



**V V COLLEGE OF ENGINEERING**  
(Approved By AICTE, New Delhi and Affiliated To Anna University Chennai)  
**V V Nagar, Arasoor, Tisaiyanvilai**

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **College Vision and Mission Statement**

#### **Vision**

“Emerge as a premier technical institution of global standards, producing enterprising, knowledgeable engineers and entrepreneurs.”

#### **Mission**

- Impart quality and contemporary technical education for rural students.
- Have the state of the art infrastructure and equipment for quality learning.
- Enable knowledge with ethics, values and social responsibilities.
- Inculcate innovation and creativity among students for contribution to society.

### **Vision and Mission of the Department of Computer Science and Engineering**

#### **Vision**

“Produce competent and intellectual computer science graduates by empowering them to compete globally towards professional excellence”.

#### **Mission**

- Provide resources, environment and continuing learning processes for better exposure in latest and contemporary technologies in Computer Science and Engineering.
- Encourage creativity and innovation and the development of self-employment through knowledge and skills, for contribution to society
- Provide quality education in Computer Science and Engineering by creating a platform to enable coding, problem solving, design, development, testing and implementation of solutions for the benefit of society.

## I. PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

Graduates of Computer Science Engineering can

- Apply their technical competence in computer science to solve real world problems, with technical and people leadership.
- Conduct cutting edge research and develop solutions on problems of social relevance.
- Work in a business environment, exhibiting team skills, work ethics, adaptability and lifelong learning.

## II. PROGRAM OUTCOMES (POs)

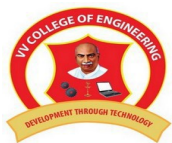
1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching 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 to 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 in 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 and 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. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

### III. PROGRAM SPECIFIC OUTCOMES (PSOs)

The Students will be able to

- Exhibit design and programming skills to build and automate business solutions using cutting edge technologies.
- Strong theoretical foundation leading to excellence and excitement towards research, to provide elegant solutions to complex problems.
- Ability to work effectively with various engineering fields as a team to design, build and develop system applications.



**V V COLLEGE OF ENGINEERING**  
(Approved By AICTE, New Delhi and Affiliated To Anna University Chennai)  
**V V Nagar, Arasoor, Tisaiyanvilai**

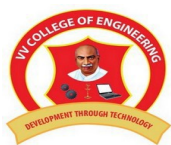
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**LABORATORY**

**LIST OF EXPERIMENTS – R2021**

<b>PRACTICAL SUBJECT NAME</b>	COMPUTER NETWORKS LABORATORY
<b>PRACTICAL SUBJECT CODE</b>	CS3591
<b>SEMESTER/ YEAR</b>	05 / THIRD
<b>TOTAL HOURS</b>	30
<b>STAFF IN-CHARGE</b>	Mrs. M. JANCYRANI MALLI
<b>LAB INSTRUCTOR</b>	Mr. VINOTH
<b>REGULATION</b>	2021

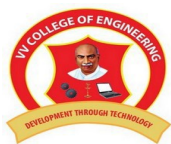
<b>CO1</b>	Explain the basic layers and its functions in Computer Networks.
<b>CO2</b>	Understand the basis of how data flows from one node to another.
<b>CO3</b>	Analyze routing Algorithms.
<b>CO4</b>	Describe protocols for various functions in the network.
<b>CO5</b>	Analyze the working of various application layer protocols.



**V V COLLEGE OF ENGINEERING**  
(Approved By AICTE, New Delhi and Affiliated To Anna University Chennai)  
**V V Nagar, Arasoor, Tisaiyanvilai**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

<b>S.No</b>	<b>Name of the Experiment</b>	<b>CO Mapping</b>	<b>PO Mapping</b>
1	Networking Commands	CO1	PO2 PSO1- PSO3
2	HTTP Web Client Program To Download a Web Page Using TCP Sockets	CO2	PO2,PO5, PSO1- PSO3
3	Applications of TCP Sockets a) TCP - Echo Client and Echo Server b) TCP – Client and Server Application for Chat c) TCP – Client and Server File Transfer	CO2	PO3,PO5, PSO1- PSO3
4	Simulation of DNS Using UDP Sockets	CO2	PO4,PO5, PSO1- PSO3
5	WIRESHARK Tool to Capture Packets	CO3	PO2-PO5, PSO1-PSO3
6	Simulation of ARP / RARP Protocols	CO4	PO5, PSO1- PSO3
7	a) Study of Network Simulator (NS) b)Simulation of Congestion Control Algorithms Using NS	CO4	PO3,PO5, PSO1- PSO3
8	Study of TCP/UDP Performance Using Simulation Tool	CO4	PO1-PO5, PSO1-PSO3
9	a) Simulation of Distance Vector Routing Algorithm b) Simulation of Link State Routing Algorithm	CO3	PO3, PSO1-PSO3
10	Simulation of Error Correction Code (Like CRC)	CO5	PO5, PSO1- PSO3

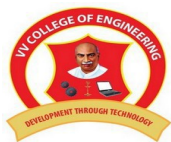


**V V COLLEGE OF ENGINEERING**  
(Approved By AICTE, New Delhi and Affiliated To Anna University Chennai)  
**V V Nagar, Arasoor, Tisaiyanvilai**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**RUBRICS FOR ASSESSING LABORATORY**

Sl. No.	Criteria	Total Marks	Excellent (25)	Good (20)	Average (10)	Poor (5)
			91% - 100%	71% - 90%	50% - 70%	<50%
1	Preparation	25	Gives clear idea about the aim and having good capability of executing experiments.	Capability of executing experiments but no proper clarification about the objective.	Gives clear idea about the target and has less capability of executing experiments.	Gives indistinct idea about the target and has less capability of executing experiments & who feel difficult to follow the objectives.
2	Viva	25	Have executed the experiments in an efficient way & make credible and unbiased judgments regarding the experiments.	Executed the experiments with less efficient & has partial judgments regarding the experiments.	Executed the experiments with less efficiency and has no judgments regarding experiments .	Incomplete experiments & lack of judgments regarding experiments.
3	Performance	25	Followed all the instructions given in the procedure and submitted the manual on time.	Followed all the instructions given in the procedure with some assisting.	Followed some of the instructions given in the procedure & late in submission of manual.	Unable to follow the instructions given in the procedure & late in submission of manual.



**V V COLLEGE OF ENGINEERING**  
(Approved By AICTE, New Delhi and Affiliated To Anna University Chennai)  
**V V Nagar, Arasoor, Tisaiyanvilai**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

Department of Computer Science and Engineering		
Preparation	25	
Viva	25	
Performance	25	
Total	75	
Lab Incharge	Date	

## INDEX

S.No	DATE	NAME OF THE EXPERIMENT	SIGN
1		Networking Commands	
2		HTTP Web Client Program To Download a Web Page Using TCP Sockets	
3		Applications of TCP Sockets a) TCP - Echo Client and Echo Server b) TCP – Client and Server Application for Chat c) TCP – Client and Server File Transfer	
4		Simulation of DNS Using UDP Sockets	
5		WIRESHARK Tool to Capture Packets	
6		Simulation of ARP / RARP Protocols	
7		a) Study of Network Simulator (NS) b)Simulation of Congestion Control Algorithms Using NS	
8		Study of TCP/UDP Performance Using Simulation Tool	
9		a) Simulation of Distance Vector Routing Algorithm b) Simulation of Link State Routing Algorithm	
10		Simulation of Error Correction Code (Like CRC)	



**Ex.No. 1****NETWORKING COMMANDS****AIM:**

To use and study the commands tcpdump, netstat, ifconfig, nslookup and traceroute. To capture ping and traceroute PDUs using a network protocol analyzer and examine.

**1. tcpdump (for LINUX)**

Tcpdump is a command line utility that allows you to capture and analyze network traffic going through your system.

**PROCEDURE:**

Check if tcpdump is installed on your system

```
$ which tcpdump
/usr/sbin/tcpdump
```

If tcpdump is not installed,

```
$ sudo yum install -y tcpdump
```

To get Supervisor Privilege

```
$ su
```

(And password vvcse@123)

(\$ is changed to # and the commands can be executed in supervisor)

Capturing packets with tcpdump

Use the command tcpdump -D to see which interfaces are available for capture.

```
[root@localhost it]# tcpdump -D
```

- 1.nflog (Linux netfilter log (NFLOG) interface)
- 2.nfqueue (Linux netfilter queue (NFQUEUE) interface)
- 3.usbmon1 (USB bus number 1)
- 4.enp2s0
- 5.usbmon2 (USB bus number 2)
- 6.any (Pseudo-device that captures on all interfaces)
- 7.lo [Loopback]

Capture all packets in any interface by running this command:

```
[root@localhost it]# tcpdump -i any
```

```
06:03:58.258143 ARP, Request who-has 172.16.51.87 tell 172.16.22.25, length 46
06:03:58.258225 ARP, Request who-has 172.16.51.88 tell 172.16.22.25, length 46
06:03:58.260828 ARP, Request who-has 172.16.51.122 tell 172.16.22.25, length 46
06:03:58.260903 ARP, Request who-has 172.16.51.123 tell 172.16.22.25, length 46
^C
```

5244 packets captured

59636 packets received by filter

54378 packets dropped by kernel

(Press ctrl+C to stop execution)

Filter packets based on the source or destination IP Address

```
[root@localhost it]#tcpdump -i any -c5 -nn src 172.16.20.138
```

```
6:10:30.712414 ARP, Request who-has 172.16.16.16 tell 172.16.20.138, length 28
```

```
06:10:31.483765 IP 172.16.20.138.47997 > 51.158.186.98.123: NTPv4, Client, length 48
```

```
5 packets captured
```

```
5 packets received by filter
```

```
0 packets dropped by kernel
```

```
[root@localhost it]#tcpdump -i any -c5 -nn dst 172.16.20.139
```

```
6:10:30.712414 ARP, Request who-has 172.16.16.16 tell 172.16.20.138, length 28
```

```
06:10:31.483765 IP 172.16.20.138.47997 > 51.158.186.98.123: NTPv4, Client, length 48
```

```
5 packets captured
```

```
5 packets received by filter
```

```
0 packets dropped by kernel
```

```
Filtering packets
```

To filter packets based on protocol, specifying the protocol in the command line. For example, capture ICMP packets only by using this command:

```
[root@localhost it]# tcpdump -i any -c5 icmp
```

```
(tcpdump captures and displays only the ICMP-related packets.)
```

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
```

```
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
```

```
06:15:07.800786 IP localhost.localdomain > ec2-54-204-39-132.compute-1.amazonaws.com: ICMP echo request, id 8180, seq 13, length 64
```

```
06:15:08.063488 IP ec2-54-204-39-132.compute-1.amazonaws.com > localhost.localdomain: ICMP echo reply, id 8180, seq 13, length 64
```

```
5 packets captured
```

```
5 packets received by filter
```

```
0 packets dropped by kernel
```

#### a). windump (for Windows)

WinDump is the Windows version of tcpdump, the command line network analyzer for UNIX. WinDump is fully compatible with tcpdump and can be used to watch, diagnose and save to disk network traffic according to various complex rules.

#### INSTALLATION PROCEDURE:

1. Download and install Windump (<http://www.winpcap.org/windump/>)
2. You will need to place your network card into promiscuous mode – for this, install WinPcap. Download and install WinPcap. (<http://www.winpcap.org/install/default.htm>)

#### Testing Procedure:

1. Open a Command Prompt with Administrator Rights
2. Change the directory to your download directory  
**cd c:\Program Files\**

3. Run windump to collect packets  
c:\Program Files\windump
4. Run windump to locate your network adapter  
c:\Program Files\windump -D
5. Windump will list your adapter with a number.

You may have several adapters listed. You select the interface to start running windump. (as shown in step 5 using interface number 1).

6. Run windump to collect packets and write out to a file

```
c:\Program Files\windump -i 1 -q -w C:\perflogs\diagTraces -n -C 30 -W
10 -U -s 0
```

**This will create a directory c:\perflogs\ and a file called diagTrace0.**

The switches mean this:

-i is the number of NIC selected in the previous step

-q is quiet mode

-w <name> is the prefix of the files to create

-n the logging will not resolve host names, all data will be in IP address format

-C the size in Millions of Bytes the logs files so grow to before moving to the next file

-W the number of circular log files to retain in addition to the current log file, specify in <path>

where the files are to be stored

-U as each packet is saved, it will be written to the output file

-s decreases the amount of packet buffering, set this to zero

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\IT HOD>cd Downloads
C:\Users\IT HOD\Downloads>windump
windump: listening on \Device\NPF_{8F64F018-0896-4E4E-BBDF-8CCC0986AD83}
13:38:14.617056 arp who-has 172.16.72.89 tell CS03C014
13:38:14.723835 28:34:a2:19:26:bd <oui Unknown> > 34:db:fd:77:e4:61 <oui Unknown>
>, ethertype Unknown (0xa0a0), length 60:
    0x0000: 0006 0101 0101 0101 0101 0101 0101 0101 .....
    0x0010: 0101 0101 0101 0101 0101 0101 0101 0101 .....
    0x0020: 0101 0101 0101 0101 0101 0101 0101 .....
13:38:14.784612 IP MECCAD33.137 > 172.16.255.255.137: UDP, length 50
13:38:14.795666 IP6 sacoeit-PC.57153 > ff02::1:3.5355: UDP, length 24
13:38:14.795937 IP sacoeit-PC.55379 > 224.0.0.252.5355: UDP, length 24
13:38:14.851049 arp who-has 172.16.73.89 tell CS03C014
13:38:14.903731 IP6 sacoeit-PC.57153 > ff02::1:3.5355: UDP, length 24
13:38:14.903865 IP sacoeit-PC.55379 > 224.0.0.252.5355: UDP, length 24
13:38:14.975530 arp who-has 172.16.64.89 tell CS03C014
13:38:14.975533 arp who-has 172.16.65.89 tell CS03C014
13:38:14.975583 arp who-has 172.16.66.89 tell CS03C014
13:38:14.975585 arp who-has 172.16.70.89 tell CS03C014
13:38:14.975849 arp who-has 172.16.74.89 tell CS03C014
13:38:15.095201 802.1d unknown version
13:38:15.107012 IP sacoeit-PC.137 > 172.16.255.255.137: UDP, length 50
13:38:15.324697 arp who-has 172.16.118.185 tell it02c16
13:38:15.324700 arp who-has 172.16.118.186 tell it02c16
13:38:15.324702 arp who-has 172.16.118.187 tell it02c16
13:38:15.324703 arp who-has 172.16.118.188 tell it02c16
```

```

C:\Windows\system32\cmd.exe - windump -i 1 -q -w C:\perflogs\diagtraces -n -C 30 -W 10 -U -s 0
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\IT HOD>cd Downloads

C:\Users\IT HOD\Downloads>windump -D
1.\Device\NPF_{8F64F018-0896-4E4E-BBDF-8CCC0986AD83} <Realtek RTL8168B/8111B PCI
-E Gigabit Ethernet NIC>

C:\Users\IT HOD\Downloads>windump -i 1 -q -w C:\perflogs\diagtraces -n -C 30 -W
10 -U -s 0
windump: listening on \Device\NPF_{8F64F018-0896-4E4E-BBDF-8CCC0986AD83}
-

```

## 2. netstat (For LINUX and Windows)

netstat (network statistics) is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc.

```

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\IT HOD>netstat

Active Connections

Proto Local Address Foreign Address State
TCP 127.0.0.1:49218 ITHOD-PC:49219 ESTABLISHED
TCP 127.0.0.1:49219 ITHOD-PC:49218 ESTABLISHED
TCP 127.0.0.1:49252 ITHOD-PC:49253 ESTABLISHED
TCP 127.0.0.1:49253 ITHOD-PC:49252 ESTABLISHED
TCP 127.0.0.1:49269 ITHOD-PC:49270 ESTABLISHED
TCP 127.0.0.1:49270 ITHOD-PC:49269 ESTABLISHED
TCP 127.0.0.1:49295 ITHOD-PC:49296 ESTABLISHED
TCP 127.0.0.1:49296 ITHOD-PC:49295 ESTABLISHED
TCP 172.16.22.129:8194 SOPHOESG:64609 ESTABLISHED
TCP 172.16.22.129:8194 ITHOD-PC:49293 ESTABLISHED
TCP 172.16.22.129:49220 74.125.24.188:5228 ESTABLISHED
TCP 172.16.22.129:49258 SOPHOESG:8194 ESTABLISHED
TCP 172.16.22.129:49293 ITHOD-PC:8194 ESTABLISHED
TCP 172.16.22.129:50690 sacoe-PC:microsoft-ds ESTABLISHED

```

Displays protocol statistics and current TCP/IP network connections.

```

NETSTAT [-a] [-b] [-e] [-f] [-n] [-o] [-p proto] [-r] [-s] [-t] [interval]

-a          Displays all connections and listening ports.
-b          Displays the executable involved in creating each connection or
           listening port. In some cases well-known executables host
           multiple independent components, and in these cases the
           sequence of components involved in creating the connection
           or listening port is displayed. In this case the executable
           name is in [] at the bottom, on top is the component it called,
           and so forth until TCP/IP was reached. Note that this option
           can be time-consuming and will fail unless you have sufficient
           permissions.
-e          Displays Ethernet statistics. This may be combined with the -s
           option.
-f          Displays Fully Qualified Domain Names (FQDN) for foreign
           addresses.
-n          Displays addresses and port numbers in numerical form.
-o          Displays the owning process ID associated with each connection.
-p proto    Shows connections for the protocol specified by proto; proto
           may be any of: TCP, UDP, TCPv6, or UDPv6. If used with the -s
           option to display per-protocol statistics, proto may be any of:
           IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, or UDPv6.
-r          Displays the routing table.
-s          Displays per-protocol statistics. By default, statistics are
           shown for IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, and UDPv6;
           the -p option may be used to specify a subset of the default.
-t          Displays the current connection offload state.
interval    Redisplays selected statistics, pausing interval seconds
           between each display. Press CTRL+C to stop redisplaying
           statistics. If omitted, netstat will print the current
           configuration information once.

```

### 3. ifconfig (For LINUX)

It displays the details of a network interface card like IP address, MAC Address, and the status of a network interface card.

```
[lit@localhost ~]$ ifconfig
```

```
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 172.16.20.138 netmask 255.255.0.0 broadcast 172.16.255.255
inet6 fe80::d884:13bc:fd22:2d43 prefixlen 64 scopeid 0x20<link>
ether a0:8c:fd:e7:10:86 txqueuelen 1000 (Ethernet)
RX packets 4474083 bytes 280780119 (267.7 MiB)
RX errors 0 dropped 353 overruns 0 frame 0
TX packets 14455 bytes 1798944 (1.7 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 4154 bytes 352264 (344.0 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 4154 bytes 352264 (344.0 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

#### a). ipconfig (For Windows)

(internet protocol configuration) is a console application of some operating systems that displays all current TCP/IP network configuration values and refresh Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings.

```
C:\Users\IT HOD>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::20dc:3b80:28fe:3432%11
    IPv4 Address. . . . . : 172.16.22.129
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : 172.16.16.16

Tunnel adapter isatap.{8F64F018-0896-4E4E-BBDF-8CCC0986AD83}:

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

C:\Users\IT HOD>
```

### 4. nslookup (For LINUX and Windows)

nslookup (stands for “Name Server Lookup”) is a useful command for getting information from DNS server. It is a network administration tool for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or any other specific DNS record.

1. nslookup annauniv.edu
2. nslookup 172.217.26.206
3. nslookup -type=any annauniv.edu

#### a). nslookup -type=ns annauniv.edu

```
C:\Users\IT HOD>nslookup -type=ns annauniv.edu
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
annauniv.edu    nameserver = ns1.annauniv.edu

C:\Users\IT HOD>_
```

```
C:\Users\IT HOD>nslookup annauniv.edu
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
Name:    annauniv.edu
Address:  14.139.161.7
```

```
C:\Users\IT HOD>nslookup 172.217.26.206
Server:  dns.google
Address:  8.8.8.8

Name:    maa03s23-in-f206.1e100.net
Address:  172.217.26.206
```

```
C:\Users\IT HOD>nslookup -type=any annauniv.edu
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
annauniv.edu    internet address = 14.139.161.7
annauniv.edu    text =

        "v=spf1 ip4:14.139.161.8 -all"
annauniv.edu
        primary name server = ns1.annauniv.edu
        responsible mail addr = root.annauniv.edu
        serial      = 20190820
        refresh     = 3000 (5 mins)
        retry       = 9000 (15 mins)
        expire      = 604800 (7 days)
        default TTL = 86400 (1 day)
annauniv.edu    nameserver = ns1.annauniv.edu
annauniv.edu    MX preference = 0, mail exchanger = sonic.annauniv.edu

C:\Users\IT HOD>nslookup -type=any annauniv.edu
```

#### 5. traceroute ( For LINUX)

The traceroute command is used in Linux to map the journey that a packet of information undertakes from its source to its destination.

[it@localhost ~]\$ **traceroute**

Usage:

traceroute [ -4dFITnreAUDV ] [ -f first\_ttl ] [ -g gate,... ] [ -i device ] [ -m max\_ttl ] [ -N squeries ] [ -p port ] [ -t tos ] [ -l flow\_label ] [ -w waittime ] [ -q nqueries ] [ -s src\_addr ] [ -z sendwait ] [ --fwmark=num ] host [ packetlen ]

Options:

-4	Use IPv4
-6	Use IPv6
-d --debug	Enable socket level debugging
-F --dont-fragment	Do not fragment packets

```
[it@localhost ~]$ traceroute annauniv.edu
```

```
traceroute to annauniv.edu (103.70.60.38), 30 hops max, 60 byte packets
```

```
 1  117.193.124.33 (117.193.124.33)  1.389 ms  1.216 ms  1.072 ms
 2  172.16.199.74 (172.16.199.74)  1.902 ms  1.834 ms  1.761 ms
 3  218.248.235.161 (218.248.235.161) 27.212 ms * *
 4  * * *
 5  218.248.178.42 (218.248.178.42) 15.521 ms * *
 6  * * *
 7  madurai-eg-175.232.249.45.powergrid.in (45.249.232.175) 16.007 ms
15.345 ms 15.867 ms
```

```
[it@localhost ~]$ traceroute 172.16.20.139
```

```
traceroute to 172.16.20.139 (172.16.20.139), 30 hops max, 60 byte packets
```

```
 1  localhost.localdomain (172.16.20.138) 3004.348 ms !H 3004.215 ms !H
3004.104 ms !H
```

#### a). tracert (For Windows)

The tracert command is a Command Prompt command that's used to show several details

about the path that a packet takes from source to destination.

1. tracert /?
2. tracert drsacoe.com
3. tracert 172.16.20.139

```
C:\Users\IT HOD>tracert /?
```

```
Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout]
              [-R] [-S srcaddr] [-4] [-6] target_name
```

Options:

-d	Do not resolve addresses to hostnames.
-h maximum_hops	Maximum number of hops to search for target.
-j host-list	Loose source route along host-list (IPv4-only).
-w timeout	Wait timeout milliseconds for each reply.
-R	Trace round-trip path (IPv6-only).
-S srcaddr	Source address to use (IPv6-only).
-4	Force using IPv4.
-6	Force using IPv6.

```
C:\Users\IT HOD>_
```

```
Trace complete.
```

```
C:\Users\IT HOD>tracert 172.16.20.139
```

```
Tracing route to 172.16.20.139 over a maximum of 30 hops
```

```
 1  ITHOD-PC [172.16.22.129] reports: Destination host unreachable.
```

```
Trace complete.
```

```
C:\Users\IT HOD>_
```

```
C:\Users\IT HOD>tracert -d 8.8.8.8
```

```
Tracing route to 8.8.8.8 over a maximum of 30 hops
```

1	1 ms	1 ms	1 ms	117.193.124.33
2	1 ms	1 ms	1 ms	172.16.199.74
3	19 ms	19 ms	19 ms	218.248.255.5
4	19 ms	*	*	218.248.255.6
5	*	*	*	Request timed out.
6	25 ms	22 ms	25 ms	72.14.211.114
7	22 ms	22 ms	22 ms	108.170.253.113
8	21 ms	21 ms	21 ms	216.239.43.173
9	22 ms	22 ms	22 ms	8.8.8.8

```
Trace complete.
```

```
C:\Users\IT HOD>_
```

```

Trace complete.
C:\Users\IT HOD>tracert drsacoe.com

Tracing route to drsacoe.com [184.168.221.39]
over a maximum of 30 hops:
  0  1 ms    1 ms    1 ms    117.193.124.33
  1  10 ms   1 ms    1 ms    172.16.199.74
  2  13 ms   *       *       218.248.255.5
  3  *       10 ms   10 ms   218.248.255.6
  4  21 ms   21 ms   21 ms   115.110.161.85.static.vsnl.net.in [115.110.161.85]
  5  21 ms   21 ms   21 ms   115.114.85.222
  6  21 ms   21 ms   21 ms   115.114.85.241
  7  149 ms  148 ms  149 ms   if-ae-3-3.tcore2.cxr-chennai.as6453.net [180.87.36.6]
  8  154 ms  154 ms  153 ms   if-ae-9-2.tcore2.mlv-mumbai.as6453.net [180.87.37.10]
  9  147 ms  147 ms  147 ms   if-ae-2-2.tcore1.mlv-mumbai.as6453.net [180.87.38.11]
 10  157 ms  155 ms  154 ms   if-ae-5-6.tcore1.wyn-marseille.as6453.net [180.87.38.126]
 11  155 ms  156 ms  155 ms   if-ae-8-1600.tcore1.pye-paris.as6453.net [80.231.217.6]
 12  156 ms  156 ms  161 ms   if-ae-11-2.tcore1.pvu-paris.as6453.net [80.231.53.49]
 13  263 ms  263 ms  263 ms   80.231.153.66
 14  *       *       *       Request timed out.
 15  324 ms  324 ms  324 ms   4.28.83.74
 16  *       *       *       Request timed out.
 17  321 ms  322 ms  321 ms   148.72.32.7
 18  327 ms  327 ms  327 ms   be38.trmc0215-01.ars.mgmt.phx3.gdg [184.168.0.69]
 19  323 ms  324 ms  325 ms   ip-184-168-0-94.ip.secureserver.net [184.168.0.94]
 20  333 ms  333 ms  333 ms   ip-184-168-221-39.ip.secureserver.net [184.168.221.39]

Trace complete.
C:\Users\IT HOD>

```

## 6. ping command

The ping command is a Command Prompt command used to test the ability of the source computer to reach a specified destination computer. The ping command is usually used as a simple way to verify that a computer can communicate over the network with another computer or network device.

The ping command operates by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination computer and waiting for a response. How many of those responses are returned, and how long it takes for them to return, are the two major pieces of information that the ping command provides.

### a). ping 127.0.0.1

To test whether or not TCP/IP is functioning on the local host, first ping the loopback address 127.0.0.1

```

C:\Users\IT HOD>ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\IT HOD>

```



### b).ping localhost

To test host name cache resolution, ping the name localhost. This is an alias for the loopback address

```
C:\Users\IT HOD>ping localhost

Pinging ITHOD-PC [::1] with 32 bytes of data:
Reply from ::1: time<1ms
Reply from ::1: time<1ms
Reply from ::1: time<1ms
Reply from ::1: time<1ms

Ping statistics for ::1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\IT HOD>
```

### c). ping <IPv4 address> (Ping the Host IPv4 Address)

To test the local host IPv4 address:

1. Use ipconfig to display the host IP address. Note the IPv4 Address displayed.

2. Type ping <IPv4 Address> where <IPv4 Address> is the IPv4 address displayed above. For example, if the IPv4 address was 192.168.1.101, you would type ping 192.168.1.101. Then press Enter.

```
C:\Users\IT HOD> ping 192.168.1.101

Pinging 192.168.1.101 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.101:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\Users\IT HOD>ping 184.168.221.39

Pinging 184.168.221.39 with 32 bytes of data:
Reply from 184.168.221.39: bytes=32 time=319ms TTL=53
Reply from 184.168.221.39: bytes=32 time=319ms TTL=53
Reply from 184.168.221.39: bytes=32 time=319ms TTL=53
Reply from 184.168.221.39: bytes=32 time=319ms TTL=53

Ping statistics for 184.168.221.39:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 319ms, Maximum = 319ms, Average = 319ms

C:\Users\IT HOD>_
```

### d).ping <hostname> (Ping the Host Name)

To test the local host name:

1. Use ipconfig /all to display the host name. Note the Host Name displayed.

2. Type ping <hostname> where <hostname> is the Host Name IPv4 address displayed above. For example, if the host name was host1, you would type ping host1. Then press Enter.

### e) . ping < default gateway address> (Ping the Default Gateway)

1. Use ipconfig to display the default gateway address. Note the Default Gateway displayed.

2. Type ping <default gateway address> where <default gateway address> is the default gateway address displayed above. For example, if the default gateway address was 192.168.1.1, you would type ping 192.168.1.1. Then press Enter.

**f). Ping an Internet Host by IPv4 Address**

8.8.8.8 is the IPv4 address of one of Google's public DNS servers. To test Internet connectivity:

1. Type ping 8.8.8.8 and press Enter.

**g). Ping an Internet Host by Name**

google-public-dns-a.google.com is the host name of one of Google's public DNS servers. To test Internet connectivity with host name resolution:

1. Type ping google-public-dns-a.google.com and press Enter.

**h). Ping an Internet Host by IPv6 Address**

ipv6.google.com is the IPv6-only host name of Google's web servers. To test Internet connectivity with IPv6 host name resolution:

1. Type ping ipv6.google.com and press Enter.

**i). Ping an Internet Host by IPv6 Name**

ipv6.google.com is the IPv6-only host name of Google's web servers. To test Internet connectivity with IPv6 host name resolution:

```
C:\Users\IT HOD>ipconfig/all

Windows IP Configuration

Host Name . . . . . : ITHOD-PC
Primary Dns Suffix . . . . . : 
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . : 
Description . . . . . : Realtek RTL8168B/8111B Family PCI-E Gigabit Ethernet NIC (NDIS 6.20)
Physical Address. . . . . : 00-1C-C0-85-81-17
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::20dc:3b80:28fe:343111<Preferred>
IPv4 Address. . . . . : 172.16.22.129<Preferred>
Subnet Mask . . . . . : 255.255.0.0
Default Gateway . . . . . : 172.16.16.16
DHCPv6 IAID . . . . . : 234888384
DHCPv6 Client DUID. . . . . : 00-01-00-01-09-68-19-88-00-1C-C0-85-81-17

DNS Servers . . . . . : 8.8.8.8
                       4.4.8.8
NetBIOS over Tcpip. . . . . : Enabled

Tunnel adapter isatap.{8F64F018-0896-4E4E-BBDF-8CCC0986AD83}:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Description . . . . . : Microsoft ISATAP Adapter
Physical Address. . . . . : 00-00-00-00-00-00-00-E0
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes

C:\Users\IT HOD>ping 172.16.16.16

Pinging 172.16.16.16 with 32 bytes of data:
Reply from 172.16.16.16: bytes=32 time<1ms TTL=64
Reply from 172.16.16.16: bytes=32 time<1ms TTL=64
Reply from 172.16.16.16: bytes=32 time<1ms TTL=64
Reply from 172.16.16.16: bytes=32 time<1ms TTL=64

Ping statistics for 172.16.16.16:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\IT HOD>
```

```
C:\Users\IT HOD>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=21ms TTL=56
Reply from 8.8.8.8: bytes=32 time=21ms TTL=56
Reply from 8.8.8.8: bytes=32 time=21ms TTL=56
Reply from 8.8.8.8: bytes=32 time=21ms TTL=56

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 21ms, Maximum = 21ms, Average = 21ms

C:\Users\IT HOD>ping google-public-dns-a.google.com

Pinging google-public-dns-a.google.com [8.8.8.8] with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=21ms TTL=56
Reply from 8.8.8.8: bytes=32 time=21ms TTL=56
Reply from 8.8.8.8: bytes=32 time=21ms TTL=56
Reply from 8.8.8.8: bytes=32 time=21ms TTL=56

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 21ms, Maximum = 21ms, Average = 21ms

C:\Users\IT HOD>ping ipv6.google.com
Ping request could not find host ipv6.google.com. Please check the name and try
again.

C:\Users\IT HOD>ping 2001:4860:4860::8888

Pinging 2001:4860:4860::8888 with 32 bytes of data:
PING: transmit failed. General failure.
PING: transmit failed. General failure.
PING: transmit failed. General failure.
PING: transmit failed. General failure.

Ping statistics for 2001:4860:4860::8888:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\Users\IT HOD>
```

**RESULT:**

## Ex.No.2

### HTTP WEB CLIENT PROGRAM TO DOWNLOAD A WEB PAGE USING TCP SOCKETS

#### AIM:

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

#### ALGORITHM:

1. Start the program
2. Read the file to be downloaded from webpage
3. To download an image, use java URL class which can be found under java.net package.
4. The file is downloaded from server and is stored in the current working directory.
5. Stop the program

#### PROGRAM:

##### Download.java

```
import java.io.*;
import java.net.URL;
public class Download
{
    public static void main(String[] args) throws Exception
    {
        try
        {
            //String fileName = "Assessment_Schedule_ND19.pdf";
            //String website =
            "https://coe1.annauniv.edu/aucoe/pdf/2019_nd"+fileName;
            //String fileName = "Rainbow_Rose_%283366550029%29.jpg";
            //String website =
            "https://upload.wikimedia.org/wikipedia/commons/6/61/"+fileName;
            String fileName = "JPEG_example_JPG_RIP_100.jpg";
            String website =
            "https://upload.wikimedia.org/wikipedia/commons/b/b4/"+fileName;
            System.out.println("Downloading File From: " + website);
            URL url = new URL(website);
            InputStream inputStream = url.openStream();
            OutputStream outputStream = new FileOutputStream(fileName);
            byte[] buffer = new byte[2048];
            int length = 0;
            while ((length = inputStream.read(buffer)) != -1)
```

```
    {  
        System.out.println("Buffer Read of length: " + length);  
        outputStream.write(buffer, 0, length);  
    }  
    inputStream.close();  
    outputStream.close();  
    }  
    catch(Exception e)  
    {  
        System.out.println("Exception: " + e.getMessage());  
    }  
}  
}
```

**OUTPUT:**

**RESULT:**

**Ex.No.3a****TCP - ECHO CLIENT AND ECHO SERVER****AIM:**

To write a program to implement an applications using TCP Sockets like echo client and echo server

**ALGORITHM:**

1. Start the Program
2. In Server
  - a) Create a server socket and bind it to port.
  - b) Listen for new connection and when a connection arrives, accept it.
  - c) Read the data from client.
  - d) Echo the data back to the client.
  - e) Close all streams.
  - f) Close the server socket.
  - g) Stop.
3. In Client
  - a)** Create a client socket and connect it to the server's port number.
  - b)** Send user data to the server.
  - c)** Display the data echoed by the server.
  - d)** Close the input and output streams.
  - e)** Close the client socket.
  - f)** Stop.
4. Stop the program

**PROGRAM:****SERVER:**

```
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdio.h>
#include<strings.h>
#define SERV_TCP_PORT 3785
int main(int argc,char *argv[])
```

```

{
int sockfd,new,clen;
struct sockaddr_in serv_add,cli_add;
char buffer[4096];
sockfd=socket(AF_INET,SOCK_STREAM,0);
serv_add.sin_family=AF_INET;
serv_add.sin_addr.s_addr=inet_addr("127.0.0.1");
serv_add.sin_port=htons(SERV_TCP_PORT);
bind(sockfd,(struct sockaddr*)&serv_add,sizeof(serv_add));
listen(sockfd,5);
clen=sizeof(cli_add);
new=accept(sockfd,(struct sockaddr*)&cli_add,&clen);
printf("ECHO SERVER connected...");
do
{
bzero(buffer,4096);
read(new,buffer,4096);
write(new,buffer,4096);
}while(strcmp(buffer,"bye\n")!=0);
close(sockfd);
return 0;
}

```

#### **CLIENT:**

```

#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<strings.h>
#define SERV_TCP_PORT 3785
int main(int argc,char*argv[])
{
int sockfd;
struct sockaddr_in serv_add;
char buffer[4096];
sockfd=socket(AF_INET,SOCK_STREAM,0);
serv_add.sin_family=AF_INET;
serv_add.sin_addr.s_addr=inet_addr("127.0.0.1");
serv_add.sin_port=htons(SERV_TCP_PORT);
connect(sockfd,(struct sockaddr*)&serv_add,sizeof(serv_add));
do{
bzero(buffer,4096);
printf("\n Client:");
fgets(buffer,4096,stdin);
write(sockfd,buffer,4096);
bzero(buffer,4096);
read(sockfd,buffer,4096);
}

```

```
printf("\n Echo Server  Message:%s",buffer);  
}while(strcmp(buffer,"bye\n")!=0);  
close(sockfd);  
return 0;  
}
```

**OUTPUT:**

**RESULT:**



**Ex.No.3b****TCP – CLIENT AND SERVER APPLICATION FOR CHAT****AIM:**

To write a program to implement a CHAT application using TCP Sockets.

**ALGORITHM:**

1. Start the Program
2. In Server
  - a) Create a server socket and bind it to port.
  - b) Listen for new connection and when a connection arrives, accept it.
  - c) Read Client's message and display it
  - d) Get a message from user and send it to client
  - e) Repeat steps 3-4 until the client sends "end"
  - f) Close all streams
  - g) Close the server and client socket
  - h) Stop
3. In Client
  - a) Create a client socket and connect it to the server's port number
  - b) Get a message from user and send it to server
  - c) Read server's response and display it
  - d) Repeat steps 2-3 until chat is terminated with "end" message
  - e) Close all input/output streams
  - f) Close the client socket
  - g) Stop
4. Stop the program

**PROGRAM:****SERVER:**

```
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdio.h>
#include<strings.h>
#define SERV_TCP_PORT 3771
int main(int argc,char*argv[])
{
```

```

int sockfd,new,clen;
struct sockaddr_in serv_add,cli_add;
char buffer[4096];
sockfd=socket(AF_INET,SOCK_STREAM,0);
serv_add.sin_family=AF_INET;
serv_add.sin_addr.s_addr=inet_addr("127.0.0.1");
serv_add.sin_port=htons(SERV_TCP_PORT);
bind(sockfd,(struct sockaddr*)&serv_add,sizeof(serv_add));
listen(sockfd,5);
clen=sizeof(cli_add);
new=accept(sockfd,(struct sockaddr*)&cli_add,&clen);
do
{
bzero(buffer,4096);
read(new,buffer,4096);
printf("\n Client message:%s",buffer);
bzero(buffer,4096);
printf("server:");
fgets(buffer,4096,stdin);
write(new,buffer,4096);
}
while(strcmp(buffer,"bye\n")!=0);
close(sockfd);
return 0;
}

```

#### **CLIENT:**

```

#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<strings.h>
#define SERV_TCP_PORT 3771
int main(int argc,char*argv[])
{
int sockfd;
struct sockaddr_in serv_add;
char buffer[4096];
sockfd=socket(AF_INET,SOCK_STREAM,0);
serv_add.sin_family=AF_INET;
serv_add.sin_addr.s_addr=inet_addr("127.0.0.1");
serv_add.sin_port=htons(SERV_TCP_PORT);
connect(sockfd,(struct sockaddr*)&serv_add,sizeof(serv_add));
do{
bzero(buffer,4096);
printf("\n Client:");
fgets(buffer,4096,stdin);

```

```
write(sockfd,buffer,4096);
bzero(buffer,4096);
read(sockfd,buffer,4096);
printf("\n server Message:%s",buffer);
}
while(strcmp(buffer,"bye\n")!=0);
close(sockfd);
return 0;
}
```

**OUTPUT:**

**RESULT:**

**Ex.No.3c**

## **TCP - CLIENT AND SERVER FILE TRANSFER**

### **AIM:**

To write a program to implement an applications using TCP Sockets like file transfer.

### **ALGORITHM:**

1. Start the program
2. In Client side:
  - a) Import all the necessary packages.
  - b) Create a client socket and connect to server using port & IP address.
  - c) Send a file request to server.
  - d) Then server responses are retrieved using input & output streams.
  - e) Close the socket.
3. At Server side:
  - a) Import all the necessary packages.
  - b) Create a client and server socket and accept client connection.
  - c) Transfer file that user requested to transfer using output stream.
  - d) Close the socket and trace the output.
4. Stop the program

### **PROGRAM**

#### **SERVER:**

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/types.h>
#define SERV_PORT 3322
int main(int argc,char **argv)
{
int i,j;
ssize_t n;
FILE*fp;
char s[80],f[80];
```

```

struct sockaddr_in servaddr,cliaddr;
int listenfd,connfd,clilen;
listenfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(SERV_PORT);
bind(listenfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
listen(listenfd,1);
clilen=sizeof(cliaddr);
connfd=accept(listenfd,(struct sockaddr*)&cliaddr,&clilen);
printf("\n Client Connected");
read(connfd,f,80);
fp=fopen(f,"r");
printf("\nName of the requested file:%s",f);
printf("\nContent of the requested file\n");
while(fgets(s,80,fp)!=NULL)
{
printf("%s",s);
write(connfd,s,sizeof(s));
}
}

```

#### **CLIENT:**

```

#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/types.h>
#define SERV_PORT 3322
int main(int argc,char **argv)
{
int i,j;
ssize_t n;
char filename[80],recvline[80];
struct sockaddr_in servaddr;
int sockfd;
sockfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(SERV_PORT);
inet_pton(AF_INET,argv[1],&servaddr.sin_addr);
connect(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
printf("Enter the file name");
scanf("%s",filename);
write(sockfd,filename,sizeof(filename));
printf("\n Data of the file from server:\n");

```

```
while(read(sockfd,recvline,80)!=0)
{
puts(recvline);
}
```

**OUTPUT:**

**RESULT:**

**Ex.No.4****SIMULATION OF DNS USING UDP SOCKETS****AIM:**

To write a program to implement the DNS application using UDP Sockets.

**ALGORITHM:**

1. Start the Program
2. In Server
  - a) Create an array of hosts and its ip address in another array
  - b) Create a datagram socket and bind it to a port
  - c) Create a datagram packet to receive client request
  - d) Read the domain name from client to be resolved
  - e) Lookup the host array for the domain name
  - f) If found then retrieve corresponding address
  - g) Create a datagram packet and send ip address to client
  - h) Repeat steps 3-7 to resolve further requests from clients
  - i) Close the server socket
  - j) Stop
3. In Client
  - a) Create a datagram socket
  - b) Get domain name from user
  - c) Create a datagram packet and send domain name to the server
  - d) Create a datagram packet to receive server message
  - e) Read server's response
  - f) If ip address then display it else display "Domain does not exist"
  - g) Close the client socket
  - h) Stop
4. Stop the program

**PROGRAM:****SERVER :**

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<fcntl.h>
#include<arpa/inet.h>
int main()
{
int std,i,len,port;
char content[100],str[100];
char
```

```

ip[50][50]={"127.0.0.1","127.0.0.2","127.0.0.3","127.0.0.4","127.0.0.5"};
char eth[50][50]={"google","yahoo","rediff","rocket","hotmail"};
struct sockaddr_in ser,cli;
printf("\n Enter your port number");
scanf("%d",&port);
if((std=socket(AF_INET,SOCK_DGRAM,0))<0)
{
printf("\n Socket not created");
exit(0);
}
bzero((char*)&ser,sizeof(ser));
bzero((char*)&cli,sizeof(cli));
ser.sin_family=cli.sin_family=AF_INET;
ser.sin_port=cli.sin_port=htons(port);
ser.sin_addr.s_addr=cli.sin_addr.s_addr=htons(INADDR_ANY);
i=bind(std,(struct sockaddr*)&ser,sizeof(ser));
if(i<0)
{
printf("\n Connection problem\n");
exit(0);
}
len=sizeof(cli);
while(1)
{
printf("\n Received from client");
recvfrom(std,content,100,0,(struct sockaddr*)&cli,&len);
printf("%s/n",content);
if(strcmp(content,"bye")==0)
break;
for(i=0;i<5;i++)
{
if(!strcmp(content,ip[i]))
{
strcpy(str,"");
printf("\n Received");
strcat(str,"ethernet name is:");
strcat(str,eth[i]);
sendto(std,str,100,0,(struct sockaddr*)&cli,len);
}
else
if(!strcmp(content,eth[i]))
{
strcpy(str,"");
printf("\n Received");
strcat(str,"IP address is:");
strcat(str,ip[i]);
sendto(std,str,100,0,(struct sockaddr*)&cli,len);
}
}
}

```



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

#### **CLIENT:**

```
#include<stdio.h>  
#include<stdlib.h>  
#include<string.h>  
#include<sys/socket.h>  
#include<sys/types.h>  
#include<netinet/in.h>  
#include<fcntl.h>  
int main()  
{  
int std,i,len,port;  
char content[100];  
struct sockaddr_in ser,cli;  
printf("\n Enter your port number");  
scanf("%d",&port);  
if((std=socket(AF_INET,SOCK_DGRAM,0))<0)  
{  
printf("\n socket not created");  
exit(0);  
}  
bzero((char*)&ser,sizeof(ser));  
bzero((char*)&cli,sizeof(cli));  
ser.sin_family=cli.sin_family=AF_INET;  
ser.sin_port=cli.sin_port=htons(port);  
ser.sin_addr.s_addr=cli.sin_addr.s_addr=htons(INADDR_ANY);  
len=sizeof(ser);  
for(;;)  
{  
printf("\n Enter ip address\n");  
scanf("\n %s",content);  
sendto(std,content,100,0,(struct sockaddr*)&ser,len);  
printf("\n sent to server\n");  
printf("%s\n",content);  
if(strcmp(content,"bye")==0)  
break;  
recvfrom(std,content,100,0,(struct sockaddr*)&ser,&len);  
printf("\nReceive from server\n");  
printf("\n%s",content);  
}  
close(std);  
return(0);  
}
```

**OUTPUT:**

**RESULT:**

**Ex.No.5****WIRESHARK TOOL TO CAPTURE PACKETS****AIM:**

To Capture ping and traceroute PDUs using a network protocol analyzer and examine.

**Network protocol analyzer - Wireshark**

Wireshark is free & Open source network packet analyzer that is used for network analysis, troubleshooting, etc.

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets.

**Installation on CentOS:**

```
# yum install gcc gcc-c++ bison flex libpcap-devel qt-devel gtk3-devel rpm-  
build libtool c-ares-devel qt5-qtbase-devel qt5-qtmultimedia-devel qt5-linguist  
desktop-file-utils
```

```
# yum install wireshark wireshark-qt
```

```
# yum install wireshark-gnome.
```

**To Open Wireshark**

Open directly or use the following commands

```
#su -root
```

```
# wireshark
```

In wireshark filter icmp packets

In a konsole execute

```
# ping www.sudo.com
```

```
# traceroute www.google.com
```

**Installation on Windows:**

1. Download the wireshark package for 32 bit – windows os.
2. Copy the downloaded package in the Program Files folder of C drive,
3. Execute the .exe file
- 4.

**Testing Procedure:**

Use Wireshark to Open the file for analysis (In Wireshark window)

1. File > Open > C:\perflogs > diagTraces0
2. Capture > options > select (choose or type) the filter type( tcp, udp ....)

Capturing from Local Area Connection

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-F> Expression...

No.	Time	Source	Destination	Protocol	Length	Info
6739	74.579415	IntelCor_2d:af:e9	Broadcast	ARP	60	Who has 172.16.27.161? Tell 172.16.11.102
6740	74.599461	172.16.22.108	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
6741	74.630085	Pegatron_0b:ad:bf	Broadcast	ARP	60	Who has 172.16.21.4? Tell 172.16.21.47
6742	74.630129	IntelCor_2d:af:e9	Broadcast	ARP	60	Who has 172.16.28.161? Tell 172.16.11.102
6743	74.633383	Pegatron_c8:ff:bf	Broadcast	ARP	60	Who has 172.16.21.44? Tell 172.16.22.108
6744	74.634713	Pegatron_0c:5d:65	Broadcast	ARP	60	Who has 172.16.22.108? Tell 172.16.21.44
6745	74.655143	Pegatron_0c:5d:65	Broadcast	ARP	60	Who has 172.16.244.65? Tell 172.16.21.44
6746	74.705167	Pegatron_0c:5d:65	Broadcast	ARP	60	Who has 172.16.245.65? Tell 172.16.21.44
6747	74.736594	172.16.22.129	40.100.138.2	TCP	54	49391 → 443 [ACK] Seq=1 Ack=177 Win=16117 Len=0

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0  
 ▶ Ethernet II, Src: IntelCor\_2d:af:e9 (00:19:d1:2d:af:e9), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
 ▶ Address Resolution Protocol (request)

```

0000  ff ff ff ff ff ff 00 19 d1 2d af e9 08 06 00 01  .....
0010  08 00 06 04 00 01 00 19 d1 2d af e9 ac 10 0b 66  .....
0020  00 00 00 00 00 00 ac 10 15 9e 00 00 00 00 00  .....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
  
```

Local Area Connection: <live capture in progress>      Packets: 6747 • Displayed: 6747 (100.0%)      Profile: Defa

12:20 PM  
10/29/2019

**RESULT:**

**Ex.No.6****SIMULATION OF ARP /RARP PROTOCOLS****AIM:**

To write a program to simulate ARP/RARP protocols.

**ALGORITHM:**

1. Start the Program
2. Enter the list of IP addresses and its corresponding MAC (Ethernet) addresses.
3. Perform the Address Translations.
  - a. If ARP, then enter IP address it returns the corresponding MAC Address
  - b. If RARP, then enter MAC address, it returns the corresponding IP address
4. Stop the program

**PROGRAM:**

```
#include<stdio.h>
#include<string.h>
int main()
{
int i,a,flag=0;
char
ip1[5][32]={ "192.168.0.5","192.168.0.6","192.168.0.8","192.168.0.9"};
char
mac1[5]
[48]={ "2c:41:38:9a:75:f5","2c:41:38:9a:75:f6","2c:41:38:9a:75:f7","2c:41:38:9a:75:f8","2c:
41:38:9a:75:f9"};
char ip[32],mac[48];
printf("\n The ARP table is \n");
printf("\n IP address\t MAC address");
for(i=0;i<5;i++)
{
printf("\n%s\t%s",ip1[i],mac1[i]);
}
printf("\n 1.ARP 2.RARP\n EXIT\n Enter your choice:");
scanf("%d",&a);
switch(a)
{
case 1:
printf("\n Enter Ip address:");
scanf("%s",&ip);
for(i=0;i<5;i++)
{
if(strcmp(ip,ip1[i])==0)
{
printf("\n mac addr is %s",mac1[i]);
printf("\n");
flag=1;
}
```

```
}  
}  
if(flag==0)  
printf("\n Host not found in ARP table");  
break;  
case 2:  
printf("\n Enter MAC address:");  
scanf("%s",&mac);  
for(i=0;i<5;i++)  
{  
if(strcmp(mac,mac1[i])==0)  
{  
printf("\n IP addr is %s",ip1[i]);  
printf("\n");  
flag=1;  
}  
}  
if(flag==0)  
printf("\n Host not found in ARP table");  
break;  
case 3:  
printf("EXIT");  
}  
return 0;  
}
```

**OUTPUT:**

**RESULT:**

## Ex.No.7a

### STUDY OF NETWORK SIMULATOR (NS)

#### AIM:

To Study the Network Simulator (NS) and the Simulation of Congestion Control Algorithms Using NS.

#### INTRODUCTION:

Network Simulator (Version 2), widely known as NS2, is simply an eventdriven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator, 1 the foundation which NS is based on.

Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual InterNetwork Testbed (VINT) project currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of Researchers and developers in the community are constantly working to keep NS2 strong and versatile.

#### BASIC ARCHITECTURE:

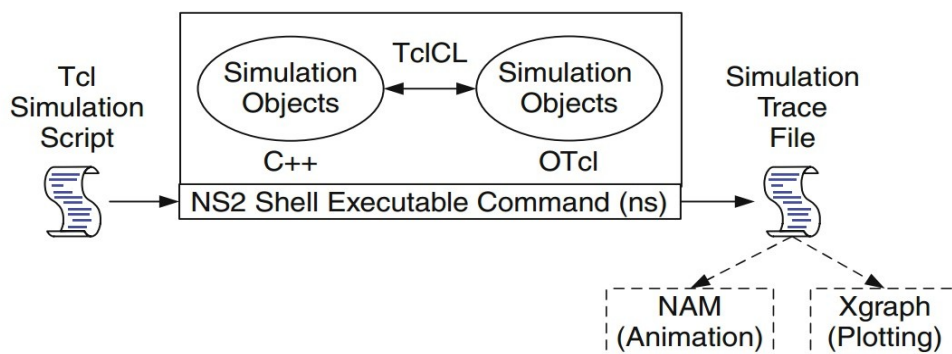


Fig. 2.1. Basic architecture of NS.

Figure shows the basic architecture of NS2. NS2 provides users with executable command ns which take on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns.

In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend).

The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g., \_o10) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may define its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively. Before proceeding further, the readers are encouraged to learn C++ and OTcl languages.

NS2 provides a large number of built-in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects insufficient. They need to develop their own C++ objects, and use a OTcl configuration interface to put together these objects. After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behavior of the network, users can extract a relevant subset of text-based data and transform it to a more conceivable presentation.



## CONCEPT OVERVIEW

ns uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols require a systems programming language which can efficiently manipulate bytes, packet headers and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring number of scenarios.

In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. ns meets both of these needs with two languages, C++ and OTcl.

## BASIC COMMANDS IN NS2

- Create event scheduler
  - **set ns [new Simulator]**
- Trace packets on all links
  - **set nf [open out.nam w]**
  - **\$ns trace-all \$nf**
  - **\$ns namtrace-all \$nf**
- Nodes
  - **set n0 [\$ns node]**
  - **set n1 [\$ns node]**
- Links and queuing
  - **\$ns duplex-link \$n0 \$n1 <bandwidth> <delay> <queue\_type>**
  - **<queue\_type>: DropTail, RED, etc.**
  - **\$ns duplex-link \$n0 \$n1 1Mb 10ms RED**

- Creating a larger topology

```

for {set i 0} {$i < 7} {incr i} {
    set n($i) [$ns node]
}
for {set i 0} {$i < 7} {incr i} {
    $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms RED
}

```

- Link failures

- **\$ns rtmodel-at <time> up|down \$n0 \$n1**

- Creating UDP connection

- **set udp [new Agent/UDP]**
- **set null [new Agent/Null]**
- **\$ns attach-agent \$n0 \$udp**
- **\$ns attach-agent \$n1 \$null**
- **\$ns connect \$udp \$null**

- Creating Traffic (On Top of UDP)

- **set cbr [new Application/Traffic/CBR]**
- **\$cbr set packetSize\_ 500**
- **\$cbr set interval\_ 0.005**
- **\$cbr attach-agent \$udp**

- Post-Processing Procedures

```

proc finish {}
{
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}

```

- Schedule Events
  - **\$ns at <time> <event>**
- Call 'finish'
  - **\$ns at 5.0 "finish"**
- Run the simulation
  - **\$ns run**

#### INSTALLATION PROCEDURE FOR NS2:

1. Install winrar
2. Unzip NS2.35 Part-I (Extract here)
3. It creates a folder NS2.35. Open the folder and run the setup file.
  - a. Click next
  - b. Select Install from Local directory and click next
  - c. Select Path C:\cygwin and click next
  - d. In Select local package change the path as NS2.35\nslocal\release and click next
  - e. Installation continues for 15 to 20 min
  - f. Select Create icon on Desktop and Add icon to start menu and click finish.
4. Run cygwin from desktop and then close cygwin.
5. In C:\cygwin\home create a new folder in the name Nouredine
6. Now extract the file ns-allinone-2.35-RC7avecgraph available in NS2.35. It creates a folder ns-allinone-2.35-RC7.
7. Copy the ns-allinone-2.35-RC7 folder and paste with in C:\cygwin\home\Nouredine
8. Now copy the .bashrc file available in NS2.35 and paste in C:\cygwin\home\admin by replacing the already existing .bashrc file
9. Open cygwin from desktop and execute the following commands:
 

```
admin@admin-PC ~
$ ns
% ns-version
2.35-RC6
% exit
```
10. Copy NSG2.1 and nam-1.0a11a-win32 from NS2 Installation file and paste in desktop.

```
admin@admin-PC ~
$ cd ..
admin@admin-PC /home
$ cd ..
```

```
admin@admin-PC /  
$ ls  
Cygwin.bat  Cygwin.ico  sampleprogram  bin  cygdrive  dev  etc  home  lib  opt  
proc  tmp  usr  var  
admin@admin-PC /  
$ cd sampleprogram  
admin@admin-PC /sampleprogram  
$ns sample.tcl
```

**RESULT:**

## Ex No. 7b SIMULATION OF CONGESTION CONTROL ALGORITHMS USING NS

### AIM:

To study the simulation of Congestion Control Algorithm using NS2.

### ALGORITHM:

1. Run NSG 2.1
2. Create two nodes n0 to n5.
3. Create a duplex-link and set the following parameters
  - a. Queue type : Droptail
  - b. Capacity 0.2 Mbps
  - c. Propagation delay: 200 ms
  - d. Queue size: 10
4. Create link between the nodes n0 to n5.
  - a. Set node n0 as source (Reno agent)
  - b. Set node n4 as sink
5. In Application tab do the following.
  - a. Application Type: ftp
  - b. Start time: 1
  - c. Stop time: 40
6. In parameters tab do the following
  - a. Simulation time: 50
  - b. Data record file: **congestion.xg**
  - c. Click done
7. Click TCL and save the file as congestion.tcl in C:\cygwin\Sampleprogram
8. Run the congestion.tcl file
9. Verify the congestion.xg file using Xgraph command

## **PROGRAM:**

### **#create simulator**

```
set ns [new Simulator]
```

### **#to create nodes**

```
set n0 [$ns node]
```

```
set n1 [$ns node]
```

```
set n2 [$ns node]
```

```
set n3 [$ns node]
```

```
set n4 [$ns node]
```

```
set n5 [$ns node]
```

### **# to create the link between the nodes with bandwidth, delay and queue**

```
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
```

```
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
```

```
$ns duplex-link $n2 $n3 0.3Mb 200ms DropTail
```

```
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
```

```
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail
```

### **# Sending node is 0 with agent as Reno Agent**

```
set tcp1 [new Agent/TCP/Reno]
```

```
$ns attach-agent $n0 $tcp1
```

### **# receiving (sink) node is n4**

```
set sink1 [new Agent/TCPSink]
```

```
$ns attach-agent $n4 $sink1
```

### **# establish the traffic between the source and sink**

```
$ns connect $tcp1 $sink1
```

### **# Setup a FTP traffic generator on "tcp1"**

```
set ftp1 [new Application/FTP]
```

```
$ftp1 attach-agent $tcp1
```

```
$ftp1 set type_ FTP
```

### **# start/stop the traffic**

```
$ns at 0.1 "$ftp1 start"
```

```
$ns at 40.0 "$ftp1 stop"
```

```

# Set simulation end time
$ns at 50.0 "finish"

# procedure to plot the congestion window
proc plotWindow {tcpSource outfile} {
    global ns
    set now [$ns now]
    set cwnd [$tcpSource set cwnd_]
    # the data is recorded in a file called congestion.xg (this can be plotted # using
    # xgraph or #gnuplot. this example uses xgraph to plot the cwnd_
    puts $outfile "$now $cwnd"
    $ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"
}
set outfile [open "congestion.xg" w]
$ns at 0.0 "plotWindow $tcp1 $outfile"
proc finish {} {
    exec xgraph congestion.xg -geometry 300x300 &
    exit 0
}
# Run simulation
$ns run

```

**OUTPUT:**

**RESULT:**

## **Ex. No. 8      STUDY OF TCP/UDP PERFORMANCE USING SIMULATION TOOL**

### **AIM:**

To study the performance of TCP/UDP using simulation tool NS2.

### **ALGORITHM:**

1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create 4 number of nodes
5. Create duplex links between the nodes and set Queue size
6. Setup TCP connection between node n0 ( Source) and n3 (Sink)
7. Apply FTP Traffic over TCP connection.
8. Setup UDP Connection between n1 and n3
9. Apply CBR Traffic over UDP connections
10. Schedule events (Start and Stop) for the TCP and UDP connections
11. Set the simulation time. And write command to print CBR packet size and CBR interval.
12. Run the program. Run the awk file to display the total number of dropped packets from out1.tr . Open and run the out.nam file.

### **Program:**

**#Create a simulator object**

```
set ns [new Simulator]
```

**#Define different colors for data flows (for NAM)**

```
$ns color 1 Blue
```

```
$ns color 2 Red
```

**#Open the NAM trace file**

```
set nf [open out1.nam w]
```

```
$ns namtrace-all $nf
```

**#Open the Trace file**

```
set tf [open out1.tr w]
```



```
$ns trace-all $tf
#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the NAM trace file
    close $nf
    #Execute NAM on the trace file
    exec nam out.nam &
    exit 0
}
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5

#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

**#Setup a UDP connection**

```
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
```

**#Setup a CBR over UDP connection**

```
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
```

**#Schedule events for the CBR and FTP agents**

```
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
```

**#Detach tcp and sink agents (not really necessary)**

```
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"
```

**#Call the finish procedure after 5 seconds of simulation time**

```
$ns at 5.0 "finish"
```

**#Print CBR packet size and interval**

```
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"
```

**#Run the simulation**

```
$ns run
```

**#Trace11.awk program (To display the total number of dropped packets from trace file**

```
BEGIN {
c=0;
}
{
if ($1=="d")
{
c++;
/*printf("%s\t%s\n",$5,$11);*/
}
}
}
```

```
END{  
printf("The number of packets dropped =%d\n",c);  
}
```

**OUTPUT:**

**RESULT:**

**Ex.No.9.a****SIMULATION OF DISTANCE VECTOR ROUTING ALGORITHM****AIM:**

To perform the Simulation of Distance Vector Routing algorithm.

**ALGORITHM:**

1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create 12 number of nodes using for loop
5. Create duplex links between the nodes
6. Setup UDP Connection between the nodes
7. Apply CBR Traffic over UDP connections
8. Choose distance vector routing protocol to transmit data from sender to receiver.
9. Schedule events and run the program.

**PROGRAM (DISTANCE VECTOR PROTOCOL)**

```
set ns [new Simulator]
set nr [open thro.tr w]
$ns trace-all $nr
set nf [open thro.nam w]
$ns namtrace-all $nf
proc finish { } {
    global ns nr nf
    $ns flush-trace
    close $nf
    close $nr
    exec nam thro.nam &
    exit 0
}
for { set i 0 } { $i < 12 } { incr i 1 } {
```

```
set n($i) [$ns node]}
for {set i 0} {$i < 8} {incr i} {
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }

$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail

set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp0 $null0

set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0

$ns rtproto DV

$ns rtmodel-at 10.0 down $n(11) $n(5)
$ns rtmodel-at 15.0 down $n(7) $n(6)

$ns rtmodel-at 30.0 up $n(11) $n(5)
$ns rtmodel-at 20.0 up $n(7) $n(6)
```

```
$udp0 set fid_ 1
$udp1 set fid_ 2
$ns color 1 Red
$ns color 2 Green
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 45 "finish"
$ns run
```

**OUTPUT:**

**RESULT:**

**Ex.No.9.b****SIMULATION OF LINK STATE ROUTING ALGORITHM****AIM:**

To perform the Simulation of Link State Routing algorithm.

**ALGORITHM:**

1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create 12 number of nodes using for loop
5. Create duplex links between the nodes
6. Setup UDP Connection between the nodes
7. Apply CBR Traffic over UDP connections
8. Choose Link state routing protocol to transmit data from sender to receiver.
9. Schedule events and run the program.

**PROGRAM (LINK STATE PROTOCOL)**

```
set ns [new Simulator]
set nr [open link.tr w]
$ns trace-all $nr
set nf [open link.nam w]
$ns namtrace-all $nf
proc finish { } {
    global ns nr nf
    $ns flush-trace
    close $nf
    close $nr
    exec nam link.nam &
    exit 0
}
for { set i 0 } { $i < 12 } { incr i 1 } {
```

```
set n($i) [$ns node]}

for {set i 0} {$i < 8} {incr i} {
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }
$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail

set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp0 $null0

set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0

$ns rtproto LS
$ns rtmodel-at 10.0 down $n(11) $n(5)
$ns rtmodel-at 15.0 down $n(7) $n(6)
$ns rtmodel-at 30.0 up $n(11) $n(5)
$ns rtmodel-at 20.0 up $n(7) $n(6)

$udp0 set fid_ 1
$udp1 set fid_ 2
```



```
$ns color 1 Red  
$ns color 2 Green
```

```
$ns at 1.0 "$cbr0 start"  
$ns at 2.0 "$cbr1 start"  
$ns at 45 "finish"  
$ns run
```

**OUTPUT:**

**RESULT:**

**Ex.No.10****SIMULATION OF ERROR CORRECTION CODE (LIKE CRC)****AIM:**

To simulate the Cyclic Redundancy Code(CRC) encoder and decoder.

**ALGORITHM:**

1. Start the program.
2. Perform the CRC simulation.
  - a. If CRC Encoder, enter the dataword length, dataword, the divisor length and divisor.
    1. Concatenate the dataword with (divisor length -1 ) number of zeros.
    2. Perform modulo2 division (i.e step 1 result divided by divisor).
    3. (divisor length -1 ) number of remainder Least significant digits is considered to be CRC. The codeword is generated by concatenating the dataword with CRC.
  - b. If CRC Decoder, enter the codeword length, codeword, the divisor length and divisor.
    1. Perform modulo2 division(i.e codeword divided by divisor)
    2. If the remainder is zero, then the received codeword has no error. Otherwise the received codeword has error.
3. Stop the program

**PROGRAM:**

```
#include<stdio.h>
int * mod2div(int n[], int div[], int clength, int divlength);
int f[20];
void main()
{
int j,i,a,temp2,clength,dlength,divlength,c[20],n[20],div[50];
a=temp2=0;
int *f;

printf("\n 1.CRC ENCODER 2.CRC DECODER \n EXIT\n Enter your choice:");
scanf("%d",&a);
switch(a)
{
case 1:
```

```

printf("enter the length of dataword\n");
scanf("%d",&dlength);

printf("enter the data word\n");
for(i=0;i<dlength;i++)
{
scanf("%d",&n[i]);
}
for(i=0;i<dlength;i++)
{
c[i]=n[i];
}
printf("enter the length of divisor\n");
scanf("%d",&divlength);
printf("enter the divisor\n");
for(i=0;i<divlength;i++)
{
scanf("%d",&div[i]);
}
clength=dlength+divlength-1;
for(i=dlength;i<clength;i++)
{
n[i]=0;
}
printf("\nthe data with zeros is \n");
for(i=0;i<clength;i++)
printf("%d",n[i]);
f=mod2div(n,div,clength,divlength);
for(i=dlength;i<clength;i++)
{
c[i]=n[i];
}
printf("\n The codeword is \n ");
for(i=0;i<clength;i++)
{
printf("%d",c[i]);
}
break;
case 2:
printf("enter the length of codeword\n");
scanf("%d",&clength);
printf("enter the code word\n");
for(i=0;i<clength;i++)
{
scanf("%d",&n[i]);
}
printf("enter the length of divisor\n");
scanf("%d",&divlength);

```

```

printf("enter the divisor\n");
for(i=0;i<divlength;i++)
{
scanf("%d",&div[i]);
}
f=mod2div(n,div,clength,divlength);
for(j=0;j<divlength;j++)
{
temp2=temp2+f[j];
}
if(temp2==0)
printf("\n No error in received code");
else
printf("\n error occurs in received code");
break;
case 3:
printf("EXIT");
}
return 0;
}
// function description mod2div
int * mod2div(int n[], int div[], int clength, int divlength)
{
int i,M,temp,dlength,j,quot[20];
dlength=clength-divlength+1;

for(i=0;i<dlength;i++)
{
temp=i;
if(n[i]==1)
{
quot[i]=1;
for (j=0;j<divlength;j++)
{
M=n[temp] ^ div[j];
n[temp]=f[j]=M;
temp=temp+1;
}
}
else
{
quot[i]=0;
for(j=0;j<divlength;j++)
f[j]=n[i+j];
}
}
printf("\n and the quotient is \n ");
for(j=0;j<dlength;j++)

```

```
printf("%d",quot[j]);

printf("\n and the remainder is \n ");
for(j=0;j<divlength;j++)
printf("%d",f[j]);
return f;
}
```

**OUTPUT:**

**RESULT:**

# **CS3591 – COMPUTER NETWORKS**

## **VIVA QUESTIONS**

### **1. What is Computer Networks?**

A computer network is a group of interconnected nodes or computing devices that exchange data and resources with each other. A network connection between these devices can be established using cable or wireless media.

### **2. What is the use of arp -a?**

It will show the IP address of your computer along with the IP address and MAC address of your router.

### **3. What is the use of netstat?**

Netstat displays a variety of statistics about a computer's active TCP/IP connections. This tool is most useful when you're having trouble with TCP/IP applications such as HTTP, and FTP.

### **4. What is the use of ipconfig?**

The ipconfig command displays information about the host (the computer you're sitting at) computer TCP/IP configuration.

### **5. What is the use of ping?**

The Ping command's main purpose is to place a phone call to another computer on the network, and request an answer. Ping has 2 options it can use to place a phone call to another computer on the network. It can use the computer's name or IP address.

### **6. What is the use of tracert?**

The tracert command displays a list of all the routers that a packet has to go through to get from the computer where tracert is run to any other computer on the internet.

### **7. What is meant by socket?**

A socket is a software object that acts as an end point establishing a bidirectional network communication link between a server-side and a client-side program.

### **8. Define HTTP.**

HTTP means HyperText Transfer Protocol. HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

### **9. What is meant by client server communication?**

Client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

### **10. Define webpage.**

A web page or webpage is a document commonly written in HTML (Hypertext Markup Language) that is accessible through the Internet or other networks using an Internet browser.

### **11 Define URL.**

URL stands for Uniform Resource Locator, and is used to specify addresses on the World Wide Web. A URL is the fundamental network identification for any resource connected to the web (e.g., hypertext pages, images, and sound files). The protocol specifies how information from the link is transferred.

### **12. What is meant by socket programming?**

Socket programs are used to communicate between various processes usually running on different systems. It is mostly used to create a client-server environment.

### **13. what is the use of java.net?**

The java.net package provides 4 kinds of Sockets: Socket is a TCP client API, and will typically be used to connect to a remote host. ServerSocket is a TCP server API, and will typically accept connections from client sockets. DatagramSocket is a UDP endpoint API and is used to send and receive datagram packets.

### **14. Define echo.**

Echo describes when data is sent to a computer or other network devices, and that information is sent back to verify the information was received. Echo can be a command used with operating systems, network devices, or a function of a device.

### **15. Define chat.**

A real-time communication via keyboard between two or more users on a local network (LAN) or over the Internet

### **16. Define TCP.**

TCP/IP stands for Transmission Control Protocol/Internet Protocol, which is a set of networking protocols that allows two or more computers to communicate. The Defense Data Network, part of the Department of Defense, developed TCP/IP, and it has been widely adopted as a networking standard.

**17. What is socket API?**

A socket API is an application programming interface (API), usually provided by the operating system, that allows application programs to control and use network sockets. Internet socket APIs are usually based on the Berkeley sockets standard.

**18. What is socket address?**

A socket address is the combination of an IP address and a port number, much like one end of a telephone connection is the combination of a phone number and a particular extension. Based on this address, internet sockets deliver incoming data packets to the appropriate application process or thread.

**19. What are the local socket address?**

Local socket address: Local IP address and port number

**20. Define File transfer.**

File transfer is the process of copying or moving a file from one computer to another over a network or Internet connection. It enables sharing, transferring or transmitting a file or a logical data object between different users and/or computers both locally and remotely.

**21. what is called as socket pairs?**

Communicating local and remote sockets are called socket pairs .

**22. Define DNS.**

DNS. (Domain Name System) The Internet's system for converting alphabetic names into numeric IP addresses. For example, when a Web address (URL) is typed into a browser, DNSservers return the IP address of the Web server associated with that name.

**23. Define UDP.**

User Datagram Protocol (UDP) is part of the Internet Protocol suite used by programs running on different computers on a network. UDP is used to send short messages called datagrams but overall, it is an unreliable, connectionless protocol.

**24. what is meant by MACaddress?**

A MAC address is a hardware identification number that uniquely identifies each device on a network.

**25. What is meant by IP address?**

An Internet Protocol address (IP address) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two principal functions: host or network interface identification and location addressing.



**26. List the IP address class.**

Class	1 st Octet Decimal	Range Default Subnet Mask
A	1 – 126*	255.0.0.0
B	128 – 191	255.255.0.0
C	192 – 223	255.255.255.0
D	224 – 239	Reserved for Multicasting

**27. Define ARP.**

The address resolution protocol (arp) is a protocol used by the Internet Protocol (IP) [RFC826], specifically IPv4, to map IP network addresses to the hardware addresses used by a data link protocol.

**28. Define IPv4.**

IPv4 (Internet Protocol Version 4) is the fourth revision of the Internet Protocol (IP) used to identify devices on a network through an addressing system.

**29. Define RARP.**

RARP (Reverse Address Resolution Protocol) is a protocol by which a physical machine in a local area network can request to learn its IP address from a gateway server's Address Resolution Protocol (ARP) table .

**30. What is the use of RARP?**

RARP is used for the opposite purpose; namely, to convert an ethernet address to an IP address. Its linux-network-administration purpose is to enable diskless machines, such as X workstations, to find out their IP address at boot time.

**31. Define NS.**

A Network Simulator is an object oriented, event driven discrete simulator written in C++, with an object oriented tool command language (OTCL) interpreter as a front end.

**32. What protocols does NS support?**

A lot! Almost all variants of TCP, several forms of multicast, wired networking, several ad hoc routing protocols and propagation models (but not cellular phones), data diffusion, satellite, and other stuff. See the documentation (described above) for details, or download ns and look.

**33. Define data communication.**

Data communication is a term referred when the sender and receiver are digital devices, which communicate with each other by means of binary information.

#### **34. List of Network Simulators:**

- Ns2 (Network Simulator 2).
- Ns3 (Network Simulator 3).
- OPNET.
- OMNeT++.
- NetSim.
- REAL.
- QualNet.
- J-Sim.

#### **35. Define OPNET.**

OPNET Network Simulator. OPNET Network simulator is a tool to simulate the behavior and performance of any type of network. The main difference Opnet Network Simulator comparing to other simulators lies in its power and versatility. IT Guru provides pre-built models of protocols and devices .

#### **36. What is distance vector routing protocol?**

In distance vector routing the least cost route between any two nodes is the route with minimum distance. In this each node maintains a vector table of minimum distances to every node. The table at each node also guides the packets to the desired node by showing the next stop in the route.

#### **37. What is advantage in distance vector routing protocol?**

In this each node shares its routing table with its immediate neighbors periodically and when there is a change.

#### **38. What are the routing protocols?**

- Intra domain routing protocols
- Interdomain routing protocols

#### **39. What is intradomain routing protocols?**

Routing inside an autonomous system is referred to as intradomain routing protocols.

#### **40. What is RIP?**

RIP(routing information protocol) is an intradomain routing protocol used inside an autonomous system. it is very simple protocol based on distance vector routing .

**41. What is link state routing protocol?**

It has a different concept from that of distance vector routing. In this each node in the domain has the entire topology of the domain-the list of nodes and links, how they are connected including the type, cost and condition of the links.

**42. What is other name of link state routing protocol?**

Dijkstra's algorithm.

**43. What is advantage of link state routing protocol?**

The link state packet can carry a large amount of information. It has topology

**44. How the routing table is builded?**

By formation of a shortest path tree for each node. Calculation of a routing table based on the shortest path tree. Creation of the states of the links by each node.

**45. What is periodic update?**

A node sends its routing table, normally every 30's in a periodic update. The period depends on the protocol that is using distance vector routing.

**46. Define CRC.**

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.

**47. Define error detection.**

Error detection is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver. Error correction is the detection of errors and reconstruction of the original, error-free data. 3. Define error correcting.

**48. List error detecting methods.**

There are three main techniques for detecting errors in frames: Parity Check, Checksum and Cyclic Redundancy Check (CRC).

**49. Define parity check.**

A parity check is the process that ensures accurate data transmission between nodes during communication. A parity bit is appended to the original data bits to create an even or odd bit number; the number of bits with value one.

**50. Define Checksum.**

Checksum is a simple error detection mechanism to determine the integrity of the data transmitted over a network. Communication protocols like TCP/IP/UDP implement this scheme in order to determine whether the received data is corrupted along the network.