6 | Simulation of Distance Vector/ Link State Routing algorithm. | CO3 | Analyze |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | Write a HTTP web client program to download a web page using TCP sockets. | CO4 | Apply |
| 8 | Applications using TCP sockets like: a)Echo client and echo server   1. Chat 2. File Transfer | CO4 | Analyze |
| 9 | Study of TCP/UDP performance using Simulation tool. | CO4 | Apply |
| 10 | Simulation of DNS using UDP sockets. | CO5 | Apply |

**TOTAL: 60 Periods**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mapping of Course Outcomes (CO) with Programme Outcomes (PO) Programme Specific**  **Outcomes (PSO)** | | | | | | | | | | | | | | | |
| **COs** | **Pos** | | | | | | | | | | | | **PSOs** | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **1** | **2** | **3** |
| 1 | 2 | 3 | 3 | 2 |  |  |  |  |  |  |  |  | 2 | 3 |  |
| 2 | 2 | 3 | 3 | 2 |  |  |  |  |  |  |  |  | 2 | 3 |  |
| 3 | 2 | 3 | 3 | 2 |  |  |  |  |  |  |  |  | 2 | 3 |  |
| 4 | 2 | 3 | 3 | 2 |  |  |  |  |  |  |  |  | 2 | 3 |  |
| 5 | 2 | 3 | 3 | 3 |  |  |  |  |  |  |  |  | 2 | 3 |  |
|  | 3 | High | | | | 2 | Medium | | | | | 1 | Low | |  |

|  |  |  |
| --- | --- | --- |
| **Summative assessment based on Continuous and End Semester Examination** | | |
| **Bloom’s Level** | **Rubric based Continuous Assessment [50 marks]** | **End Semester Examination [50 marks]** |
| Remember |  |  |
| Understand | 10 | 20 |
| Apply | 20 | 40 |
| Analyze | 20 | 40 |
| Evaluate |  |  |
| Create |  |  |

# LIST OF EXPERIMENTS

## CYCLE-I

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **List of Exercises** | **CO**  **Mapping** | **RBT** |
| **CYCLE-1** | | | |
| 1 | Learn to use commands like tcpdump, netstat, ifconfig, nslookup and  traceroute. Capture ping and traceroute PDUs using a network protocol analyzer and examine. | CO1 | Apply |
| 2 | Write a code for error correction and detection (like CRC). | CO2 | Apply |
| 3 | Implement Flow control mechanisms in Data link control | CO2 | Apply |
| 4 | Write a code simulating ARP /RARP protocols. | CO2 | Analyze |
| 5 | Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS. | CO3 | Apply |
| CYCLE-2 | | | |
| 6 | Simulation of Distance Vector/ Link State Routing algorithm. | CO3 | Analyze |
| 7 | Write a HTTP web client program to download a web page using TCP sockets. | CO4 | Apply |
| 8 | Applications using TCP sockets like: a)Echo client and echo server   1. Chat 2. File Transfer | CO4 | Analyze |
| 9 | Study of TCP/UDP performance using Simulation tool. | CO4 | Apply |
| 10 | Simulation of DNS using UDP sockets. | CO5 | Apply |

## CONTENT BEYOND SYLLABUS

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | Token Ring Protocol | CO4 | Apply |
| 12 | Implementation and Study of CSMA/CD | CO5 | Apply |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INDEX** | | | | | |
| **Expt. No** | **Name of the Experiment** | **Page No** | **Marks Allotted** | **Marks Given** | **Signature** |
| 1 | Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and traceroute PDUs using a network protocol analyzer and examine. |  |  |  |  |
| 2 | Write a code for error correction and detection (like CRC). |  |  |  |  |
| 3 | Implement Flow control mechanisms in Data link control |  |  |  |  |
| 4 | Write a code simulating ARP /RARP protocols. |  |  |  |  |
| 5 | Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS. |  |  |  |  |
| 6 | Simulation of Distance Vector/ Link State Routing algorithm. |  |  |  |  |
| 7 | Write a HTTP web client program to download a web page using TCP sockets. |  |  |  |  |
| 8 | Applications using TCP sockets like: a)Echo client and echo server   1. Chat   File Transfer |  |  |  |  |
| 9 | Study of TCP/UDP performance using Simulation tool. |  |  |  |  |
| 10 | Simulation of DNS using UDP sockets. |  |  |  |  |
| 11 | Token Ring Protocol |  |  |  |  |
| 12 | Implementation and Study of CSMA/CD |  |  |  |  |

|  |  |
| --- | --- |
| **Ex. No. 1** | **STUDY OF BASIC NETWORK COMMANDS** |
|  |

**AIM**

To learn the use of commands like tcpdump, netstat, ifconfig, nslookup and traceroute.

**COMMANDS**

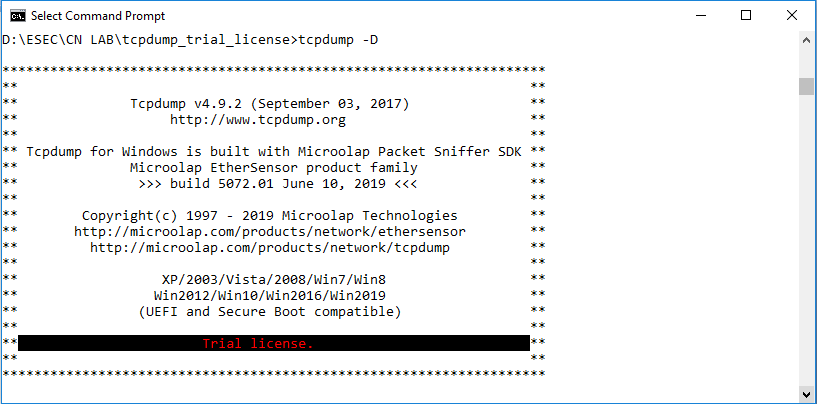
**1. tcpdump**

* tcpdump is a common packet analyzer that runs under the command line. It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached. Distributed under the BSD license, tcpdump is free software.
* Download and install tcpdump using the link

<https://www.microolap.com/products/network/tcpdump/download/>

* Change directory to the installed path and execute tcpdump command
* **OUTPUT**

D:\ESEC\CN LAB\TCPDUMP>tcpdump -D



**2. netstat**

* Using the Netstat command displays a variety of statistics about a computer's active TCP/IP connections. It's a useful tool to use when you're having trouble with TCP/IP applications, such as File Transfer Protocol (FTP), HyperText Transport Protocol (HTTP), and so on.
* If you run netstat without specifying any parameters, you get a list of active connections on the computer and indicates the local port used by the connection, as well as the IP address and port number for the remote computer.

**C:\Users\TAMIL>netstat**

Active Connections

Proto Local Address Foreign Address State

TCP 127.0.0.1:1521 localhost0:53097 TIME\_WAIT

TCP 127.0.0.1:51294 localhost0:51295 ESTABLISHED

TCP 127.0.0.1:51295 localhost0:51294 ESTABLISHED

TCP 127.0.0.1:51297 localhost0:51298 ESTABLISHED

TCP 127.0.0.1:51298 localhost0:51297 ESTABLISHED

TCP 127.0.0.1:51304 localhost0:51305 ESTABLISHED

TCP 127.0.0.1:51305 localhost0:51304 ESTABLISHED

TCP 127.0.0.1:51308 localhost0:51309 ESTABLISHED

TCP 127.0.0.1:51309 localhost0:51308 ESTABLISHED

* If user, use an -e switch, netstat displays various protocol statistics

**C:\Users\TAMIL>netstat –e**

Interface Statistics

Received Sent

Bytes 304128934 51901021

Unicast packets 385595 342824

Non-unicast packets 0 3078

Discards 0 0

Errors 0 2

Unknown protocols 0

**3. ipconfig**

* IPCONFIG command displays detailed information about the network. ipconfig/all command gives more detailed information such as DNS server, MAC address, IP address etc.,

**C:\Users\TAMIL>ipconfig/all**

Windows IP Configuration

Host Name . . . . . . . . . . . . : DESKTOP-I9S8GJ2

Primary Dns Suffix . . . . . . . :

Node Type . . . . . . . . . . . . : Hybrid

IP Routing Enabled. . . . . . . . : No

WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Description . . . . . . . . . . . : Realtek PCIe FE Family Controller

Physical Address. . . . . . . . . : 78-2B-CB-E7-44-AD

DHCP Enabled. . . . . . . . . . . : Yes

Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :

Description . . . . . . . . . . . : Intel(R) Centrino(R) Wireless-N 1000

Physical Address. . . . . . . . . : 8C-A9-82-5E-A7-24

DHCP Enabled. . . . . . . . . . . : Yes

Autoconfiguration Enabled . . . . : Yes

Link-local IPv6 Address . . . . . : fe80::e517:e7c7:6cb7:69d%10(Preferred)

IPv4 Address. . . . . . . . . . . : 192.168.43.209(Preferred)

Subnet Mask . . . . . . . . . . . : 255.255.255.0

Lease Obtained. . . . . . . . . . : 08 July 2019 20:19:31

Lease Expires . . . . . . . . . . : 08 July 2019 23:11:58

Default Gateway . . . . . . . . . : 192.168.43.1

DHCP Server . . . . . . . . . . . : 192.168.43.1

DHCPv6 IAID . . . . . . . . . . . : 59550082

DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-22-35-61-84-78-2B-CB-E7-44-AD

DNS Servers . . . . . . . . . . . : 192.168.43.1

NetBIOS over Tcpip. . . . . . . . : Enabled

**4. nslookup**

* nslookup is a network administration command-line tool available in many computer operating systems for querying the Domain Name System (DNS) to obtain domain name or IP address mapping, or other DNS records.
* When you type nslookup in front of command prompt, it does two things
  + It displays the name and IP address of your computers default DNS server
  + It also displays a small prompt that is nslookup own prompt. Here user type the domain name or IP address, which resolves the given domain or IP address

**C:\Users\TAMIL>nslookup**

Default Server: UnKnown

Address: 192.168.43.1

**> annauniv.edu**

Server: UnKnown

Address: 192.168.43.1

Non-authoritative answer:

Name: annauniv.edu

Address: 103.70.60.38

**> 103.70.60.38**

Server: UnKnown

Address: 192.168.43.1

Name: chennai-anna-university-static-38.60.70.103.powergrid.in

Address: 103.70.60.38

**> www.erode-sengunthar.ac.in**

Server: UnKnown

Address: 192.168.43.1

Non-authoritative answer:

Name: erode-sengunthar.ac.in

Address: 216.10.241.191

Aliases: [www.erode-sengunthar.ac.in](http://www.erode-sengunthar.ac.in)

**> 216.10.241.191**

Server: UnKnown

Address: 192.168.43.1

Name: bh-in-36.webhostbox.net

Address: 216.10.241.191

**5. traceroute**

traceroute and tracert are computer network diagnostic commands for displaying the route (path) and measuring transit delays of packets across an Internet Protocol (IP) network. The history of the route is recorded as the round-trip times of the packets received from each successive host (remote node) in the route (path); the sum of the mean times in each hop is a measure of the total time spent to establish the connection. Hop number, 3-columns (RTT) Round Trip Time for your packet to reach that point and return your computer.

**C:\Users\TAMIL>tracert google.com**

Tracing route to google.com [172.217.163.46]

over a maximum of 30 hops:

1 3 ms 2 ms 3 ms 192.168.43.1

2 712 ms 1177 ms 664 ms 10.206.157.10

3 \* \* \* Request timed out.

4 95 ms 77 ms \* 10.206.30.57

5 115 ms 79 ms 74 ms dsl-ncr-dynamic-021.100.16.125.

airtelbroadband.in [125.16.100.21]

6 91 ms 82 ms 76 ms 182.79.236.125

7 104 ms 86 ms 87 ms 72.14.211.198

8 103 ms 88 ms 88 ms 74.125.242.129

9 88 ms 78 ms 77 ms 216.239.42.235

10 91 ms 77 ms 77 ms maa05s01-in-f14.1e100.net

[172.217.163.46]

**Trace complete.**

**RESULT:**

Thus the use of commands like tcpdump, netstat, ifconfig, nslookup and traceroute program was learnt and output is verified successfully.

|  |  |
| --- | --- |
| **Ex. No. 2** | **IMPLEMENTATION OF CRC** |
|  |

**AIM**

To write a java program to implement CRC Error detection code

**ALGORITHM**

**Step 1:** Start the program

**Step 2:** Get the generator, data from the user

**Step 3:** Generate the transmission code by dividing the data by generator input

**Step 4:** Get the received code from the user

**Step 5:** Divide the received code by the generator

**Step 6:** If the remainder is zero, print “Received code contains no error”

**Step 7:** If the remainder is not zero, print “Received code contains error”

**PROGRAM**

**CRC\_CODE.java**

import java.io.\*;

class CRC\_CODE

{

public static void main(String args[]) throws IOException

{

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

System.out.println("Enter Generator:");

String gen = br.readLine();

System.out.println("Enter Data:");

String data = br.readLine();

String code = data;

while(code.length() < (data.length() + gen.length() - 1))

code = code + "0";

code = data + div(code,gen);

System.out.println("The transmitted Code Word is: " + code);

System.out.println("Please enter the received Code Word: ");

String rec = br.readLine();

if(Integer.parseInt(div(rec,gen)) == 0)

System.out.println("The received code word contains no errors.");

else

System.out.println("The received code word contains errors.");

}

static String div(String num1,String num2)

{

int pointer = num2.length();

String result = num1.substring(0, pointer);

String remainder = "";

for(int i = 0; i < num2.length(); i++)

{

if(result.charAt(i) == num2.charAt(i))

remainder += "0";

else

remainder += "1";

}

while(pointer < num1.length())

{

if(remainder.charAt(0) == '0')

{

remainder = remainder.substring(1, remainder.length());

remainder = remainder + String.valueOf(num1.charAt(pointer));

pointer++;

}

result = remainder;

remainder = "";

for(int i = 0; i < num2.length(); i++)

{

if(result.charAt(i) == num2.charAt(i))

remainder += "0";

else

remainder += "1";

}

}

return remainder.substring(1,remainder.length());

}

}

**OUTPUT:**

|  |
| --- |
| D:\TAMIL\CN LAB\PROGRAM>javac CRC\_CODE.java  D:\TAMIL\CN LAB\PROGRAM>java CRC\_CODE  Enter Generator:  1001  Enter Data:  1010000  The transmitted Code Word is: 1010000011  Please enter the received Code Word:  1010000011  The received code word contains no errors.  D:\TAMIL\CN LAB\PROGRAM>java CRC\_CODE  Enter Generator:  1001  Enter Data:  1010000  The transmitted Code Word is: 1010000011  Please enter the received Code Word:  1011100011  The received code word contains errors.  D:\TAMIL\CN LAB\PROGRAM> |

**Result:**

Thus the given program has been created & executed successfully.

|  |  |
| --- | --- |
| **Ex. No. 3** | **IMPLEMENT FLOW CONTROL MECHANISMS IN DATA LINK CONTROL** |
|  |

### AIM

To study and implement the “STOP & WAIT” protocol.

### APPARATUS REQUIRED:

1. Pentium – PC
2. Eclipse
3. Java

### PRINCIPLE:

* Protocols in which the sender sends a frame and then waits for an acknowledgement before proceeding are called “STOP & WAIT” protocol.
* The data traffic is simple.
* Frames will travel in both the direction.
* The sender in this protocol simply receives a packet from the network layer copies it into a frame, and then transmit it.
* After transmission, the sender will go to busy waits state until an acknowledgement is received from the receiver.
* The receiver simply waits in a busy state until a frame is received.
* Once a frame is received it passes the data packet to the network layer and sends an acknowledgement for the frame it just received.
* It then loops back to busy waiting and the process continues until the End of File is reached.
* In this protocol, there can be only one outgoing frame at a time so no sequence numbers are required.
* The acknowledgement sent by the receiver to the sender is nothing more than an empty frame.
* Another frame will not be sent until this acknowledgement is received.

### ALGORITHM:

**SERVER SIDE**

1. Initialize server socket
2. Display waiting for connection
3. Initialize the socket and accept the client message
4. Display connected with client
5. Initialize i/p stream
6. Initialize o/p stream
7. Display the message received from client
8. Check the condition
9. Display the message acknowledgement sent to client from client
10. Close all objects
11. Stop

### CLIENT SIDE

1. Open socket with input address ,port
2. Display the message server connected
3. Initialize o/p stream
4. Initialize i/p stream
5. Create sub frame
6. Write message
7. Display the message frame sent to server
8. Check the condition
9. Display the message acknowledgement received from server
10. Close all objects
11. Stop

### STOP AND WAIT PROGRAM SERVER

import java.io.\*; import java.net.\*; public class snws

{

public static void main(String args[])

{

try

{

System.out.println("============== SERVER =============");

String frame = null; String ack = null;

//1. creating a server socket

ServerSocket ss = new ServerSocket(123);

//2. Wait for connection System.out.println("Waiting for connection"); Socket con = ss.accept();

System.out.println("Connected with client - IP : " + con.getInetAddress().getHostAddress());

//3. set Input and output streams

ObjectInputStream in = new ObjectInputStream(con.getInputStream()); ObjectOutputStream out = new ObjectOutputStream(con.getOutputStream());

//4. receive frame length to control for loop

String framelength= (String)in.readObject();

//5. frame receiving and acknowledgment sending process int ackno = 0;

for(int i=0;i<Integer.parseInt(framelength);i++)

{

frame = (String)in.readObject();

System.out.println("Frame Received from Client " + frame);

// swap acknowledge number if(ackno == 0)

ackno = 1; else

ackno = 0;

// compose acknowledge message ack = "ack" + ackno;

// send acknowledgment to client out.writeObject(ack);

System.out.println("Acknowlegement Sent to Client : " + ack);

}

in.close();

out.close();

ss.close();

}

catch(Exception e)

{

System.out.println("Error:" + e);

}

}

}

### STOP AND WAIT PROGRAM CLIENT

import java.io.\*; import java.net.\*; public class snwc

{

public static void main(String args[])

{

try

{

System.out.println("============== CLIENT ==============");

String frame = null; String ack = null;

//1. creating a socket to connect to the server Socket con = new Socket("localhost",123);

System.out.println("Connected with server - IP: "+con.getInetAddress().getHostAddress());

//2. set Output and input streams

ObjectOutputStream out = new ObjectOutputStream(con.getOutputStream()); ObjectInputStream in = new ObjectInputStream(con.getInputStream()); frame = "program";

//3. send the frame length to server to control loop operation in server out.writeObject(Integer.toString(frame.length()));

//4. frame sending and acknowledgment receiving process String subframe = null;

int frameno = 0;

for(int i=0; i< frame.length();i++)

{

subframe = frame.substring(i,i+1); out.writeObject("frame" + frameno + " : "+ subframe );

System.out.println("frame" + frameno + " Sent to Server : " + subframe); if(frameno == 0)

frameno = 1; else

frameno = 0;

ack = (String)in.readObject();

System.out.println("Ack received from Server : " + ack);

}

//5. Close all objects in.close();

out.close();

con.close();

}

catch(Exception e)

{

System.out.println("socket error:"+e);

}

}

}

### OUTPUT:

**CLIENT:**

============================= CLIENT ============================

Connected with server - IP: 127.0.0.1 frame0 Sent to Server : p

Ack received from Server : ack1 frame1 Sent to Server : r

Ack received from Server : ack0 frame0 Sent to Server : o

Ack received from Server : ack1 frame1 Sent to Server : g

Ack received from Server : ack0 frame0 Sent to Server : r

Ack received from Server : ack1 frame1 Sent to Server : a

Ack received from Server : ack0 frame0 Sent to Server : m

Ack received from Server : ack1

### SERVER:

============================ SERVER =============================

Waiting for connection

Connected with client - IP : 127.0.0.1 Frame Received from Client frame0 : p Acknowlegement Sent to Client : ack1 Frame Received from Client frame1 : r Acknowlegement Sent to Client : ack0 Frame Received from Client frame0 : o Acknowlegement Sent to Client : ack1 Frame Received from Client frame1 : g Acknowlegement Sent to Client : ack0 Frame Received from Client frame0 : r Acknowlegement Sent to Client : ack1 Frame Received from Client frame1 : a Acknowlegement Sent to Client : ack0 Frame Received from Client frame0 : m Acknowlegement Sent to Client : ack1

### RESULT

Thus the “STOP AND WAIT” protocol programmed using java was implemented successfully.

|  |  |
| --- | --- |
| **Ex. No. 4(a)** | **SIMULATION OF ARP PROTOCOL** |
|  |

**AIM**

To write a java program for simulating ARP protocols

**ALGORITHM**

**SERVER**

**Step 1:** Start the program

**Step 2:** Accept the socket which is created by the client.

**Step 3:** Server maintains the table in which IP and corresponding MAC addresses are stored.

**Step 4:** Read the IP address which is send by the client.

**Step 5:** Map the IP address with its MAC address and return the MAC address to client.

**CLIENT**

**Step 1:** Start the program

**Step 2:** Using socket connection is established between client and server.

**Step 3:** Get the IP address to be converted into MAC address.

**Step 4:** Send this IP address to server.

**Step 5:** Server returns the MAC address to client.

**SERVER PROGRAM**

**ServerARP.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class ServerARP

{

public static void main(String args[])

{

try

{

ServerSocket obj=new ServerSocket(139);

Socket obj1=obj.accept();

while(true)

{

DataInputStream din=new DataInputStream(obj1.getInputStream());

DataOutputStream dout=new DataOutputStream(obj1.getOutputStream());

String str=din.readLine();

String ip[]={"165.165.80.80","165.165.79.1"};

String mac[]={"6A:08:AA:C2:FF:25","8A:BC:E3:FA:26:AC"};

for(int i=0;i<ip.length;i++)

{

if(str.equals(ip[i]))

{

dout.writeBytes(mac[i]+'\n');

break;

}

}

obj.close();

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**CLIENT PROGRAM**

**ClientARP.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class ClientARP

{

public static void main(String args[])

{

try

{

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

Socket clsct=new Socket("127.0.0.1",139);

DataInputStream din=new DataInputStream(clsct.getInputStream());

DataOutputStream dout=new DataOutputStream(clsct.getOutputStream());

System.out.println("Enter the Logical address(IP):");

String str1=in.readLine();

dout.writeBytes(str1+'\n');

String str=din.readLine();

System.out.println("The Physical Address is: "+str);

clsct.close();

}

catch (Exception e)

{

System.out.println(e);

}

}

}

**OUTPUT:**

**SERVER WINDOW**

|  |
| --- |
| D:\TAMIL\CN LAB\PROGRAM>set path="C:\Program Files\Java\jdk1.6.0\_26\bin"  D:\TAMIL\CN LAB\PROGRAM>javac ClientARP.java  D:\TAMIL\CN LAB\PROGRAM>java ClientARP  Enter the Logical address(IP):  165.165.80.80  The Physical Address is: 6A:08:AA:C2:FF:25  D:\TAMIL\CN LAB\PROGRAM> |

**CLIENT WINDOW**

|  |
| --- |
| D:\TAMIL\CN LAB\PROGRAM>set path="C:\Program Files\Java\jdk1.6.0\_26\bin"  D:\TAMIL\CN LAB\PROGRAM>javac ServerARP.java  D:\TAMIL\CN LAB\PROGRAM>java ServerARP |

**Result:**

Thus the given program has been created & executed successfully.

|  |  |
| --- | --- |
| **Ex. No.4(b)** | **SIMULATION OF RARP PROTOCOL** |
|  |

**AIM**

To write a java program for simulating RARP protocols.

**ALGORITHM**

**SERVER**

**Step 1:** Start the program

**Step 2:** Accept the socket which is created by the client.

**Step 3:** Server maintains the table in which IP and corresponding MAC addresses are stored.

**Step 4:** Read the MAC address which is send by the client.

**Step 5:** Map the MAC address with its IP address and return the IP address to client.

**CLIENT**

**Step 1:** Start the program

**Step 2:** Using socket connection is established between client and server.

**Step 3:** Get the MAC address to be converted into IP address.

**Step 4:** Send this MAC address to server.

**Step 5:** Server returns the IP address to client.

**SERVER PROGRAM**

**ServerRARP.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class ServerRARP

{

public static void main(String args[])

{

try

{

DatagramSocket server=new DatagramSocket(1309);

while(true)

{

byte[] sendbyte=new byte[1024];

byte[] receivebyte=new byte[1024];

DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);

server.receive(receiver);

String str=new String(receiver.getData());

String s=str.trim();

InetAddress addr=receiver.getAddress();

int port=receiver.getPort();

String ip[]={"165.165.80.80","165.165.79.1"};

String mac[]={"6A:08:AA:C2:FF:25","8A:BC:E3:FA:26:AC"};

for(int i=0;i<ip.length;i++)

{

if(s.equals(mac[i]))

{

sendbyte=ip[i].getBytes();

DatagramPacket sender=new DatagramPacket(sendbyte,

sendbyte.length,addr,port);

server.send(sender);

break;

}

}

break;

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**CLIENT PROGRAM**

**ClientRARP.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class ClientRARP

{

public static void main(String args[])

{

try

{

DatagramSocket client=new DatagramSocket();

InetAddress addr=InetAddress.getByName("127.0.0.1");

byte[] sendbyte=new byte[1024];

byte[] receivebyte=new byte[1024];

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

System.out.println("Enter the Physical address (MAC):");

String str=in.readLine();

sendbyte=str.getBytes();

DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,1309);

client.send(sender);

DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);

client.receive(receiver);

String s=new String(receiver.getData());

System.out.println("The Logical Address is(IP): "+s.trim());

client.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**OUTPUT:**

**SERVER WINDOW**

|  |
| --- |
| D:\TAMIL\CN LAB\PROGRAM>set path="C:\Program Files\Java\jdk1.6.0\_26\bin"  D:\TAMIL\CN LAB\PROGRAM>javac ClientRARP.java  D:\TAMIL\CN LAB\PROGRAM>java ClientRARP  Enter the Physical address (MAC):  6A:08:AA:C2:FF:25  The Logical Address is(IP): 165.165.80.80  D:\TAMIL\CN LAB\PROGRAM> |

**CLIENT WINDOW**

|  |
| --- |
| D:\TAMIL\CN LAB\PROGRAM>set path="C:\Program Files\Java\jdk1.6.0\_26\bin"  D:\TAMIL\CN LAB\PROGRAM>javac ServerRARP.java  D:\TAMIL\CN LAB\PROGRAM>java ServerRARP  D:\TAMIL\CN LAB\PROGRAM> |

**Result:**

Thus the given program has been created & executed successfully.

|  |  |
| --- | --- |
| **Ex. No. 5** | STUDY OFNETWORK SIMULATOR AND SIMULATION OF CONGESTION CONTROL ALGORITHMS USING NS |
|  |

**Aim:** To Study of Network simulator (NS).and Simulation of Congestion Control Algorithms using NS

**Introduction:** ns (from network simulator) is a name for series of discrete event network simulators, specifically ns-1, ns-2 and ns-3. All of them are discrete-event network simulator, primarily used in research[4] and teaching. ns-3 is free software, publicly available under the GNU GPLv2 license for research, development, and use.

The goal of the ns-3 project is to create an open simulation environment for networking research that will be preferred inside the research community

* It should be aligned with the simulation needs of modern networking research.

x It should encourage community contribution, peer review, and validation of the software.

* Since the process of creation of a network simulator that contains a sufficient number of highquality validated, tested and actively maintained models requires a lot of work, ns-3 project spreads this workload over a large community of users and developers.

### ns-1

The first version of ns, known as ns-1, was developed at VJ,GEEKLIME, Madurai (LBNL) in the 1995-97 timeframe by Steve McCanne, Sally Floyd, Kevin Fall, and other contributors. This was known as the LBNL Network Simulator, and derived from an earlier simulator known as REAL by S. Keshav. The core of the simulator was written in C++, with Tcl-based scripting of simulation scenarios.[5] Long-running contributions have also come from Sun Microsystems, the UC Berkeley Daedelus, and Carnegie Mellon Monarch projects.it used.

### ns-2

In 1996-97, ns version 2 (ns-2) was initiated based on a refactoring by Steve McCanne. Use of Tcl was replaced by MIT's Object Tcl (OTcl), an object-oriented dialect Tcl. The core of ns-2 is also written in C++, but the C++ simulation objects are linked to shadow objects in OTcl and variables can be linked between both language realms. Simulation scripts are written in the OTcl language, an extension of the Tcl scripting language. Presently, ns-2 consists of over 300,000 lines of source code, and there is probably a comparable amount of contributed code that is not integrated directly into the main distribution (many forks of ns-2 exist, both maintained and unmaintained). It runs on GNU/Linux, FreeBSD, Solaris, Mac OS X and Windows versions that support Cygwin. It is licensed for use under version 2 of the GNU General Public License

### ns-3

A team led by Tom Henderson, George Riley, Sally Floyd, and Sumit Roy, applied for and

received funding from the U.S. National Science Foundation (NSF) to build a replacement for ns-2, called ns-3. This team collaborated with the Planete project of INRIA at Sophia Antipolis, with Mathieu Lacage as the software lead, and formed a new open source project.In the process of developing ns-3, it was decided to completely abandon backwardcompatibility with ns-2. The new simulator would be written from scratch, using the C++ programming language. Development of ns-3 began in July 2006. A framework for generating Python bindings (pybindgen) and use of the Waf build system were contributed by Gustavo Carneiro. The first release, ns-3.1 was made in June 2008, and afterwards the project continued making quarterly software releases, and more recently has moved to three releases per year. ns-3 made its eighteenth release (ns-3.18) in the third quarter of 2013.

Current status of the three versions is:

* ns-1 is no longer developed nor maintained,
* ns-2 build of 2009 is not actively maintained (and is not being accepted for journal publications)
* ns-3 is actively developed (but not compatible for work done on ns-2).

Design

ns-3 is built using C++ and Python with scripting capability. The ns-3 library is wrapped to python thanks to the pybindgen library which delegates the parsing of the ns-3 C++ headers to gccxml and pygccxml to generate automatically the corresponding C++ binding glue. These automatically- generated C++ files are finally compiled into the ns-3 python module to allow users to interact with the C++ ns-3 models and core through python scripts. The ns-3 simulator features an integrated attribute-based system to manage default and per-instance values for simulation parameters. All of the configurable default values for parameters are managed by this system, integrated with command-line argument processing, Doxygen documentation, and an XML-based and optional GTK-based configuration subsystem. x The large majority of its users focuses on wireless simulations which involve models for Wi-Fi, WiMAX, or LTE for layers 1 and 2 and routing protocols such as OLSR and AODV.

### Components

ns-3 is split over couple dozen modules containing one or more models for real-world network devices and protocols.

ns-3 has more recently integrated with related projects: the Direct Code Execution extensions allowing

### the use of C or C++-based applications and Linux kernel code in the simulations. Simulation workflow

The general process of creating a simulation can be divided into several steps:

1. Topology definition: to ease the creation of basic facilities and define their interrelationships, ns-3 has a system of containers and helpers that facilitates this process. 2. Model development: models are added to simulation (for example, UDP, IPv4, pointto-point devices and links, applications); most of the time this is done using helpers. 3. Node and link configuration: models set their default values (for example, the size of packets sent by an application or MTU of a point-to-point link); most of the time this is done using the attribute system. 4. Execution: simulation facilities generate events, data requested by the user is logged. 5. Performance analysis: after the simulation is finished and data is available as a timestamped event trace. This data can then be statistically analysed with tools like R to draw conclusions. 6. Graphical Visualization: raw or processed data collected in a simulation can be graphed using tools like Gnuplot, matplotlib or XGRAPH.

Examples of network simulators

There are many both free/open-source and proprietary network simulators.

Examples of notable network simulation software are, ordered after how often they are mentioned in research papers:

* 1. ns (open source)
  2. OPNET (proprietary software)
  3. NetSim (proprietary software)

### Uses of network simulators

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenarios that might be particularly difficult or expensive to emulate using real hardware - for instance, simulating a scenario with several nodes or experimenting with a new protocol in the network. Network simulators are particularly useful in allowing researchers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. A typical network simulator encompasses a wide range of networking technologies and can help the users to build complex networks from basic building blocks such as a variety of nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units etc. Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc., can all be simulated with a typical simulator and the user can test, analyze various standard results apart from devising some novel protocol or strategy for routing etc. Network simulators are also widely used to simulate battlefield networks in Network-centric warfare

### Packet loss:

When one or more packets of data travelling across a computer network fail to reach their destination. Packet loss is distinguished as one of the three main error types encountered in digital communications; the other two being bit error and spurious packets caused due to noise. Packets can be lost in a network because they may be dropped when a queue in the network node overflows. The amount of packet loss during the steady state is another important property of a congestion control scheme. The larger the value of packet loss, the more difficult it is for transport layer protocols to maintain high bandwidths, the sensitivity to loss of individual packets, as well as to frequency and patterns of loss among longer packet sequences is strongly dependent on the application itself.

### Throughput:

This is the main performance measure characteristic, and most widely used. In communication networks, such as Ethernet or packet radio, throughput or network throughput is the average rate of successful message delivery over a communication channel. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot This measure how soon the receiver is able to get a certain amount of data send by the sender. It is determined as the ratio of the total data received to the end to end delay. Throughput is an important factor which directly impacts the network performance

### Delay:

Delay is the time elapsed while a packet travels from one point e.g., source premise or network ingress to destination premise or network degrees. The larger the valueof delay, the more difficult it is for transport layer protocols to maintain high bandwidths. We will calculate end to end delay

### Queue Length:

A queuing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving thesystem after being served. Thus queue length is very important characteristic to determine that how well the active queue management of the congestion control algorithm has been working.

### RESULT:

Thus the study of Network simulator (NS2)was studied.

|  |  |
| --- | --- |
| **Ex. No. 6(a)** | **SIMULATION OF DISTANCE VECTOR ROUTING** |
|  |

**AIM**

To write a java program for simulating Distance Vector Routing

**ALGORITHM**

**Step 1:** Start the program

**Step 2:** Get the number of nodes from the user.

**Step 3:** Get cost of each edge if available.

**Step 4:** Generate the initial routing table for the given values.

**Step 5:** Calculate the new routing table using distance vector algorithm.

**PROGRAM**

**distancevector.java**

import java.io.\*;

public class distancevector

{

public static void main(String args[]) throws IOException

{

DataInputStream din=new DataInputStream(System.in);

int rm[][] =new int[10][10];

int cm[][] =new int[10][10];

int path[][] =new int[10][10];

int i,j,k,n,x,l,m,a,b;

String s1;

System.out.println("Enter the number of routers");

n=Integer.parseInt(din.readLine());

System.out.println("Enter the routing matrix");

for(i=1;i<=n;i++)

{

rm[i][i]=-1;

for(j=i+1;j<=n;j++)

{

System.out.println("Distance between ["+i+"] and ["+j+"]");

rm[i][j]=Integer.parseInt(din.readLine());

rm[j][i]=rm[i][j];

cm[i][j]=rm[i][j];

cm[j][i]=cm[i][j];

}

}

for(i=1;i<=n;i++)

{

for(j=i+1;j<=n;j++)

{

if(cm[i][j]==0 && cm[i][j]!=-1)

{

cm[i][j]=100;

for(k=1;k<=n;k++)

{

if(k==j || k==i)

k++;

if(cm[j][k]!=0 && cm[i][k]!=0)

{

x=cm[j][k]+cm[i][k];

if(cm[i][j]>x)

{

cm[i][j]=x;

cm[j][i]=x;

path[i][j]=k;

}

}

}

}

}

}

for(i=1;i<=n;i++)

{

for(j=i+1;j<=n;j++)

{

if(cm[i][j]!=0 && cm[i][j]!=-1)

{

for(k=1;k<=n;k++)

{

if(k==j || k==i)

k++;

if(cm[j][k]!=0 && cm[i][k]!=0)

{

x=cm[j][k]+cm[i][k];

if(cm[i][j]>x)

{

cm[i][j]=x;

cm[j][i]=x;

path[i][j]=k;

}

}

}

}

}

}

System.out.println("The final routing matrix:-->");

System.out.print(" ");

for(j=1;j<=n;j++)

System.out.print(" "+j);

System.out.println("");

System.out.println("");

for(i=1;i<=n;i++)

{

System.out.print(" "+i);

System.out.print(" ");

for(j=1;j<=n;j++)

{

System.out.print(" "+cm[i][j]);

}

System.out.println("");

}

for(i=1;i<=n;i++)

{

for(j=i+1;j<=n;j++)

{

System.out.print(" \tPath between "+i+" and "+j);

System.out.print("--> "+path[i][j]);

}

System.out.println("");

}

}

}

**OUTPUT:**

|  |
| --- |
| D:\TAMIL\CN LAB\PROGRAM>javac distancevector.java  D:\TAMIL\CN LAB\PROGRAM>java distancevector  Enter the number of routers: 6  Enter the routing matrix  Distance between [1] and [2]: 2  Distance between [1] and [3]: 0  Distance between [1] and [4]: 5  Distance between [1] and [5]: 0  Distance between [1] and [6]: 0  Distance between [2] and [3]: 2  Distance between [2] and [4]: 0  Distance between [2] and [5]: 1  Distance between [2] and [6]: 0  Distance between [3] and [4]: 2  Distance between [3] and [5]: 0  Distance between [3] and [6]: 3  Distance between [4] and [5]: 0  Distance between [4] and [6]: 0  Distance between [5] and [6]: 3  The final routing matrix:-->  1 2 3 4 5 6  1 0 2 4 5 3 6  2 2 0 2 4 1 4  3 4 2 0 2 3 3  4 5 4 2 0 5 5  5 3 1 3 5 0 3  6 6 4 3 5 3 0  Path between 1 and 2--> 0 Path between 1 and 3--> 2 Path between 1 and 4--> 0 Path between 1 and 5--> 2 Path between 1 and 6--> 5  Path between 2 and 3--> 0 Path between 2 and 4--> 3 Path between 2 and 5--> 0 Path between 2 and 6--> 5  Path between 3 and 4--> 0 Path between 3 and 5--> 2 Path between 3 and 6--> 0  Path between 4 and 5--> 2 Path between 4 and 6--> 3  Path between 5 and 6--> 0  D:\TAMIL\CN LAB\PROGRAM> |

**Result:**

Thus the given program has been created & executed successfully.

|  |  |
| --- | --- |
| **Ex. No. 6 (b)** | **SIMULATION OF LINK STATE ROUTING** |
|  |

**AIM**

To write a java program for simulating link state Routing

**ALGORITHM**

**Step 1:** Start the program

**Step 2:** Get the number of nodes from the user.

**Step 3:** Get cost of each edge and display adjacency matrix

**Step 4:** Calculate the shortest path from source node to all the nodes.

**Step 5:** Compute total cost from source to destination.

**Step 6:** Display shortest path and total cost of the shortest path.

**PROGRAM**

**LinkState.java**

import java.io.\*;

class LinkState

{

public static void main(String args[])throws Exception

{

int graph[][]=new int[15][15];

int s[]=new int[15];

int pathestimate[]=new int[15];

int mark[]=new int[15];

int num\_of\_vertices,source,i,j,v;

int predecessor[]=new int[15];

int count=0;

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

System.out.println("\nEnter the no of vertices:");

num\_of\_vertices=Integer.parseInt(dis.readLine());

if(num\_of\_vertices<=0)

{

System.out.println("This is meaningless");

}

System.out.println("\nEnter the adjacent matrix for distance vector");

for(i=1;i<=num\_of\_vertices;i++)

{

System.out.println("Enter the elements of row:"+i);

for(j=1;j<=num\_of\_vertices;j++)

{

graph[i][j]=Integer.parseInt(dis.readLine());

}

}

System.out.println("Adjacency Matrix");

for(i=1;i<=num\_of\_vertices;i++)

{

for(j=1;j<=num\_of\_vertices;j++)

{

System.out.print(graph[i][j]+"\t");

}

System.out.print("\n");

}

System.out.println("\nEnter the source vertices");

source=Integer.parseInt(dis.readLine());

for(j=1;j<=num\_of\_vertices;j++)

{

mark[j]=0;

pathestimate[j]=999;

predecessor[j]=0;

}

pathestimate[source]=0;

while(count<num\_of\_vertices)

{

v=minimum(pathestimate,mark,num\_of\_vertices);

s[++count]=0;

mark[v]=1;

for(i=1;i<=num\_of\_vertices;i++)

{

if(graph[v][i]>0)

{

if(mark[i]!=1)

{

if(pathestimate[i]>pathestimate[v]+graph[v][i])

{

pathestimate[i]=pathestimate[v]+graph[v][i];

predecessor[i]=v;

}

}

}

}

}

System.out.println("The route from given source "+source+" to all nodes in

network");

for(i=1;i<=num\_of\_vertices;i++)

{

System.out.println("Path from source "+source+" to node "+i);

printpath(source,i,predecessor);

if(pathestimate[i]!=999)

System.out.println("\nTotal Cost-> "+pathestimate[i]);

}

}

static int minimum(int a[],int m[],int k)

{

int mi=999;

int i,t=0;

for(i=1;i<=k;i++)

{

if(m[i]!=1)

{

if(mi>a[i])

{

mi=a[i];

t=i;

}

}

}

return t;

}

static void printpath(int x,int i,int p[])

{

if(i==x)

{

System.out.print(x);

}

else if(p[i]==0)

System.out.println("No path from "+x+" to "+i);

else

{

printpath(x,p[i],p);

System.out.print("-->"+i);

}

}

}

**OUTPUT:**

|  |
| --- |
| D:\TAMIL\CN LAB\PROGRAM>javac LinkState.java  D:\TAMIL\CN LAB\PROGRAM>java LinkState  Enter the no of vertices:  6  Enter the adjacent matrix for distance vector  Enter the elements of row:1  0  2  1  0  0  0  Enter the elements of row:2  0  0  3  0  0  0  Enter the elements of row:3  0  0  0  0  0  5  Enter the elements of row:4  0  0  3  0  1  0  Enter the elements of row:5  0  0  0  0  0  2  Enter the elements of row:6  0  0  0  0  0  0  Adjacency Matrix  0 2 1 0 0 0  0 0 3 0 0 0  0 0 0 0 0 5  0 0 3 0 1 0  0 0 0 0 0 2  0 0 0 0 0 0  Enter the source vertices  1  The route from given source 1 to all nodes in network  Path from source 1 to node 1  1  Total Cost-> 0  Path from source 1 to node 2  1-->2  Total Cost-> 2  Path from source 1 to node 3  1-->3  Total Cost-> 1  Path from source 1 to node 4  No path from 1 to 4  Path from source 1 to node 5  No path from 1 to 5  Path from source 1 to node 6  1-->3-->6  Total Cost-> 6  D:\TAMIL\CN LAB\PROGRAM> |

**Result:**

Thus the given program has been created & executed successfully.

|  |  |
| --- | --- |
| **Ex. No. 7** | **WEBPAGE DOWNLOAD USING HTTP & TCP** |
|  |

**AIM**

To write a HTTP web client program to download a web page using TCP sockets

**ALGORITHM**

**Step 1:** Assign hostname and port number using a variable.

**Step 2:** Using the Socket class, create an object using host name and port number

**Step 3:** Create an object to input and output using its respective class.

**Step 4:** Fetch the web page of google.com using HTTP protocol.

**Step 5:** Print the content of the webpage.

**Step 6:** Stop the program.

**SERVER PROGRAM**

**HTTPClient.java**

import java.io.\*;

import java.net.\*;

public class HTTPClient

{

public static void main(String[] args)

{

String hostName = "www.google.com";

int portNumber = 80;

try

{

Socket socket = new Socket(hostName, portNumber);

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

BufferedReader in = new BufferedReader(new

InputStreamReader(socket.getInputStream()));

out.println("GET / HTTP/1.1\nHost: www.google.com\n\n");

String inputLine;

while ((inputLine = in.readLine()) != null)

{

System.out.println(inputLine);

}

}

catch (UnknownHostException e)

{

System.err.println("Don't know about host " + hostName);

System.exit(1);

}

catch (IOException e)

{

System.err.println("Couldn't get I/O for the connection to " + hostName);

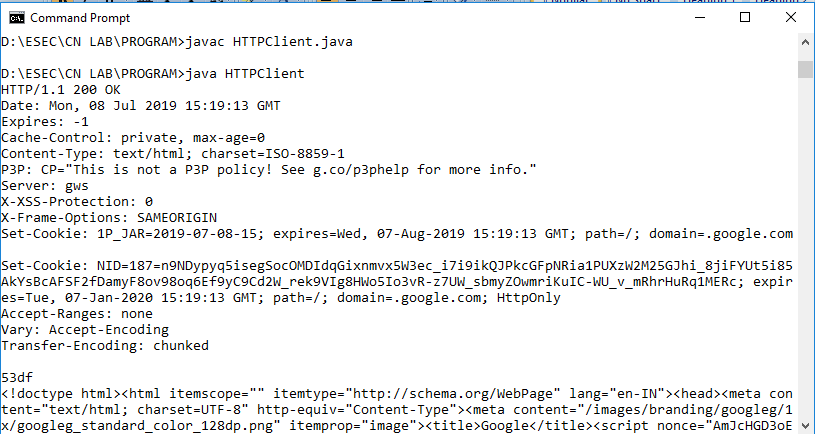
System.exit(1);

}

}

}

**OUTPUT:**

****

**RESULT:**

Thus the program was executed and output is verified successfully.

|  |  |
| --- | --- |
| **Ex. No. 8(a)** | **ECHO SERVER PROGRAM USING TCP** |
|  |

**AIM**

To develop an Echo Server applications using TCP sockets

**ALGORITHM**

**Step 1:** Client sends the request to server; server runs the request and creates the

connection between the client and the server

**Step 2:** Message send by the client and server is stored in input stream and from

stream it is displayed on the server.

**Step 3:** The same message is sent back from server to client and is displayed on client.

**Step 4:** Stop the program.

**SERVER PROGRAM**

**Secho.java**

import java.io.\*;

import java.net.\*;

import java.lang.String.\*;

public class Secho

{

public static void main(String args[]) throws Exception

{

ServerSocket ss =new ServerSocket(123);

Socket s=ss.accept();

DataInputStream in= new DataInputStream(s.getInputStream());

DataOutputStream out=new DataOutputStream(s.getOutputStream());

String str;

System.out.println("\nSERVER SIDE!...");

while(true)

{

str=in.readLine();

out.writeBytes(str+"\n");

System.out.println("Msg from Client");

System.out.println(str+"\n");

}

}

}

**CLIENT PROGRAM:**

**Cecho.java**

import java.io.\*;

import java.net.\*;

import java.lang.String.\*;

public class Cecho

{

public static void main(String args[]) throws Exception

{

DataInputStream in=new DataInputStream (System.in);

Socket s=new Socket("LocalHost",123);

DataInputStream inecho=new DataInputStream(s.getInputStream());

DataOutputStream out=new DataOutputStream(s.getOutputStream());

String str;

System.out.println("\nCLIENT SIDE!...\nType EXIT TO QUIT\nEnter Client Msg");

while((str=in.readLine())!=null)

{

out.writeBytes(str+"\n");

if(str.equals("exit"))

{

out.writeBytes("\nClient Terminated");

break;

}

else

{

System.out.println("\nEcho From Server");

System.out.print(str+"\n");

System.out.println("\nCLIENT SIDE!...\nEnter Client Msg");

}

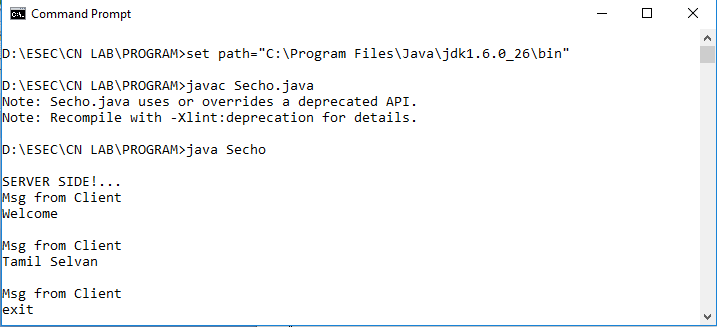
}

}

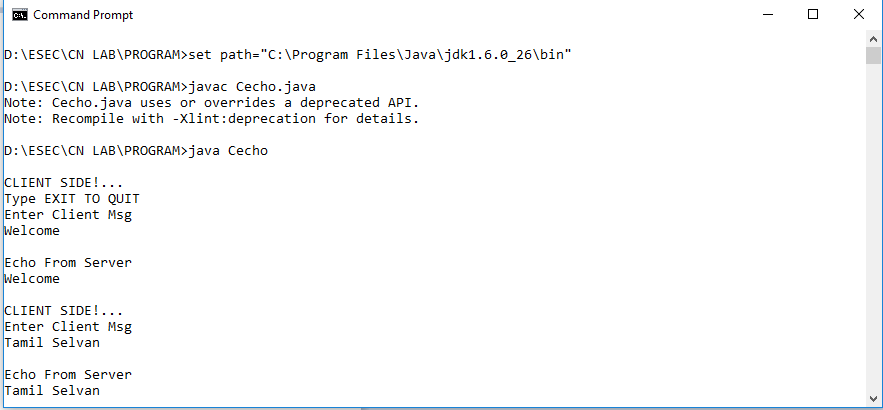
}

**OUTPUT:**

**SERVER OUTPUT:**

****

**CLIENT OUTPUT:**

****

**RESULT:**

Thus the program was executed and output is verified successfully.

|  |  |
| --- | --- |
| **Ex. No. 8(b)** | **CHAT APPLICATION USING TCP** |
|  |

**AIM**

To develop a chat applications using TCP sockets

**ALGORITHM**

**Step 1:** Client sends the request to server

**Step 2:** Server run the request and the connection with the client that has requested

the server**.**

**Step 3:** The client send the message to server.

**Step 4:** The server process it and it is displayed (ie)replier by sending the message to

client also displayed.

**Step 5:** Stop the program.

**SERVER PROGRAM:**

**chatserver.java**

import java.io.\*;

import java.net.\*;

public class chatserver

{

public static void main(String args[])throws Exception

{

DataInputStream din=null;

DataOutputStream dout=null;

Socket c=null;

ServerSocket m=null;

DataInputStream stdin=new DataInputStream(System.in);

try

{

m=new ServerSocket(68);

c=m.accept();

din=new DataInputStream(c.getInputStream());

dout=new DataOutputStream(c.getOutputStream());

}

catch(Exception e)

{ }

while(c!=null)

{

String m2;

System.out.println("Server");

while(true)

{

String m1=din.readLine();

System.out.println("Message from client.."+m1);

System.out.println("\n\n Enter the message...");

m2=stdin.readLine();

dout.writeBytes(""+m2);

dout.writeBytes("\n");

}

}

din.close();

dout.close();

c.close();

m.close();

}

}

**CLIENT PROGRAM:**

**chatclient.java**

import java.io.\*;

import java.net.\*;

public class chatclient

{

public static void main(String args[])throws Exception

{

Socket c=null;

DataInputStream uin=null;

DataInputStream din=null;

DataOutputStream dout=null;

try

{

c=new Socket("localhost",68);

uin=new DataInputStream(System.in);

din=new DataInputStream(c.getInputStream());

dout=new DataOutputStream(c.getOutputStream());

}

catch(Exception e)

{ }

if(c!=null)

{

String inp;

System.out.println("Enter the message:");

while((inp=uin.readLine())!=null)

{

dout.writeBytes(""+inp);

dout.writeBytes("\n");

System.out.println("Echoed message from server.."+din.readLine());

System.out.println("Enter ur message:");

}

}

din.close();

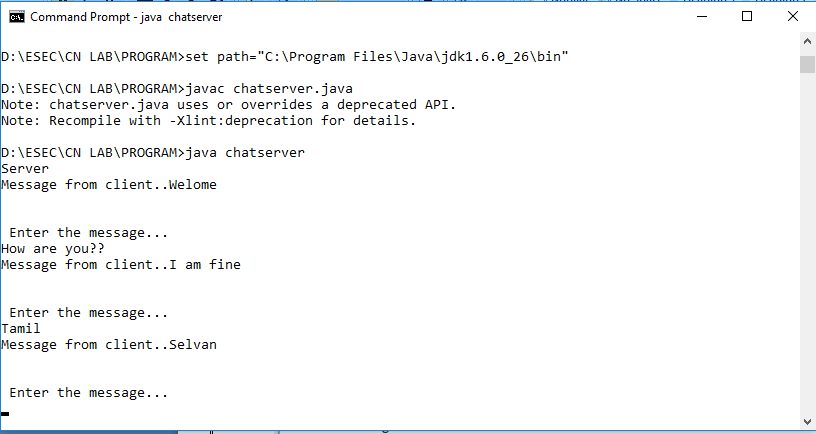
dout.close();

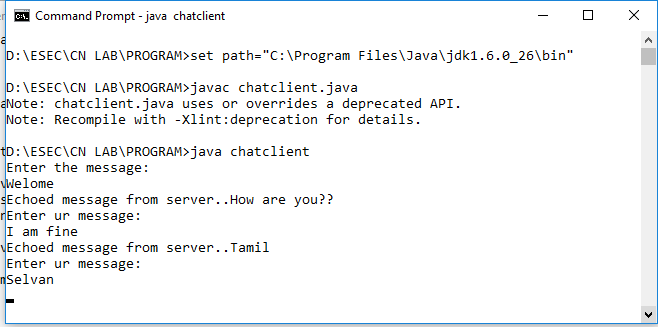
c.close();

}

}

**OUTPUT:**

****



**RESULT:**

Thus the program was executed and output is verified successfully.

|  |  |
| --- | --- |
| **Ex. No. 8(c)** | **FILE TRANSFER USING TCP** |
|  |

**AIM**

To develop a chat applications using TCP sockets

**ALGORITHM**

**SERVER**

**Step 1:** Open the socket and waits

**Step 2:** Accept the client request ,when it opens the socket

**Step 3:** Get the file sent by the client from the socket

**Step 4:** Read file stream and create a file in the current working directory and write

the stream to it

**Step 5:** Closes the connection

**CLIENT**

**Step 1:** Open the client TCP socket and establish connection with server

**Step 2:** Read the pathname of file to be uploaded from client

**Step 3:** Read the file and write to the socket

**Step 4:** Close the connection as the file is transferred completely

**Coding**

**SERVER PROGRAM**

**FtpS.java**

import java.io.\*;

import java.net.\*;

class FtpS

{

public static void main(String a[]) throws Exception

{

Socket s=null;

ServerSocket ss=null;

DataOutputStream sso=null;

DataInputStream sin=null;

ss=new ServerSocket(55555);

System.out.println("waiting");

s=ss.accept();

sso=new DataOutputStream(s.getOutputStream());

sin=new DataInputStream(s.getInputStream());

FileOutputStream fos=new FileOutputStream("abc1.txt");

int str1;

while((str1=sin.read())!=-1)

{

fos.write((char)str1);

System.out.println((char)str1);

}

sso.close();

s.close();

}

}

**CLIENT PROGRAM**

**FtpC.java**

import java.io.\*;

import java.net.\*;

public class FtpC

{

public static void main(String a[]) throws IOException

{

Socket s=null;

DataInputStream si=null;

s=new Socket("localhost",55555);

si=new DataInputStream(s.getInputStream());

DataInputStream inp=new DataInputStream(System.in);

DataOutputStream so=new DataOutputStream(s.getOutputStream());

String str;

System.out.println("enter the file name(path)");

str=inp.readLine();

int i;

FileInputStream fos=new FileInputStream(str);

while((i=fos.read())!=-1)

so.writeBytes(" "+(char)i);

si.close();

}

}

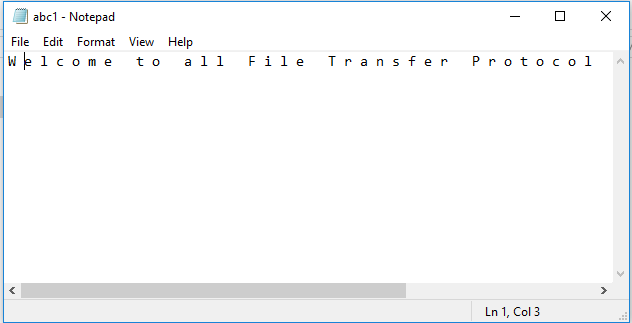
**OUTPUT:**

**SERVER WINDOW**

|  |
| --- |
| D:\ESEC\CN LAB\PROGRAM>set path="C:\Program Files\Java\jdk1.6.0\_26\bin"  D:\ESEC\CN LAB\PROGRAM>javac FtpS.java  D:\ESEC\CN LAB\PROGRAM>java FtpS  waiting  Welcome to |

**CLIENT WINDOW**

|  |
| --- |
| D:\ESEC\CN LAB\PROGRAM>set path="C:\Program Files\Java\jdk1.6.0\_26\bin"  D:\ESEC\CN LAB\PROGRAM>javac FtpC.java  Note: FtpC.java uses or overrides a deprecated API.  Note: Recompile with -Xlint:deprecation for details.  D:\ESEC\CN LAB\PROGRAM>java FtpC  enter the file name(path)  sample.txt  D:\ESEC\CN LAB\PROGRAM> |



**Result:**

Thus the given program has been created & executed successfully.

|  |  |
| --- | --- |
| **Ex. No. 9** | **DNS USING UDP SOCKETS** |
|  |

**AIM**

To develop simulate DNS using UDP sockets in Java program

**ALGORITHM**

**Step 1:** Start the program

**Step 2:** Create a Menu

1.DNS

2. Reserve DNS

3.Exit

**Step 3:** Get the choice from the user

**Step 4:** If option == 1:

1. Get the host from user
2. Get the IP address of the respective host name.

**Step 5:** If option ==2:

1. Get the IP address from the user
2. Find the host name of the respective IP address

**Step 6:** If option == 3, Stop the program

**Coding**

**DNS.java**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class DNS

{

public static void main(String[] args)

{

int n;

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

do

{

System.out.println("\n Menu: \n 1. DNS 2. Reverse DNS 3. Exit \n");

System.out.println("\n Enter your choice");

n = Integer.parseInt(System.console().readLine());

if(n==1)

{

try

{

System.out.println("\n Enter Host Name ");

String hname=in.readLine();

InetAddress address;

address = InetAddress.getByName(hname);

System.out.println("Host Name: " + address.getHostName());

System.out.println("IP: " + address.getHostAddress());

}

catch(IOException ioe)

{

ioe.printStackTrace();

}

}

if(n==2)

{

try

{

System.out.println("\n Enter IP address");

String ipstr = in.readLine();

InetAddress ia = InetAddress.getByName(ipstr);

System.out.println("IP: "+ipstr);

System.out.println("Host Name: " +ia.getHostName());

}

catch(IOException ioe)

{

ioe.printStackTrace();

}

}

}while(!(n==3));

}

}

**OUTPUT:**

|  |
| --- |
| D:\TAMIL\CN LAB\PROGRAM>javac DNS.java  D:\TAMIL\CN LAB\PROGRAM>java DNS  Menu:  1. DNS 2. Reverse DNS 3. Exit  Enter your choice: 1  Enter Host Name  www.erode-sengunthar.ac.in  Host Name: www.erode-sengunthar.ac.in  IP: 216.10.241.191  Menu:  1. DNS 2. Reverse DNS 3. Exit  Enter your choice: 2  Enter IP address  216.10.241.191  IP: 216.10.241.191  Host Name: bh-in-36.webhostbox.net  Menu:  1. DNS 2. Reverse DNS 3. Exit  Enter your choice: 3  D:\TAMIL\CN LAB\PROGRAM> |

**Result:**

Thus the given program has been created & executed successfully

|  |  |
| --- | --- |
| **Ex. No. 10** | Simulation of DNS using UDP sockets |
|  |

**Aim:**

To write a java program for Dns application program

## Problem Description:

The purpose of DNS is to resolve the IP address for the given Domain Name. It is a dependent protocol.DNS is simulated in this experiment by having a process which fetches the domain name as an argument and resolves the IP address by using inet address class and local DNS cache.

## Algorithm:

* Start the program
* Define two character arrays with hostnames and IP addresses
* Get the hostname from user
* Match the hostname with array and get the index.
* Display the corresponding IP Address.
* End of the program

## Program:

import javax.naming.directory.Attributes; import javax.naming.directory.InitialDirContext; import javax.naming.NamingEnumeration; import javax.naming.NamingException;

import java.net.InetAddress;

## import java.net.UnknownHostException; public class DNS

## 

## 

## Result:

Thus the above program has been executed and verified successfully.

## CONTENT BEYOND THE SYLLABUS EXPERIMENTS

|  |  |
| --- | --- |
| **Ex. No. 11** | Implementation of Token Ring Protocol. |
|  |

### AIM:

To study and implement the token ring protocol.

### APPARATUS REQUIRED:

* Pentium pc
* Java and eclipse software

### PRINCIPLE:

* In the token passing method, the stations in a network are organized In a logical ring.
* In a physical ring topology, when a station sends the token to its successor, the token cannot be seen by other stations.
* In this, each device has a dedicated point-to-point connection with only the two devices on either side of it.
* In this method, a special packet called token circulates throughout the ring.
* When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data.
* When the station has no more data to send, it releases the token, passing it to the next logical station in the ring

### ALGORITHM:

**Client 1:**

* Start the program
* Open socket with input address and port
* Establish a connection between client 1 and client2
* Pass the token to client 2
* Stop the program

### Client 2:

* Start
* Initialize server server socket
* Wait to connect with client2
* Initialize the socket and accept the client message
* Display connected with client1
* Receive the token sent by client 1
* Establish a connection between client 2 and client 3
* Open socket with input address and port
* Pass the token to client 3
* Stop

### Client 3:

* Start
* Initialize server socket
* Wait to connect with client2
* Initialize the socket and accept the client2 mess
* Receive the token which has sent by client2

### TOKEN RING PROGRAM CLIENT 1

import java.io.IOException; import java.net.Socket;

import java.net.SocketException; public class cl1

{

public static void main (String args [])

{

try

{

System.out.println("========== Client 1 ========="); Socket con = new Socket("192.168.5.2",140);

System.out.print("Token Sent to Client 2 : "+ con.getInetAddress().getHostAddress()); con.close();

}

catch (SocketException e)

{

System.out.print("\n Clinet 2 is disconnected from LAN"); System.out.print("\n Token Ring breaks");

}

catch(IOException e)

{

System.out.println("io error:"+e);

}

}

}

### TOKEN RING PROGRAM CLIENT 2

import java.io.IOException;

import java.net.ServerSocket; import java.net.Socket; public class cl2

{

public static void main (String args [])

{

try

{

System.out.println("========== Client 2 ========="); ServerSocket providersocket = new ServerSocket(140); System.out.println("waiting for connection");

Socket con = providersocket.accept();

System.out.print("Token received from Client 1 : "+ con.getInetAddress().getHostAddress()); providersocket.close();

Thread.sleep(4000); try

{

System.out.println(" \n==== Passing Token to next client ======"); Socket con2 = new Socket("192.168.5.3",140);

System.out.print("Token sent to Client 3 :"+ con2.getInetAddress().getHostAddress()); con2.close();

}

catch(IOException e)

{

System.out.println("Client 3 is disconnected from LAN"); System.out.println("\n Token Ring breaks");

}

}

catch (IOException e)

{

System.out.println("socket error:" +e);

}

catch (InterruptedException e)

{

System.out.println("socket error:" +e);

}

}

}

### TOKEN RING PROGRAM CLIENT 3

import java.io.IOException; import java.net.ServerSocket; import java.net.Socket; public class cl3

{

public static void main (String args [])

{

try

{

System.out.println("========== Client 3 ========="); ServerSocket providersocket = new ServerSocket(140); System.out.println("waiting for connection");

Socket con = providersocket.accept();

System.out.print("Token received from Client 2 : "+ con.getInetAddress().getHostAddress()); providersocket.close();

}

catch (IOException e)

{

System.out.println("socket error:" +e);

}

}

}

Case 1 : When all the clients connected in LAN Client 1:

========== Client 1 =========

Token Sent to Client 2 : 192.168.5.2 Client 2:

========== Client 2 =========

waiting for connection

Token received from Client 1 : 192.168.5.1

========== Passing Token to next client ========= Token sent to Client 3 :192.168.5.3

Client 3 :

========== Client 3 =========

waiting for connection

Token received from Client 2 : 192.168.5.2 Case 2: When Client 2 disconnected from LAN Client 1:

========== Client 1 =========

Client 2 is disconnected from LAN Token Ring breaks

Client 2:

(Don’t run program in client2, it is assumed like client 2 disconnected from LAN) Client 3:

========== Client 3 =========

waiting for connection

Case 3: When Client 3 disconnected from LAN Client 1:

========== Client 1 =========

Token Sent to Client 2 : 192.168.5.2 Client 2:

========== Client 2 =========

waiting for connection

Token received from Client 1 : 192.168.5.1

========== Passing Token to next client =========

Client 3 is disconnected from LAN Token Ring breaks

Client 3:

(Don’t run program in client 3 it is assumed like client 3 disconnected from LAN) Case 4: When Client 2 and Client 3 disconnected from LAN

Client 1:

========== Client 1 =========

Client 2 is disconnected from LAN Client 2:

(Don’t run program in client 2 it is assumed like client 2 disconnected from LAN) Client 3:

(Don’t run program in client 3 it is assumed like client 3 disconnected from LAN)

### PROCEDURE:

1. Type the Client 1, Client 2 and Client 3 program in three different computers.
2. To verify Case 1: first execute Client 3 then Client 2 then Client 1, so Client 3waiting for token from Client 2, Client 2 waiting for Token from Client 1, Client 1 starts the token sending, it sends token to Client 2, now client 2 holds token for 40 seconds then it sends token to Client 3.
3. To verify Case 2: first execute Client 3 then Client 1, don’t execute program in Client 2 (It is assumed like Client 2 is disconnected from the LAN), Client 1 starts the token sending, it sends token to Client 2, but Client 2 is disconnected from the LAN, so that, the token ring breaks.
4. To verify Case 3: Don’t execute Client 3 (It is assumed like Client 3 is disconnected from LAN), execute program in Client 2 then Client 1, Client 1 sends token to Client 2, now Client 2 trying to send token to Client 3, but Client 3 is disconnected from the LAN, so that, token ring breaks.
5. To verify Case 4 : Don’t execute Client 3 and Client 2 (It is assumed like Client 3 and Client 2 are disconnected from the LAN). So the token ring process is not initiated at all.

**RESULT:** Thus, the token ring protocol was programmed using java and implemented successfully in the laboratory.

|  |  |
| --- | --- |
| **Ex. No. 12** | IMPLEMENTATION AND STUDY OF CSMA/CD |
|  |

### AIM

To study and implement the Carrier Sense Multiple Access with Collision Detection APPARATUS REQUIRED:

1. Pentium – PC
2. Eclipse
3. Java PRINCIPLE:

The Carrier Sense Multiple Access is based on the principle of “SENSE BEFORETRANSMIT”. This CSMA/CD is generally used in wired networks. It is mainly focused to detect the collision if it has occurred.

ALGORITHM:

SERVER:

1. Initialize server socket
2. Display waiting for connection
3. Initialize the socket and accept the client message
4. Display connected with client
5. Initialize i/p stream
6. Read message
7. Display message from client
8. Close all objects
9. Stop CLIENT:
10. Open socket with input address ,port
11. Initialize o/p stream
12. Send the message, if message sent collision not occurred.
13. If message not sent, collision occurred (To occur collision don’t run not server)
14. Calculate back of time using random number selection and wait for that time
15. Again send the message, if message sent collision not occurred.
16. If message not sent, collision occurred, Again calculate back off time by selecting random number, this trail can be done for 15 times.
17. If not succeeded with 15 trails transmission will be stopped.

CSMA/CD PROGRAM SERVER:

import java.io.\*; import java.net.\*; public class Server

{

public static void main(String[] args)

{

try

{

System.out.println("============ Client 2 ==============="); ServerSocket ss = new ServerSocket(137); System.out.println("Waiting for connection");

Socket con = ss.accept();

System.out.println("Connected");

ObjectInputStream in = new ObjectInputStream(con.getInputStream()); System.out.println((String)in.readObject());

in.close();

ss.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

}

CSMA/CD PROGRAM

CLEINT:

import java.io.\*; import java.net.\*; public class client1

{

public static void main(String[] args)

{

try

{

System.out.println("============ Client 1 ==============="); client1 cli = new client1();

int Tp = 2000; int R = 0;

int Tb = 0;

for(int i=1; i<=15;i++)

{

System.out.println("attempt : "+i); if(cli.send() == "sent")

{

break;

}

else

{

R = 2^i-1;

System.out.println("Selected Random number :"+R); Tb = R\*Tp;

System.out.println("waiting for next attempt with back time (in seconds): "+Tb); Thread.sleep(Tb);

}

}

}

catch (InterruptedException e)

{

System.out.println(e);

}

}

String send()

{

String str=null; try

{

Socket soc = new Socket("localhost",137);

ObjectOutputStream out = new ObjectOutputStream(soc.getOutputStream()); String msg = "CNLAB";

out.writeObject(msg); System.out.println("Message sent : "+msg); str = "sent";

}

catch(Exception e)

{

str = "collision occured"; System.out.println("Message sent : "+msg);

}

return str;

}

} OUTPUT

============ Client 1 ===============

attempt : 1 collision occured

Selected Random number :2

waiting for next attempt with back time (in seconds): 4000

attempt : 2 collision occured

Selected Random number :3

waiting for next attempt with back time (in seconds): 6000 attempt : 3

collision occured

Selected Random number :0

waiting for next attempt with back time (in seconds): 0 attempt : 4

collision occured

Selected Random number :1

waiting for next attempt with back time (in seconds): 2000 attempt : 5

Message sent : CNLAB

============ Server ===============

Waiting for connection Connected

CNLAB

### PROCEDURE:

1. Run the Client program, don’t run server program (consider as to raise collision)
2. Client will try to send the message to server, but server is not running, so the attempt is failed, so consider as collision occurred.
3. Client will select random number and calculate Back off time and wait for the back off time and again will try to send the message. This loop continues till the 15 trails.
4. Run the server in between consider as no collision, so message will be sent from client to server.

### RESULT:

Thus the CSMA/CD is executed and studied.

### VIVA QUESTIONS:

1. **What is Routing algorithm?**

Routing is the process of selecting paths in a network along which to send network traffic. Routing isperformed for many kinds of networks, including the telephone network (Circuit switching),electronicdata networks (such as the Internet), and transportation networks.

### What is Distance vector routing algorithm?

A distance-vector routing protocol is one of the two major classes of routing protocols, the other majorclass being the link-state protocol. A distance-vector routing protocol requires that a router informs itsneighbors of topology changes periodically. Compared to link-state protocols, which require a router toinform all the nodes in a network of topology changes, distance-vector routing protocols have lesscomputational complexity and message overhead.

### Define parity bit.

The simplest form of error detection is to append a single bit called a parity bit to a string of data.

### Define hamming distance.

The number of bits positions in which two codeword differ is called the hamming distance.

### What is meant by codeword & block length?

Codeword is the encoded block of ‘n’ bits. It contains message bits and redundant bits. Block length: the number of bits ‘n’ after coding is called the block length of the code

1. What is IP CONFIG?
2. How to get a valid ipconfig?
3. What is TCP/IP configuration?
4. What are the parameters ipconfig displays?

10.What is the use of DNS and DHCP in ipconfig?

11. What is nslookup?

12.What is the use of nslookup?

13.How sockets can be used to write client-server application using a connection-oriented client- server technique?

14.What this function nslookup does?

15.What this function DNS does?

16.What is traceroute?

17.How does the race condition. occur?

18.What does a socket consists of?

19.What is the difference between a NULL pointer and a void pointer?

20.What is encapsulation techniques?

21.What is PING?

22.What is the use of PING?

23.What is a message queue?

24.What are RAW sockets?

25.What are public and private ports?

26.What is a Java bean?