DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MANIPAL INSTITUTE OF TECHNOLOGY
Manipal University, Manipal-576104

LAB MANUAL

Department : **COMPUTER SCIENCE & ENGINEERING**

Subject : **CSE-310 COMPUTER NETWORKS LAB**
PROGRAMMING LAB

Semester & branch : **VI CSE**

No of contact hours/week : **3**

Submitted by : **Sri Manamohana K**

(Signature of the faculty)
Date:12-01-2015

Approved by:

(Signature of HOD)
Date:

INSTRUCTIONS TO STUDENTS

1. Students should be regular and come prepared for the lab practice.
2. In case a student misses a class, it is his/her responsibility to complete that missed experiment(s).
3. Students should bring the observation/Work book. Lab manual, prescribed textbook and class notes can be kept ready for reference if required.
4. Students should implement all experiments individually.
5. While conducting the experiments students should see that their programs would meet the following criteria:
   - Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.
   - Programs should perform input validation (Data type, range error, etc.) and give appropriate error messages and suggest corrective actions.
   - Comments should be used to give the statement of the problem and every function should indicate the purpose of the function, inputs and outputs
   - Statements within the program should be properly indented
   - Use meaningful names for variables and functions.
   - Make use of Constants and type definitions wherever needed.
   - Students are advised to follow the modular approach to write the programs.

6. Once the experiment(s) get executed, they should show the program and results to the Instructors and upload LAB REPORT to Portal.
7. Questions for lab tests and exam need not necessarily be limited to the questions in the manual, but could involve some variations and / or combinations of the questions.

PROCEDURE OF EVALUATION

1. Continuous evaluation (12 Evaluations of 5 Marks each) : 60 Max. Marks
   Continuous evaluation scheme:

i) LAB REPORT : 2.5 Marks

ii) Program execution : 1.5 Marks

iii) Viva : 1 Marks

2. End Semester Test :40 Marks.

LAB-01

1.1 Basic Linux commands

The basic Linux commands are summarized below. See the manual pages for a list of options for each command.

- **man** *command_name* : Gets online help for command name.
- **passwd** : Sets (changes) the password.
- **pwd** : Displays the current working directory.
- **ls :** Lists the contents of a directory.
- **more** *file_name* : Scrolls through a file.
    - To list the next page, press the space bar.
    - go backwards, press b.
    - To quit from more, press q.
- **mv** *old_file_ name new_file_name*: Renames a file.
- **mv** *file_name directory_name*: Moves a file to a directory.
- **mv** *old_directory name new_directory_ name*: Renames a directory.
- **rm** *file name* : Deletes(removes) a file.
- **mkdir** *directory name*: Creates a directory.
- **rmdir** *directory name*: Removes a directory.
- **cd** *directory name*: Changes the current working directory to directory name. If directory name is omitted, the shell is moved to your home directory.
- **cp** *file name new file name*: Copies a file.
  cp *file name directory name*: Copies a file into directory name.
- **chmod** *who op-code permission file or directory name*: Changes the file access permissions.
  who: **u** user, **g** group, **o** other users, **a** all;
  op-code: + add permission, − remove permission;
  permission: **r** read, **w** write, **x** execute.
- **ps**: Process status report.
- **kill** *PID*: Terminates the process with a process ID *PID*.
- Ctrl-**c** : Terminates a command before it is finished.
- **cmp** *file1 file2*: Compares *file1* and *file2* byte by byte.
- **grep** *keyword file(s)*: Search the file(s) and outputs the lines containing

the keyword.

Most of the above commands accept input from the system's *standard input* device (e.g., the keyboard) and send an output to the system's *standard output* device (e.g., the screen). Sometimes it is convenient to direct the output to another process as input for further processing, or to a file for storage. The *redirect* operator ">" directs the output to a file, as:
command > *file name*

With the pipe operator "|", two commands can be concatenated as:
command1 | command2,
where the output of **command1** is redirected as the input of **command2**.

1.2 Review of Linux File System calls

open() : opens a file in specified mode
close() : closes specified file descriptor
read() : read from a file
write(): write to a file
creat(): create a file
fork(): create a child process
(Refer to Linux **man pages** for more information)

1.3 Network daemons and services

A daemon is a process running in the background of the system. Many TCP/IP services (e.g., Telnet) are handled by a daemon called **inetd**. Rather than running several network-related daemons, the **inetd** daemon works as a dispatcher and starts the necessary server processes when requests arrive. When a client wants a particular service from a remote server, the client contacts the **inetd** daemon through the server's well- known port number, which prompts **inetd** to start the corresponding server process.

The network daemons managed by **inetd** are specified in a configuration file called **/etc/inetd.conf**. Each service has a line in the file defining the network daemon that provides the service and its configuration parameters. One can comment a line, i.e., insert a # at the beginning of the line, to disable the corresponding service. Note that there are some stand-alone network daemons that are not managed by **inetd**. For example, web service is provided by the **httpd** daemon, and DNS service is provided by the **named** daemon.

In Latest Linux distrubutions, **xinetd** replaces inetd, adding stronger

security and more functionality. **xinetd** uses a simple common configuration file **/etc/xinetd.conf**. In addition, each service managed by **xinetd** uses an individual configuration file in the **/etc/xinetd.d** directory.

Well-known port numbers are defined in the **/etc/services** file. A server can handle multiple clients for a service at the same time through the same well-known port number, while a client uses an ephemeral port number. The uniqueness of a communication session between two hosts is preserved by means of the port number and IP address pairs of the server and client hosts.

1.4 Network configurations files

When a host is configured to boot locally, certain TCP/IP configuration parameters are stored in appropriate local disk files. When the system boots up, these parameters are read from the files and used to configure the daemons and the network interfaces. A parameter may be changed by editing the corresponding configuration file.

In addition to */etc/services* and */etc/inetd.conf* discussed above, we now list other network configuration files.

Linux Configuration Files	**Purpose**
/etc/hosts	Stores the host name of this machine and other machines.
/etc/sysconfig/network	Stores the host name and the default gateway IP address.
/etc/sysconfig/network-scripts/ifcfg-eth0	Stores the IP address of the first Ethernet interface.
/etc/default-route	Stores a default gateway, i.e., the IP address or the domain name of the default router.
/etc/resolv.conf	Stores the IP addresses of the DNS servers
/etc/nsswitch.conf	Configures the means by which host names are resolved.

1.5 Understanding Basic Networking Commands in Linux

1.5.1 Configuring a network interface

The **netstat** command can be used to display the configuration information and statistics on a network interface. The same command is also used to display the host routing table. Several **netstat** options are listed below that will be used frequently in the experiments.

netstat -*a* : Shows the state of all sockets, routing table entries, and interfaces.
netstat -*r* : Displays the routing table.
netstat -*i* : Displays the interface information.
netstat -*n* : Displays numbers instead of names.
netstat -*s* : Dis plays per-protocol statistics.

The **ifconfig** command is used to configure a network interface. The following options are used for the reconfiguration of the IP address and network mask.

ifconfig -*a :* Shows the states of all interfaces in the system.
ifconfig <*interface name*> *down* : Disables the network interface, where interface name is the name of the Ethernet interface.
ifconfig <*interface name*> <*new IP address*> *up* : Assigns a new IP address to the interface and brings it up.
ifconfig <*interface name* > *netmask* <*new netmask*> : Assigns a new network mask for the interface.

1.6 Diagnostic tools

Diagnostic tools are used to identify problems in the network, as well as to help understand the behavior of network protocols. We will use the following tools extensively in the experiments.

1.6.1 Tcpdump

Tcpdump is a network traffic sniffer built on the packet capture library *libpcap*. While started, it captures and displays packets on the LAN segment. By analyzing the traffic flows and the packet header fields, a great deal of information can be gained about the behavior of the protocols and their operation within the network. Problems in the network can also be identified. A packet filter can be defined in the command line with different options to obtain a desired output.

1.6.2 WireShark

Wireshark is a network protocol analyzer built on the packet capture library *pcap*. In addition to capturing network packets as in *Tcpdump*, Wireshark provides a user friendly graphical interface, and supports additional application layer protocols. Wireshark can also import pre-captured data files from other network monitoring tools, such as *Tcpdump* and *Sniffer*.

Exercise 1.1:

Login to the system. Enter your user account with login ID and the password. Get acquainted with the Desktop environment, the Linux commands, text editors, and the man pages.

Exercise 1.2

After logging in, open a command window, Show your login ID by typing **whoami** in the console.

Create a directory of your own, using **mkdir** *name_of your_directory*. Change to your directory, using **cd** *name_of_your_directory*. You can save your data files for all your laboratory experiments here.

Open another command window. Run **pwd** in this and the previously opened consoles. Save the outputs in both consoles.

LAB REPORT

What is the default directory when you open a new command window?
What is your working directory?

Exercise 1.3

Run **ps -e** to list the processes running in your host. After starting a new process by running telnet in another command window, execute **ps -e** again in a third window to see if there is any change in its output.
Find the process id of the telnet process you started, by:

ps -e | grep telnet

Then use **kill** process id of telnet to terminate the **telnet** process.

LAB REPORT

Is the Internet service daemon, **xinetd**, started in your system? Is **inetd** started in your system? Why? Document issued commands and its output.

Exercise 1.4

Display the file */etc/services* on your screen, using:

more /etc/services

Then in another console, use the redirect operator to redirect the **more** output to a file using **more /etc/services > ser_more**. Compare the file *ser_more* with the original more output in the other command window.

Copy */etc/services* file to a local file named *ser_cp* in your working directory, using **cp /etc/services ser_cp**. Compare files ser_more and ser_cp, using **cmp ser_more ser_cp**. Are these two files identical?

Concatenate these two files using **cat ser_more ser cp > ser_cat**.

Display the file sizes using **ls -l ser***. Save the output. What are the sizes of files *ser_more, ser_ cp, and ser_cat?*

LAB REPORT

Submit the **ls** output you saved in this exercise and answer the above questions.

Exercise 1.5

Read the **man** pages for the following programs and learn the usage

arp	**arping**	**ifconfig**	**tcpdump**
ping	**netstat**	**route**	**wireshark**

LAB REPORT

Explain each of the above commands briefly. Two or three sentences per command would be adequate. Document the output observed while working with above commands.

Exercise 1.6

Write a C program to copy the content of one file of unknown size to another using Linux File system calls. (Use separate file name to copied file.)

LAB REPORT

Copy files of different file-types(Text Files,Object codes). Compare file

sizes of original and copied file.

Exercise 1.7

Write a program to demonstrate the concept of parent & child processes using ***fork()*** system call in which the parent reads a file name from the keyboard and put it in a buffer. Child gets the file name from the buffer and reads first 10 characters from the file and puts into the same buffer. Parent reads the content from the buffer and display on the screen. (Use modular approach to develop the program)

LAB REPORT

What is the Process ID of parent and Child process in your computer?

Exercise 1.7

Write **one C program each** to demonstrate the working of following forms of IPC.

a)Pipes

b)FIFO

LAB REPORT

Explain the working of your program and differentiate between pipe and FIFO IPC with respect to your program.

[Note: All program Source codes and REPORT to be uploaded to Department Portal before Leaving the Lab]

LAB-02

Socket Programming – Using TCP

Exercise 2.1

In this experiment, our goal is to make two machines talk to each other, using a stream socket (i.e. a TCP connection). You will have to write two programs: client.c and server.c. This is a simplistic implementation of an FTP client and server. The client and server interact as follows:

The server essentially runs on a machine waiting for incoming TCP connections on a specific port (say 5000).

On an incoming TCP connection from a client, it should accept it.

It should then read a string from the client, representing the filename requested by the client. You may assume that the string is NULL terminated, and is of maximum length 256, including the trailing NULL character. The server should read this string, and send the corresponding file to the client, and then close the connection.

If the file does not exist, then the server simply closes the connection without sending anything back to the client. Once a connection is closed, the server should continue waiting for the next incoming TCP connection, for the next file transfer.

The client, in a loop, does the following:

(1) Read filename from user, via STDIN

(2) form a TCP connection to the server

(3) send the filename to the server

(4) store the file contents from the server received via the TCP connection, onto the local file system.

To make things easy for you, we have provided the skeletons of the programs in subject page of Department Portal http://mycse in folder **LAB_02_RESOURCES** .Your job is to fill in the blanks. The provided data sheet contains the syntax of all the functions you would need for this exercise. Note that the variables used in the syntax may be very different from those used in the skeleton, so use caution.

Initially, you will run both client and server on the same machine utilizing the loop back feature. This will help in debugging the programs. For this, you will need to specify the destination address (server's) as 127.0.0.1.

Once the programs are running correctly, you can pair with one of the other

teams and test that your client runs with their server and vice-versa. In this case change the destination IP address appropriately.

Exercise 2.2

Develop UDP ECHO Server and Client. Demonstrate the working with multiple instances of clients.

Exercise 2.3

Develop UDP **Concurrent** Server that handles **multiple** Clients. Server computes square of a number sent by clients and returns the result to clients for display. Also display server process ID (PID) in each transaction.(Hint: Use fork() at Server)

Exercise 2.4

Develop a client/server based application using UDP to execute the program at remote server. i.e. the client sends the executable file to the server, server executes the file , stores the result in a file and sends back to the client.

LAB REPORT

Upload the Source Codes and Output with neat comments.

LAB-03

Socket Programming -Using TCP

Exercise 3.1

Modify the Exercise 2.1 with TCP

Exercise 3.23

Modify the Exercise 2.2 with TCP

Exercise 3.3

Modify the Exercise 2.3 with TCP. Use **select()** system call in this exercise.

LAB REPORT

Upload the Source Codes and Output with neat comments.

LAB-04 and LAB-05

Implement a Small Network Application in C using knowledge obtained from Lab-Lab-3 and Lab-4. (Max. Duration:2 Hours for Coding)
Eg: File Server, Time Server, File Down-loader, Multi-user Chat Server
(Topic is subject to approval from faculty).

LAB-06

Network interface exercises

Exercise 6.1

Use the **ifconfig -a** command to display information about the network interfaces on your host. Find the IP address and the net mask of your machine.

LAB REPORT

How many interfaces does the host have? List all the interfaces found, give their names, and explain their functions briefly.

What are the MTUs of the interfaces on your host?

Is network subnetted? What is the reasoning for your answer? What are the reasons for subnetting?

Exercise 6.2

While **tcpdump host** *your_host* is running in one command window, run **ping** 127.0.0.1 from another command window.

LAB REPORT

From the ping output, is the 127.0.0.1 interface on? Can you see any ICMP message sent from your host in the tcpdump output? Why?

Exercise 6.3

By using **netstat -in** command, collect the statistics from all the hosts on the network. Since we use the same login name and password, we can **telnet** to other workstations and run **netstat -in** there

Save the **netstat -in** outputs.

If you don't see a significant amount of output packets in the **netstat** output, the machine was probably restarted recently. You may do this experiment later, or use the following **sock** command to generate some network traffic:

sock -u -i -n200 *remote_host* **echo**

LAB REPORT

Calculate the average collision rate over all the hosts for the set of statistics you collected in this exercise.

ARP exercises

Exercise 6.4

Use **arp -a** to see the entire ARP table. Observe that all the IP addresses displayed are on the same subnet.

If you find that all the remote hosts are in your host's ARP table, you need to delete a remote host (not your workstation) from the table, using,

arp -d *remote_host*

Save the ARP table for your lab report.

While **tcpdump -enx -w exe3_4.out** is running, **ping** a remote host that has no entry in your host ARP table. Then terminate the **tcpdump** program.

Next, run **wireshark -r exe2.out&** to load the **tcpdump** trace file.

Observe the first few lines of the packet trace to see how ARP is used to resolve an IP address.

Run **arp -a** to see a new line added in your host's ARP table. Save the new ARP table for your lab report.

Note down **ARP Request** packet and **ARP Reply** Packet in **wireshark** window.

LAB REPORT

From the saved **tcpdump** output, explain how ARP operates. Draw the format of a captured, ARP request and reply including each field and the value.

Your report should include the answers for the following questions.

- What is the target IP address in the ARP request?
- At the MAC layer, what is the destination Ethernet address of the frame carrying the ARP request?
- What is the frame type field in the Ethernet frame?
- Who sends the ARP reply?

Exercise 6.5

While **tcpdump host** *your_host* is running to capture traffic from your machine, execute **telnet 128.238.66.200**. Note there is no host with this IP address in the current configuration of the lab network.

Save the **tcpdump** output of the first few packets for the lab report.

After getting the necessary output, terminate the telnet session.

LAB REPORT

From the saved tcpdump output, describe how the ARP timeout and retransmission were performed. How many attemps were made to resolve a non-existing IP address?

Exercise with ICMP and Ping

Exercise 6.6

Use **ping -sv** remote host to test whether the remote host is reachable, while running: **tcpdump -enx** host your_host **and** remote_host.

Save the **tcpdump** and **ping** output for the future study on **ping.**

LAB REPORT

What ICMP messages are used by ping?

Exercise 6.7

While running **tcpdump -x -s 70 host** *your_host* **and** *remote_host*, execute the following **sock** command to send a UDP datagram to the remote host:

sock -i -u -n1 -w1000 remote host **88888**.

Save the **tcpdump** output for the lab report.

LAB REPORT

Study the saved ICMP port unreachable error message.

Why are the first 8 bytes of the original IP datagram payload included in the ICMP message?

Exercise 6.8

While **tcpdump** is running to capture the ICMP messages, **ping** a host with IP address 128.238.60.100. Save the **ping** output.

LAB REPORT

Can you see any traffic sent on the network? Why? Explain what happened from the ping output.

List the different ICMP messages you captured in previous Exercises (if any).

Give the values of the type and code fields.

LAB-07

Objective:

1. Get acquainted with some commonly used networking commands and TCP/IP diagnostic tools.
2. Understand the concept of layering/encapsulation by looking at Link, IP and TCP headers.
3. Understand the concept of multiplexing using Ethernet "frame type" field, IP "protocol field", transport "port number" field.

Exercise 7.1 Play Time

Play around with *tcpdump*, *wireshark*, *ping*, *arp*, *route*, *ifconfig*, *host*
Look at */etc/hostname*; */etc/hosts*; */etc/network/interfaces*; */etc/resolv.conf*; */etc/protocols*; */etc/services* and understand what the files are for.
At the end of this exercise, you should have some basic understanding of how a host manages network information as well as gain some experience on using networking tools. You should be able to collect a trace (write to a file) via tcpdump and view the trace in wireshark (using the -r option).

Exercise 7.2 Simple Stuff

1. Whats your machine's host name and IP address? How did you get this information?
2. What is the next hop router's IP address and MAC address? How did you get this information?
3. What is the local DNS server's host name and IP address? How did you get this information?
4. What do the numbers in the file /etc/protocols represent?
5. What is the port number associated with applications: ssh, ftp, nfs, smtp (email)?

Exercise 7.3 Encapsulation and Demultiplexing

Goal: To understand layering and demultiplexing, Ms. Lux wants to capture packets. She also wants to understand how web flows operate at the same time. So, help her design an experiment that captures only those packets that are exchanged between her machine and CSE web server when she clicks the url http://mycse

Guidance:

1. Run tcpdump with -n option to avoid name lookup.
2. Use wget (command: wget --no-proxy http://mycse) to download the url. You could also use firefox/chrome, but this is cleaner and simpler.
3. Your trace should not capture any background traffic.
4. Before answering the questions, explore different packets by clicking on the individual packets. Also note the sequence of packet exchange.

LAB REPORT

1. Explain your experimental design by specifying the exact commands (with options) you will run and in which order. Avoid description unless absolutely necessary.
2. Select the first TCP packet listed.
    a) Which next-hop node is it destined to? Specify the next-hop node's MAC and IP address. How did you determine this information?
    b) Who is the packet's final destination? Specify the final destinations' MAC and IP address? How did you determine this information?
    c) What are the fields used at the link(Ethernet), IP and TCP headers to demux the packet at the destination? Specify the values of these fields in decimal format and the corresponding process (protocol) the packet is passed to.
3. Apart from the above reporting, name your trace file as “exercise7_1.out” and upload the file to portal.

Exercise 7.4 More Demultiplexing

Goal: With the success of the previous experiment, Ms. Rani now wants to capture and examine different types of traffic, basically arp, ICMP (protocol used by ping) and ssh. She wants to capture all of the above in just one single trace. Help her design an experiment to do the same.

Guidance:

1. For ssh, you could **ssh** to your neighbor's machine.
2. In wireshark, click on the protocol field to order the packets according to the protocol.

LAB REPORT

1. Explain your experimental design by specifying the exact commands (with options) you will run and in which order. Avoid description unless absolutely necessary.
2. **Arp protocol:** Click on any one of the ARP packets.

a) Trace the flow of this packet up the protocol stack i.e specify what all processes/protocols handle this packet.
b) What is the value of the field used in Ethernet header to pass packets to the ARP module? Express it in decimal format.

3. **ICMP protocol:** Click on any one of the ICMP packets.
   a) Trace the flow of this packet up the protocol stack i.e specify what all processes handle this packet.
   b) Expand the “Ethernet” header. Which higher level process (protocol) is this packet passed to and what is the value in decimals?
   c) Expand the IP header. What is the value of the field used in this header to pass packets to the ICMP module? Express it in decimal format.
4. **SSH protocol:** Click on any one of the SSH packets.
   a) Click on the IP header field. Specify the source and destination IP addresses.
   b) Expand the TCP header. Specify the source and destination port numbers.
   c) Which machine (IP address) is the SSH server? Hint: SSH server's listen on designated ports as specified in /etc/services.
5. Name your trace file as “exercise7_4.out” and add the file to your roll number directory.

LAB-08

Reference:

1. Man pages of the commands.
2. Man page of a new tool 'arping'
3. Video/Slides of 'OSI Protocol stack' and 'Inter-Layer Communication' under Introduction.
4. Video/slides of 'Supporting Protocols' under 'Network Layer'.

Exercise 8.1: Some More Demultiplexing

Goal: Ms. Rani finds ssh protocols fascinating and now wants to capture what happens when two ssh sessions are established (at about the same time) between her machine and the same remote host. Since the sessions have same source and destination IP address, she wants to figure out how the sessions are uniquely identified. Help her capture such a trace. There should be no other background traffic.

Guidance:

1. Use two windows to run each ssh session, type the command in both windows and run these commands as close in time as possible. (ssh to a neighboring machine)
2. When opening the trace in wireshark, we will use filters to filter out unnecessary packets and capture just the first packet of both sessions in either direction (client to server and back). Type tcp.flags.syn == 0x02 in the filter's field (this esentially makes use of the syn flag of TCP header, which is set to 1 only in the first packet of the TCP connection). You should see 4 packets listed.

LAB REPORT

1. Explain your design by specifying the exact commands (with options) you will run and in which order. Avoid description unless absolutely necessary.
2. What is the port number used by the remote machine for the first and the second ssh session? Are both sessions connected to the same port number on the remote machine? How do you think the ssh application at remote machine distinguishes between the two sessions?
3. When your machine receives packets from the remote host, how does the TCP layer figure out to which ssh session this packet has to be passed? Specify the value of the fields used by TCP to do this.
4. Name your trace file as “exercise1.out” and add the file to your roll-number

directory.

ARP Protocol

Col. Raaja wants to counter cyber attacks and has read up on network layer protocols. He wants to relate theory to practice. He has many questions, help him design the right experiments that will answer his questions. The colonel has little patience with irrelevant details, so ensure in each experiment that you capture the right set of packets.

Exercise 8.2: Address Resolution Protocol (ARP)

When Col. Raaja read up on how packet forwarding is done at a host, he came to know that in cases where the destination IP address belongs to the host's own subnet, the packet goes directly, otherwise it goes via a router. He wants to check this out. Can you design an experiment that illustrates this? Also, he wants to know what happens if you try to send a packet to “a non existent host within the same subnet”. Help him with that as well.

Guidance:

1. As the name of the exercise suggests, this experiment is about exploring the ARP protocol in the context of forwarding. You do not have permissions to delete or set the arp entries but you should be able to view the arp entries currently in the cache.
2. In all cases, you do NOT want the arp entry of the target in the cache. So, ensure this is the case.
3. When you are sending packets to a 'non existent host within the same subnet', do wait for at least 10 sec before closing your packet capture tool. The underlying action in this case, takes time.
4. “Within subnet” -- hosts within lab (e.g. 172.16.59.* but not all may be reachable, so check your neighborhood IP addresses).
   “Outside subnet” -- hosts outside lab (e.g. 10.109.1.11; 10.107.1.1; 10.129.50.11; 10.129.50.12; 10.104.1.1). Non existent hosts (e.g. 10.105.12.1 to 10.105.12.7) (Note: This IP addresses may vary depending on campus network settings. Consult faculty for correct IPs)

Questions:

1. For the first case, where you sent packets to a host within the same subnet, specify the command sequence used. How many arp messages were exchanged? Does the arp entry correspond to the remote host IP address you contacted? Explain your observations.
2. For the second case, where you sent packets to a remote host outside the

same subnet, specify the command sequence used. How many arp messages were exchanged? Does the arp entry correspond to the remote host IP address you contacted? Explain your observations.

3. For the third case where you sent packets to a non-existing host, specify the command sequence used. How many ARP attempts were made to resolve the non-existing IP address? After what time did ARP give up? Hint: In this case -c option of ping is important to use.

Exercise 8.3: Gratuitous ARP

Col. Raaja read that ARP is a good candidate for spoofing. An intruder can generate a gratuitous ARP advertising his MAC address and some one else's IP address and capture the some one's traffic. Given its relevance to security, he wants to know more about this aspect of ARP. Design an experiment to capture gratuitous ARPs.

Guidance:

1. 'arping' is a tool that generates gratuitous ARPs.
2. Note that you cannot spoof another person's IP address via ARPing. The goal of this experiment is not to spoof, but to understand gratuitous ARPs. So generate a gratuitous ARP with your own IP address.
3. Gratuitous ARPs can be sent as both ARP requests and ARP replies. Capture both types of packets via appropriate arguments.
   For requests: e.g. **arping -I** *eth0* **-c 5** *your_ip_address*. For replies: e.g. **arping -A -I** *eth0* **-c 5** *your_ip_address*

Questions:

1. How is it ensured that the gratuitous ARPs reach all hosts within the physical network?
2. What difference did you observe when gratuitous ARP was sent as a request versus it being sent as a reply?

LAB-09

Exercise 9.1 TCP/IP headers

In this exercise, we will use tcpdump to capture a packet containing the link, IP, and TCP headers and use ethereal to analyze this packet.

First, run **tcpdump -enx -w exe7_1.out** You will not see any *tcpdump* output, since the -w option is used to write the output to the exe2_1.out file.

Then, you may want to run **ssh** to a remote host 6 to generate some TCP traffic.

After you login the remote machine, terminate the telnet session and terminate the tcpdump program.

Next, you will use wireshark to open the packet trace captured by tcpdump and analyze the captured packets.

To do this, run **wireshark -r exe7_1.out &**. The wireshark Graphical User Interface (GUI) will pop up and the packets captured by *tcpdump* will be displayed.

For your report, you need to save any one of the packets that contain the link, IP, and TCP headers.

LAB REPORT

Draw the format of the packet you saved, including the link, IP, and TCP headers, and identify the value of each field in these headers. Express the values in the decimal format.

What is the value of the protocol field in the IP header of the packet you saved? What is the use of the protocol field?

Exercise 9.2

In a manner similar to the previous exercise, run tcpdump to capture an ARP request and an ARP reply, and then use wireshark to analyze the frames.

Run **tcpdump -enx -w exe7_2.out** to capture all the packets on the LAN segment.

If there is no *arp* requests and replies in the network, generate some using *arping* remote machine.

After you see several ARP replies in the arping output, terminate the arping and the tcpdump program. Open the tcpdump trace using wireshark **-r exe7_2.out &**. Print one ARP request and one ARP reply using **wireshark**.

LAB REPORT

What is the value of the frame type field in an Ethernet frame carrying an **ARP request** and in an Ethernet frame carrying an **ARP reply**, respectively?

What is the value of the *frame type* field in an Ethernet frame carrying an IP datagram captured in the previous exercise?

What is the use of the *frame type* field?

Exercise 9.3

Using the tcpdump utility, capture any packet on the LAN and see the output format for different command-line options. Study the various expressions for selecting which packets to be dumped.

For this experiment, use the man page for tcpdump to find out the options and expressions that can be used.

If there is no traffic on the network, you may generate traffic with some applications (e.g. telnet, ping, etc.).

LAB REPORT

Explain briefly the purposes of the following tcpdump expressions.

tcpdump udp port 520

tcpdump -x -s 120 ip proto 89

tcpdump -x -s 70 host ip addr1 and (ip addr2 or ip addr3)

tcpdump -x -s 70 host ip addr1 and not ip addr2

Exercise 9.4

Start tcpdump in a command window to capture packets between your machine and a remote host using:

tcpdump -n -nn host *your_host* and *remote_host*

Execute a TCP utility, *telnet* for example, in another command window. When you see a TCP packet in the *tcpdump* output, terminate *tcpdump* and save its output.

LAB REPORT

What are the port numbers used by the remote and the local computer?

Which machine's port number matches the port number listed for telnet in the */etc/services* file?

Exercise 9.5

Start tcpdump in one command window using:

tcpdump -n -nn host *your_ host* **and** *remote_host*.

Then, telnet to the remote host from a second command window by typing telnet remote host. Again issue the same telnet remote host command from a third command window. Now you are opening two telnet sessions to the same remote host simultaneously, from two different command windows.

Check the port numbers being used on both sides of the two connections from the output in the *tcpdump* window. Save a TCP packet from each of the connections.

LAB REPORT

When you have two telnet sessions with your machine, what port number is used on the remote machine?

Are both sessions connected to the same port number on the remote machine?

What port numbers are used in your machine for the first and second telnet, respectively?

What is the range of Internet-wide well-known port numbers? What is the range of well-known port numbers for Unix/Linux specific service?

What is the range for a client port number? Compare your answer to the well-known port numbers defined in the */etc/services* file. Are they consistent?

LAB-10
Action at Network Layer

Objective:

1. Understand the operation of various mechanisms/protocols that operate at network layer: IP fragmentation, DHCP, ICMP.

Reference:

1. Man pages of the commands.
2. Video/slides of 'IP packet format', 'Obtaining IP Addresses' and 'Supporting Protocols', all under 'Network Layer'.

Exercise 10.1: IP Fragmentation

The Colonel saw the IP packet format and is particularly fascinated by the fragmentation fields and wants to see it in action. Design an experiment where 1) No fragmentation occurs and 2) Fragmentation occurs.

Guidance:

1. You are yet to cover socket programming. “sendUDP.c” is a simple socket program (provided in the directory) that generates a single IP packet of a given size and sends it to the specified destination IP address. Compile the program and design your experiment around it.
2. Which destination IP address should you use? Try both an existing host and non-existant host within subnet and see what happens.

Questions:

Answer the following questions in your report.

1. Specify the exact commands used to conduct the experiment.
2. For the case when no fragmentation happened, note down the values corresponding to the following fields: Total length, Identifier, flags and fragment offset. Why is the total length field so?
3. For the case when fragmentation happened, for each fragment, note down the values corresponding to the following fields: Total length, Identifier, flags and fragment offset. Do the values make sense based on what was covered in theory?

Exercise 10.2: Dynamic Host Configuration Protocol (DHCP)

The Colonel is equally fascinated by how hosts obtain IP address and wants to look at the message exchange of this process. One of his staff already procured such a trace, help him interpret the trace.

Guidance:

1. Configuring IP addresses requires root permission. Since you do not have these privileges, for this exercise you will have to make do with a generated trace file “dhcp.out”. This trace file was generated using tcpdump and running “dhclient eth0” on a terminal with root permissions. Explore it via wireshark.

Questions:

Answer the following questions

1. What is the IP address of the DHCP server and on what port is it listening on?
2. Was any DHCP relay involved in forwarding the DHCP packets? How did you determine the answer.
3. When the DHCP server replied, which IP address did it reply to? And why?
4. What is the offered IP address to the client and for how long is this address valid?
5. Apart from the IP address, what additional information has the client received from the dhcp server?

Exercise 10.3: Internet Control Message Protocol (ICMP)

The Colonel read about the different types of ICMP messages and wants to look at them in a packet trace. He wants to look at the following 3 types.

- Type 0, code 0
- Type 3, code 3
- Type 8, code 0

Design experiments that will produce ICMP messages of the above type in a packet trace.

Question:

Which commands did you use to generate the above ICMP message types.

Exercise 10.4: More Internet Control Message Protocol (ICMP)

The Colonel's collegue (who in far away as an undercover agent) often complains that he doesn't get time- critical messages on time. The colonel who now knows enough theory wants to debug this. He feels its likely due to too many routers in between and feels 'tracepath' is the right command to use. Help him interpret the output of the command and the trace collected.

Guidance:

1. Traceroute may not available on the lab machines (permission issues). So for this, use tracepath which is very similar to traceroute.

2. When tracing the path to a host, select a host that is not on the same physical network as your mac hine.(eg. www.manipal.edu, www.yahoo.com) Otherwise, its too trivial with a hop count of 1.

Questions:

1. Specify the command used and cut/paste the output you saw.
2. What ICMP message type/code is involved in the process?
3. Explain how this command works by looking at the trace file and the output produced by running the command.
4. What is the IP address of your machine's default router?

LAB 11

Objective:

1. Get familiarized with NS2 simulator
2. Understand popular performance metrics: throughput and loss.

General instructions:

1. Download relevant files needed for this lab from Department Portal http://mycse , available under "lab08- files.zip" and unzip using the command "unzip lab04-files.zip".
2. In some of the exercises, we will specify a goal. Your job is to design an experiment that meets the goal, conduct it and answer the questions asked.
3. As you proceed with the lab instructions below, for each exercise, note down the answers to the exercise along with any interesting observations in the file. Take care to ensure that the content in the file is neatly organized.

Reference: (Available in Department Portal)

1. ns-tcp.tcl, ns-simple.tcl, (example commands)
2. NS tutorial slides
3. Goals and metric slides
4. sample.gnu (example Gnuplot file for plotting)

Lab Instructions:

When evaluating the traces, you can get the desired results with the use of ***grep***, ***pipe (|)***, ***redirection (>)***, **wc** and some manual inspection/calculations. There is no need to write any other code or use complex bash scripting.

Exercise 11.1: Warm Up

Two scripts *ns-tcp.tcl* and *ns-simple.tcl* have been provided. Run them and see the action in NAM. Also examine the output trace file (simple.tr, out.tr). Understand what is happening.

LAB REPORT

1. For ns-tcp.tcl:
    a) What is the size of the TCP packet, what is the size of the ack?
    b) What application feeds TCP?
    c) What is the queue-size at the intermediate node?
    d) What is the data rate and propogation delay of the two links?

e) How many tcp data packets and how many ack packets were dropped?

2. For ns-simple.tcl

a) How many flows and of what type (distinguish based on transport protocol) have been set-up as part of the simulation?

b) What application feeds the two flows?

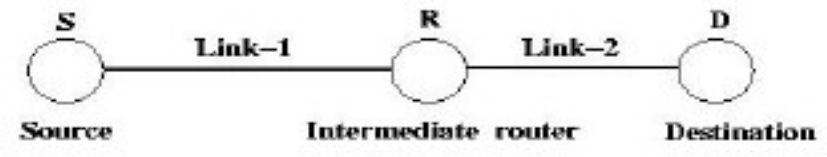
c) By looking at the 'tcl' file, can you figure out at what rate the two flows are injecting data (source data-rate) into the network? Specify the rate also.

d) When do the flows end?

e) Determine the throughput (as observed at the receiver) of the two flows? Throughput: The rate at which bits (unique not duplicate) are being received at the destination? Eg: If 10 packets of size 100 bytes were received in a duration of 100 seconds at receiver, we say the throughput is 10 * 100 /100 = 10Bps or 80bps or 0.08kbps.

Exercise 11.2: Metrics

Goal: Metric Mahadeva wants to understand performance of protocols on networks. Since this is his first foray in this area, he decides to keep the topology and protocol simple. His smart friend Raju Topiwala gave him the following topology to use and suggested that he use UDP since its much simpler than TCP. UDP just adds sequence numbers to packets and pretty much does nothing more (as far as this experiment is concerned).



There is a source node, a destination node, and an intermediate router (marked "S", "D", and "R" respectively). The link between nodes S and R (Link-1) has a bandwidth of 1Mbps and 50ms latency. The link between nodes R and D (Link-2) has a bandwidth of 100kbps and 5ms latency.

Mahadeva wants to understand the following:

- If the source data rate (rate at which source injects data into the network) varies, what happens to the packets in the network?
- At what point does it get congested?
- How do the throughput and loss vary as a function of the source data rate.

Help him to write a ns2 tcl script (name it ns-udp.tcl) that helps him conduct this experiment. Help him also interpret the results of the experiment.

Guidance:

1. Vary the source data rate (termed Offered Load hence forth) to take on values of 40kbps, 80 kbps, 120kbps and 160kbps. Note each value will result in one run. Thus you have 4 runs and hence 4 trace files. Ensure proper naming of the trace files (Eg: udp-40k.tr, udp-80k.tr, udp-120k.tr and udp-160k.tr). Include a tcl file of one of the runs as ns-udp.tcl in the directory.
2. Routers have limited buffer space and drop packets during congestion. Use the "queue-limit" command to model this. Limit the queue at the bottleneck link (link-2) between node "R" and node "D" to 10 packets. Its fun to monitor the queue at Link-2 (between R and D). You can do so using the duplex-link-op command.
3. Let each experiment run for at least 10sec.
4. **Metrics:** For each run, calculate/measure the following metrics by processing the output trace file.
   a) Offered Load: The rate at which source traffic is being injected into the network. (You set this value in the tcl script but confirm via trace that you got it correct.)
   b) Packet Loss: The number of packets that were sent by the sender, but didn't reach the destination. Express it in percentage.
   c) Throughput: The rate at which bits are being received at the destination? Eg: If 100 packets of size 100 bytes were received in a duration of 100 seconds, we say the throughput is 100 * 100 *8/100 = 800bps or 0.8kbps. Express it in kbps.

LAB REPORT

Plot the following graphs. Ensure the axis are properly labeled, with proper legend and correct units.

1. Offered Load vs percentage packet loss
2. Offered load vs throughput

In the report, comment on what you observe in each graph and the reasons for the same.

Exercise 11.3: Additional Work(To be evaluated Separately)

This exercise is not compulsory but time permitting you can explore it. This does involve bash scripting or using C code.

This is a continuation of the previous exercise. If the source data rate (rate at

which source injects data into the network) varies, how does the delay vary as a function of the source data rate. That is plot the Offered Load vs Average end-to-end delay.

Average end-to-end delay for a run: If t1 is the time the packet was generated at the source and t2 was the time the packet was received at the destination. Delay of a packet is t2-t1. Calculate average of these values for packets that reached the destination. Express it in ms. Then plot the Offered Load vs Average end-to-end delay for the different runs.

LAB 12

Hands on experiments using switches,routers,wireless access points. Students will practice design and configuration of LANs with *IP address asaignment*, Interconnection between LANs, Subnetting, Supernetting.

Topology will given by the teacher.

LAB 13

END SEM EXAM