



Lab Manual

Second Year Semester-IV

Department of Information Technology

Networking Lab



Institute Vision & Mission

Vision

To foster and permeate higher and quality education with value added engineering, technology programs, providing all facilities in terms of technology and platforms for all round development with societal awareness and nurture the youth with international competencies and exemplary level of employability even under highly competitive environment so that they are innovative adaptable and capable of handling problems faced by our country and world at large.

Mission

The Institution is committed to mobilize the resources and equip itself with men and materials of excellence thereby ensuring that the Institution becomes pivotal center of service to Industry, academia, and society with the latest technology. RAIT engages different platforms such as technology enhancing Student Technical Societies, Cultural platforms, Sports excellence centers, Entrepreneurial Development Center and Societal Interaction Cell. To develop the college to become an autonomous Institution & deemed university at the earliest with facilities for advanced research and development programs on par with international standards. To invite international and reputed national Institutions and Universities to collaborate with our institution on the issues of common interest of teaching and learning sophistication.



Department Vision & Mission

Vision

To pervade higher and quality education with value added engineering, technology programs to deliver the IT graduates with knowledge, skills, tools and competencies necessary to understand and apply the technical knowledge and to become competent to practice engineering professionally and ethically in tomorrow's global environment. To contribute to the overall development by imparting moral, social and ethical values.

Mission

- The mission of the IT department is to prepare students for overall development including employability, entrepreneurship and the ability to apply the technology to real life problems by educating them in the fundamental concepts, technical skills/programming skills, depth of knowledge and development of understanding in the field of Information Technology.
- To develop entrepreneurs, leaders and researchers with exemplary level of employability even under highly competitive environments with high ethical, social and moral values.



Departmental Program Educational Objectives

PEO1: To produce IT graduates who have strong foundation in mathematics, sciences and basic engineering and prepare them with strong engineering knowledge.

PEO2: To provide technical competence to use techniques, skills and modern engineering tools that allows them to work effectively in areas such as Algorithms, Computer Organization, Information Systems, Networks and Multimedia.

PEO3: To train students with good practical knowledge by use of modern equipped laboratory and conduct the experiments, record, analyze and interpret data and also in scientific and engineering breadth that cover multi-disciplinary subjects enabling them to comprehend, analyze the Information Technology problems and develop solutions.

PEO4: To inculcate the students in professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach and an ability to relate engineering.

PEO5: To produce graduates who have the ability to pursue advanced studies and research in disciplines and develop consciousness on the issues of social concerns and enabling them to upgrade their personality and experiences through community services to have a meaningful linkage with the society.

PEO6: To provide graduates with the ability to upgrade their personality and develop an attitude for self-learning and life-long learning.



Index

Sr. No.	Contents	Page No.
1.	List of Experiments	1
2.	Experiment Plan and Lab Outcomes	2
3.	Study and Evaluation Scheme	4
4.	Experiment No. 1	4
5.	Experiment No. 2	28
6.	Experiment No. 3	34
7.	Experiment No. 4	39
8.	Experiment No. 5	42
9.	Experiment No. 6	47
10.	Experiment No. 7	55
11.	Experiment No. 8	59
12.	Experiment No. 9	65
13.	Experiment No. 10	72
14.	Experiment No. 11	82
15.	Experiment No. 12	90
16.	Experiment No. 13	99

List of Experiments

Sr. No.	Experiments Name
1	Study and installation of Windows 7 and Ubuntu.
2	Understand various networking commands – ARP, RARP, PING, TELNET, TRACRT, NSLOOKUP
3	To study and installation of NS2 Simulator.
4	Introduction to TCL Hello programming.
5	TCL script for creating Nodes.
6	TCL script for TCP configuration between four nodes.
7	Implement Ring topology in Ns2 simulator.
8	Study and analysis routing protocol and Shortest Path routing by DSDV.
9	Analysis of network performance by configuring the queue size and capacity of links for measuring QoS of generated traffic.
10	Comparative QoS parameters using Xgraph/gnuplot for different load conditions.
11	Socket Programming with C Client Server Model.
12	Installation of Wire shark and Analysis of Packet headers.
13	A case study to design and configure organization network.



Experiment Plan & Lab Outcome

Lab Outcomes:

LO1	Execute and evaluate network administration commands and demonstrate their use in different network scenarios
LO2	Demonstrate the installation and configuration of network simulator.
LO3	Demonstrate and measure different network scenarios and their performance behavior.
LO4	Analyze the contents the packet contents of different protocols.
LO5	Implement the socket programming for client server architecture.
LO6	Design and setup a organization network using packet tracer.

Module No.	Week No.	Experiments Name	Lab Outcome
1	W1	Study and installation of Windows 7 and Ubuntu.	LO2
2	W2	Understand various networking commands – ARP, RARP, PING, TELNET, TRACRT, NSLOOKUP	LO1
3	W3	To study and installation of NS2 Simulator.	LO2
4	W4	Introduction to TCL Hello programming.	LO3
5	W5	TCL script for creating Nodes.	LO3
6	W6	TCL script for TCP configuration between four nodes.	LO3
7	W7	Implement Ring topology in Ns2 simulator.	LO3
8	W8	Study and analysis routing protocol and Shortest Path routing by DSDV.	LO3
9	W9	Analysis of network performance by configuring the queue size and capacity of links for measuring QoS of generated traffic.	LO3
10	W10	Comparative QoS parameters using	LO3

		Xgraph/gnuplot for different load conditions.	
11	W12	Installation of Wire shark and Analysis of Packet headers.	LO4
12	W11	Socket Programming with C Client Server Model.	LO5
13	W13	A case study to design and configure organization network.	LO6

Study and Evaluation Scheme

Course Code	Course Name	Teaching Scheme			Credits Assigned			
ITL401	Networking LAB	Theory	Practical	Tutorial	Theory	TW/Practical	Tutorial	Total
		-	02	-	-	01	-	01

Course Code	Course Name	Examination Scheme		
ITL401	Networking LAB	Term Work	Oral	Total
		25	25	50

Term Work:

Term Work shall consist of at least 10 to 12 practical's based on the above list. Also Term work Journal must include at least 2 assignments.

Term Work Marks:

25 Marks (Total marks) = 15 Marks (Experiment) + 5 Marks (Assignments) + 5 Marks (Attendance)

Oral Exam: An Oral exam will be held based on the above syllabus.

Networking Lab

Experiment No. : 1

**Study and installation of Windows 7 and
Ubuntu.**

Experiment No. 1

1. Aim: Study and installation of Windows 7 and Ubuntu.

2. What will you learn by performing this experiment?

We will learn how to install or format our computer. How the partition is performed for different disk.

3. Software Required: Windows, Ubuntu

4. Theory :

Performing a New Installation of Windows 7

The three basic types of clean installation procedures are as follows:

- Install on a brand new disk or computer system
- Erase the disk, format it, and install
- Install into a new directory for dual-booting

If you intend to use either of the first two methods, be sure your computer can boot from a DVD (most newer computers support booting from a DVD drive). Doing so might require changing the drive boot order in the BIOS or CMOS, but try it first as-is. With no floppy disk inserted and a clean hard disk, try the DVD drive next. The Windows 7 DVD is bootable and should run the Setup program automatically. Installation takes 15 to 30 minutes, depending on the speed of your machine. Refer to the following sections if you have questions about any steps in this process.

Note: Windows 7 automatically applies the NTFS format to any disk partition upon which it is installed during a clean installation.

Typical Clean Setup Procedure

If you're installing into an empty partition and you can boot an operating system that is supported for the purpose of Setup (Windows Vista or XP), just boot up, insert the DVD and choose Install Now from the resulting dialog box. Then you can follow the installation step-by-step procedure.

If Windows doesn't detect the DVD automatically upon insertion, you must run the Setup program, setup.exe, from the Start, Run dialog box (after opening the Run dialog box, type D:/setup.exe; on Vista use the Start menu Search box instead [using the correct letter for your DVD drive if it isn't D]). The setup.exe application is located in the Sources directory on the DVD. After the Setup routine starts, you can follow the installation procedure step by step.

If your computer has a blank hard disk or your current OS isn't supported, this process changes. You must launch the installation process from the Windows 7 DVD (this works only if you can boot from the DVD drive). Setup automatically runs if you boot from the DVD.

Yet another setup method involves the network. To initiate a network installation, you must create a network share of the distribution DVD or a copy of the DVD on a hard drive. The destination system must have network access, and the user account must have at least read access to the installation files. Initiate Setup by executing setup.exe from the network share. For example, from the Start, Run command, or the Vista Start menu Search box, types this path: \\ \sources\Setup. Setup recognizes an over-the-network installation and automatically copies all files from the network share to the local system before the first reboot.

There are two options to choose from during the Windows 7 installation process:

Upgrade. This option replaces your current version of Windows with Windows 7, and keeps your files, settings, and programs in place on your computer.

Custom. This option replaces your current version of Windows with Windows 7, but doesn't preserve your files, settings, and programs. It's sometimes referred to as a clean installation for that reason.

Windows 7 DVD is bootable. In order to boot from the DVD you need to set the boot sequence. Look for the boot sequence under your BIOS setup and make sure that the first boot device is set to CD-ROM/DVD-ROM.

Step 1 - Place Windows 7 DVD in your DVD-ROM drive and start your PC. Windows 7 will start to boot up and you will get the following progress bar.

Step 2 - The next screen allows you to setup your language time and currency format, keyboard or input method. Choose your required settings and click next to continue.

Step 3 - The next screen allows you to install or repair Windows 7. Since we are doing a clean install we will click on "install now".

Step 4 - Read the license terms and tick I accept license terms. Then click next to continue.

Step 5 - You will now be presented with two options. Upgrade or Custom (Advanced). Since we are doing a clean install we will select Custom (Advanced).

Step 6 - Choose where you would like to install Windows 7. If you have one hard drive you will get a similar option to the image below. You can click next to continue. If you have more than one drive or partition then you need to select the appropriate drive and click next. If you need to format or partition a drive then click Drive options (advance) before clicking next.

Step 7 - Windows 7 starts the installation process and starts copying all the necessary files to your hard drive as shown on the image below.

Step 8 - It will go through various stages of the setup and will reboot your system few times.

Step 9 - When your PC reboots it attempts to boot from DVD as its the first boot device. Do not press any key during the boot prompt so Windows 7 will continue with the installation by booting from the hard drive.

Step 10 - After the reboot your computer will be prepared for first use.

Step 11 - At this stage you need to choose a user name and computer name. Click next to continue. The user account you create here is the Administrator account which is the main account for your Windows 7 that has all the privileges.

Step 12 - Choose your password and password hint just in case you forget your password and need to jog your memory.

Step 13 - You can now type the product key that came with Windows 7 and click next. If you do not enter the product key you can still proceed to the next stage. However Windows 7 will run in trial mode for 30 days. You must therefore activate Windows within 30 days otherwise you can not access your computer after 30 days.

Step 14 - Help protect your computer and improve Windows automatically. Choose Use recommended settings.

Step 15 - Review your time and date settings. Select your time zone, correct the date and time and click next to continue.

Step 16 - Select your computer's current location. If you are a home user then choose Home network otherwise select the appropriate option.

Step 17 - Windows will now finalize the settings for your computer and restart.

Step 18 - After the final restart Windows 7 will start to boot up.

Step 19 - Finally you have the logon screen. Just type your password and press enter or click on the arrow to logon to Windows 7 for the first time.

Step 20 - After you have logged on to Windows 7 for the first time, you will see similar desktop to the image below. At this point you can start using your computer. However it may not be fully configured. You need to make sure that all the hardware is detected correctly and the necessary device drivers are installed. This can be done from the device manager.

Step 21 - To go to device manager click - Start Menu -> Control Panel -> System and Security -> System -> Device Manager. You will see all your hardware listed as shown on the image below. You need to check if you have any yellow exclamation marks next to the name of the devices, similar to "Multimedia Audio Controller" on the image below. This indicates that the driver has not been installed for this device.

At this stage you can install the driver for this device. To do so, Right Mouse click on Multimedia Audio Controller -> Update Driver Software...

Step 22 - You can choose to "Search automatically for updated driver software" or "Browse my computer for driver software". If you have the driver CD or if the driver is on a USB drive then choose "browse my computer for driver software". Windows 7 will search and install the driver from the CD or you can locate the driver manually.

Once you have removed all the yellow exclamation marks from the device manager your Windows 7 configuration would be fully complete.

Performing a New Installation of Linux

As the open source revolution grows around the world, more and more people are starting to switch over to the Linux Operating System and pre-eminent of all the Linux OS is the Red Hat Linux, owned and distributed by the Red Hat Inc. However, installation of Linux itself is seen as a rather arduous and herculean task among many beginners/inexperienced users. Steps to easily install red hat linux :

Step 1 – Insert the Red Hat Linux DVD into the DVD-drive of your computer. As soon as the following screen pops up, press ‘Enter’ to install Red Hat Enterprise Linux (RHEL) through GUI mode.

Step 2- RHEL installer would then prompt you conduct a check as to whether the CD media from which you’re installing is functioning correctly or not. Choose ‘Skip’, press enter and the installation would begin.

Step 3- Next, we need to select the language- English or any other language as per your preference, and then press ‘Next’ .

Step 4- In this step, the RHEL installer would ask you about the appropriate type of keyboard for the system. We take the ‘US English’ keyboard, you can pick any other

option depending on the type of your keyboard. Then press 'Next' to move to the next step.

Step 5- Next, the installer would ask for an 'installation number' if you wish to install full set of Red Hat functionalities. Enter the installation number and press 'OK' if you have an officially licensed installation number(for corporate clients that buy Red Hat's backup support and full features).

Others can select 'Skip entering installation number' and press 'OK' to proceed. RHEL would show a warning message, press 'Skip' in it to continue.

Step 6- The Red Hat installer would then require you to create partitions in your computer's hard disk for the installation. You can do it in four ways but the simplest way is to select 'Use free space on selected drives and create default layout' as this option will not affect any other OS residing in your system.

Check the 'review and modify portioning layout' to create partitions and click next.

Step 7- In this step you must create the required system partitions and mount points such as '/boot', '/home', 'swap' etc which are required for the Linux's proper functioning.

To create different partitions such as /home, /var etc, click on 'New' to create the partitions.

Then, select /home in the mount point and choose 'ext3' as the file system and give the desired size for it and then click 'OK'. Similarly also create /boot and /var.

Also, create a swap partition by clicking on 'New' and then choosing the file system as 'swap' and also give the size of Swap partition.(Usually size of swap partition SHOULD BE twice the size of RAM available to the system but you can keep its size less than that too)

Once you have made all the desired partitions and given their mount points, click 'Next' to continue installation.

Step 8- This step pertains to the default OS that will be loaded by the GRUB loader

(Note- If you have multiple Operating Systems installed, you would see multiple options here and you have to check in front of the OS name that you want to be loaded by default when the system is started.)

Click 'Next' to continue.

Step 9- This step pertains to the network settings of the Linux system that you are going to install. You can select the Ethernet devices through which the system would communicate with other devices in the network.

You can also provide the hostname, Gateway address and DNS address to the system during this step. (However it's better to adjust these settings once the system has been fully installed).

Step 10- The next step is to adjust the system clock to your particular time zone. Select your time zone and then click 'Next'.

Step 11 – This is a very important step that deals with the root(super-user) password for the system . Type the password and confirm it and then click next.

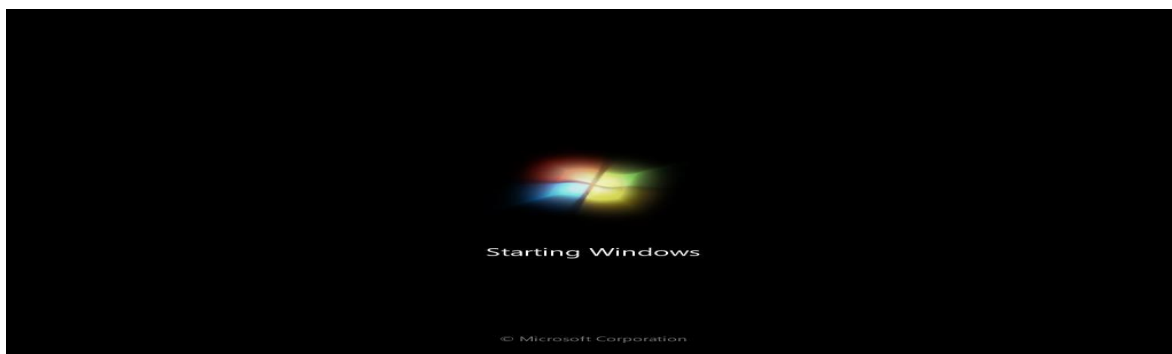
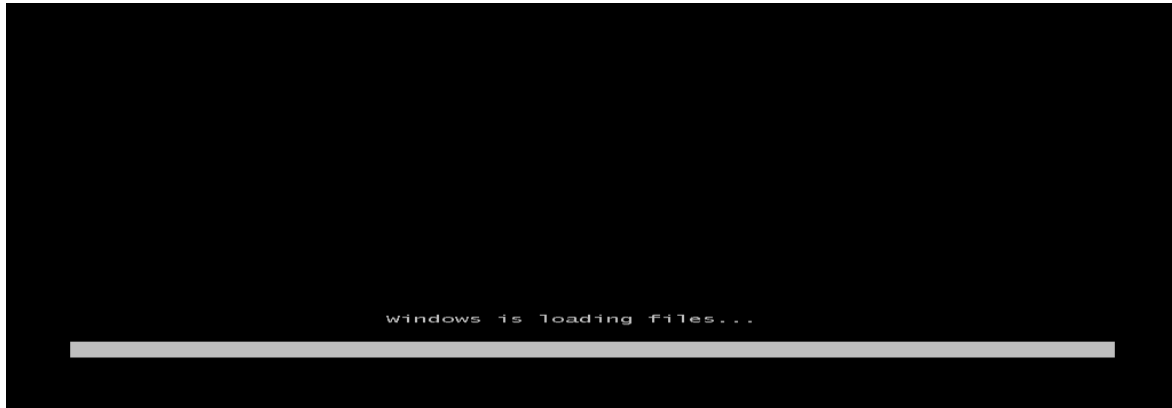
Step 12 – The RHEL installer would then prompt you about if you wish to install some extra 'Software Development' or 'Web Server' features. By default, keep it at 'Customize later' and press 'Next'.

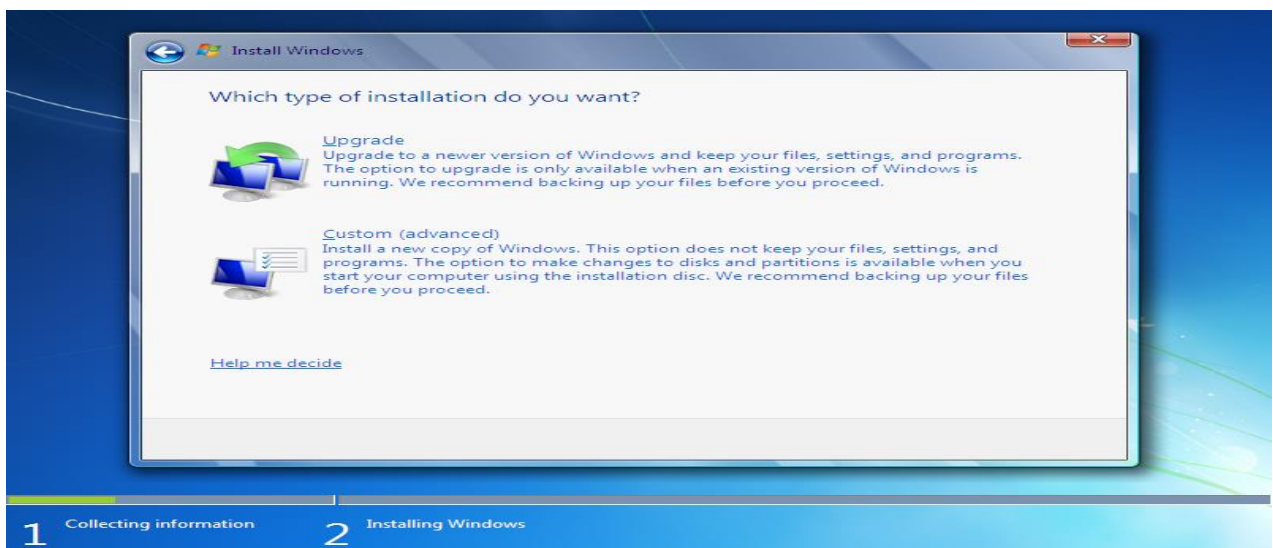
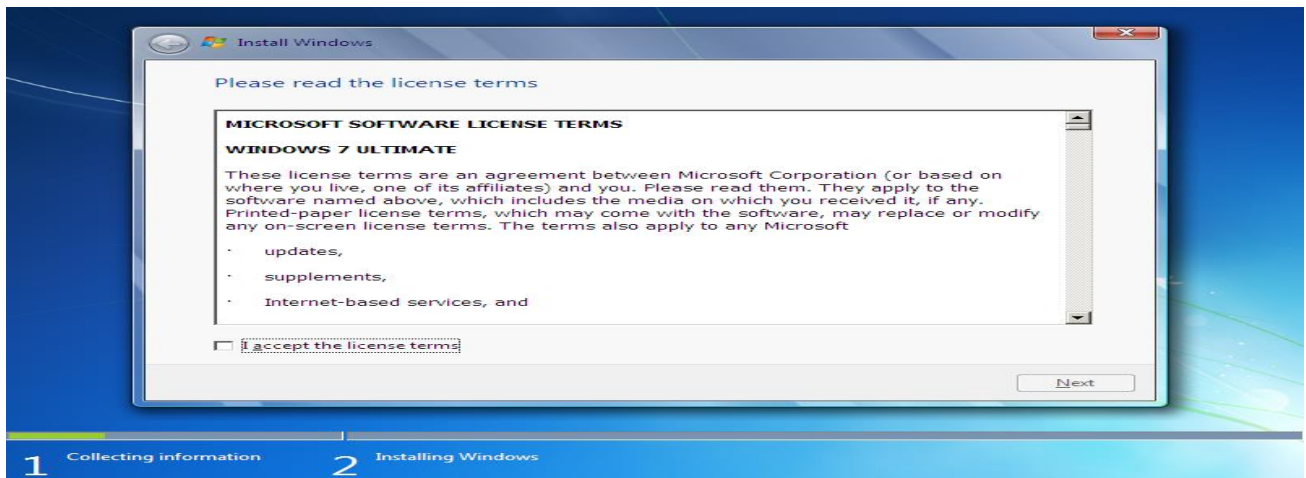
Step 13- This next step will initiate the installation of Red Hat Linux, press 'Next' to begin the process.

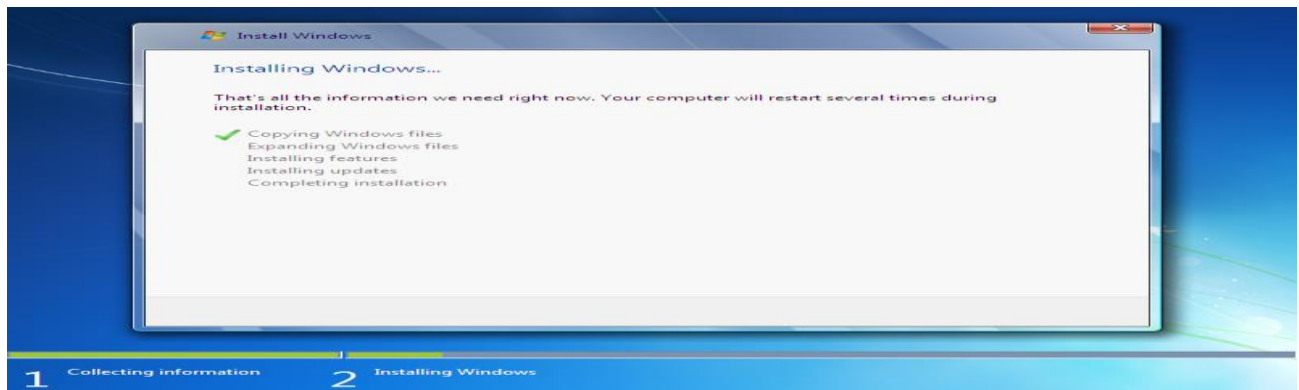
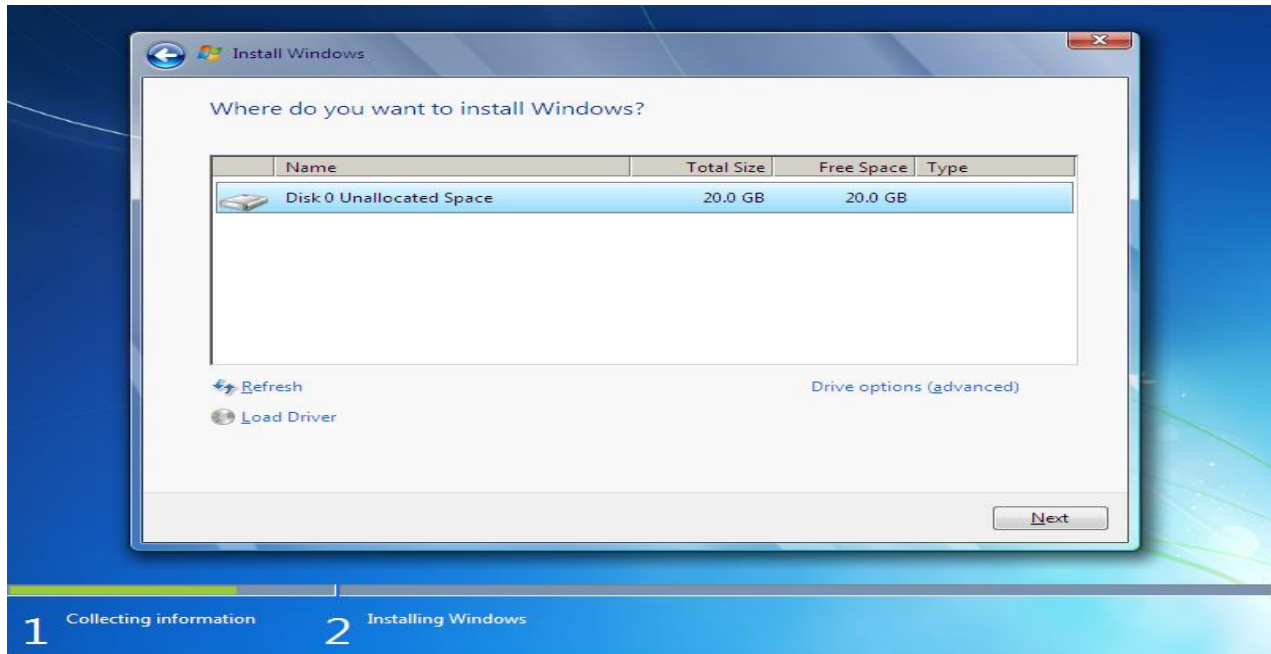
Step 14- Upon the completion of installation you should the following screen. Press Reboot and you'd be ready to use your newly installed Red Hat Linux OS.

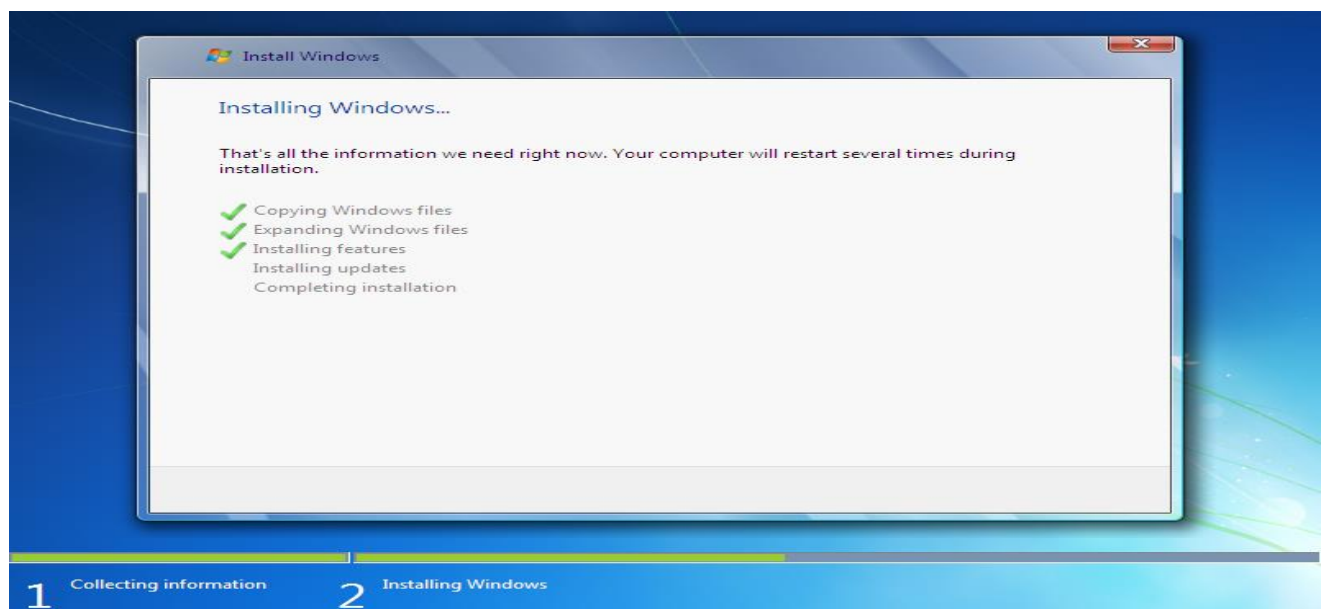
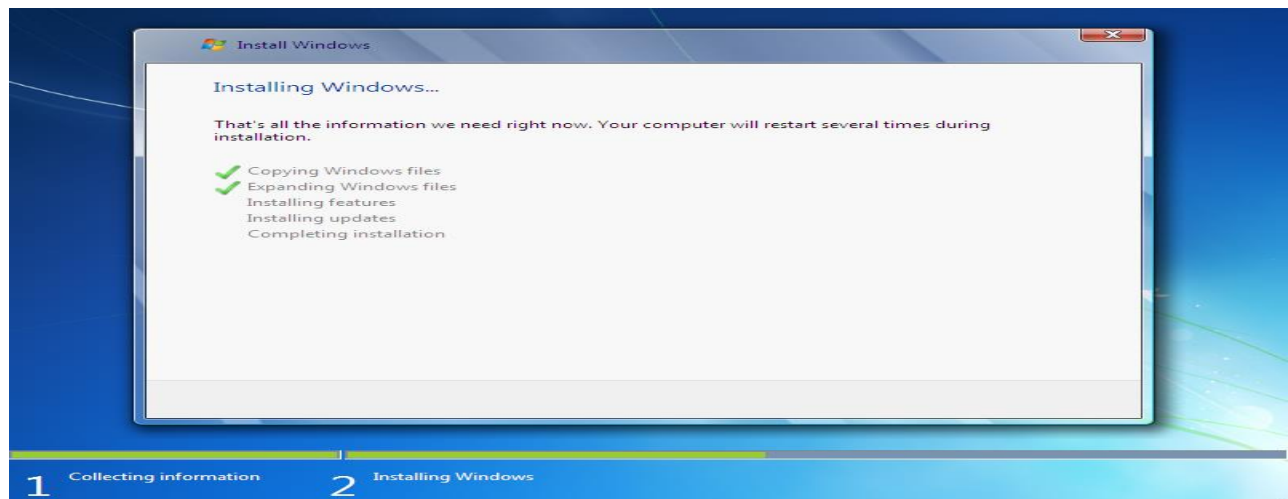
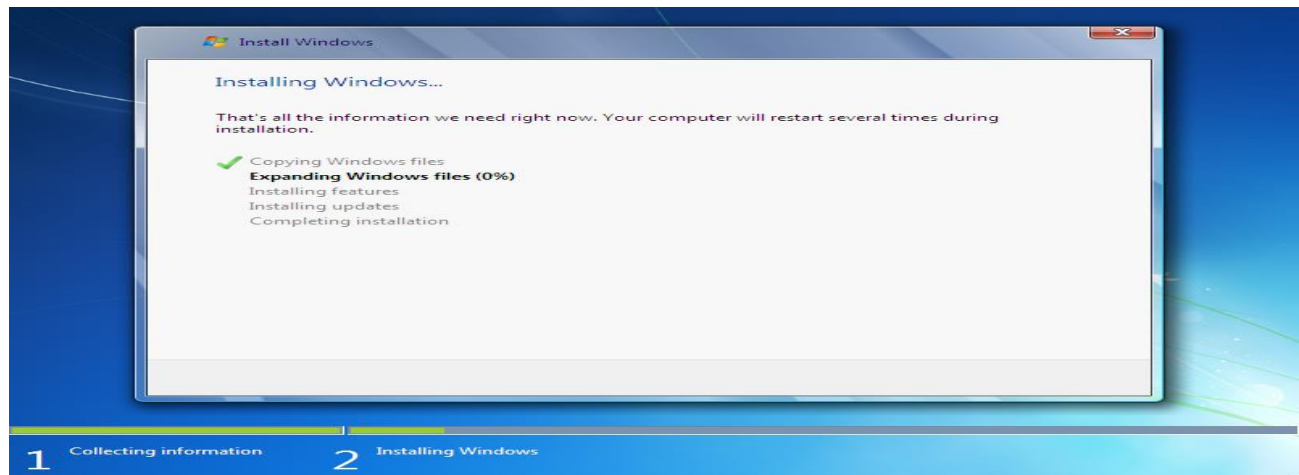
Conclusion and Discussion: Installation of Window 7 and ubuntu is perform in dual boot form. For windows we required original software of windows instead Ubuntu is open source.

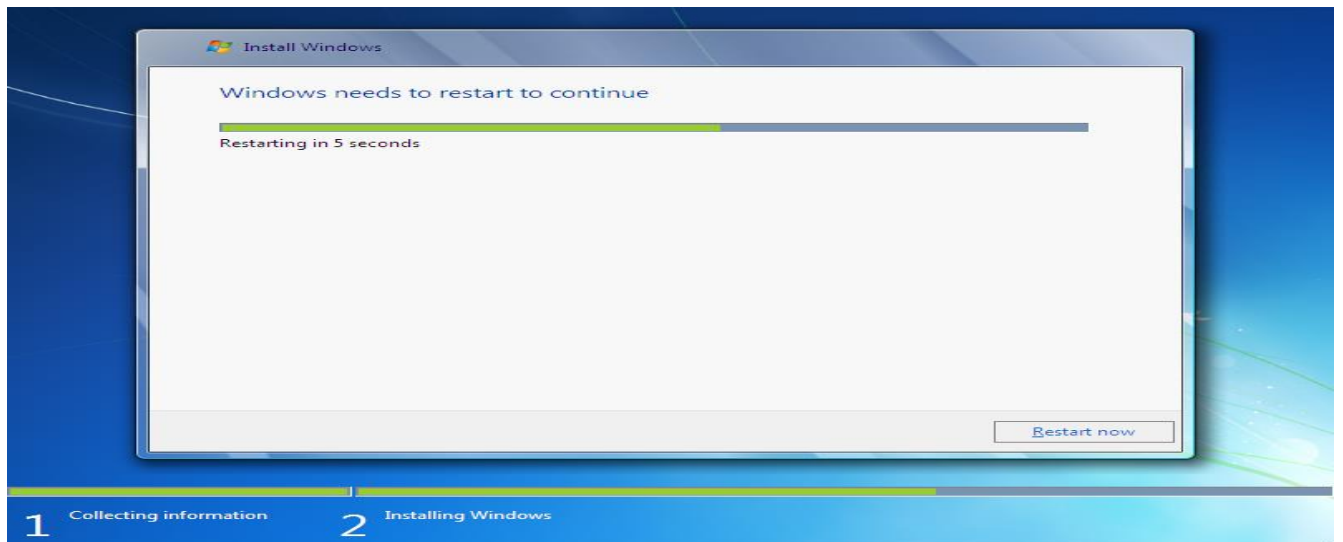
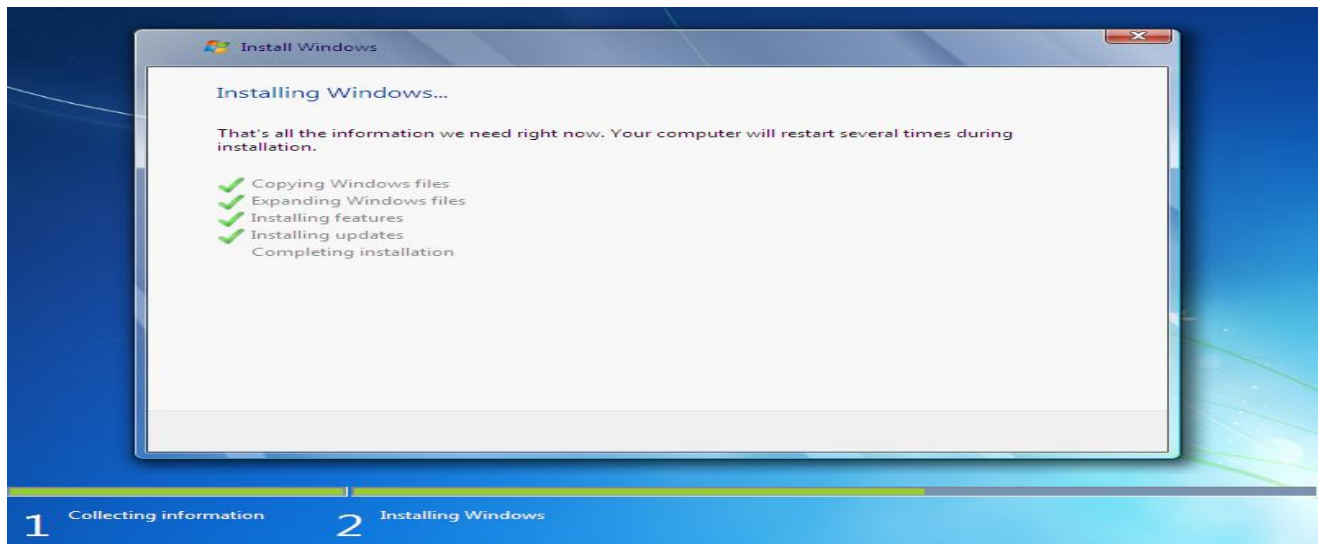
Results:

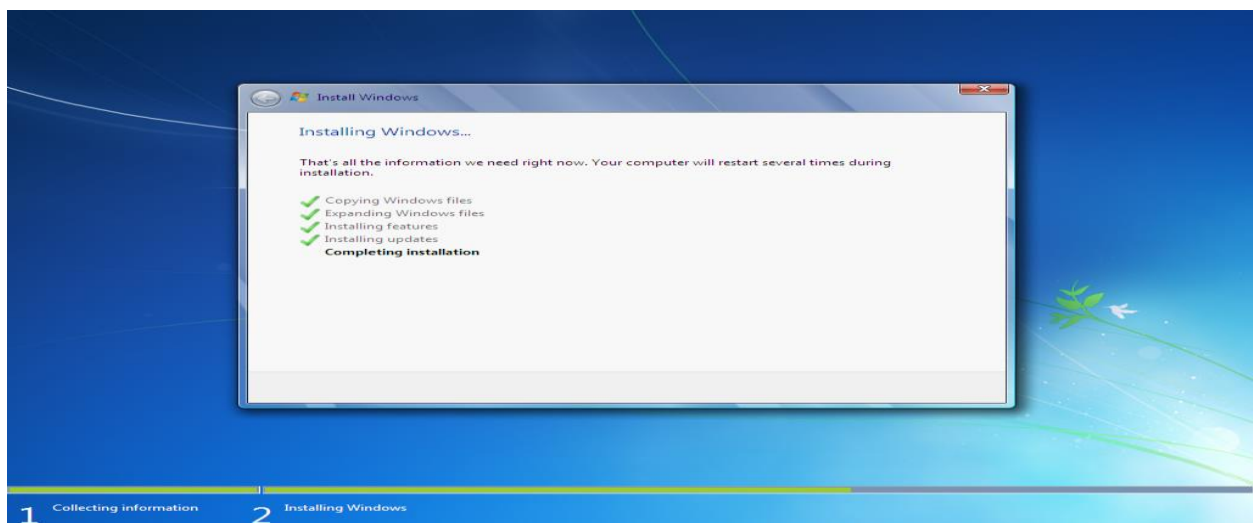
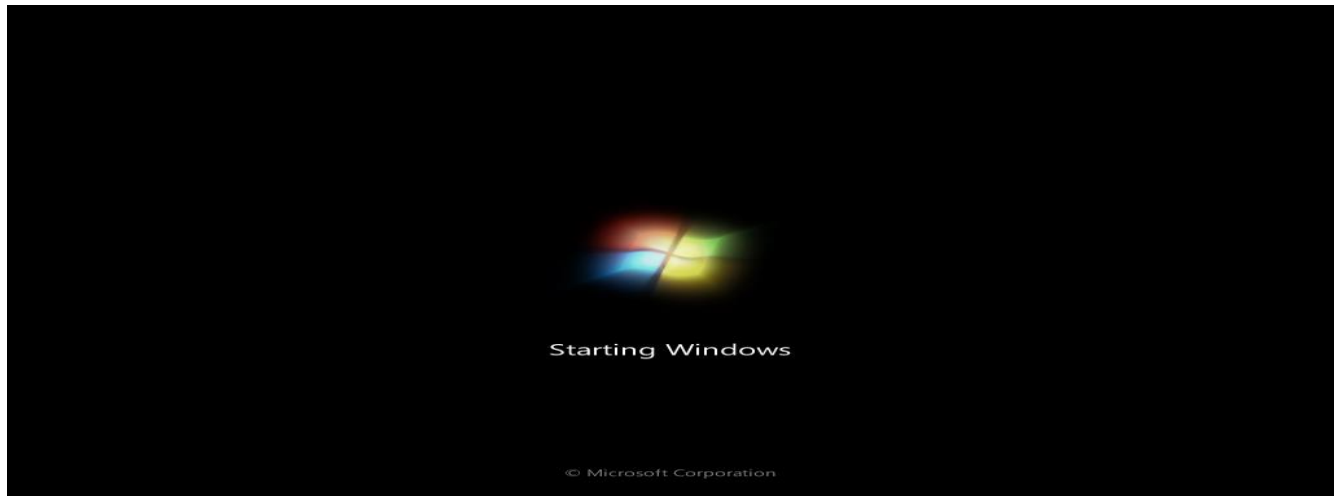


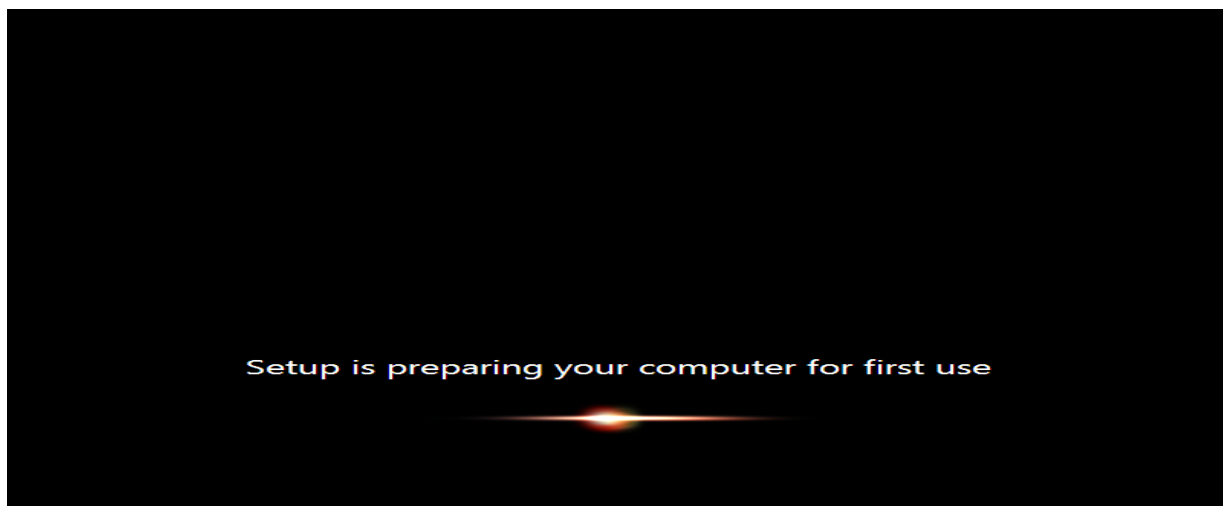
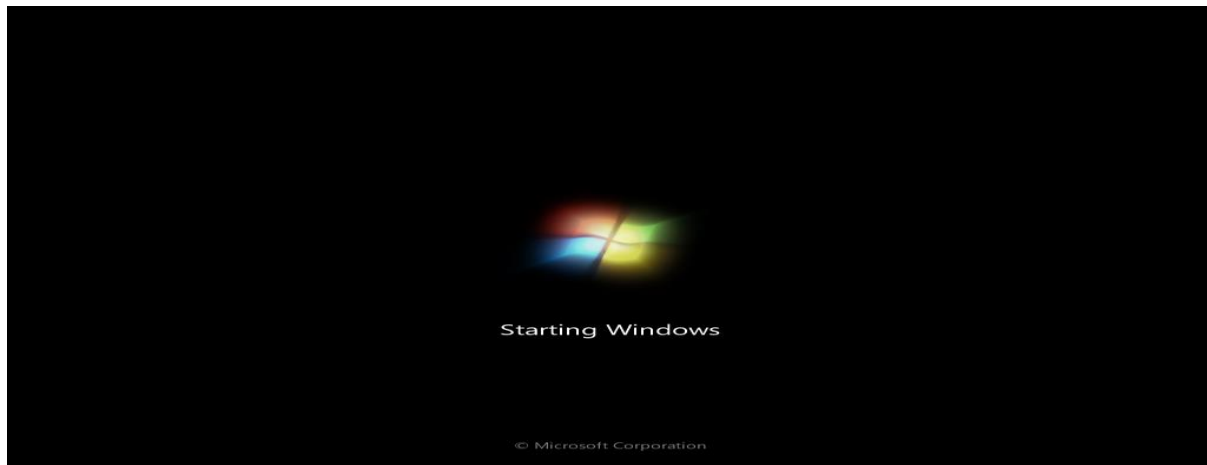
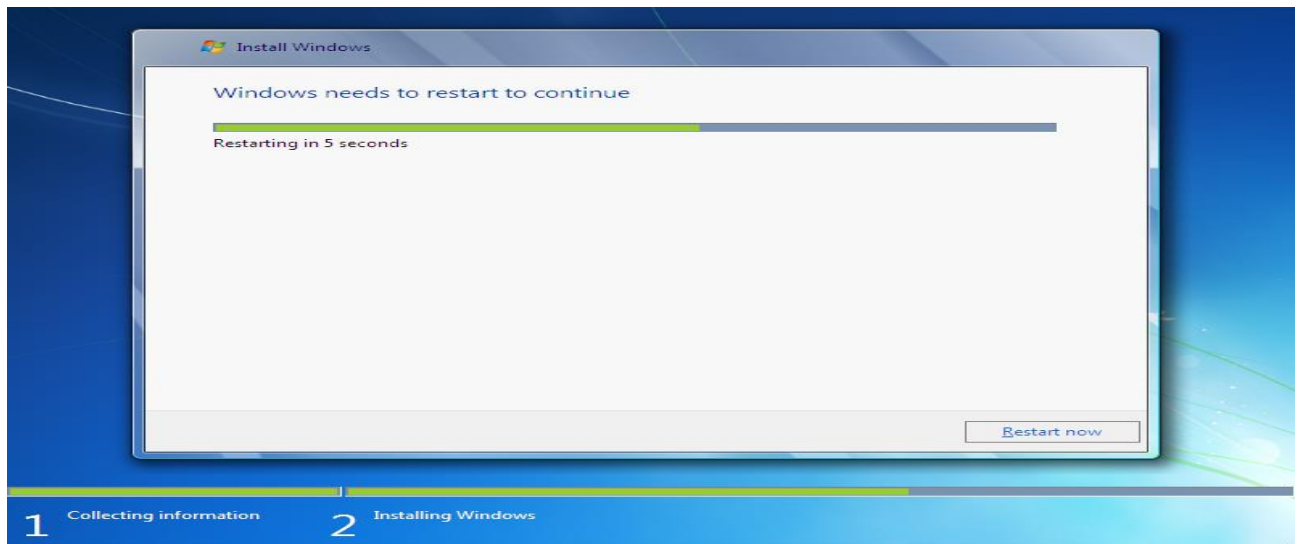




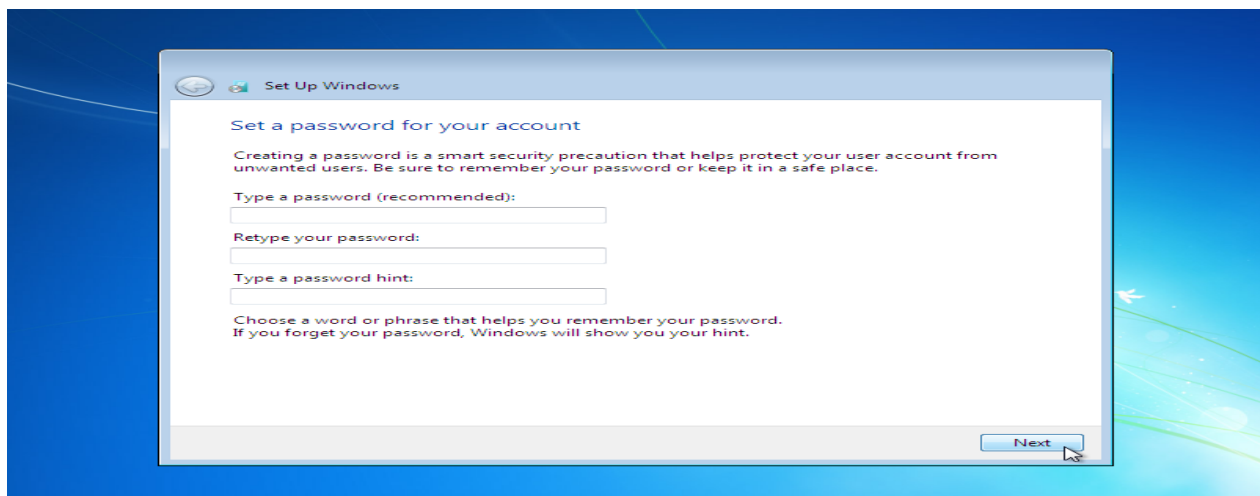
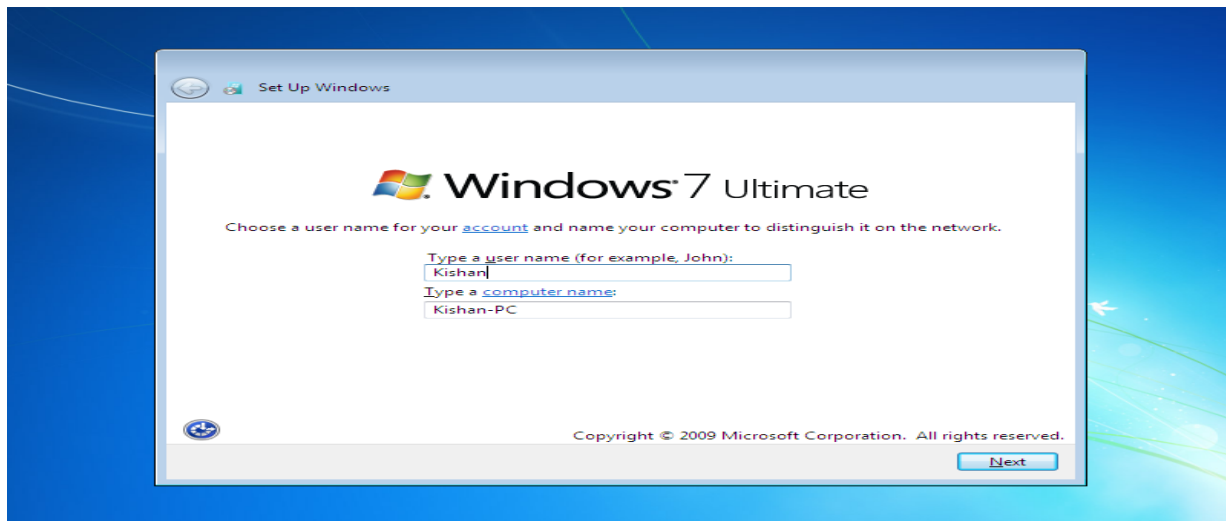


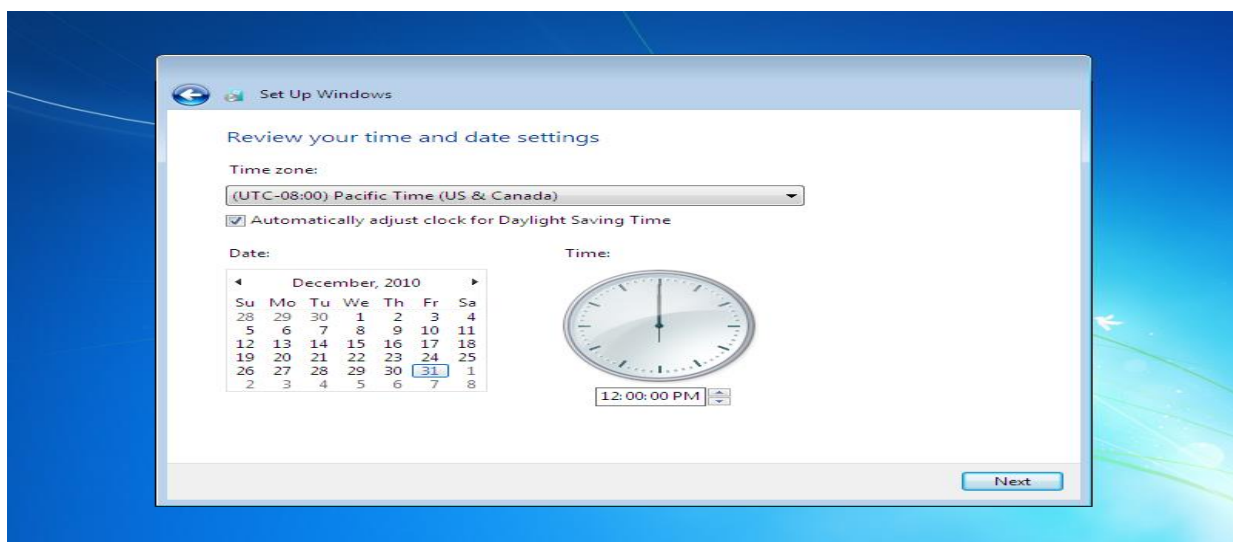
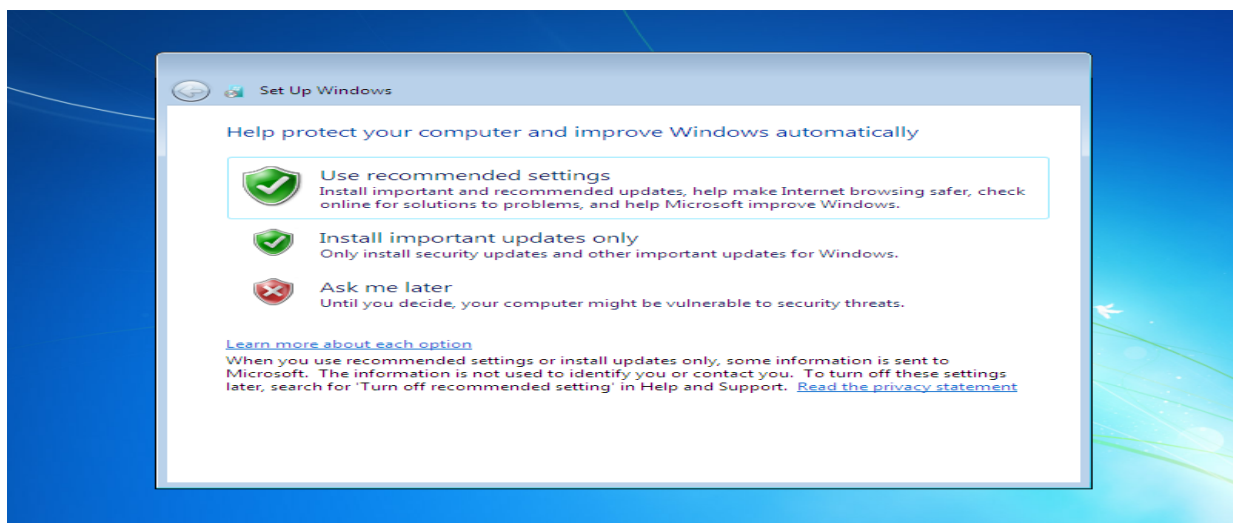
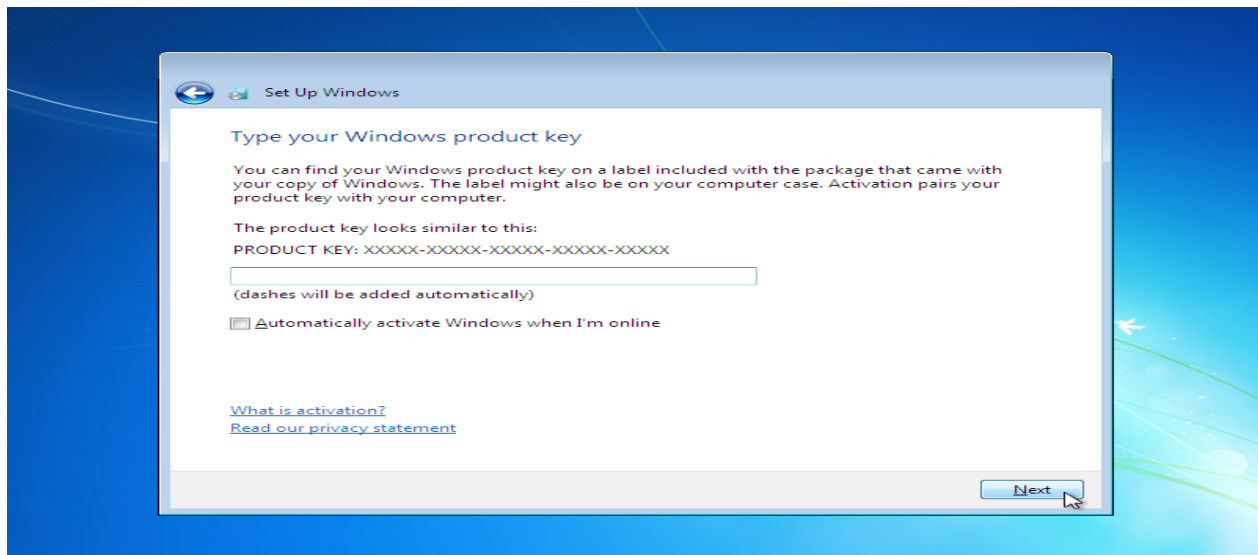


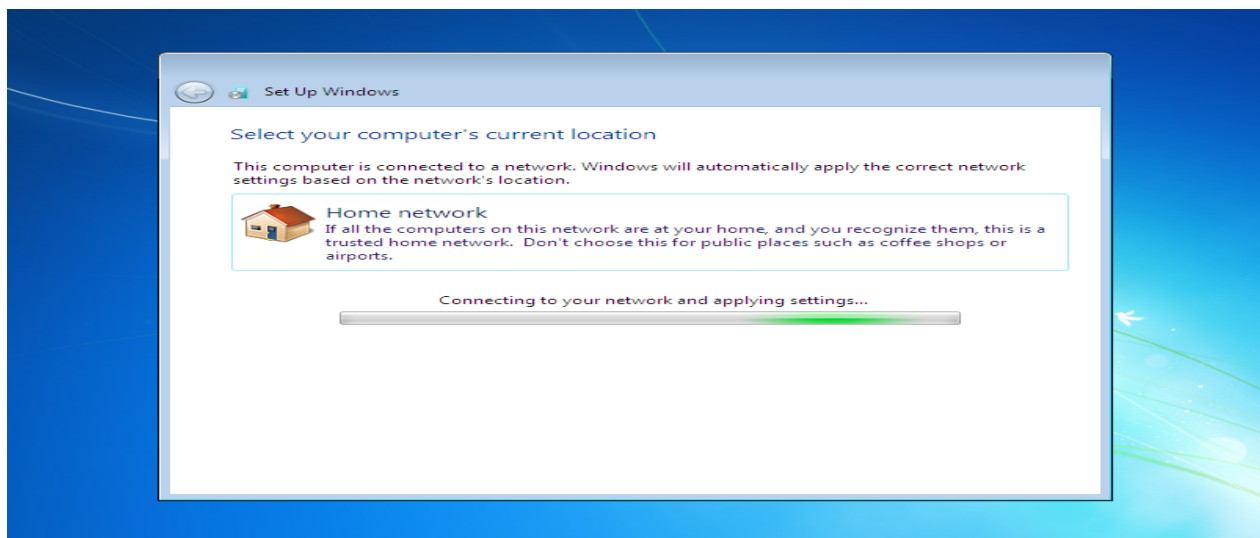
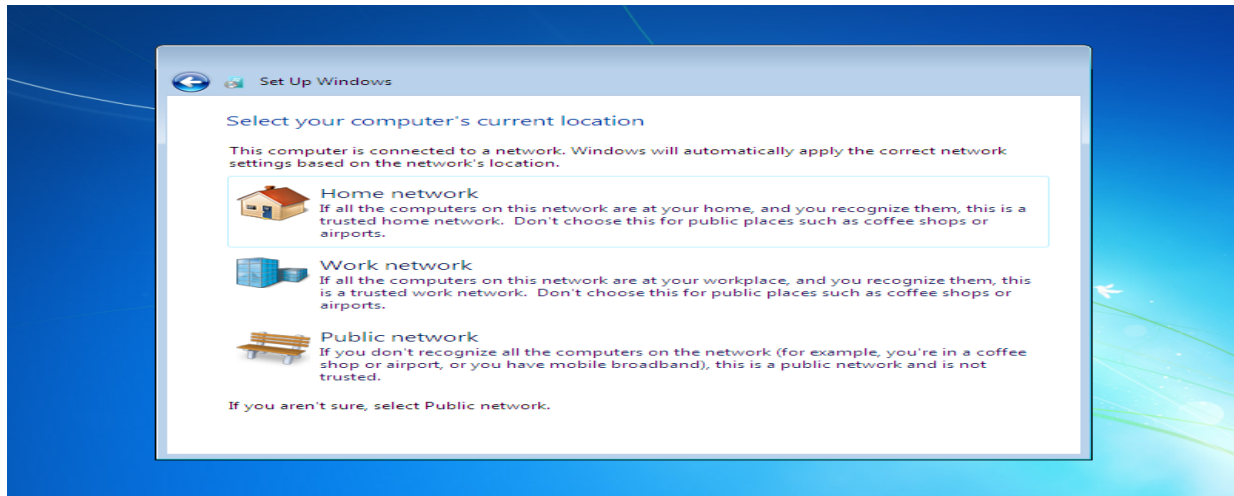


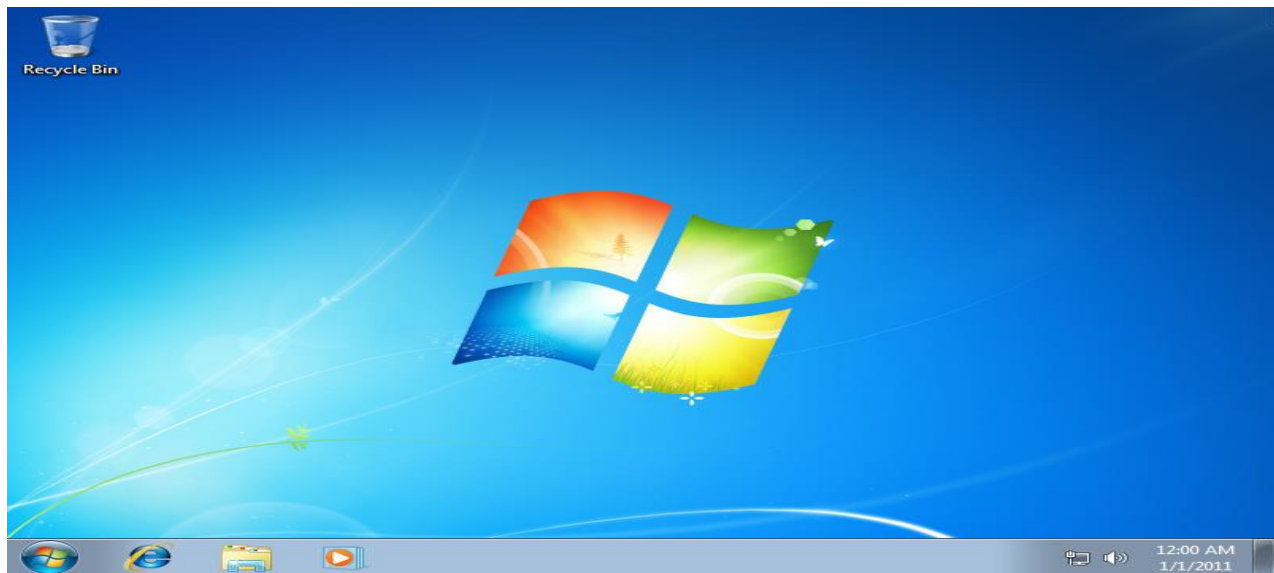


Setup is checking video performance

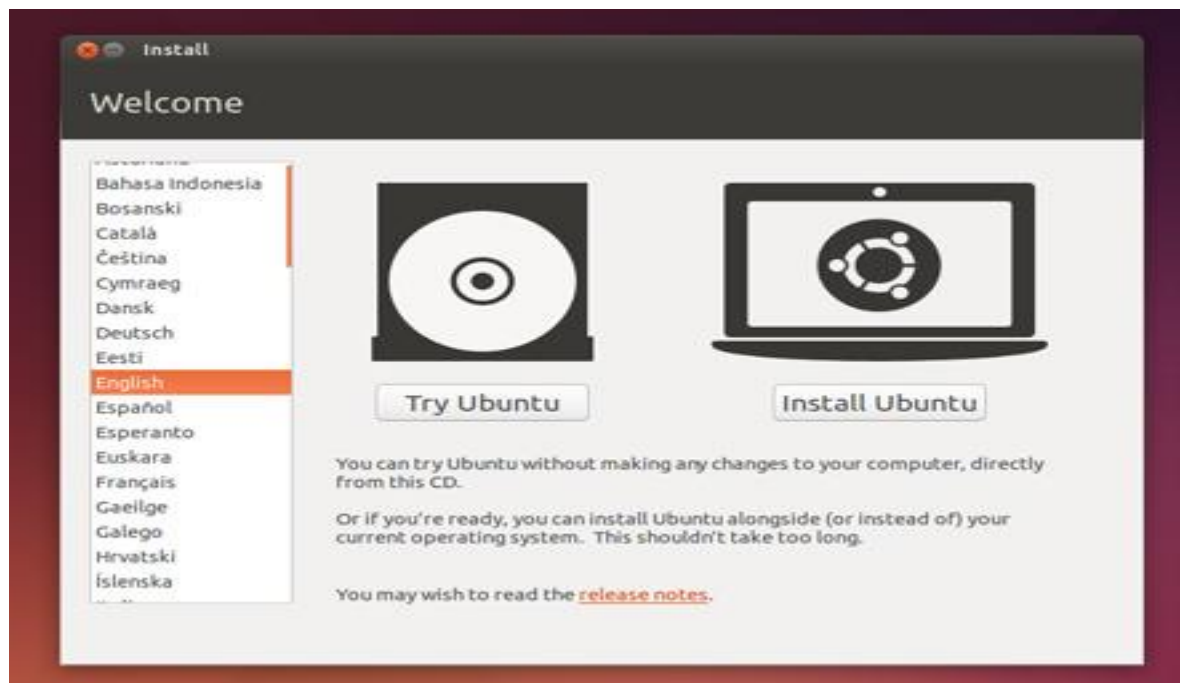


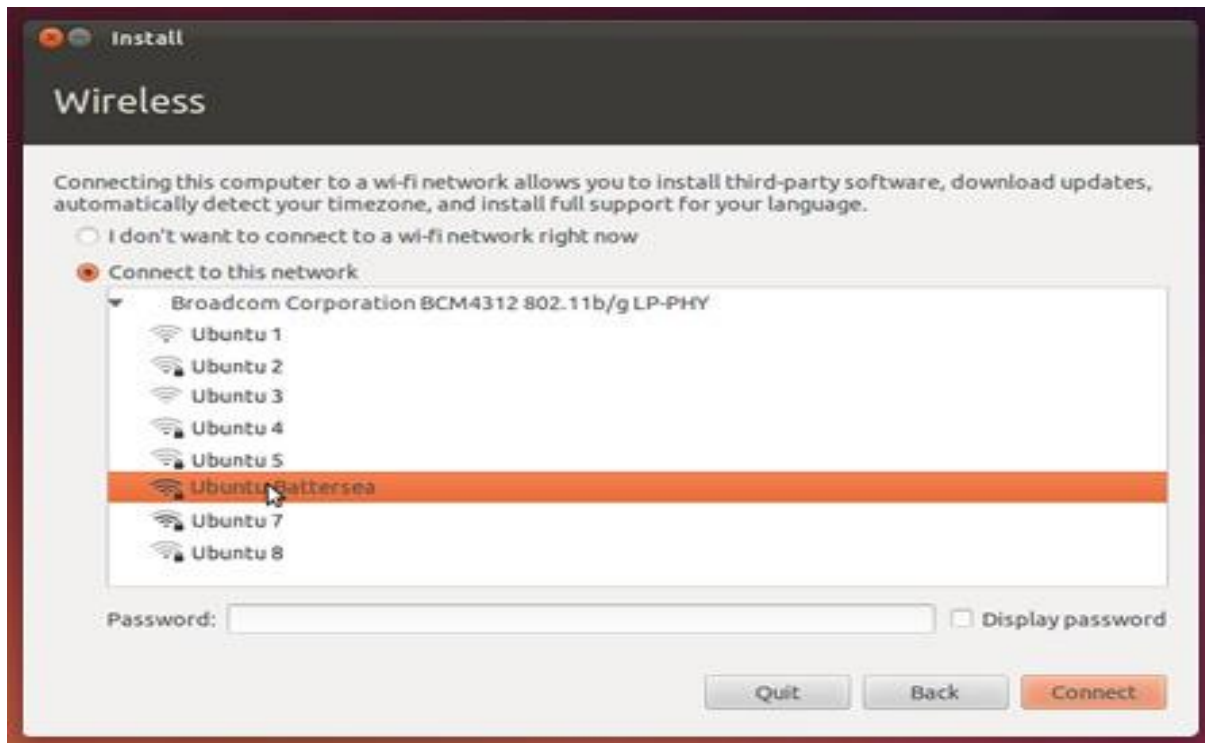


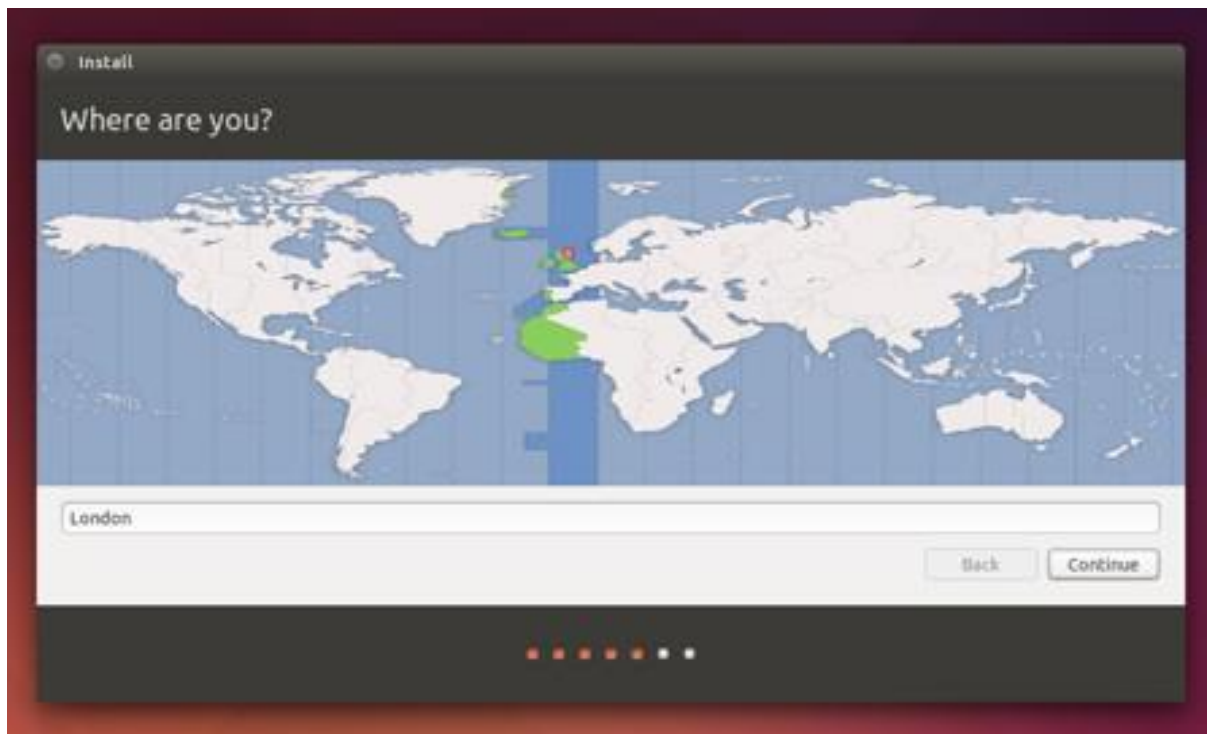


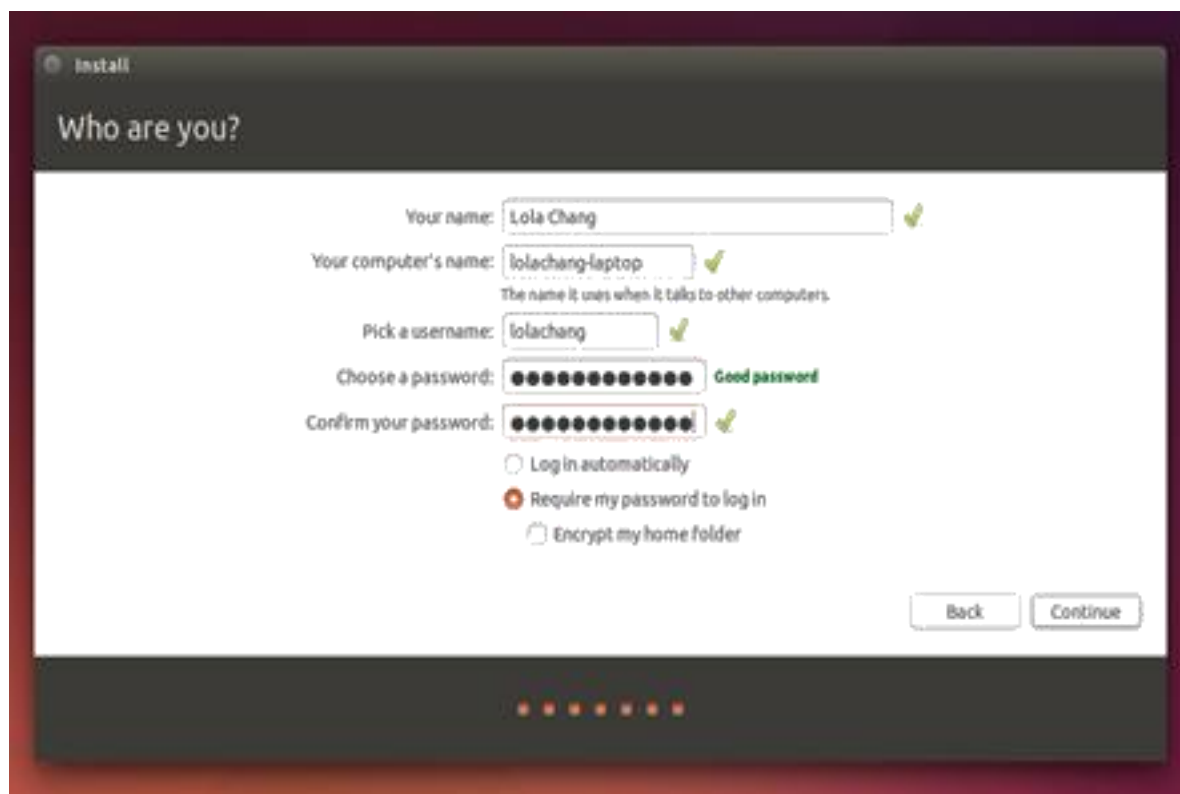
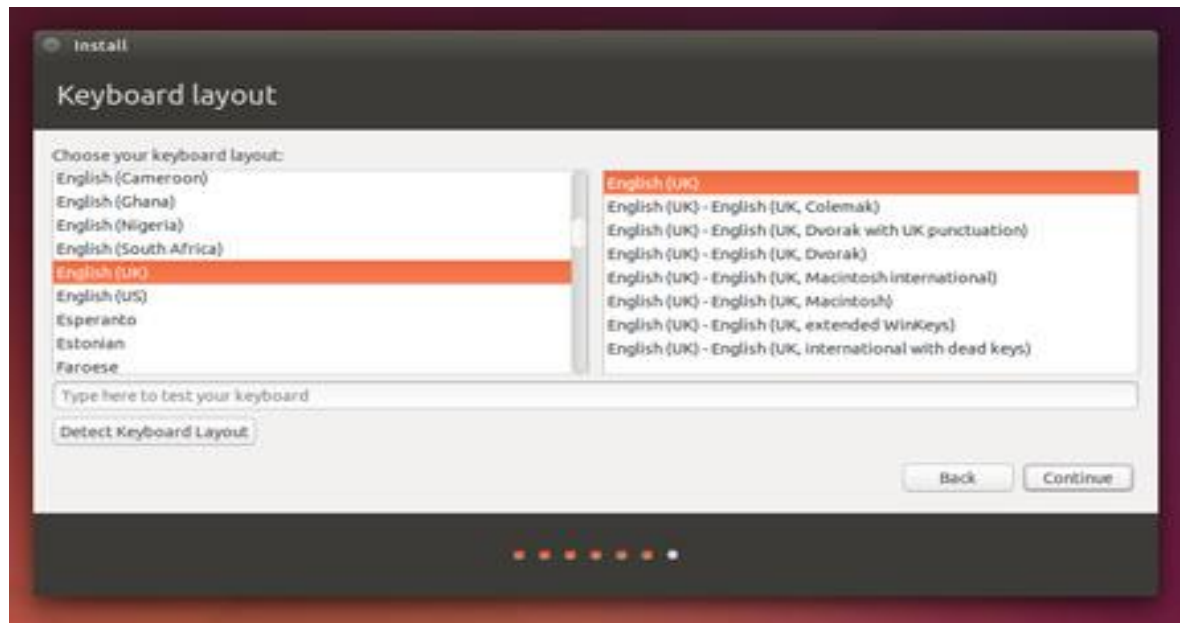


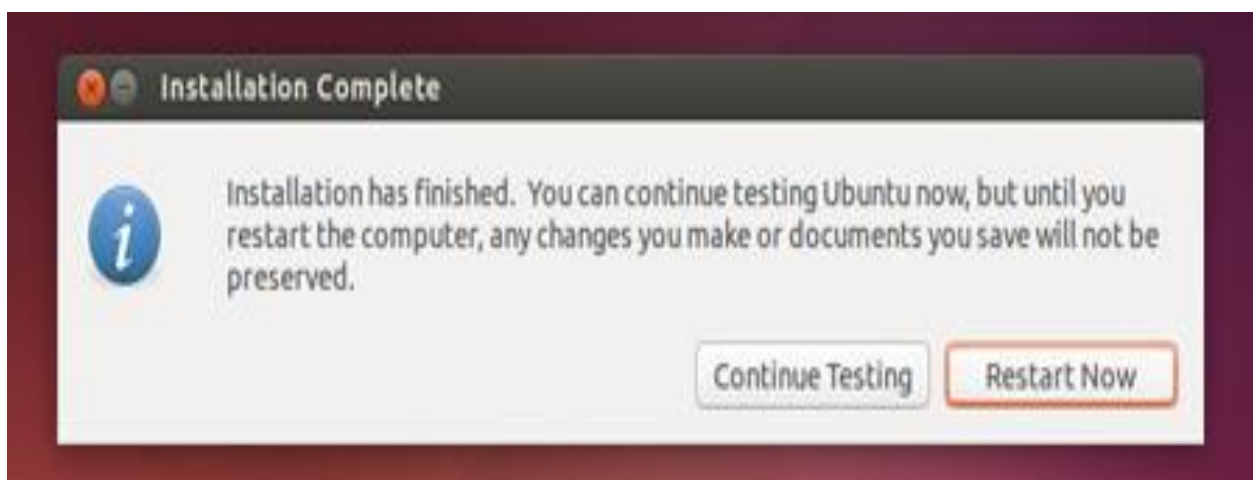
Ubuntu











5. Viva Questions:

- Q1: Dual boot installation is performed? Explain?
- Q2: Base of Ubuntu
- Q3: Who developed windows and Ubuntu

6. References:

1. A. S. Tanenbaum, "Computer Networks", 4th edition, Prentice Hall
2. B. F. Ferouzan, "Data and Computer Communication", Tata McGraw Hill.
3. dtsupport.ztsystems.com/windows7/WINDOWS7INSTALL.pdf
4. <https://help.ubuntu.com/16.04/installation-guide/i386/install.en.pdf>

Networking Lab

Experiment No. : 2

Understand various networking
commands – ARP, RARP, PING,
TELNET, TRACET, NSLOOKUP

Experiment No. 2

1. **Aim:** Understand various networking commands – ARP, RARP, PING, TELNET, TRACET, NSLOOKUP

2. **What will you learn by performing this experiment?**

All are basic commands which are required for networking. Used in troubleshooting utilities, checking online status, etc..

3. **Software Required:** Windows, Ubuntu

4. **Theory:**

ARP:

The Arp command lets you view and manage the Address Resolution Protocol (ARP) cache. In other words, Arp displays and modifies the IP address-to-MAC address translation tables used by the ARP protocol. In order for the Arp command to be meaningful and helpful, you need to first understand the purpose of the Address Resolution Protocol. As DNS translates between host names and IP addresses, ARP translates between MAC addresses (Layer 2) and IP addresses (Layer 3). When a host attempts to communicate with another host on the same subnet, it must first know the destination host's MAC address. If there is no entry in the sending host's ARP cache for the destination MAC address, ARP sends out a broadcast (to all hosts in the subnet) asking the host with the target IP address to send back its MAC address. These IP-to-MAC mappings build up in the ARP cache which the arp command lets you view and modify.

Be aware that the ARP cache is a tempting target for hackers. It can be vulnerable to cache poisoning attacks in which false entries are inserted into the ARP cache, causing the compromised host to unknowingly send data (often unencrypted) to the attacker.

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 or cache. A network administrator creates a table in a local area network's gateway router that maps the physical machine (or Media Access Control - MAC address) addresses to corresponding Internet Protocol addresses. When a new machine is set up, its RARP client program requests from the RARP server on the router to be sent its IP address. Assuming that an entry has been set up in the router table, the RARP server will return the IP address to the machine which can store it for future use.

PING:

The ping (Packet InterNet Groper) command is arguably the most useful networking troubleshooting utility; it is definitely the simplest and most used. Ping tests the online status of a host on an IP network and measures the total round-trip time (in milliseconds) for packets sent from the source host to a destination host and back. Ping does this by sending Internet Control Message Protocol (ICMP) packets (usually). ICMP is a protocol that works with IP to provide error checking and reporting functionality. Ping sends ICMP echo requests to a remote host. If the host is able to respond, it replies with echo reply.

When you ping a destination name, the replies will tell you the host's IP address, the number of bytes sent, round-trip time, and the packets' Time to Live (TTL). When you ping a destination IP address, it responds with all the above except for the host name (unless you provide the -a switch). Ping doesn't just test connectivity – it can also verify that TCP/IP is installed correctly and that DNS name resolution is working properly. Standard ICMP pings do not utilize port numbers or use TCP or UDP; ICMP is a Layer 3 (Network layer) protocol. TCP and UDP operate at Layer 4, the Transport layer. The ping command can be misused in a variety of denial of service attacks, such as the Ping of Death, Smurf attack, and Ping flood. However, ICMP serves more functions than just enabling ping requests and replies. ICMP is important in several other ways for proper IP function (such as MTU discovery, unreachable notifications, etc.) so disabling it altogether is not advised.

TELNET:

Telnet is a network protocol that allows for connections and user sessions on remote hosts. It is called a 'terminal emulator' because it lets you work on a remote host as if you were seated in front of it and using its monitor, keyboard, and mouse. A computer terminal is defined as "an electronic or electromechanical hardware device that is used for entering data into, and displaying data from, a computer or a computing system." The input and output devices described above create a terminal environment, and telnet lets you mimic that environment on another computer (also referred to as "getting a shell" on the remote host). Any information security professional worth his salt will tell you that using telnet is not recommended as all logins, passwords and commands are transferred in clear text. An attacker may eavesdrop on a telnet session and obtain the credentials and sensitive data of other users. The Secure Shell (SSH) protocol has effectively replaced telnet; however, in closed, restricted networks where there is zero chance of an attacker performing packet sniffing, telnet can be used. You should still make efforts to phase telnet out and replace it with SSH because it is a very important and useful protocol to learn. This page supposedly lists free telnet servers that you can connect to and practice on. By running a telnet connection to the open ports on a host, you can view the banners that reveal which service is answering on those specific ports. Many email, FTP, and web servers will respond to telnet connection requests with the name and version of their email, FTP, and web software (such as Exchange, IIS, etc).

TRACEROUTE:

The traceroute command is used to discover the routes that packets actually take when traveling to their destination. The device (for example, a router or a PC) sends out a sequence of User Datagram Protocol (UDP) datagram's to an invalid port address at the remote host. Three datagram's are sent, each with a Time-To-Live (TTL) field value set to one. The TTL value of 1 causes the datagram to "timeout" as soon as it hits the first router in the path; this router then responds with an ICMP Time Exceeded Message (TEM) indicating that the datagram has expired. Another three UDP messages are now sent, each with the TTL value set to 2, which causes the second router to return ICMP TEMs. This process continues until the packets actually reach the other destination. Since these datagram's are trying to access an invalid port at the destination host, ICMP Port Unreachable Messages are returned, indicating an unreachable port; this event signals the Traceroute program that it is finished. The purpose behind this is to record the source of each ICMP Time Exceeded Message to provide a trace of the path the packet took to reach the destination.

NSLOOKUP:

Nslookup ("name server lookup") is a DNS query tool. This means that nslookup checks DNS records, domain host aliases, domain host services, and operating system information by querying DNS servers. Nslookup can also be used to perform DNS zone transfers and is useful when performing network foot printing during ethical hacking efforts. It has two modes: interactive and non interactive. Interactive mode queries DNS servers for details about various hosts and domains. Non interactive mode prints only the name and requested information for a host or domain.

Although it is still available by default on Windows and Linux/Unix, nslookup has been deprecated and further use is discouraged. It has effectively been replaced by its successors, the dig (Domain Information Groper) and host utilities. Unlike nslookup, they are not available natively on Windows and must be installed manually. There is a host command in Windows Power Shell but that is something different. You can install the Windows versions of dig and host by extracting them from BIND

Dig is basically an improved version of nslookup. Host enables quick lookups of DNS server information and is used to find 1) the IP address of a given domain name and 2) the domain name of a given IP address.

5. **Conclusion and Discussion:** we executed all commands in windows and Ubuntu also. All commands are useful in networking to perform some operation.

Results :

```

C:\Windows\system32>arp -a

Interface: 192.168.0.117 --- 0xa
Internet Address      Physical Address      Type
192.168.0.1           00-21-70-c0-9e-01    dynamic
192.168.0.255         ff-ff-ff-ff-ff-ff    static
224.0.0.22            01-00-5e-00-00-16    static
239.255.255.250       01-00-5e-7f-ff-fa    static
255.255.255.255       ff-ff-ff-ff-ff-ff    static

C:\Windows\system32>arp -g

Interface: 192.168.0.117 --- 0xa
Internet Address      Physical Address      Type
192.168.0.1           00-21-70-c0-9e-01    dynamic
192.168.0.255         ff-ff-ff-ff-ff-ff    static
224.0.0.22            01-00-5e-00-00-16    static
239.255.255.250       01-00-5e-7f-ff-fa    static
255.255.255.255       ff-ff-ff-ff-ff-ff    static

C:\Windows\system32>arp -s 192.168.0.22 aa-aa-aa-11-22-33

C:\Windows\system32>arp -g

Interface: 192.168.0.117 --- 0xa
Internet Address      Physical Address      Type
192.168.0.1           00-21-70-c0-9e-01    dynamic
192.168.0.22          aa-aa-aa-11-22-33    static
192.168.0.255         ff-ff-ff-ff-ff-ff    static
224.0.0.22            01-00-5e-00-00-16    static
239.255.255.250       01-00-5e-7f-ff-fa    static
255.255.255.255       ff-ff-ff-ff-ff-ff    static

C:\Windows\system32>

```

```

C:\>tracert mediacollege.com

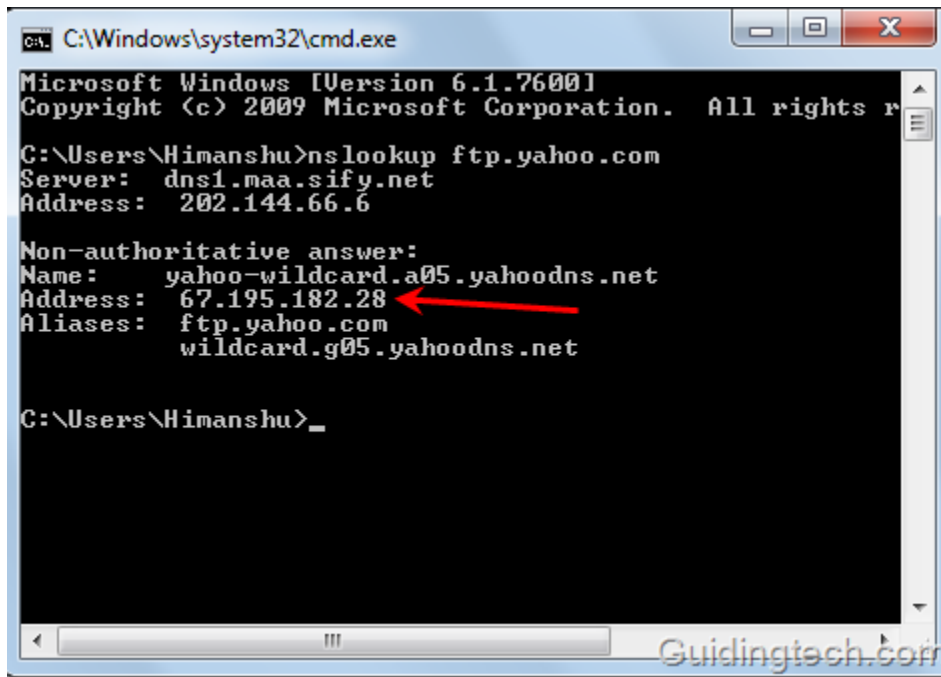
Tracing route to mediacollege.com [66.246.3.197]
over a maximum of 30 hops:

 1  <10 ms  <10 ms  <10 ms  192.168.1.1
 2  240 ms  421 ms  70 ms  219-88-164-1.jetstream.xtra.co.nz [219.88.164.1]
 3  20 ms   30 ms   30 ms  210.55.205.123
 4  *      *      *      Request timed out.
 5  30 ms   30 ms   40 ms  202.50.245.197
 6  30 ms   40 ms   40 ms  g2-0-3.tkbr3.global-gateway.net.nz [202.37.245.140]
 7  30 ms   30 ms   40 ms  so-1-2-1-0.akbr3.global-gateway.net.nz [202.50.116.161]
 8  160 ms  161 ms  160 ms  p1-3.sjbri.global-gateway.net.nz [202.50.116.178]
 9  160 ms  171 ms  160 ms  so-1-3-0-0.pabr3.global-gateway.net.nz [202.37.245.230]
10  160 ms  161 ms  170 ms  pao1-br1-g2-1-101.gnaps.net [198.32.176.165]
11  180 ms  181 ms  180 ms  lax1-br1-p2-1.gnaps.net [199.232.44.5]
12  170 ms  170 ms  171 ms  lax1-br1-ge-0-1-0.gnaps.net [199.232.44.50]
13  240 ms  241 ms  240 ms  nyc-n20-ge2-2-0.gnaps.net [199.232.44.21]
14  240 ms  251 ms  250 ms  ash-n20-ge1-0-0.gnaps.net [199.232.131.36]
15  241 ms  240 ms  250 ms  0503.ge-0-0-0.gbr1.ash.nac.net [207.99.39.157]
16  251 ms  260 ms  250 ms  0.so-2-2-0.gbr2.nwr.nac.net [209.123.11.29]
17  250 ms  260 ms  261 ms  0.so-0-3-0.gbr1.oct.nac.net [209.123.11.233]
18  250 ms  260 ms  261 ms  209.123.182.243
19  250 ms  260 ms  261 ms  sol.yourhost.co.nz [66.246.3.197]

Trace complete.

C:\>

```

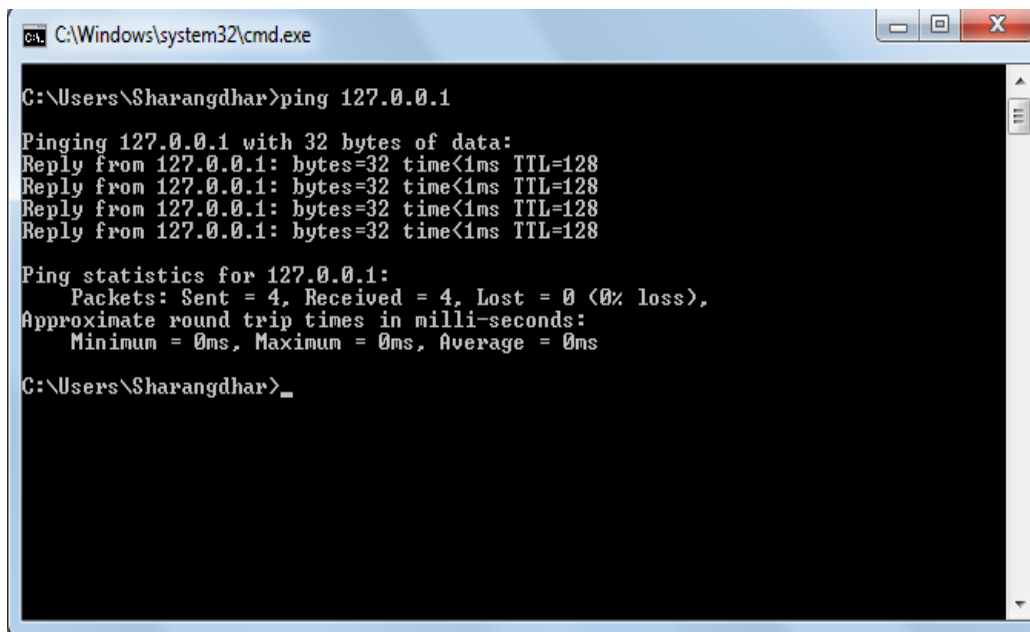


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Himanshu>nslookup ftp.yahoo.com
Server: dns1.maa.sify.net
Address: 202.144.66.6

Non-authoritative answer:
Name: yahoo-wildcard.a05.yahoodns.net
Address: 67.195.182.28
Aliases: ftp.yahoo.com
wildcard.g05.yahoodns.net

C:\Users\Himanshu>
```



```
C:\Windows\system32\cmd.exe
C:\Users\Sharangdhar>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\Sharangdhar>
```

6. Viva Questions:

- Q1: What are the maximum bytes is send to the ping command.
Q2: traceroute is used for static or dynamic? Explain?

7. References:

- www.cs.virginia.edu/~cs757/slidespdf/cs757-ns2-tutorial1.pdf

Networking Lab

Experiment No. : 3

**To study and installation of NS2
Simulator**

Experiment No. 3

1. **Aim:** To study and installation of NS2 Simulator
2. **What will you learn by performing this experiment?**

NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. Although NS is fairly easy to use once you get to know the simulator, it is quite difficult for a first time user, because there are few user-friendly manuals. Even though there is a lot of documentation written by the developers which has in depth explanation of the simulator, it is written with the depth of a skilled NS user.

3. **Software Required:** Ubuntu , NS 2.34

4. **Theory:**

NS is an event driven network simulator developed at UC Berkeley that simulates variety of IP networks. It implements network protocols such as TCP and UPD, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations. The NS project is now a part of the VINT project that develops tools for simulation results display, analysis and converters that convert network topologies generated by well-known generators to NS formats. Currently, NS (version 2) written in C++ and OTcl (Tcl script language with Object-oriented extensions developed at MIT) is available.

NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. Although NS is fairly easy to use once you get to know the simulator, it is quite difficult for a first time user, because there are few user-friendly manuals. Even though there is a lot of documentation written by the developers which has in depth explanation of the simulator, it is written with the depth of a skilled NS user. The purpose of this project is to give a new user some basic idea of how the simulator works, how to setup simulation networks, where to look for further information about network components in simulator codes, how to create new network components, etc., mainly by giving simple examples and brief explanations based on our experiences.

Although all the usage of the simulator or possible network simulation setups may not be covered in this project, the project should help a new user to get started quickly.

Figure 1. Simplified User's View of NS

Following are the steps for installation of ns-allinone-2.34 on Linux environment.

Step 1:

Getting ready your Ubuntu for ns-2 installation

```
$ sudo apt-get update
```

```
$ sudo apt-get install build-essential autoconf automake libxmu-dev
```

```
$ sudo apt-get install xorg-dev g++ xgraph
```

Step 2:

Downloading ns-2.34

<http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.34/ns-allinone-2.34.tar.gz/download>

Step 3:

Extract the file using the command

```
$ tar -xvf ns-allinone-2.34.tar.gz
```

Changing directory

```
$ cd ns-allinone-2.34
```

Step 4:

Check your gcc compiler version

```
$ gcc --version
```

Step 5:

Setting path and installation

```
$ export CC=gcc-4.7.2 CXX=g++-4.7.2 && ./install
```

Step 6:

Setting Environment Variables

```
$ gedit ~/.bashrc
```

```
# LD_LIBRARY_PATH
```

```
OTCL_LIB=/your/path/ns-allinone-2.34/otcl-1.14
```

```
NS2_LIB=/your/path/ns-allinone-2.34/lib
```

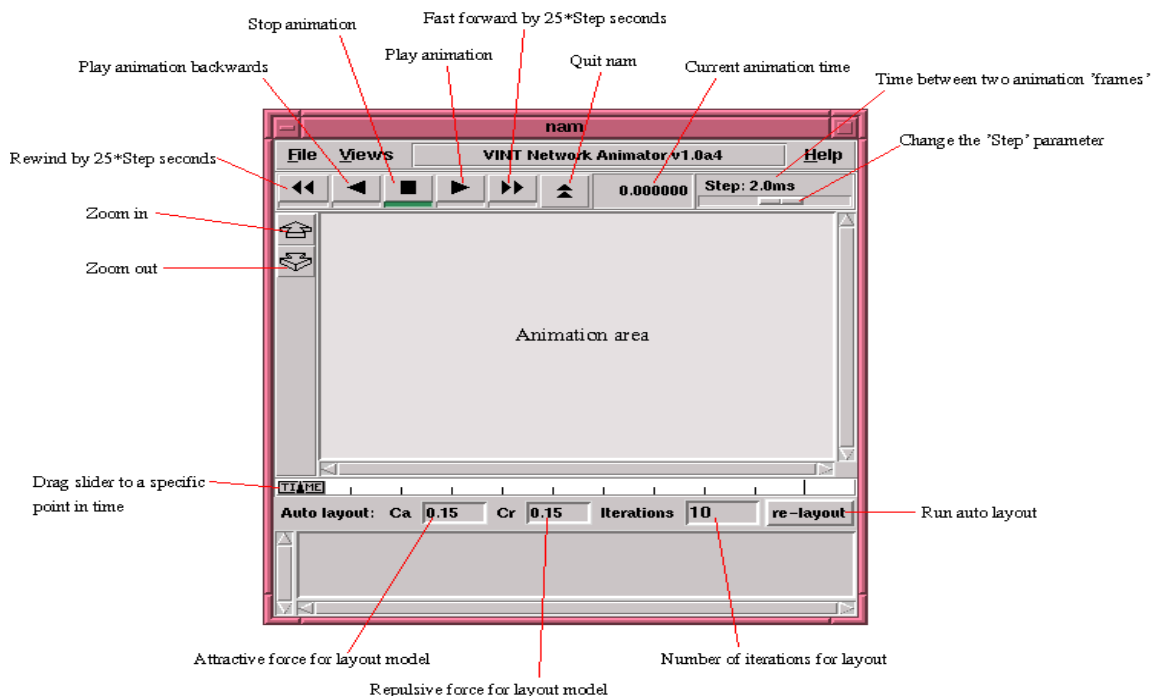
```
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$
USR_LOCAL_LIB
# TCL_LIBRARY
TCL_LIB=/your/path/ns-allinone-2.34/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
# PATH
XGRAPH=/your/path/ns-allinone-2.34/bin:/your/path/ns-allinone-
2.35/tcl8.5.10/unix:/your/path/ns-allinone-2.34/tk8.5.10/unix
NS=/your/path/ns-allinone-2.34/ns-2.34/
NAM=/your/path/ns-allinone-2.34/nam-1.15/
export PATH=$PATH:$XGRAPH:$NS:$NAM
Save and use the command $ source ~/.bashrc
```

Installation completed use ns command and you should see % symbol in your terminal

5. Conclusion and Discussion:

Hence, we installed NS2 in Linux.

6. Result :



7. Viva Questions:

Q1: Which language is use as front end and back end in NS2.

Q2: NS2 is installed in windows? It is user friendly?

8. References:

- www.cs.virginia.edu/~cs757/slidespdf/cs757-ns2-tutorial1.pdf

Networking Lab

Experiment No. : 4

Introduction to TCL Hello programming.

Experiment No. 4

7. **Aim:** Introduction to TCL Hello programming.

8. **What will you learn by performing this experiment?**

We will learn about the TCL scripting language. The way of execution of tcl program and a basic knowledge about NS-2 Architecture.

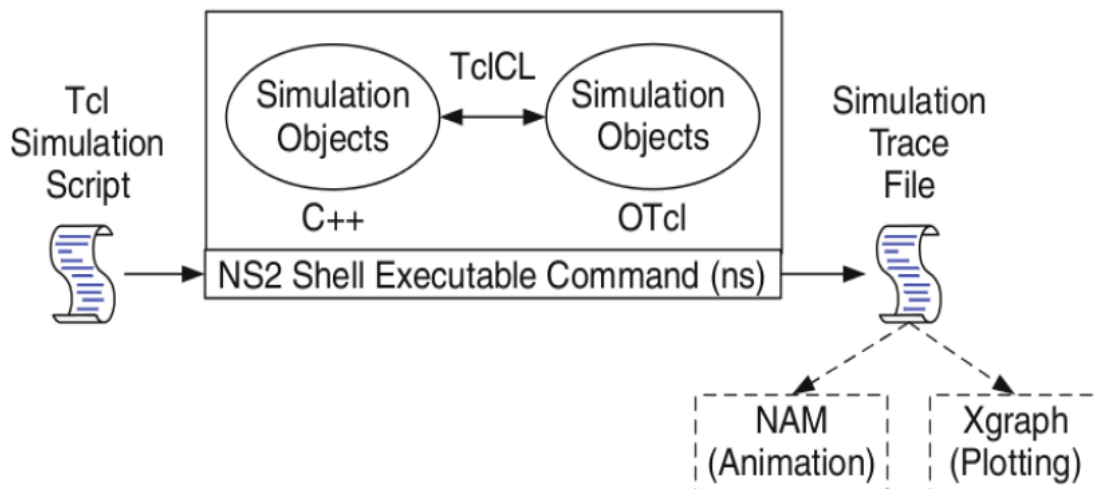
9. **Software Required:** Ubuntu, NS 2.34

10. **Theory :**

- **Introduction to NS-2:**

Widely known as NS2, is simply an event driven simulation tool. 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.

- **Basic Architecture of NS2**



- **Tcl scripting**

Tcl is a general purpose scripting language. [Interpreter]. Tcl runs on most of the platforms such as Unix, Windows, and Mac. The strength of Tcl is its simplicity. It is not necessary to declare a data type for variable prior to the usage.

Basics of TCL

Syntax: command arg1 arg2 arg3

- **Hello World!**

```
puts stdout{Hello, World!}
Hello, World!
```

- **Variables Command Substitution**

```
set a 5 set len [string length foobar]
set b $a set len [expr [string length foobar] + 9]
```

- **Simple Arithmetic**

```
expr 7.2 / 4
```

- **Procedures**

```
proc Diag {a b} {
    set c [expr sqrt($a * $a + $b * $b)]
    return $c }
puts —Diagonal of a 3, 4 right triangle is [Diag 3 4]
```

Output: Diagonal of a 3, 4 right triangle is 5.0

- **Loops**

```
while{$i < $n} { for {set i 0} {$i < $n} {incr i} {
    .....
} }
```

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

set ns [new Simulator]

Which is thus the first line in the tcl script? This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

File name: **simple.tcl**

```
set sim [new Simulator]
$sim at 1 "puts \"Hello World!\""
$sim at 1.5 "exit"
$sim run
```

Output:

```
# ns simple.tcl
Hello World!
```

Networking Lab

Experiment No. : 5

TCL script for creating Nodes

Experiment No. 5

1. Aim: TCL script for creating Nodes

2. What will you learn by performing this experiment?

Simple NS simulation script is an OTcl script that creates the simple network configuration and runs the simulation.

3. Software Required: Ubuntu , NS 2.34

4. Theory:

Definition of a network of links and nodes

The way to define a node is

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

The node is created which is printed by the variable n0. When we shall refer to that node in the script we shall thus write \$n0.

Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

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

Which means that \$n0 and \$n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction. To define a directional link instead of a bi-directional one, we should replace —duplex-link by —simplex-link.

In NS, an output queue of a node is implemented as a part of each link whose input is that node. The definition of the link then includes the way to handle overflow at that queue. In our case, if the buffer capacity of the output queue is exceeded then the last packet to arrive is dropped. Many alternative options exist, such as the RED (Random Early Discard) mechanism, the FQ (Fair Queuing), the DRR (Deficit Round Robin), the stochastic Fair Queuing (SFQ) and the CBQ (which including a priority and a round-robin scheduler). In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

```
#set Queue Size of link (n0-n2) to 20
$ns queue-limit $n0 $n2 20
```

5. Program:

```
#-----Event scheduler object creation-----#
```



```
set ns [new Simulator]

#-----creating nam objects-----#

set nf [open tcp1.nam w]
$ns namtrace-all $nf

#open the trace file
set nt [open tcp1.tr w]
$ns trace-all $nt

set proto rlm

$ns color 1 blue
$ns color 2 yellow
$ns color 3 red

#----- creating client- router- end server node-----#

set Client1 [$ns node]
set Router1 [$ns node]
set Endserver1 [$ns node]

#---creating duplex link-----#

$ns duplex-link $Client1 $Router1 2Mb 100ms DropTail
$ns duplex-link $Router1 $Endserver1 200Kb 100ms DropTail

#-----creating orientation-----#

$ns duplex-link-op $Client1 $Router1 orient right
$ns duplex-link-op $Router1 $Endserver1 orient right

#-----Labelling-----#

$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Router1 label Router1"
$ns at 0.0 "$Endserver1 label Endserver1"

#-----Configuring nodes-----#

$Endserver1 shape hexagon
$Router1 shape square

#-----Establishing queues-----#

#$ns duplex-link-op $Client1 $Router1 queuePos 0.1
#$ns duplex-link-op $Router1 $Endserver1 queuePos 0.5
```

```
#-----finish procedure-----#
```

```
proc finish {} {  
    global ns nf nt  
    $ns flush-trace  
    close $nf  
    close $nt  
  
    puts "running nam..."  
    exec nam tcp1.nam &  
    exit 0  
}
```

```
#Calling finish procedure  
$ns at 6.0 "finish"  
$ns run
```

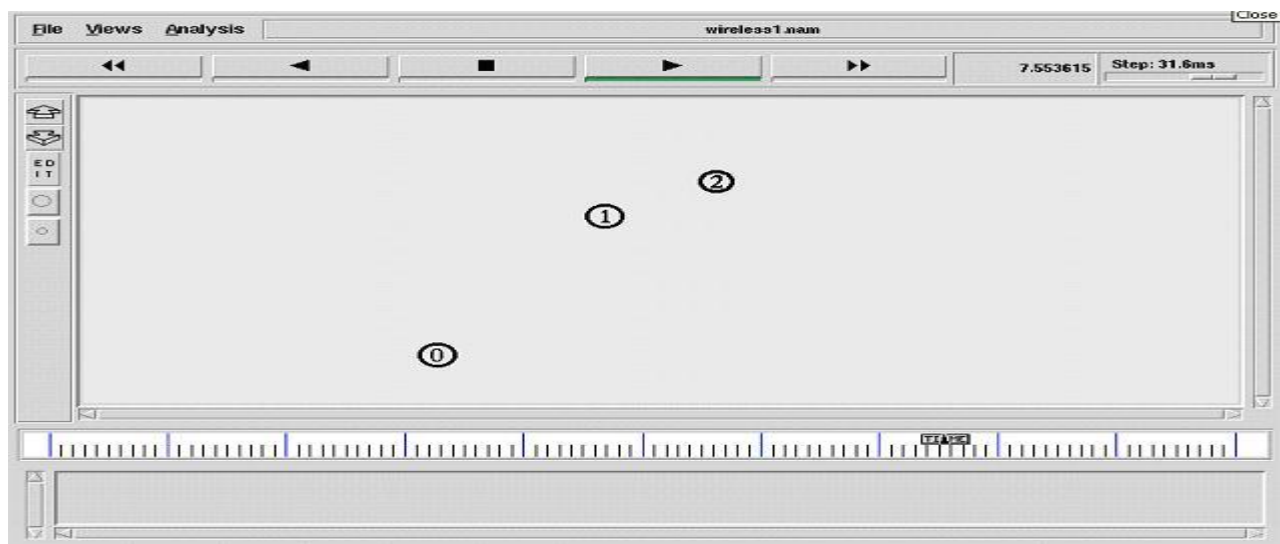
```
#----How to run----#
```

```
$ns tcp1.tcl
```

6. Conclusion and Discussion:

We learn the scripting language of Network simulator. It is similar to C++ language.

7. Result :



8. Questions for Practice:

Q: Create the following topology in ns2

Marking flows

Add the following two lines to your CBR agent definitions

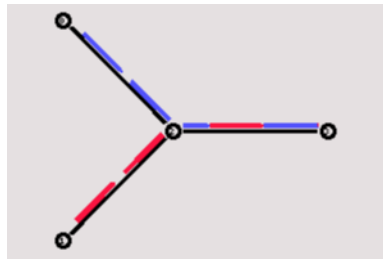
```
$udp0 set class_ 1
```

```
$udp1 set class_ 2
```

Now add the following piece of code to your Tcl script, preferably at the beginning after the simulator object has been created, since this is a part of the simulator setup

```
$ns color 1 Blue
```

```
$ns color 2 Red
```



9. References:

- www.cs.virginia.edu/~cs757/slidespdf/cs757-ns2.pdf

Networking Lab

Experiment No. : 6

**TCL script for TCP configuration
between nodes**

Experiment No. 6

1. Aim: TCL script for TCP configuration between nodes

2. What will you learn by performing this experiment?

Scripting is performing as connection oriented or connectionless services between different nodes. We are going to use different colors to the traffic in network.

3. Software Required: Ubuntu , NS 2.34

4. Theory:

Beginning of every NS2 simulator program starts with
ns[new simulator]

Creation of nodes

```
set n0 [$ns node] # create nodes and a link
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

- These commands will create four nodes namely n0, n1,n2,n3.

- Further they should be linked with each other proper flow.

Creating link

```
# create links between wired and BaseStation 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
```

These commands will create links with specified nodes as mentioned in commands with 2 megabit, a delay of 10ms and droptail queue

Tracing :

Stimulator is used to see results using trace. Tracing is done with nam trace for use with visualization.

```
Set tracefile [open.out.trw]
$ns trace-all $trace file
```

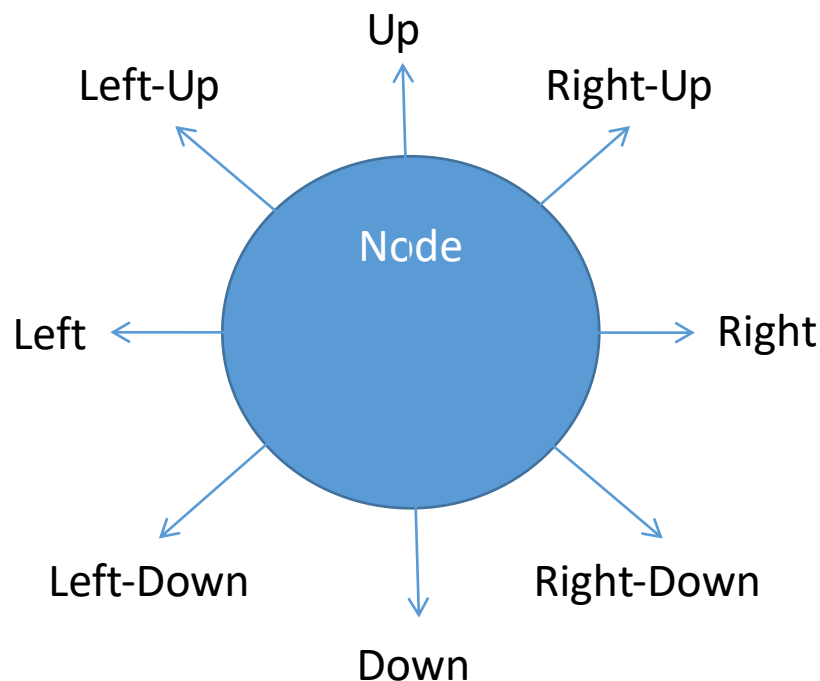
- As it is started, it should be end and hence for ending trace , finish procedure is used

Define a finish procedure

```
proc finish {} {
    global ns tracefile
    $ns flush-trace
    close $nf
close $tracefile
#close tracefile
exit 0
}
```

Orientation

-Placing of four nodes should be done



```
$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
```

Agents and Applications

We need to define routing (sources, destinations) the agents (protocols) the application that use them.

FTP over TCP

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received. There are number

variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line:

```
set tcp [new Agent/TCP]
```

The command `$ns attach-agent $n0 $tcp` defines the source node of the tcp connection. The command

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

Defines the behavior of the destination node of TCP and assigns to it a pointer called sink.

#Setup a UDP connection

```
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_2
```

#setup a CBR over UDP connection

The below shows the definition of a CBR application using a UDP agent. The command `$ns attach-agent $n4 $sink` defines the destination node. The command `$ns connect $tcp $sink` finally makes the TCP connection between the source and destination nodes.

```
set cbr [new
Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetsize_ 100
$cbr set rate_ 0.01Mb
$cbr set random_ false
```

TCP has many parameters with initial fixed default values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of

1000bytes. This can be changed to another value, say 552bytes, using the command `$tcp set packetSize_ 552`.

When we have several flows, we may wish to distinguish them so that we can identify them with different colors in the visualization part. This is done by the command `$tcp set fid_ 1` that assigns to the TCP connection a flow identification of “ 1 ”. We shall later give the flow identification of “ 2 ” to the UDP connection.

CBR over UDP

A UDP source and destination is defined in a similar way as in the case of TCP.

Instead of defining the rate in the command `$cbr set rate_ 0.01Mb`, one can define the time interval between transmission of packets using the command.

```
$cbr set interval_ 0.005
```

The packet size can be set to some value using

```
$cbr set packetSize_ <packet size>
```

Traffic Generator:

For actual data to flow, we need traffic generators. They stimulate some application traffic using CBR.

Set cbr [new application/traffic/CBR]

Hence this setup CBR connection over UDP connection

Event Scheduling :

This will schedule events for CBR and FTP agents using following commands:

```
$ns at 0.1 “ $cbr start”
```

```
$ns at 0.1 “ $ftp start”
```

```
$ns at 0.1 “ $ftp stop”
```

And then at the end `$ns run` command is required for successful running of program that includes nodes with links sending packet data.

5. Program:

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

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

```
$ns color 1 Blue
```

```
$ns color 2 Red
```


#Open the Trace files

set file1 [open out.tr w]

\$ns trace-all \$file1

#Open the NAM trace file

set file2 [open out.nam w]

\$ns namtrace-all \$file2

#Define a 'finish' procedure

proc finish {} {

global ns file1 file2

\$ns flush-trace

close \$file1

close \$file2

exec nam out.nam &

exit 0

}

#Create six 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]

\$ns at 0.1 "\$n1 label \"CBR\""

\$ns at 1.0 "\$n0 label \"FTP\""

#Create links between the nodes

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

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

\$ns simplex-link \$n2 \$n3 0.3Mb 100ms DropTail

\$ns simplex-link \$n3 \$n2 0.3Mb 100ms DropTail

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

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

#Give node position (for NAM)

\$ns duplex-link-op \$n0 \$n2 orient right-down

\$ns duplex-link-op \$n1 \$n2 orient right-up

\$ns simplex-link-op \$n2 \$n3 orient right

\$ns simplex-link-op \$n3 \$n2 orient left

```
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down
#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10
```

```
#Setup a TCP connection
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552
```

```
#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 $n5 $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_ 0.01mb
$cbr set random_ false
```

```
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 624.0 "$ftp stop"
$ns at 624.5 "$cbr stop"
```

Trace Congestion Window and RTT

```
set file [open cwnd_rtt.tr w]
```

```
$tcp attach $file
```

```
$tcp trace cwnd_
```

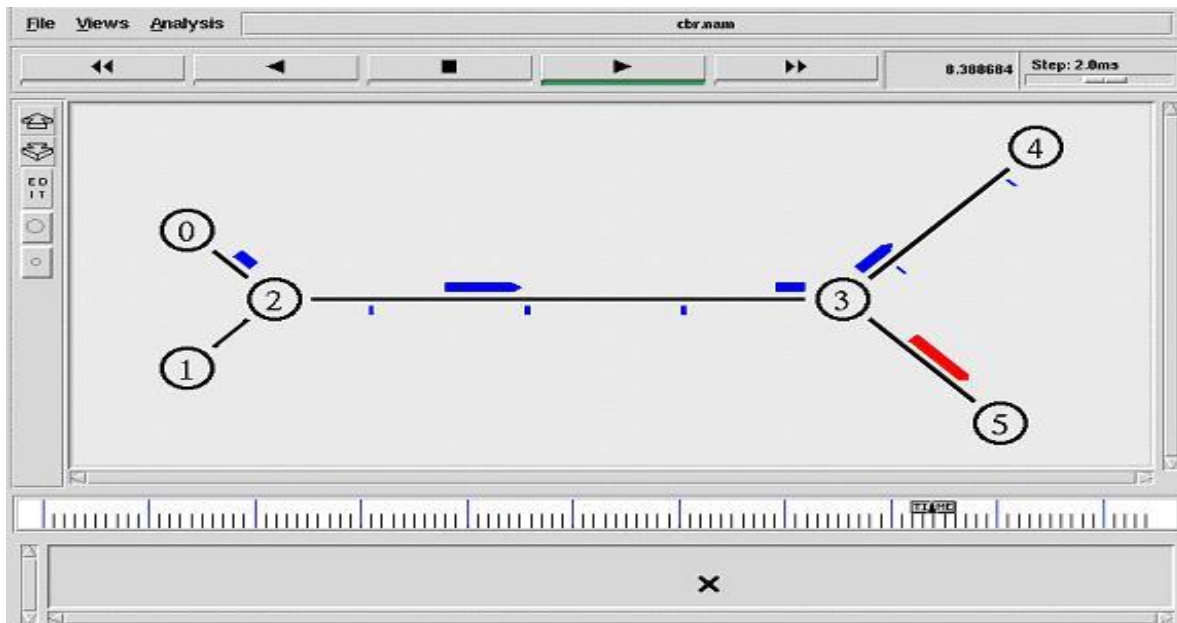
```
$tcp trace rtt_
```

```
$ns at 625.0 "finish"
```

```
$ns run
```

6. **Conclusion and Discussion:** we use different services, colors in networks. Schedule can be managed by different packet size. Events are handled by agents.

7. Result:



8. Viva Questions:

- What are the source and destination used for the services
- Is packet size is fixed or can be variable?

9. References:

- www.cs.virginia.edu/~cs757/slidespdf/cs757-ns2.pdf

Networking Lab

Experiment No. : 7

**Implement Ring topology in Ns2
simulator**

Experiment No. 7

1. **Aim:** Implement Ring topology in Ns2 simulator
2. **What will you learn by performing this experiment?**

Ring topologies typically utilize a token passing scheme, used to control access to the network. By utilizing this scheme, only one machine can transmit on the network at a time.

3. **Software Required:** Ubuntu , NS 2.34

4. **Theory:**

In local area networks where the ring topology is used, each computer is connected to the network in a closed loop or ring. Each machine or computer has a unique address that is used for identification purposes. The signal passes through each machine or computer connected to the ring in one direction. Ring topologies typically utilize a token passing scheme, used to control access to the network. By utilizing this scheme, only one machine can transmit on the network at a time. The machines or computers connected to the ring act as signal boosters or repeaters which strengthen the signals that transverse the network.

In Ring Topology, all the nodes are connected to each-other in such a way that they make a closed loop. Each workstation is connected to two other components on either side, and it communicates with these two adjacent neighbors. Data travels around the network, in one direction. Sending and receiving of data takes place by the help of TOKEN.

Token passing (in brief): Token contains a piece of information which along with data is sent by the source computer. This token then passes to next node, which checks if the signal is intended to it. If yes, it receives it and passes the empty token into the network, otherwise passes token along with the data to next node. This process continues until the signal reaches its intended destination.

The nodes with token are the ones only allowed to send data. Other nodes have to wait for an empty token to reach them. This network is usually found in offices, schools and small buildings.

Advantages of Ring Topology

- 1) This type of network topology is very organized. Each node gets to send the data when it receives an empty token. This helps to reduce chances of collision. Also in ring topology all the traffic flows in only one direction at very high speed.

- 2) Even when the load on the network increases, its performance is better than that of Bus topology.
- 3) There is no need for network server to control the connectivity between workstations.
- 4) Additional components do not affect the performance of network.
- 5) Each computer has equal access to resources.

Disadvantages of Ring Topology

- 1) Each packet of data must pass through all the computers between source and destination. This makes it slower than Star topology.
- 2) If one workstation or port goes down, the entire network gets affected.
- 3) Network is highly dependent on the wire which connects different components.
- 4) MAU's and network cards are expensive as compared to Ethernet cards and hubs

5. Program:

```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish { } {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Executenam 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]
set n4 [$ns node]
set n5 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

```
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail

#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0

#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

6. **Conclusion and Discussion:** Hence we studied and executed a program for ring topology

7. **Viva Questions:**

What is topology?

8. **References:**

- www.cs.virginia.edu/~cs757/slidespdf/cs757-ns2.pdf

Networking Lab

Experiment No. : 8

Study and analysis routing protocol and Shortest Path routing by DSDV.

Experiment No. 8

1. **Aim:** Study and analysis routing protocol and Shortest Path routing by DSDV.

2. **What will you learn by performing this experiment?**

Routing is usually performed by a dedicated device called a router. Routing is a key feature of the Internet because it enables messages to pass from one computer to another and eventually reach the target machine. Each intermediary computer performs routing by passing along the message to the next computer. Part of this process involves analysing a routing table *to* determine the best path.

3. **Software Required:** Ubuntu , NS 2.34

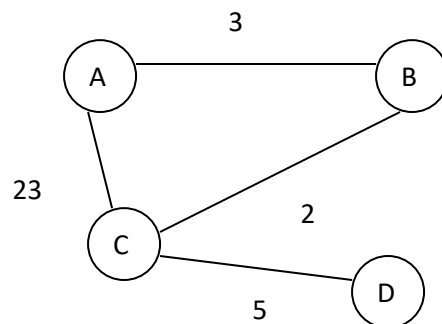
4. **Theory:**

A **distance-vector routing protocol** is one of the two major classes of routing protocols used in packet-switched networks for computer communications, the other major class being the link-state protocol. A distance-vector routing protocol uses the Bellman-Ford algorithm to calculate paths.

Examples of distance-vector routing protocols include RIPv1 and 2 and IGRP.

A distance-vector routing protocol requires that a router informs its neighbors of topology changes periodically and, in some cases, when a change is detected in the topology of a network. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead.

In this network we have 4 routers A, B, C, and D:



We shall mark the current time (or iteration) in the algorithm with T, and shall begin (at time 0, or T=0) by creating distance matrices for each router to its immediate neighbors. As we build the routing tables below, the shortest path is highlighted with the color green, a new shortest path is highlighted with the color yellow.

T=0 **from** via via via via **from** via via via via **from** via via via via **fro** vi vi vi vi

A	A	B	C	D	B	A	B	C	D	C	A	B	C	D	m D	a	a	a	a
to A					to A	3				to A	23				A	A	B	C	D
to B		3			to B					to B		2			to A				
to C			23		to C			2		to C					to B				
to D					to D					to D				5	to C			5	
															to D				

At this point, all the routers (A,B,C,D) have new "shortest-paths" for their DV (the list of distances that are from them to another router via a neighbor) They each broadcast this new DV to all their neighbors: A to B and C, B to C and A, C to A, B, and D, and D to C As each of these neighbors receives this information, they now recalculate the shortest path using it

For example: A receives a DV from C that tells A there is a path via C to D, with a distance (or cost) of 5 Since the current "shortest-path" to C is 23, then A knows it has a path to D that costs $23+5=28$ As there are no other shorter paths that A knows about, it puts this as its current estimate for the shortest-path from itself (A) to D, via C

from	via	via	via	via	from	via	via	via	via	from	via	via	via	via	from	via	via	via	via
A	A	B	C	D	B	A	B	C	D	C	A	B	C	D	D	A	B	C	D
to A					to A	3		25		to A	23	5			to A			28	
to B		3	25		to B					to B	26	2			to B			7	
to C		5	23		to C	26		2		to C					to C			5	
to D			28		to D			7		to D				5	to D				

Again, all the routers have gained in the last iteration (at T=1) new "shortest-paths", so they all broadcast their DVs to their neighbors; This prompts each neighbor to re-calculate their shortest distances again

For instance: A receives a DV from B that tells A there is a path via B to D, with a distance (or cost) of 7 Since the current "shortest-path" to B is 3, then A knows it has a path to D that costs $7+3=10$ This path to D of length 10 (via B) is shorter than the existing "shortest-path" to D of length 28 (via C), so it becomes the new "shortest-path" to D

from	via	via	via	via	from	via	via	via	via	from	via	via	via	via	from	via	via	via	via
A	A	B	C	D	B	A	B	C	D	C	A	B	C	D	D	A	B	C	D
to A					to A	3		7		to A	23	5		33	to A			10	
to B		3	25		to B					to B	26	2		12	to B			7	
to C		5	23		to C	8		2		to C					to C			5	
to D		10	28		to D	31		7		to D	51	9		5	to D				

This time, only routers A and D have new shortest-paths for their DVs So they broadcast

their new DVs to their neighbors: A broadcasts to B and C, and D broadcasts to C. This causes each of the neighbors receiving the new DVs to re-calculate their shortest paths. However, since the information from the DVs doesn't yield any shorter paths than they already have in their routing tables, then there are no changes to the routing tables.

	from	via	via	via	via		from	via	via	via	via		from	via	via	via	via		from	via	via	via	via
	A	A	B	C	D		B	A	B	C	D		C	A	B	C	D		D	A	B	C	D
T=3	to A						to A	3		7			to A	23	5		15		to A			10	
	to B		3	25			to B						to B	26	2		12		to B			7	
	to C		5	23			to C	8		2			to C						to C			5	
	to D		10	28			to D	13		7			to D	33	9		5		to D				

None of the routers have any new shortest-paths to broadcast. Therefore, none of the routers receive any new information that might change their routing tables. So the algorithm comes to a stop.

Unicast Routing Configurations :

Static route strategy

\$ns rtproto Static

Session Routing

\$ns rtproto Session

Distance Vector Routing

\$ns rtproto DV

Link State

\$ns rtproto LS

Manual Routing

\$ns rtproto Manual

set n1 [\$ns node]

set n2 [\$ns node]

\$ns duplex-link \$n1 \$n2 10Mb 100ms DropTail

\$n1 add-route-to-adj-node -default \$n2

\$n2 add-route-to-adj-node -default \$n1

5. Program :

set ns [new Simulator]

```
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the Trace file
set file1 [open unicastDV.tr w]
$ns trace-all $file1

#Open the NAM trace file
set file2 [open unicastDV.nam w]
$ns namtrace-all $file2

#Define a 'finish' procedure
proc finish {} {
    global ns file1 file2
    $ns flush-trace
    close $file1
    close $file2
    exec nam unicastDV.nam &
    exit 0
}

# Next line should be commented out to have the static
routing
$ns rtproto DV

#Create six 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]

#Create links between the nodes
$ns duplex-link $n0 $n1 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n2 0.3Mb 10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n4 0.3Mb 10ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 10ms DropTail
$ns duplex-link $n4 $n5 0.5Mb 10ms DropTail

#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n1 $n4 orient up-left
$ns duplex-link-op $n3 $n5 orient left-up
$ns duplex-link-op $n4 $n5 orient right-up

#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
```

```
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n5 $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

$ns rtmodel-at 1.0 down $n1 $n4
$ns rtmodel-at 4.5 up $n1 $n4

$ns at 0.1 "$ftp start"

$ns at 6.0 "finish"

$ns run
```

6. Conclusion and Discussion: DSDV routing algorithms are analysed and studied to find out the shortest path in network.

7. Viva Questions:

- What is best and worst case for any of the dynamic algorithms?
- Explain infinity problem?

8. References:

- www.cs.ccsu.edu/~stan/classes/cs490/slides/networks4-ch4-4.pdf
- www.cs.cornell.edu/skeshav/book/slides/routing/routing.pdf

Networking Lab

Experiment No. : 9

**Analysis of network performance by
configuring the queue size and capacity
of links for measuring QoS of generated
traffic**

Experiment No. 9

1. **Aim:** Analysis of network performance by configuring the queue size and capacity of links for measuring QoS of generated traffic
2. **What will you learn by performing this experiment?**

Routing is usually performed by a dedicated device called a router. Routing is a key feature of the Internet because it enables messages to pass from one computer to another and eventually reach the target machine. Each intermediary computer performs routing by passing along the message to the next computer. Part of this process involves analysing a routing table *to* determine the best path.

3. **Software Required:** Ubuntu , NS 2.34

4. **Theory:**

Definition of a network of links and nodes

The way to define a node is

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

The node is created which is printed by the variable n0. When we shall refer to that node in the script we shall thus write \$n0.

Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

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

Which means that \$n0 and \$n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction.

To define a directional link instead of a bi-directional one, we should replace —duplex-link by —simplex-link.

In NS, an output queue of a node is implemented as a part of each link whose input is that node. The definition of the link then includes the way to handle overflow at that queue. In our case, if the buffer capacity of the output queue is exceeded then the last packet to arrive is dropped. Many alternative options exist, such as the RED (Random Early Discard) mechanism, the FQ (Fair Queuing), the DRR (Deficit Round Robin), the stochastic Fair Queuing (SFQ) and the CBQ (which including a priority and a round-robin scheduler).

In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

```
#set Queue Size of link (n0-n2) to 20
$ns queue-limit $n0 $n2 20
```

Structure of Trace Files

When tracing into an output ASCII file, the trace is organized in 12 fields as follows in fig shown below, The meaning of the fields are:

Event	Time	From Node	To Node	PKT Type	PKT Size	Flags	Fid	Src Addr	Dest Addr	Seq Num	Pkt id
-------	------	--------------	------------	-------------	-------------	-------	-----	-------------	--------------	------------	-----------

1. The first field is the event type. It is given by one of four possible symbols r, +, -, d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.
2. The second field gives the time at which the event occurs.
3. Gives the input node of the link at which the event occurs.
4. Gives the output node of the link at which the event occurs.
5. Gives the packet type (eg CBR or TCP)
6. Gives the packet size
7. Some flags
9. This is the flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script one can further use this field for analysis purposes; it is also used when specifying stream color for the NAM display.
10. This is the source address given in the form of —node.portl.
11. This is the destination address, given in the same form.
12. This is the network layer protocol's packet sequence number. Even though UDP implementations in a real network do not use sequence number, ns keeps track of UDP packet sequence number for analysis purposes
13. The last field shows the Unique id of the packet.

Awk- An Advanced

awk is a programmable, pattern-matching, and processing tool available in UNIX. It works equally well with text and numbers.

awk is not just a command, but a programming language too. In other words, awk utility is a pattern scanning and processing language. It searches one or more files to see if they contain lines that match specified patterns and then perform associated actions, such as writing the line to the standard output or incrementing a counter each time it finds a match.

awk option 'selection_criteria {action}' file(s)

Here, selection_criteria filters input and select lines for the action component to act upon. The selection_criteria is enclosed within single quotes and the action within the curly braces. Both the selection_criteria and action forms an awk program.

Example: \$ awk '/manager/ {print}' emp.lst

Variables

Awk allows the user to use variables of there choice. You can now print a serial number, using the variable Count, and apply it those directors drawing a salary exceeding 6700:

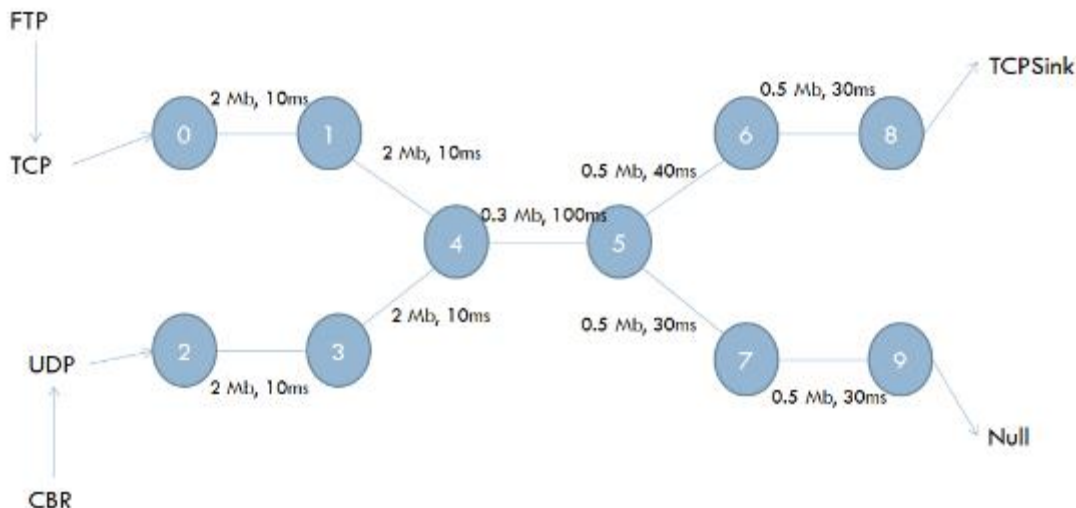
```
$ awk -F'|' ' $3 == "director" && $6 > 6700 {
```

```
Count =Count+1
```

```
printf " %3f %20s %-12s %d\n", Count,$2,$3,$6 }' empn.lst
```

5. Program:

Scenario for Program:



```
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the Trace files
set file1 [open out.tr w]
```

```
$ns trace-all $file1

#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2

#Define a 'finish' procedure
proc finish {} {
    global ns file1 file2
    $ns flush-trace
    close $file1
    close $file2
    exec nam out.nam &
    exit 0
}

#Create six 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]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]

$ns at 0.1 "$n2 label \"CBR\""
$ns at 1.0 "$n0 label \"FTP\""

#Create links between the nodes
$ns duplex-link $n0 $n1 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 2Mb 10ms DropTail
$ns duplex-link $n1 $n4 2Mb 10ms DropTail
$ns duplex-link $n3 $n4 2Mb 10ms DropTail
$ns simplex-link $n4 $n5 0.3Mb 100ms DropTail
$ns simplex-link $n5 $n4 0.3Mb 100ms DropTail
$ns duplex-link $n5 $n6 0.5Mb 40ms DropTail
$ns duplex-link $n6 $n8 0.5Mb 40ms DropTail
$ns duplex-link $n5 $n7 0.5Mb 30ms DropTail
$ns duplex-link $n7 $n9 0.5Mb 30ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n4 $n5 10

#Setup a TCP connection
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n8 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
```

```
$tcp set packetSize_ 552

#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 $n2 $udp
set null [new Agent/Null]
$ns attach-agent $n9 $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_ 0.01mb
$cbr set random_ false

$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 624.0 "$ftp stop"
$ns at 624.5 "$cbr stop"

# Trace Congestion Window and RTT
set file [open cwnd_rtt.tr w]
$tcp attach $file
$tcp trace cwnd_
$tcp trace rtt_

$ns at 625.0 "finish"
$ns run
```

Analysis

[SimpleAWK.awk](#)

```
awk -f SimpleAWK.awk out.tr > Temp
```

[Bytespersecond.awk](#)

```
awk -f Bytespersecond.awk Temp > bps.tr
```

Contents of SimpleAWK.awk

```
{  
if(($3=="0")&&($4=="1")&&($1=="r"))  
  
print $2,$6  
}
```

Contents of Bytespersecond.awk

```
BEGIN{ sum = 0;}  
{  
    if ($1>0)  
        printf("%f\t%f\n",$1,sum/$1);  
    sum = sum + $2;  
}  
END { puts Done }
```

6. Conclusion and Discussion: The way to vary the queue size and its effect over performance parameters are observed and studied.

7. Viva Questions:

- What are the performance parameters of network?
- How queue size affect throughput?

Networking Lab

Experiment No. : 10

**Comparative QoS parameters using
Xgraph/gnuplot for different load
conditions.**

Experiment No. 10

1. **Aim:** Comparative QoS parameters using Xgraph /gnuplot for different load conditions.
2. **What will you learn by performing this experiment?**
 - Different QoS parameters of network traffic and graphical way to have comparative analysis.
3. **Software Required:** Ubuntu , NS 2.34, Xgraph

4. Theory:

XGRAPH

The xgraph program draws a graph on an x-display given data read from either data file or from standard input if no files are specified. It can display upto 64 independent data sets using different colors and line styles for each set. It annotates the graph with a title, axis labels, grid lines or tick marks, grid labels and a legend.

Xgraph [options] file-name

Options are listed here

`/-bd <color> (Border)`

This specifies the border color of the xgraph window.

`/-bg <color> (Background)`

This specifies the background color of the xgraph window.

`/-fg<color> (Foreground)`

This specifies the foreground color of the xgraph window.

`/-lf <fontname> (LabelFont)`

All axis labels and grid labels are drawn using this font.

`/-t<string> (Title Text)`

This string is centered at the top of the graph.

`/-x <unit name> (XunitText)`

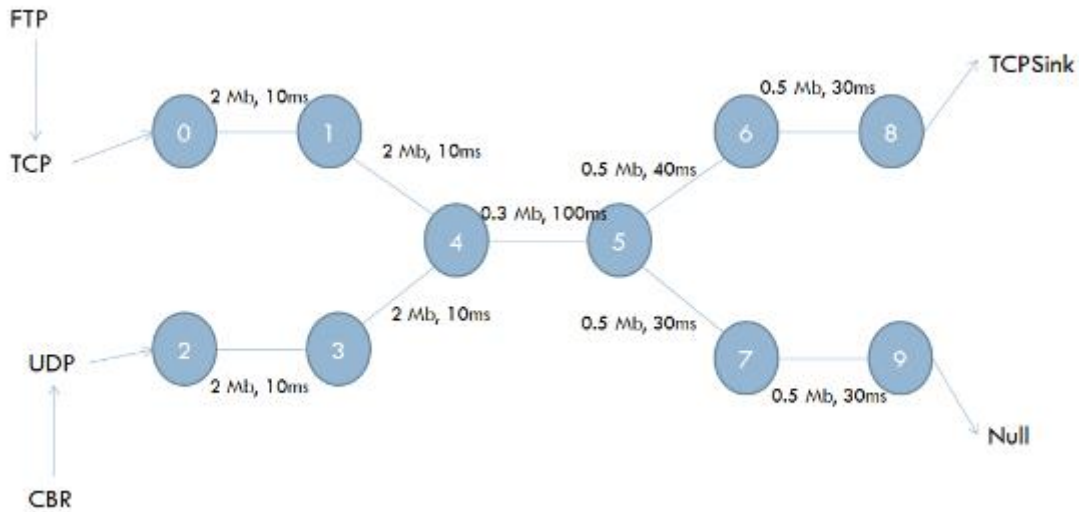
This is the unit name for the x-axis. Its default is "X".

`/-y <unit name> (YunitText)`

This is the unit name for the y-axis. Its default is "Y".

8. Program:

Scenario for Program:



```
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the Trace files
set file1 [open out.tr w]
$ns trace-all $file1

#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2

#Define a 'finish' procedure
proc finish {} {
    global ns file1 file2
    $ns flush-trace
    close $file1
    close $file2
    exec nam out.nam &
    exit 0
}

#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
```

```
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]

$ns at 0.1 "$n2 label \"CBR\""
$ns at 1.0 "$n0 label \"FTP\""

#Create links between the nodes
$ns duplex-link $n0 $n1 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 2Mb 10ms DropTail
$ns duplex-link $n1 $n4 2Mb 10ms DropTail
$ns duplex-link $n3 $n4 2Mb 10ms DropTail
$ns simplex-link $n4 $n5 0.3Mb 100ms DropTail
$ns simplex-link $n5 $n4 0.3Mb 100ms DropTail
$ns duplex-link $n5 $n6 0.5Mb 40ms DropTail
$ns duplex-link $n6 $n8 0.5Mb 40ms DropTail
$ns duplex-link $n5 $n7 0.5Mb 30ms DropTail
$ns duplex-link $n7 $n9 0.5Mb 30ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n4 $n5 10

#Setup a TCP connection
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n8 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552

#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 $n2 $udp
set null [new Agent/Null]
$ns attach-agent $n9 $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_ 0.01mb
$cbr set random_ false

$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 624.0 "$ftp stop"
$ns at 624.5 "$cbr stop"

# Trace Congestion Window and RTT
set file [open cwnd_rtt.tr w]
$tcp attach $file
$tcp trace cwnd_
$tcp trace rtt_

$ns at 625.0 "finish"
$ns run
```

➤ Analysis 1:

○ [SimpleAWK.awk](#)

Contents of SimpleAWK.awk

```
{
  if(($3=="0")&&($4=="1")&&($1=="r"))

    print $2,$6
}
```

```
awk -f SimpleAWK.awk out.tr > Temp
```

○ [Bytespersecond.awk](#)

Contents of Bytespersecond.awk

```
BEGIN{ sum = 0;}

{
    if ($1>0)
        printf("%f\t%f\n",$1,sum/$1);
    sum = sum + $2;
}

END { puts Done }
```

```
awk -f Bytespersecond.awk Temp > bps.tr
```

○ [xgraph File th : to plot graph](#)

○ [simple.gnu](#)

```
#simple.gnu
set xrange [0:700]
set yrange [0:30000]
set xlabel 'Time (sec)'
```

```
set ylabel 'Throughput (bps)'  
plot 'bps.tr' title 'Tcp' with lines  
#set term postscript  
#set output "tcp.ps"
```

- **gnuplot –persist simple.gnu**

➤ **Analysis 2 :**

- **cwnd.awk**

Contents of cwnd.awk

```
{if($6=="cwnd_") print $1,$7}
```

- **awk –f cwnd.awk cwnd_rtt.tr > cwnd**

- **cwnd.gnu**

Contents of cwnd.gnu

```
#cwnd.gnu  
set xrange [0:700]  
set yrange [0:40]  
set xlabel 'Time (sec)'  
set ylabel 'cwnd (pkts)'  
plot 'cwnd' title 'Tcp' with lines
```

- **gnuplot –persist cwnd.gnu**

➤ **Analysis 3**

- **Third script analysis.awk**

Contents of Third_Script_Analysis.awk

```
BEGIN {  
  
    seqno = -1;  
  
    droppedPackets = 0;  
  
    receivedPackets = 0;  
  
    tcps = 0;  
  
    tcpr = 0;  
  
    tcpd = 0;  
  
    udps = 0;  
  
    udpr = 0;  
  
    udpd = 0;  
  
    acks = 0;  
  
    ackr = 0;
```

```
ackd = 0;

count = 0;

}

{

#packet delivery ratio for all packets

#For counting total no of send (tcp,udp and ack)

    if (($1 == "+") && (seqno < $12))
    {
        seqno = $12;
    }

#For counting total no of receive at destination node4 and
node5 (tcp,udp and ack)

else if (($1 == "r") && (($5 == "cbr") || ($5 == "tcp") ||
($5 == "ack"))&& (($4 == "4") || ($4 == "5") || ($4 ==
"0")))

    {
        receivedPackets++;
    }

#For counting total no of drops

    else if (($1 == "d") && (($5 == "cbr") || ($5 == "tcp")
|| ($5 == "ack")))
    {
        droppedPackets++;
    }

#For individual send, receive and ack
    if (($1 == "+") && ($5 == "cbr") && ($3 == "2"))
    {
        udps++;
    }
    else if (($1 == "+") && ($5 == "tcp") && ($3 == "0"))
    {
        tcps++;
    }
    else if (($1 == "+") && ($5 == "ack") && ($3 == "4"))
    {
        acks++;
    }
    else if (($1 == "r") && ($5 == "cbr") && ($4 == "5"))
```

```
{
    udpr++;
}
else if (($1 == "r") && ($5 == "tcp") && ($4 == "4"))
{
    tcpr++;
}
else if (($1 == "r") && ($5 == "ack") && ($4 == "0"))
{
    ackr++;
}
else if (($1 == "d") && ($5 == "cbr"))
{
    udpd++;
}
else if (($1 == "d") && ($5 == "tcp"))
{
    tcpd++;
}
else if (($1 == "d") && ($5 == "ack"))
{
    ackd++;
}
#end-to-end delay
if (($1 == "+") && (seqno < $12))
{
    start_time[$12] = $2;
}
else if (($1 == "r") && ($5 == "cbr"))
{
    end_time[$12] = $2;
}
else if (($1 == "d") && ($5 == "cbr"))
{
    end_time[$6] = -1;
}
}

END {

    for(i=0; i<=seqno; i++) {

        if(end_time[i] > 0) {

            delay[i] = end_time[i] - start_time[i];

            count++;

        }

        else
```

```
        {  
            delay[i] = -1;  
        }  
    }  
  
    for(i=0; i<count; i++) {  
        if(delay[i] > 0) {  
            n_to_n_delay = n_to_n_delay + delay[i];  
        }  
    }  
  
    n_to_n_delay = n_to_n_delay/count;  
  
    print "\n";  
  
    print "Total no of GeneratedPackets          = "  
    seqno+1;  
  
    print "Total no of ReceivedPackets          = "  
    receivedPackets;  
  
    print "Total no of Dropped Packets = " droppedPackets;  
  
    print "Total Packet Delivery Ratio          = "  
    receivedPackets/(seqno+1)*100  
    "%";  
    print "Total no of TCP send                  = " tcps;  
  
    print "Total no of UDP send                  = " udps;  
  
    print "Total no of ACK send                  = " acks;  
  
    print "Total no of TCP receive              = " tcpr;  
  
    print "Total no of UDP receive              = " udpr;  
  
    print "Total no of ACK receive              = " ackr;  
  
    print "Total no of TCP drop                  = " tcpd;  
  
    print "Total no of UDP drop                  = " udpd;  
  
    print "Total no of ACK drop                  = " ackd;
```

```
        print "Average End-to-End Delay      = " n_to_n_delay"
s";

        print "\n";

    }
```

- **awk -f Third_script_analysis.awk out.tr**

9. Conclusion and Discussion: The way to plot the graph for QoS parameters.

10. Viva Questions:

- What are the performance parameters of network?
- What is the use of plotting the graph?

Networking Lab

Experiment No. : 11

Installation of Wire shark and Analysis of Packet headers.

Experiment No. 11

1. Aim: Installation of Wire shark and Analysis of Packets.

2. Objectives: To observe the performance in promiscuous & non-promiscuous mode & to find the packets based on different filters.

3. Hardware / Software Required: Wireshark, Ethereal and Tcpdump.

4. Theory:

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.

Applications:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals beside these examples can be helpful in many other situations too.

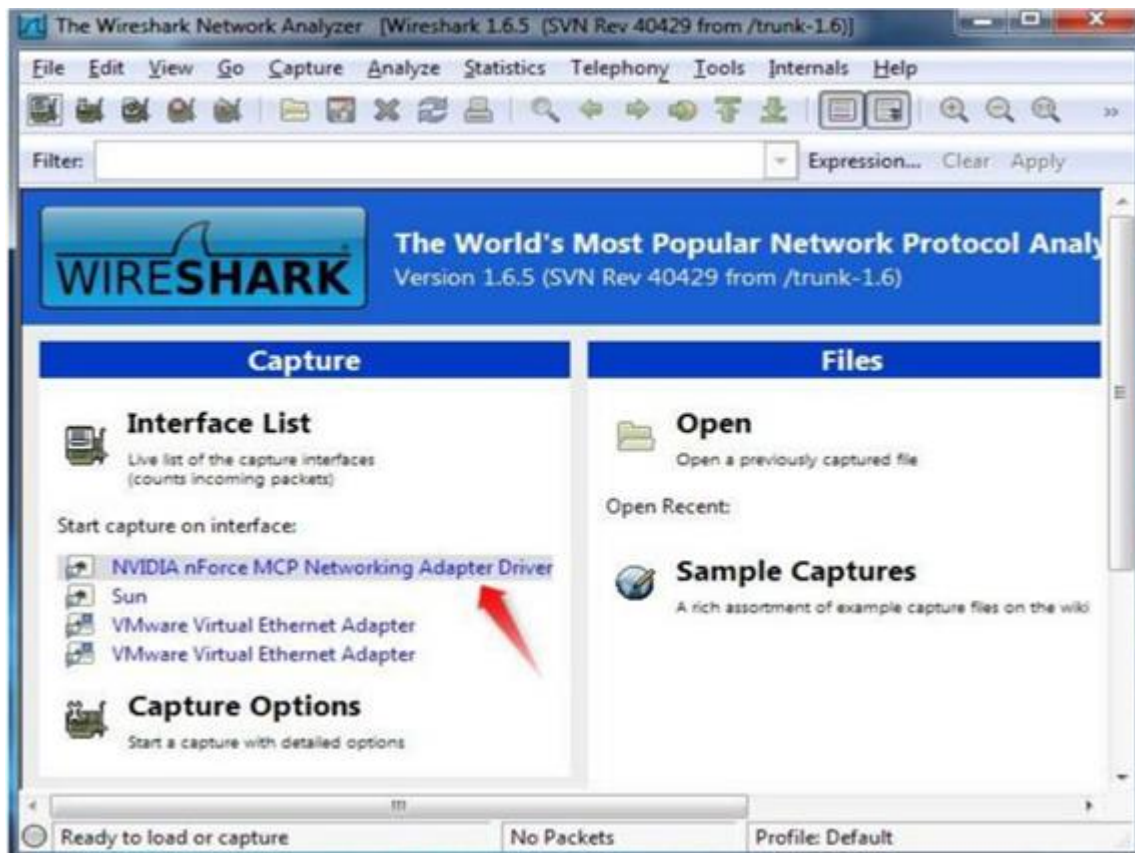
Features:

The following are some of the many features wireshark provides:

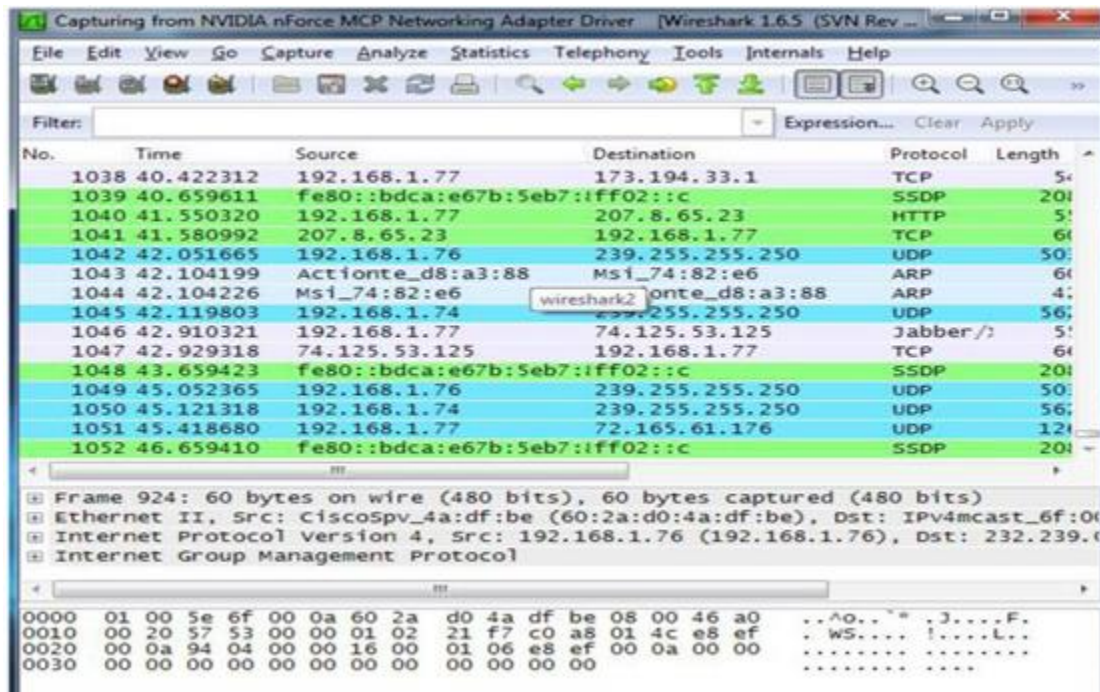
- Available for UNIX and Windows.
- Capture live packet data from a network interface.
- Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a number of other packet capture programs.
- Import packets from text files containing hex dumps of packet data. ☐ Display packets with very detailed protocol information.
- Export some or all packets in a number of capture file formats.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.
- Create various statistics.

Capturing Packets

After downloading and installing Wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface. For example, if you want to capture traffic on the wireless network, click your wireless interface. You can configure advanced features by clicking Capture Options.

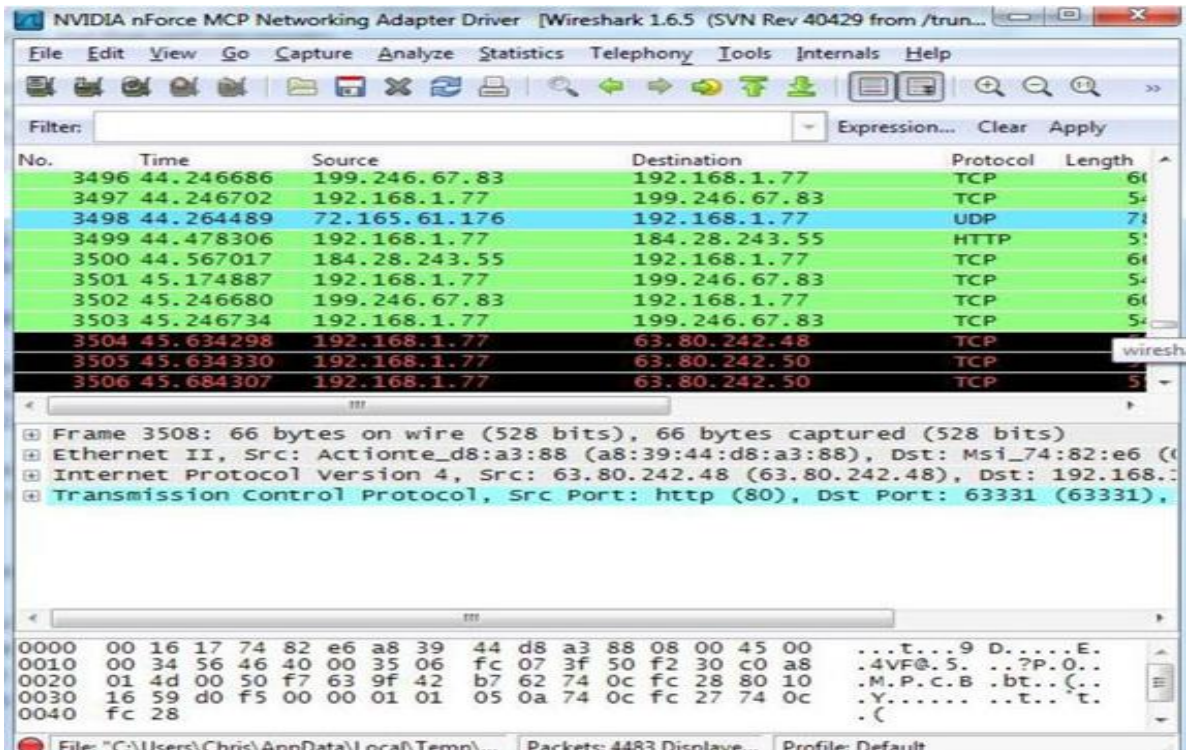


As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system. If you're capturing on a wireless interface and have promiscuous mode enabled in your capture options, you'll also see other the other packets on the network.



Click the stop capture button near the top left corner of the window when you want to stop capturing traffic

Wireshark uses colors to help you identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been delivered out-of-order.

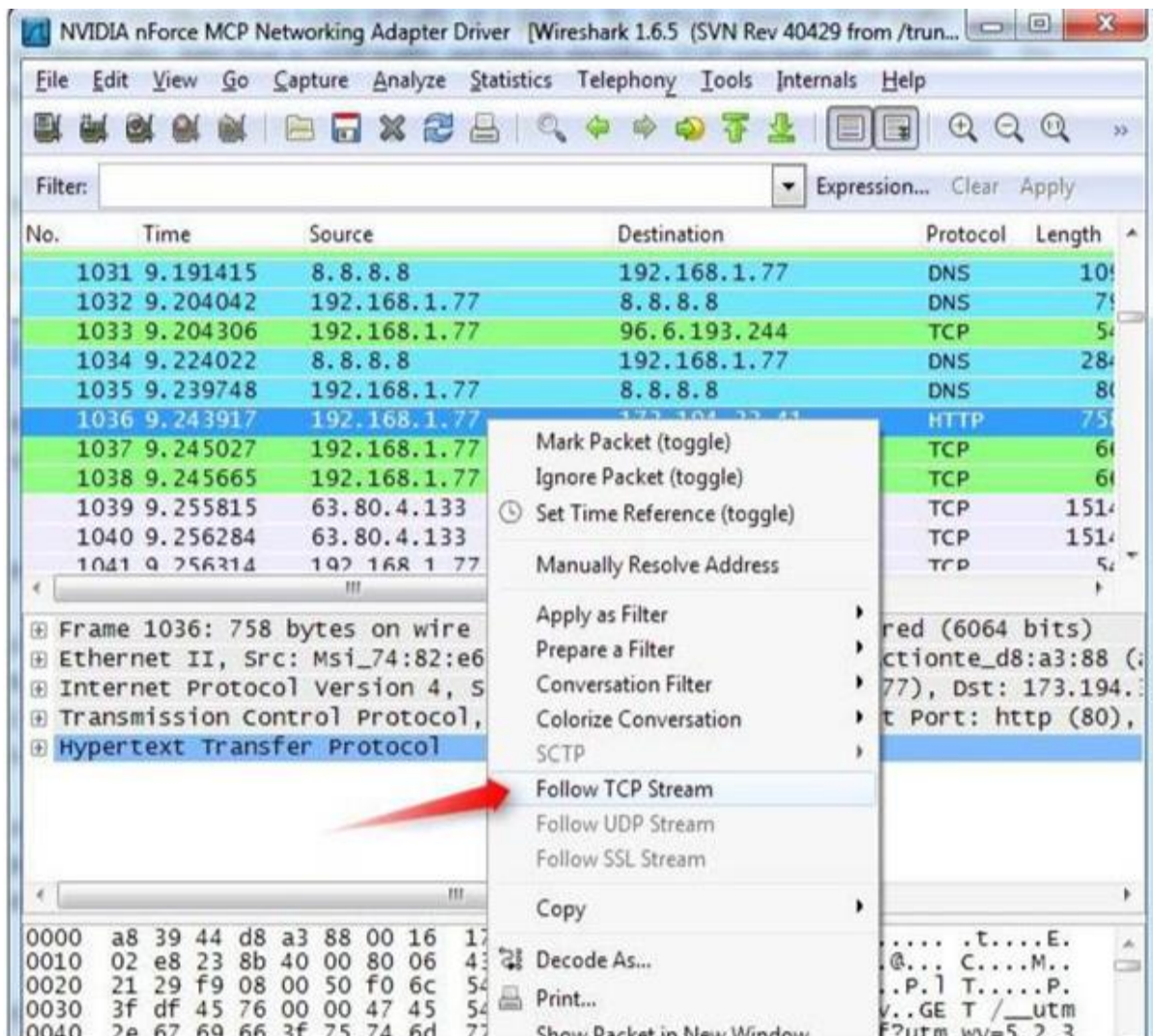


Filtering Packets

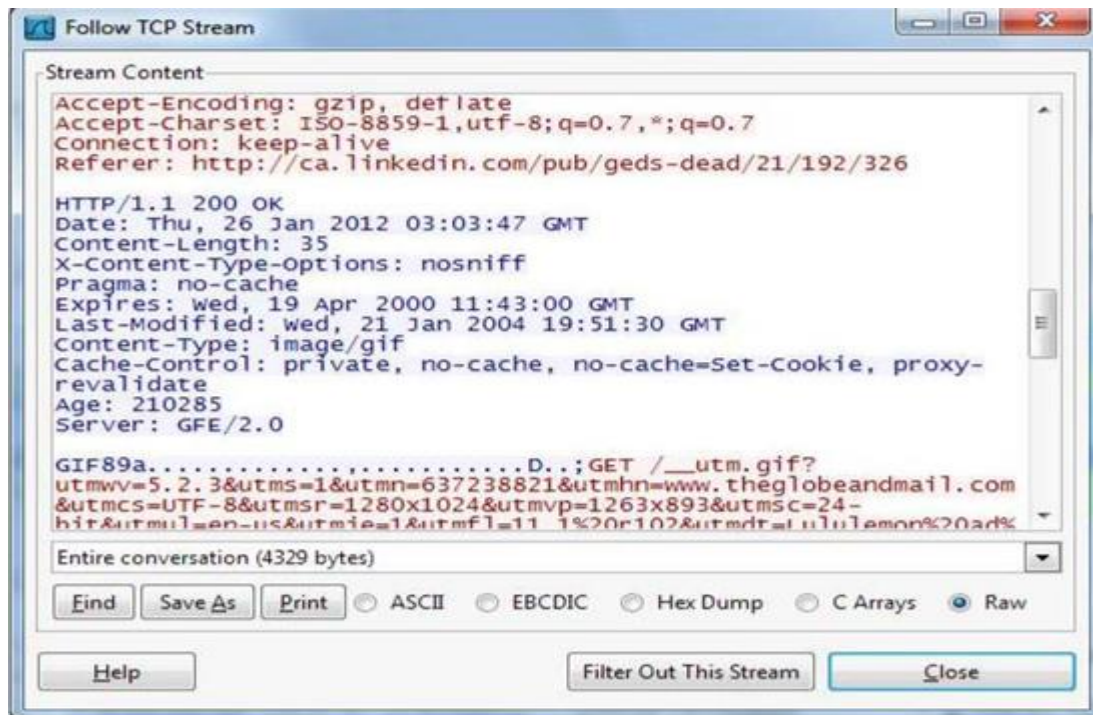
If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you'll likely have a large amount of packets to sift through. That's where Wireshark's filters come in

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type "dns" and you'll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.

Another interesting thing you can do is right-click a packet and select Follow TCP Stream



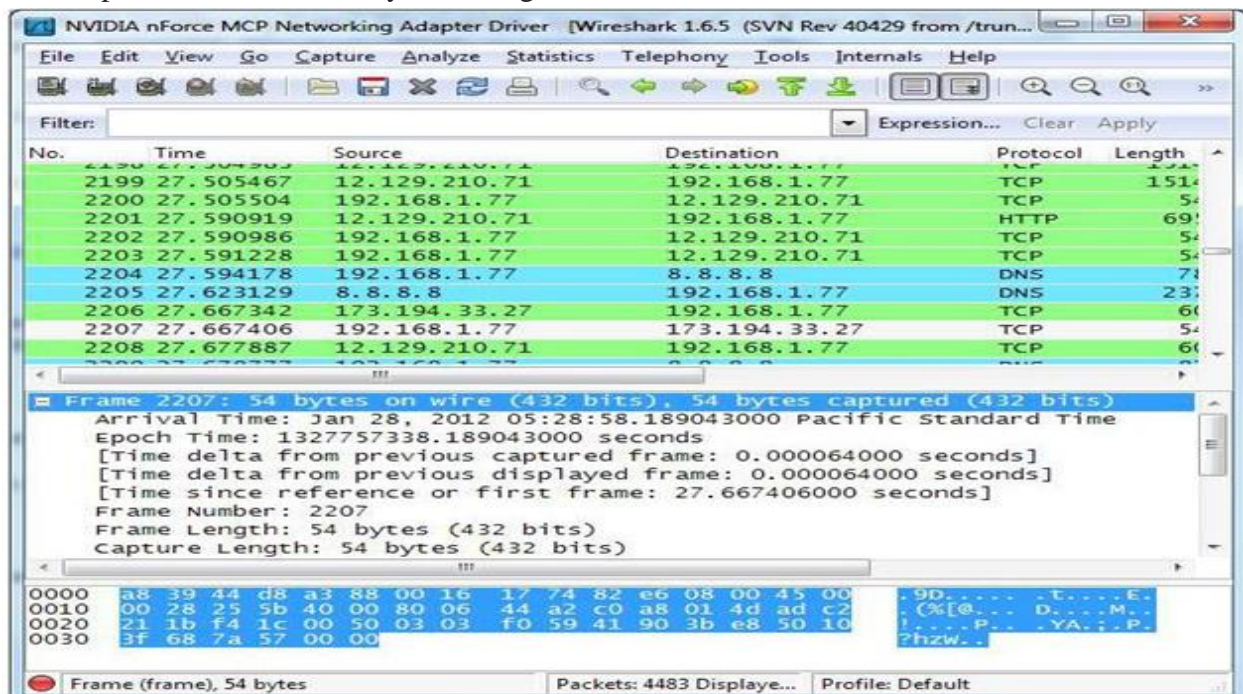
You'll see the full conversation between the client and the server.



Close the window and you'll find a filter has been applied automatically — Wireshark is showing you the packets that make up the conversation.

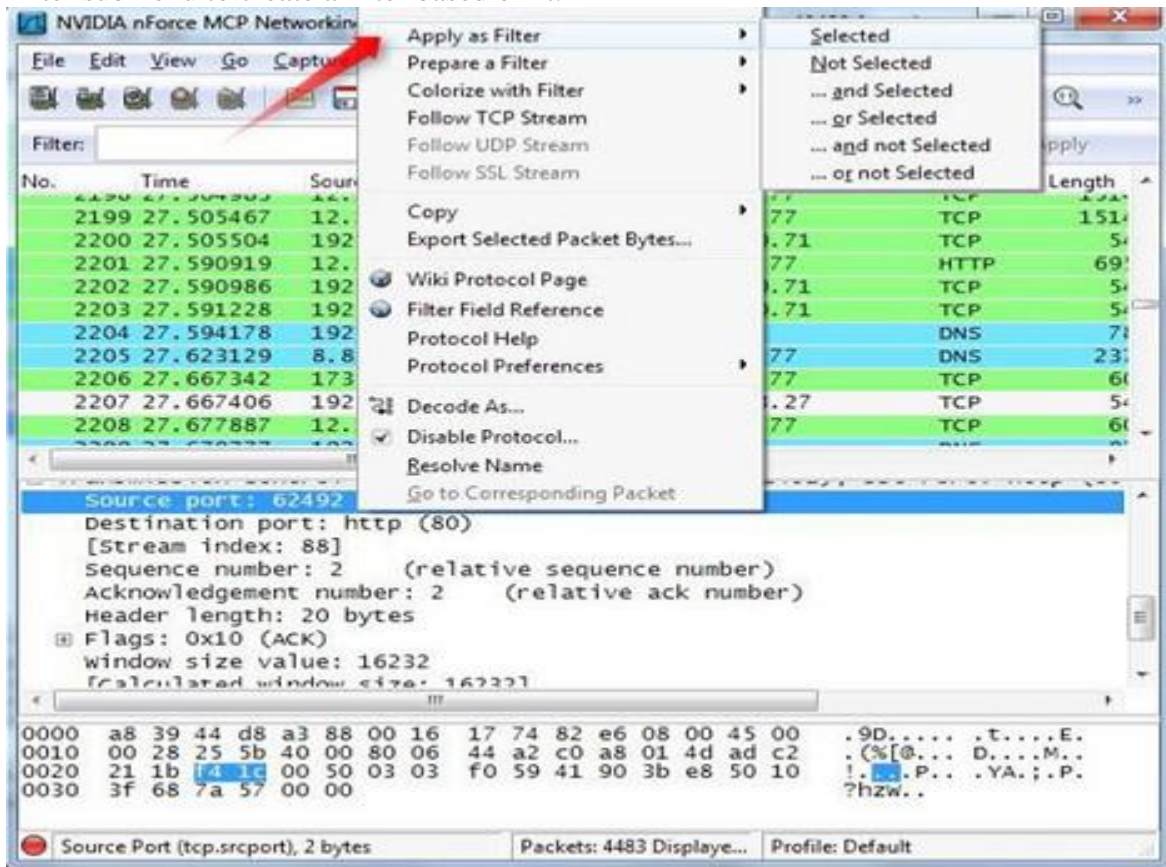
Inspecting Packets

Click a packet to select it and you can dig down to view its details.



You can also create filters from here — just right-click one of the details and use the Apply

as Filter submenu to create a filter based on it.



Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

5.Conclusion:

In this experiment we analyze packet sniffing tool that monitor network traffic transmitted between legitimate users or in the network. The packet sniffer is network monitoring tool. It is opted for network monitoring, traffic analysis, troubleshooting, Packet grapping, message, protocol analysis, penetration testing and many other purposes.

6.Viva Questions:

- 1) What is packet sniffer?
- 2) How to sniff passwords with wireshark?
- 3) List packet sniffing tools other than mentioned above?

7.References:

1. <http://netsecurity.about.com/od/informationresources/a/What-Is-A-Packet-Sniffer.htm>
2. <https://samsclass.info/120/proj/p3-wireshark.htm>
3. <http://sectools.org/tag/sniffers/>

Networking Lab

Experiment No. : 12

Socket Programming with C

Client Server Model

Experiment No. 12

1. **Aim:** Using TCP/IP Sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.
2. **What will you learn by performing this experiment?**

Sockets are a protocol independent method of creating a connection between processes. Sockets can be either

- Connection based or connectionless: Is a connection established before communication or does each packet describe the destination?
- Packet based or streams based: Are there message boundaries or is it one stream?
- Reliable or unreliable: Can messages be lost, duplicated, reordered, or corrupted?

3. **Software Required:** Ubuntu , NS 2.34

4. **Theory:**

Socket characteristics

Sockets are characterized by their domain, type and transport protocol. Common domains are:

- AF_UNIX: address format is UNIX pathname
- AF_INET: address format is host and port number

Common types are:

- virtual circuit: received in order transmitted and reliably
- datagram: arbitrary order, unreliable

Each socket type has one or more protocols. Ex:

- TCP/IP (virtual circuits)
- UDP (datagram)

Use of sockets:

- Connection-based sockets communicate client-server: the server waits for a connection from the client
- Connectionless sockets are peer-to-peer: each process is symmetric.

Socket APIs

- socket: creates a socket of a given domain, type, protocol (buy a phone)
- bind: assigns a name to the socket (get a telephone number)

- Listen: specifies the number of pending connections that can be queued for a server socket. (call waiting allowance)
- accept: server accepts a connection request from a client (answer phone)
- connect: client requests a connection request to a server (call)
- send, sendto: write to connection (speak)
- recv, recvfrom: read from connection (listen)
- shutdown: end the call

Connection-based communication

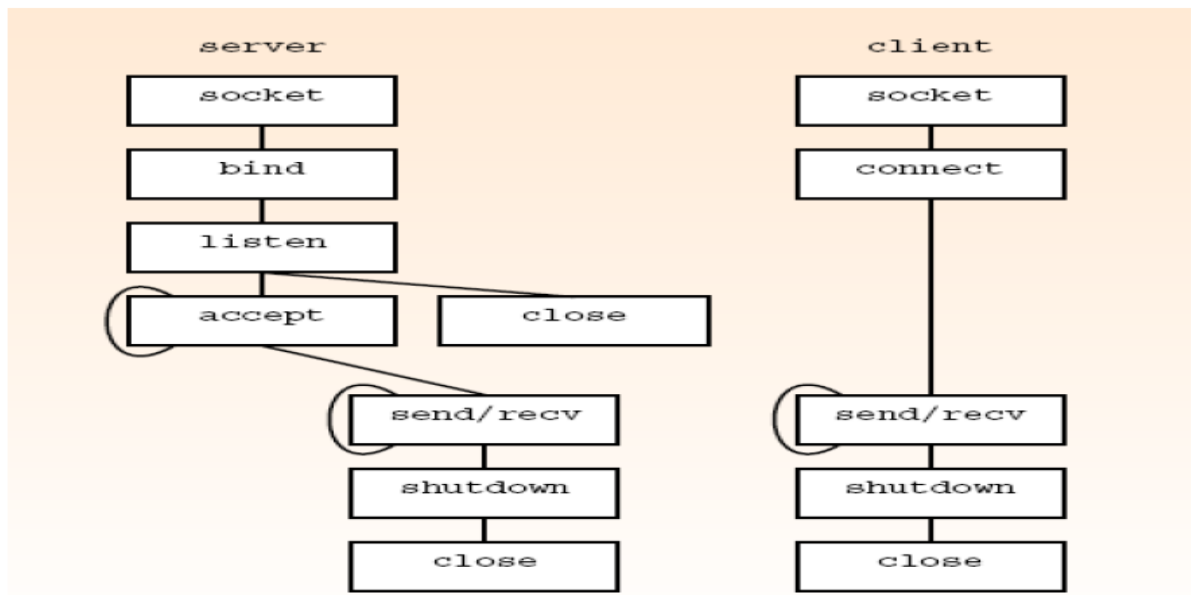
Server performs the following actions

- socket: create the socket
- bind: give the address of the socket on the server
- listen: specifies the maximum number of connection requests that can be pending for this process
- accept: establish the connection with a specific client
- send, recv: stream-based equivalents of read and write (repeated)
- shutdown: end reading or writing
- close: release kernel data structures

Client performs the following actions

- socket: create the socket
- connect: connect to a server
- send, recv: (repeated)
- shutdown
- close

TCP-based sockets



socket API

```
#include<sys/types.h>
```

```
#include<sys /socket.h>
```

```
int socket(int domain, int type, int protocol) ;
```

Returns a file descriptor (called a socket ID) if successful, -1 otherwise. Note that the socket returns a socket descriptor which is the same as a file descriptor.

The domain is AF_INET.

The type argument can be:

- SOCK_STREAM: Establishes a virtual circuit for stream
- SOCK_DGRAM: Establishes a datagram for communication
- SOCK_SEQPACKET: Establishes a reliable, connection based, two way communication with maximum message size. (This is not available on most machines.)

Protocol is usually zero, so that type defines the connection within domain.

bind

```
#include <sys / types.h>
```

```
#include<sys / socket.h>
```

```
int bind(int sid, struct sockaddr *addrPtr, int len)
```

Where

- sid: is the socket id
- addrPtr: is a pointer to the address family dependent address structure
- len: is the size of *addrPtr

Associates a socket id with an address to which other processes can connect. In internet protocol the address is [ipNumber, portNumber]

sockaddr

For the internet family:

```
struct sockaddr_in {  
sa_family_t sin_family; // = AF_INET  
in_port_t sin_port; // is a port number  
struct in_addr sin_addr; // an IP address  
}
```

listen

```
#include <sys / types.h>
```

```
#include <sys / socket.h>
```

```
int listen (int sid, int size) ;
```

Where size is the number of pending connection requests allowed (typically limited by Unix kernels to 5). Returns the 0 on success, or -1 if failure.

accept

```
#include <sys / types.h>
```

```
#include <sys / socket.h>
```

```
int accept(int sid ,struct sockaddr *addrPtr , int *lenPtr )
```

Returns the socketId and address of client connecting to socket. If lenPtr or addrPtr equal zero, no address structure is returned. lenPtr is the maximum size of address structure that can be called, returns the actual value. Waits for an incoming request, and when received creates a socket for it.

send

```
#include <sys / types.h>
```

```
#include <sys / socket.h>
```

```
int send(int sid ,const char *bufferPtr ,int len ,int flag)
```

Send a message. Returns the number of bytes sent or -1 if failure. (Must be a bound socket). Flag is either

- 0: default
- MSG OOB: Out-of-band high priority communication

recv

```
#include <sys / types.h>
```

```
#include <sys / socket.h>
```

```
int recv ( int sid , char *bufferPtr , int len , int flags)
```

Receive up to len bytes in bufferPtr. Returns the number of bytes received or -1 on failure. Flags can be either

- 0: default
- MSG OOB: out-of-bound message
- MSG PEEK: look at message without removing

Shutdown

```
#include <sys / types.h>
```

```
#include <sys / socket.h>
```

```
int shutdown ( int sid , int how)
```

Disables sending (how=1 or how=2) or receiving (how=0 or how=2). Returns -1 on failure.

Connect

-this is the first of the client calls

```
#include <sys / types.h>
```

```
#include <sys / socket.h>
```

```
int connect ( int sid , struct sockaddr *addrPtr , int len)
```

Specifies the destination to form a connection with (addrPtr), and returns a 0 if successful, -1 otherwise.

Port usage

Note that the initiator of communications needs a fixed port to target communications. This means that some ports must be reserved for these —well known ports.

Port usage:

- 0-1023: These ports can only be binded to by root
- 1024-5000: well-known ports
- 5001-64K-1: ephemeral ports

5. Source Code:

Client Side:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<string.h>
#define SERV_TCP_PORT 6880
#define SERV_HOST_ADDR "127.0.0.1"
int main()
{
    int sockfd;
    struct sockaddr_in serv_addr,cli_addr;
    char filename[100],buf[1000];
    int n;
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=inet_addr(SERV_HOST_ADDR);
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0)
        printf("Client:cant open stream socket\n");
    else
        printf("Client:stream socket opened successfully\n");
    if(connect(sockfd,(struct sockaddr *)&serv_addr, sizeof(serv_addr))<0)
        printf("Client:cant connect to server\n");
    else
        printf("Client:connected to server successfully\n");
    printf("\n Enter the file name to be displayed :");
```

```
scanf("%s",filename);
write(sockfd,filename,strlen(filename));
printf("\n filename transfered to server\n");
n=read(sockfd,buf,1000);
if(n < 0)
    printf("\n error reading from socket");
printf("\n Client : Displaying file content of %s\n",filename);
fputs(buf,stdout);
close(sockfd);
exit(0);
}
```

SERVER SIDE:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<string.h>
#define SERV_TCP_PORT 6880
#define SERV_HOST_ADDR "127.0.0.1"
int main()
{ int sockfd,newsockfd,clilen;
  struct sockaddr_in cli_addr,serv_addr;
  char filename[25],buf[1000];
  int n,m=0;
  int fd;
  if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0)
      printf("server:cant open stream socket\n");
  else
      printf("server:stream socket opened successfully\n");
  serv_addr.sin_family=AF_INET;
  serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
  serv_addr.sin_port=htons(SERV_TCP_PORT);
  if((bind(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)))<0)
      printf("server:cant bind local address\n");
  else
      printf("server:bound to local address\n");
  listen(sockfd,5);
```

```
printf("\n SERVER : Waiting for client...\n");
for (;;)
{
    clilen=sizeof(cli_addr);
    newsockfd=accept(sockfd,(struct sockaddr *) &cli_addr,&clilen);
    if(newsockfd<0)
        printf("server:accept error\n");
    else
        printf("server:accepted\n");
    n=read(newsockfd,filename,25);
    filename[n]='\0';
    printf("\n SERVER : %s is found and ready to transfer
\n",filename);
    fd=open(filename,O_RDONLY);
    n=read(fd,buf,1000);
    buf[n]='\0';
    write(newsockfd,buf,n);
    printf("\n transfer success\n");
    close(newsockfd);
    exit(0)
} }
```

6. Conclusion and Discussion:

Sockets are a mechanism for exchanging data between processes. These processes can either be on the same machine, or on different machines connected via a network. Once a socket connection is established, data can be sent in both directions until one of the endpoints closes the connection.

7. Result :

8. Output:

AT CLIENT SIDE

[root@localhost]# cc tcpc.c

[root@localhost]# ./a.out

Data Sent

File Content....

Sockets are a mechanism for exchanging data between processes. These processes can either be on the same machine, or on different machines connected via a network. Once a socket connection is established, data can be sent in both directions until one of the endpoints closes the connection.

I needed to use sockets for a project I was working on, so I developed and refined a few C++ classes to encapsulate the raw socket API calls. Generally, the application requesting the data is called the client, and the application servicing the request is called the server. I created two primary classes, ClientSocket and ServerSocket, that the client and server could use to exchange data.

Output:

```
[root@localhost ]# cc tcps.c  
[root@localhost ]# ./a.out  
Received the file name: data.txt  
File content sent
```

9. Viva Questions:

- a. What do you mean by socket?
- b. How to setup the socket communication among client and server?

10. References:

www.cs.virginia.edu/~cs757/slidespdf/cs757-ns2.pdf

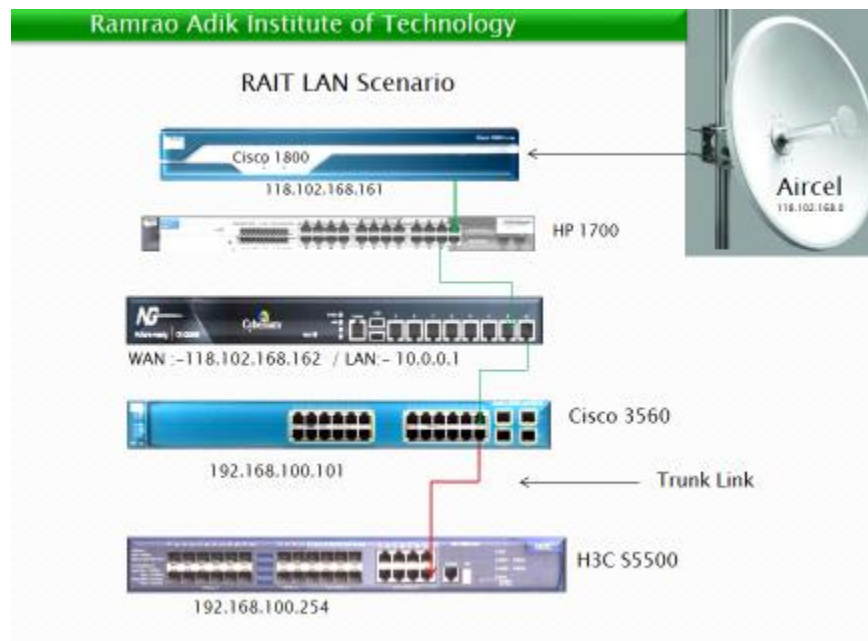
Networking Lab

Experiment No. : 13

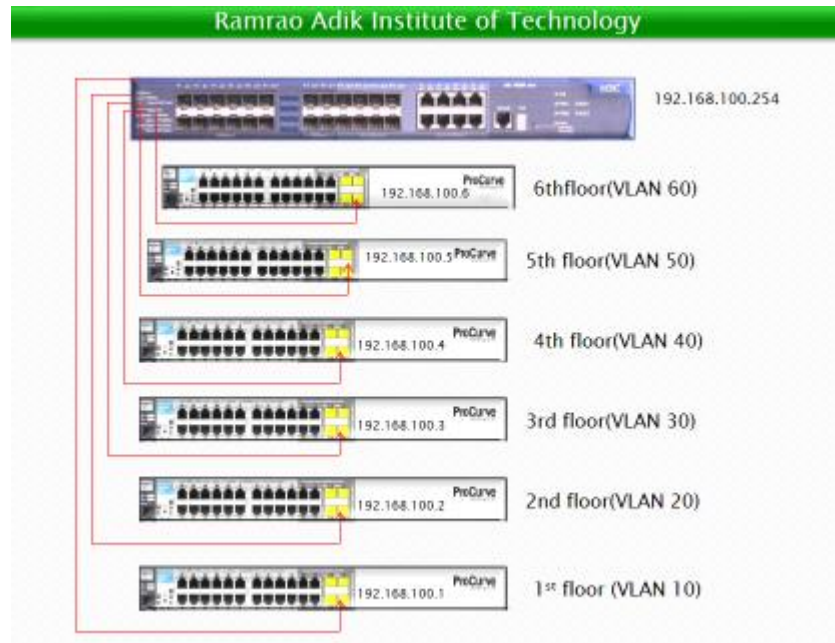
**A case study to design and configure
organization network.**

Case Study : - Design a Structured Managed Network for a Domain specific Organization.

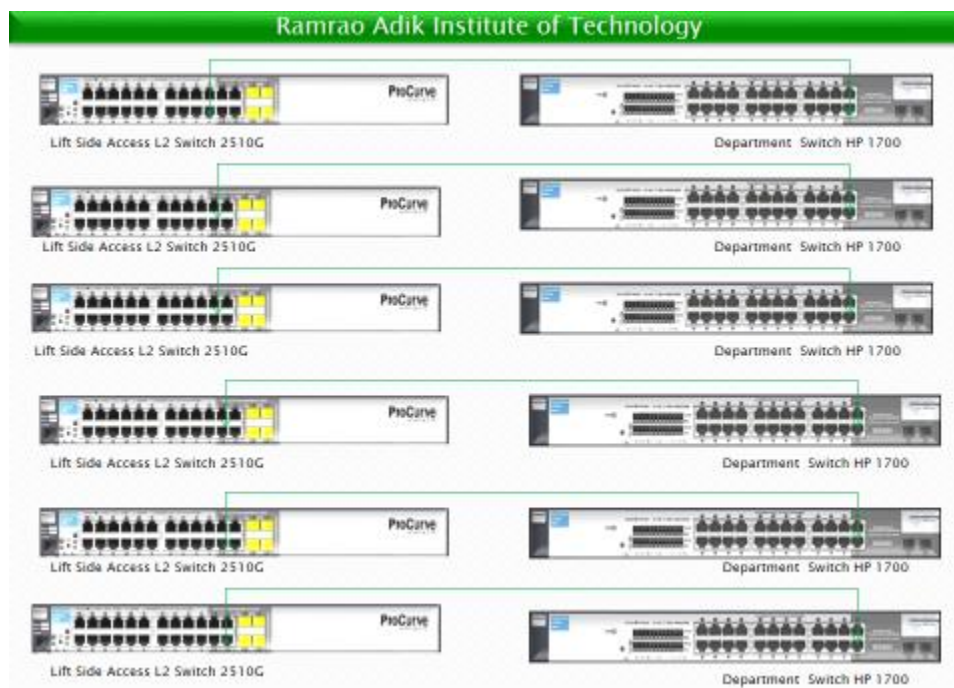
Students has to design a structured network for the selected domain specific organization like for a company in Ecommerce field or small business firm. Here while designing they have to consider the collision domain concept so that traffic can be regularize and optimal bandwidth utilization is possible. The following design are discussed for RAIT with students and given the guidance for providing solution.



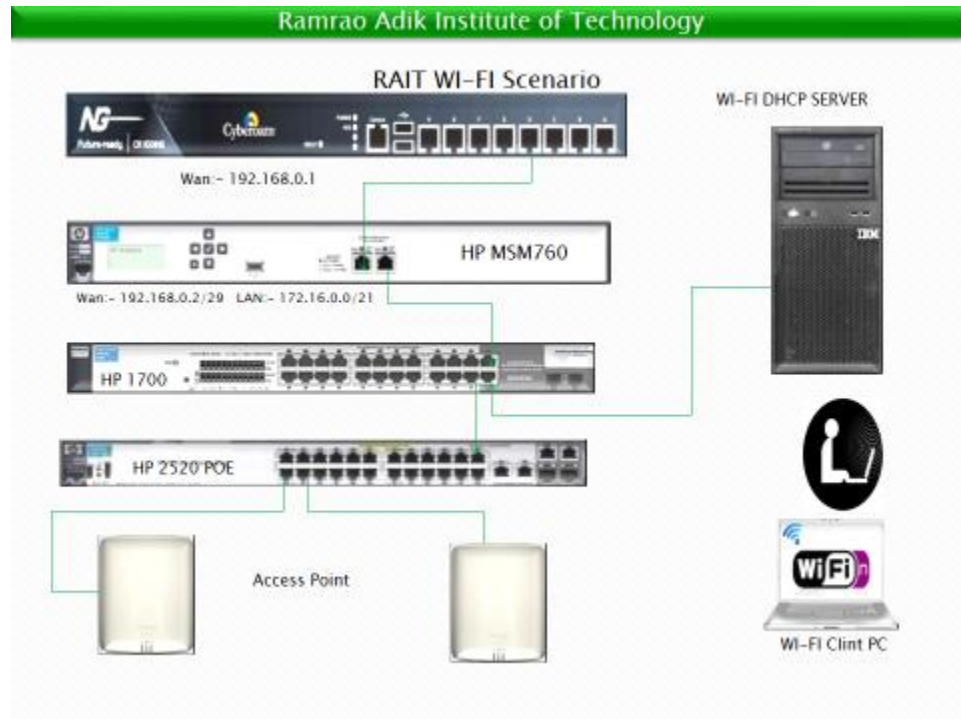
Broader Design of Network Distribution



Floor Plan



Distribution Switch Design



Core LAN. Backbone of Network