

Networking with Linux

Subject Code: MCAL26

A Practical Journal Submitted in Fulfillment
of the Degree of

MASTER

In

COMPUTER APPLICATION

Year 2023-2024

By

Mr. Jadhav Harshal Sanjay

(Application Id-79763)

Semester-II

Under the Guidance of

Prof. Kavita Chouk



Centre for Distance and Online Education
Vidya Nagari, Kalina, Santacruz East – 400098.
University of Mumbai

PCP Center

[Satish Pradhan Dyanasadhana College, Thane]



Institute of Distance and Open Learning
Vidya Nagari, Kalina, Santacruz East – 400098.

CERTIFICATE

This to certify that, **"Jadhav Harshal Sanjay "** appearing **Master's in computer application (Semester II) Application ID: 79763** has satisfactory completed the prescribed practical of **MCAL26 -Networking with Linux** as laid down by the University of Mumbai for the academic year 2023-24.

Teacher In Charge

External Examiner

Coordinator – M.C.A

Date:

Place: -

INDEX

Exercise	Topic	Page No.	Signature
1	Installation of NS3 (Network Simulator)	1	
2	Installation of NetAnim	4	
3	Installation of Wireshark	6	
4	Program to simulate UDP server client	9	
5	Program to simulate FTP using TCP protocol	12	
6	Analyze the network traffic using WireShark	16	
7	Analyze the performance parameters of network using Wire Shark	23	

Practical No. 01

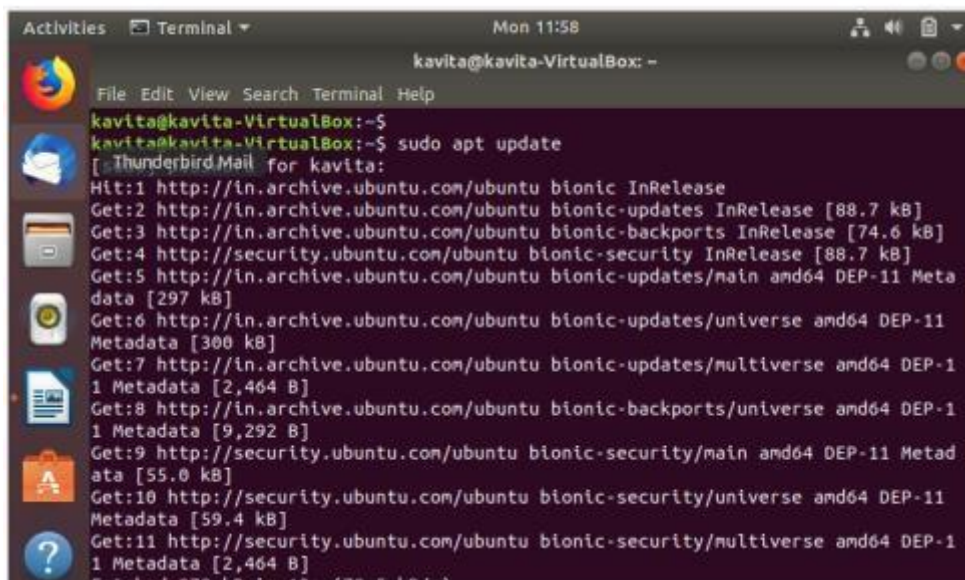
Aim: Installation of NS3 (Network Simulator):

Network Simulation is the software which tells the behaviour of computer networks. It is common useful method to calculate different network topologies exclusive of real-world implementation.

For NS3 installation on Ubuntu 18 version.

We have to execute following commands

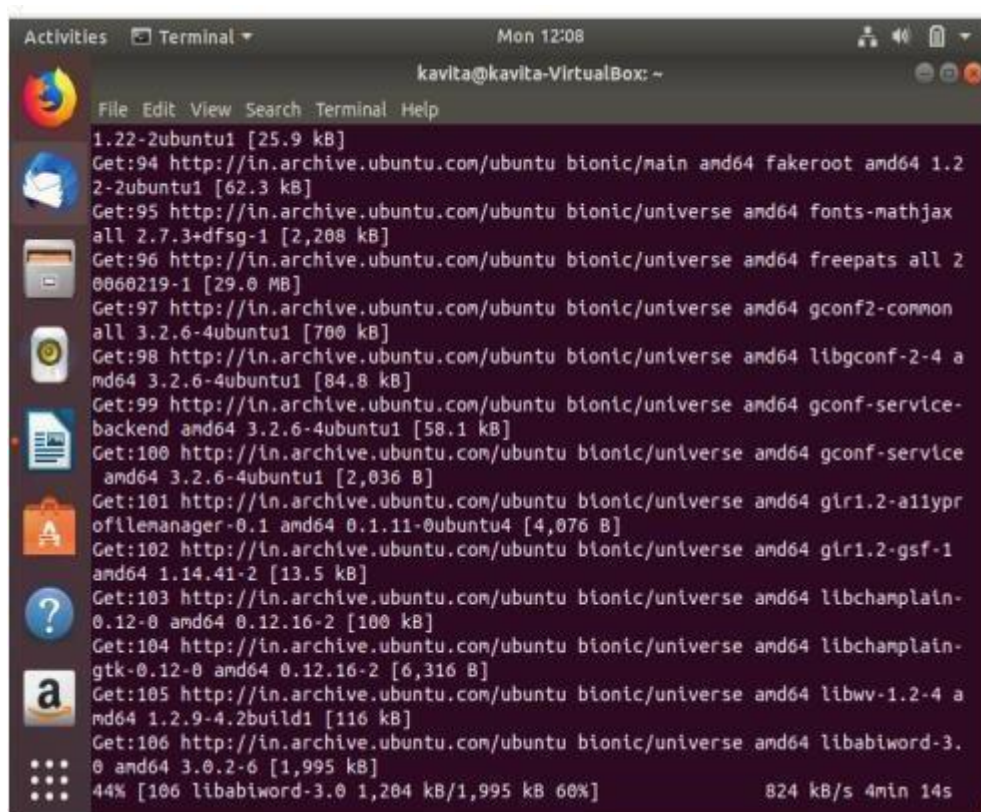
Step 1: sudo apt update



```
kavita@kavita-VirtualBox:~$ sudo apt update
[Thunderbird Mail for kavita:
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 DEP-11 Meta
data [297 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 DEP-11
Metadata [300 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 DEP-1
1 Metadata [2,464 B]
Get:8 http://in.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-1
1 Metadata [9,292 B]
Get:9 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Meta
ata [55.0 kB]
Get:10 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 DEP-11
Metadata [59.4 kB]
Get:11 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 DEP-1
1 Metadata [2,464 B]
Fetched 878 kB in 12s (70.5 kB/s)
```

Step 2: sudo apt install g++ python3 python3-dev pkg-config sqlite3 cmake python3-setuptools git qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools gir1.2-gobject-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3 openmpi-bin openmpi-common openmpi-doc libopenmpi-dev autoconf cvs bzip2 unrar gsl-bin libgsl-dev libgslcblas0 wireshark tcpdump sqlite sqlite3 libsqlite3-dev libxml2 libxml2-dev libc6-dev libc6-dev-i386 libclang-dev llvm-dev automake python3-pip libxml2 libxml2-dev libboost-all-dev

Step 3: Download ns-allinone-3.36.1.tar.bz2 from the website nsnam.org. <https://www.nsnam.org/releases/ns-allinone-3.36.1.tar.bz2>



```

kavita@kavita-VirtualBox: ~
1.22-2ubuntu1 [25.9 kB]
Get:94 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 fakeroot amd64 1.2
2-2ubuntu1 [62.3 kB]
Get:95 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-mathjax
all 2.7.3+dfsg-1 [2,208 kB]
Get:96 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 freepats all 2
0060219-1 [29.0 MB]
Get:97 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 gconf2-common
all 3.2.6-4ubuntu1 [700 kB]
Get:98 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 libgconf-2-4 a
md64 3.2.6-4ubuntu1 [84.8 kB]
Get:99 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 gconf-service-
backend amd64 3.2.6-4ubuntu1 [58.1 kB]
Get:100 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 gconf-service
amd64 3.2.6-4ubuntu1 [2,036 B]
Get:101 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 glr1.2-allipr
offilemanager-0.1 amd64 0.1.11-0ubuntu4 [4,076 B]
Get:102 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 glr1.2-gsf-1
amd64 1.14.41-2 [13.5 kB]
Get:103 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 libchamplain-
0.12-0 amd64 0.12.16-2 [100 kB]
Get:104 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 libchamplain-
gtk-0.12-0 amd64 0.12.16-2 [6,316 B]
Get:105 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 libbw-1.2-4 a
md64 1.2.9-4.2build1 [116 kB]
Get:106 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 libabiword-3.
0 amd64 3.0.2-6 [1,995 kB]
44% [106 libabiword-3.0 1,204 kB/1,995 kB 60%] 824 kB/s 4min 14s

```

Step 4: Unzip the above file content to the home folder (in my case, its /home) - Check your home folder and do it accordingly.

To unzip use the GUI with Right click and extract and select the /home folder.

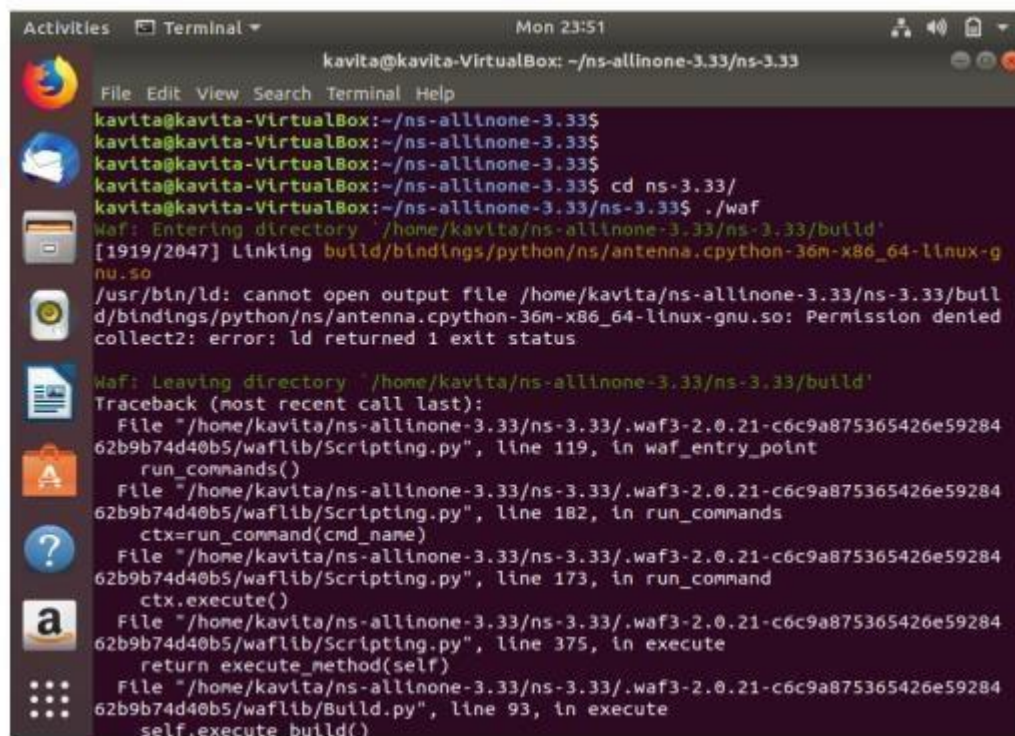
Else you can use the command

\$ tar jxvf ns-allinone-3.36.1.tar.bz2

Step 5: Step 4: Go to the folder

\$ cd ns-allinone-3.36.1/

\$./build.py --enable-examples --enable-tests

A terminal window titled 'kavita@kavita-VirtualBox: ~/ns-allinnone-3.33/ns-3.33' showing a failed build process. The user runs 'cd ns-3.33/' and './waf'. The output shows 'Waf: Entering directory "/home/kavita/ns-allinnone-3.33/ns-3.33/build"' followed by a linking error: '[1919/2047] Linking build/bindings/python/ns/antenna.cpython-36m-x86_64-linux-gnu.so' and '/usr/bin/ld: cannot open output file /home/kavita/ns-allinnone-3.33/ns-3.33/build/bindings/python/ns/antenna.cpython-36m-x86_64-linux-gnu.so: Permission denied collect2: error: ld returned 1 exit status'. A traceback follows, showing the error occurred in 'waf3-2.0.21-c6c9a875365426e5928462b9b74d40b5/waf3lib/Scripting.py' at line 119, line 182, line 173, line 375, and line 93.

```
kavita@kavita-VirtualBox: ~/ns-allinnone-3.33/ns-3.33
kavita@kavita-VirtualBox:~/ns-allinnone-3.33$
kavita@kavita-VirtualBox:~/ns-allinnone-3.33$
kavita@kavita-VirtualBox:~/ns-allinnone-3.33$ cd ns-3.33/
kavita@kavita-VirtualBox:~/ns-allinnone-3.33/ns-3.33$ ./waf
Waf: Entering directory '/home/kavita/ns-allinnone-3.33/ns-3.33/build'
[1919/2047] Linking build/bindings/python/ns/antenna.cpython-36m-x86_64-linux-g
nu.so
/usr/bin/ld: cannot open output file /home/kavita/ns-allinnone-3.33/ns-3.33/buil
d/bindings/python/ns/antenna.cpython-36m-x86_64-linux-gnu.so: Permission denied
collect2: error: ld returned 1 exit status
Waf: Leaving directory '/home/kavita/ns-allinnone-3.33/ns-3.33/build'
Traceback (most recent call last):
  File "/home/kavita/ns-allinnone-3.33/ns-3.33/.waf3-2.0.21-c6c9a875365426e59284
62b9b74d40b5/waf3lib/Scripting.py", line 119, in waf_entry_point
    run_commands()
  File "/home/kavita/ns-allinnone-3.33/ns-3.33/.waf3-2.0.21-c6c9a875365426e59284
62b9b74d40b5/waf3lib/Scripting.py", line 182, in run_commands
    ctx=run_command(cmd_name)
  File "/home/kavita/ns-allinnone-3.33/ns-3.33/.waf3-2.0.21-c6c9a875365426e59284
62b9b74d40b5/waf3lib/Scripting.py", line 173, in run_command
    ctx.execute()
  File "/home/kavita/ns-allinnone-3.33/ns-3.33/.waf3-2.0.21-c6c9a875365426e59284
62b9b74d40b5/waf3lib/Scripting.py", line 375, in execute
    return execute_method(self)
  File "/home/kavita/ns-allinnone-3.33/ns-3.33/.waf3-2.0.21-c6c9a875365426e59284
62b9b74d40b5/waf3lib/Build.py", line 93, in execute
    self.execute build()
```

Step 6: Once the installation is done. You can run the example as shown

\$ cd ns-3.36.1/

\$./ns3 run hello-simulator

Hello Simulator

Practical No. 02

Aim: Installation of NetAnim

NetAnim is the network Animator that comes with NS3. During compilation of process of NS3 NetAnim may not be compiled .so we need to compile NetAnim. It is an offline network animator tool that now comes with NS3 that nsallinone3. All versions. By using NetAnim we can animate NS3 simulator, using the xml file trace the output in the simulation. NetAnim is the software which execute xml file to generate graphical output on NS3 simulator

Method1:

NetAnim can be downloaded from the
URL <http://code.nsnam.org/jabraham3/netanim-3.104>

Method 2:

Step 1:

`sudo apt-get install NetAnim`

The above command is case sensitive (NetAnim)

Step 2:

To run once it's installed

NetAnim file.xml

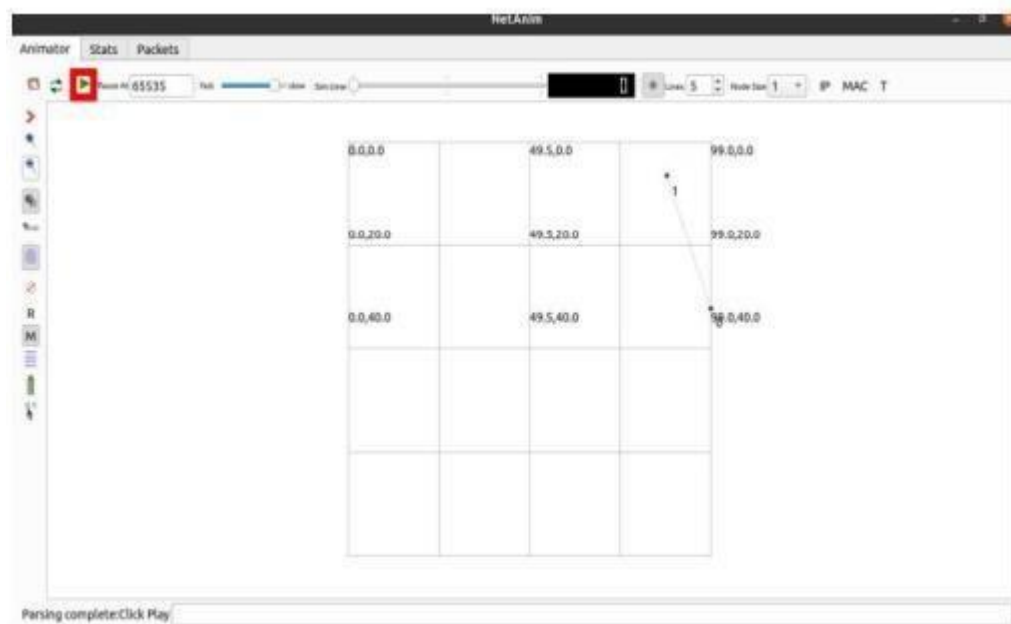
If you already has a file.xml.



Step 3: Select xml file



Step 4: Run the simulation by clicking

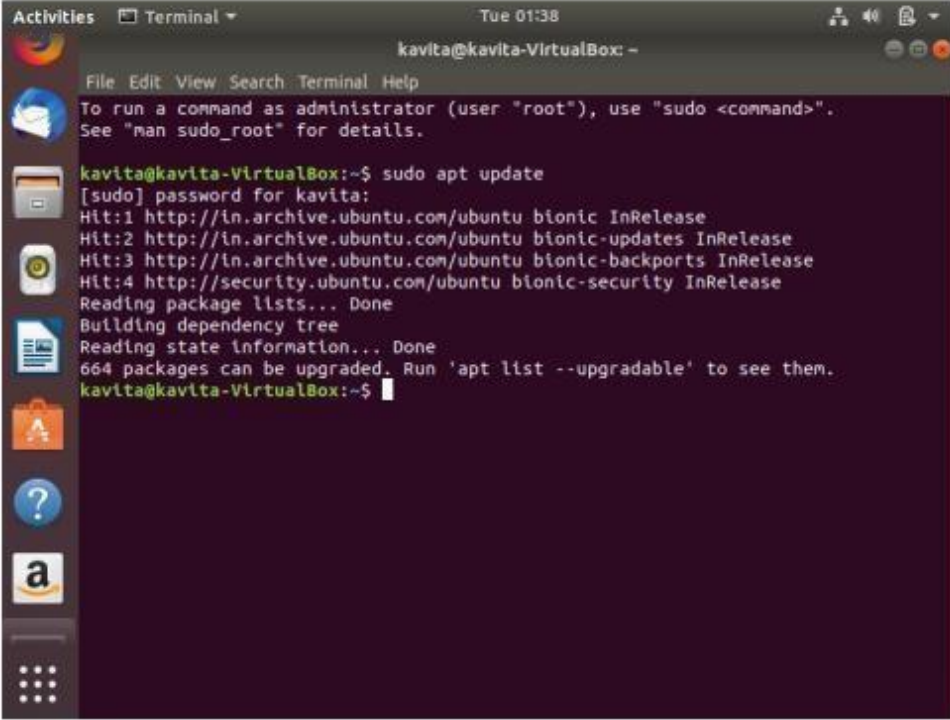


Practical No.03

Aim: Installation of Wireshark

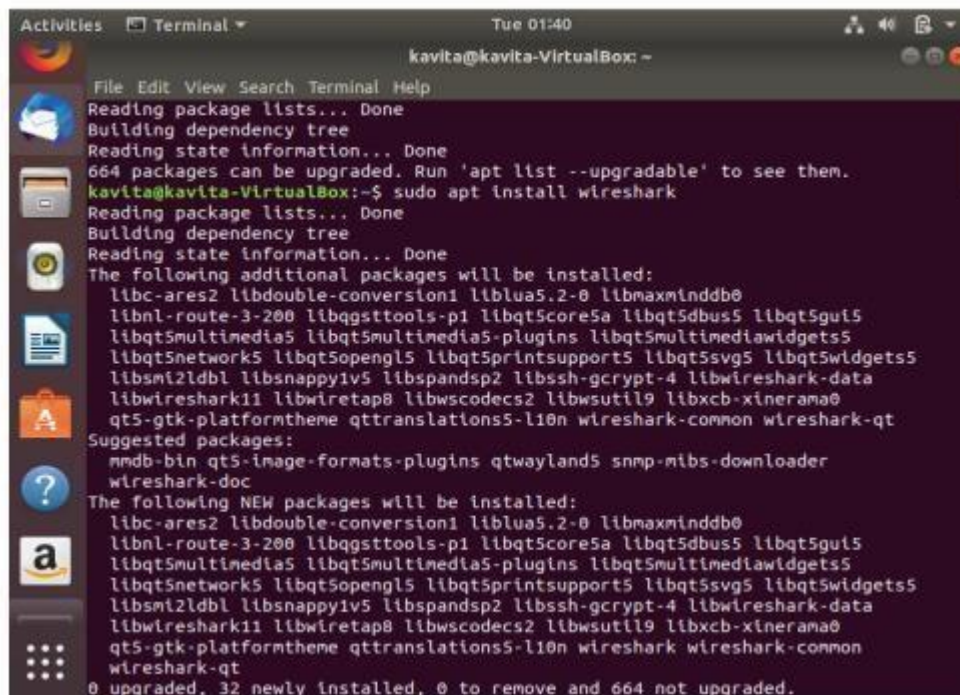
It is the network protocol for analyzing freely available packages. Wireshark is network packet analyzer. It is used to check incoming and outgoing packets in the network and save it for offline analysis. It works on Windows, Linux, macOS, FreeBSD etc.

Step 1: Update the system by using `sudo apt update`



```
Activities Terminal Tue 01:38
kavita@kavita-VirtualBox: ~
File Edit View Search Terminal Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

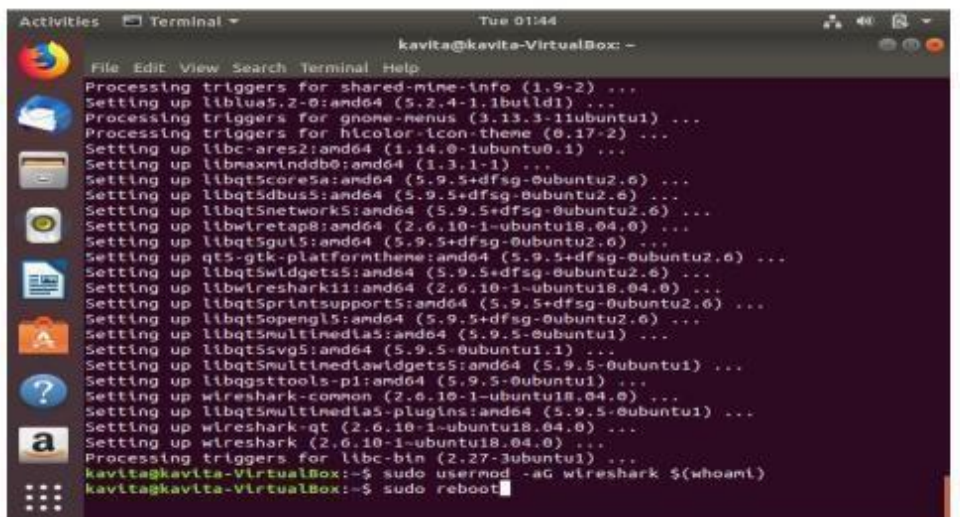
kavita@kavita-VirtualBox:~$ sudo apt update
[sudo] password for kavita:
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
664 packages can be upgraded. Run 'apt list --upgradable' to see them.
kavita@kavita-VirtualBox:~$
```

Step2: Install wireshark using sudo apt install wireshark


```

kavita@kavita-VirtualBox: ~
File Edit View Search Terminal Help
Reading package lists... Done
Building dependency tree
Reading state information... Done
664 packages can be upgraded. Run 'apt list --upgradable' to see them.
kavita@kavita-VirtualBox:~$ sudo apt install wireshark
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libbc-ares2 libdouble-conversion1 liblua5.2-0 libmaxminddb0
  libnl-route-3-200 libqgsttools-p1 libqt5core5a libqt5dbus5 libqt5gui5
  libqt5multimedia5 libqt5multimedia5-plugins libqt5multimediawidgets5
  libqt5network5 libqt5opengl5 libqt5sprintsupport5 libqt5svg5 libqt5widgets5
  libsmi2ldbl libsnappy1v5 libspandsp2 libssh-gcrypt-4 libwireshark-data
  libwireshark11 libwiretap8 libwscodec2 libwsutil9 libxcb-xinerama0
  qt5-gtk-platformtheme qttranslations5-l10n wireshark-common wireshark-qt
Suggested packages:
  mmdns-bin qt5-image-formats-plugins qtwayland5 snmp-mibs-downloader
  wireshark-doc
The following NEW packages will be installed:
  libbc-ares2 libdouble-conversion1 liblua5.2-0 libmaxminddb0
  libnl-route-3-200 libqgsttools-p1 libqt5core5a libqt5dbus5 libqt5gui5
  libqt5multimedia5 libqt5multimedia5-plugins libqt5multimediawidgets5
  libqt5network5 libqt5opengl5 libqt5sprintsupport5 libqt5svg5 libqt5widgets5
  libsmi2ldbl libsnappy1v5 libspandsp2 libssh-gcrypt-4 libwireshark-data
  libwireshark11 libwiretap8 libwscodec2 libwsutil9 libxcb-xinerama0
  qt5-gtk-platformtheme qttranslations5-l10n wireshark wireshark-common
  wireshark-qt
0 upgraded, 32 newly installed, 0 to remove and 664 not upgraded.

```

Step3: Add user in wireshark by using usermod -aG wireshark \$(whoami) Reboot the system-sudo reboot


```

kavita@kavita-VirtualBox: ~
File Edit View Search Terminal Help
Processing triggers for shared-mime-info (1.9-2) ...
Setting up liblua5.2-0:amd64 (5.2.4-1.1build1) ...
Processing triggers for gnome-menus (3.13.3-1ubuntu1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Setting up libbc-ares2:amd64 (1.14.0-1ubuntu0.1) ...
Setting up libmaxminddb0:amd64 (1.3.1-1) ...
Setting up libqt5core5a:amd64 (5.9.5+dfsg-0ubuntu2.6) ...
Setting up libqt5dbus5:amd64 (5.9.5+dfsg-0ubuntu2.6) ...
Setting up libqt5network5:amd64 (5.9.5+dfsg-0ubuntu2.6) ...
Setting up libwiretap8:amd64 (2.6.10-1-ubuntu18.04.0) ...
Setting up libqt5gui5:amd64 (5.9.5+dfsg-0ubuntu2.6) ...
Setting up qt5-gtk-platformtheme:amd64 (5.9.5+dfsg-0ubuntu2.6) ...
Setting up libqt5widgets5:amd64 (5.9.5+dfsg-0ubuntu2.6) ...
Setting up libwireshark11:amd64 (2.6.10-1-ubuntu18.04.0) ...
Setting up libqt5sprintsupport5:amd64 (5.9.5+dfsg-0ubuntu2.6) ...
Setting up libqt5opengl5:amd64 (5.9.5+dfsg-0ubuntu2.6) ...
Setting up libqt5multimedia5:amd64 (5.9.5-0ubuntu1) ...
Setting up libqt5svg5:amd64 (5.9.5-0ubuntu1.1) ...
Setting up libqt5multimediawidgets5:amd64 (5.9.5-0ubuntu1) ...
Setting up libqgsttools-p1:amd64 (5.9.5-0ubuntu1) ...
Setting up wireshark-common (2.6.10-1-ubuntu18.04.0) ...
Setting up libqt5multimedia5-plugins:amd64 (5.9.5-0ubuntu1) ...
Setting up wireshark-qt (2.6.10-1-ubuntu18.04.0) ...
Setting up wireshark (2.6.10-1-ubuntu18.04.0) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
kavita@kavita-VirtualBox:~$ sudo usermod -aG wireshark $(whoami)
kavita@kavita-VirtualBox:~$ sudo reboot

```

Step4: To run Wireshark type the command (C) on terminal Wireshark



Step5: To run Wireshark using GUI go to application and then Wireshark



Practical No. 04

Aim: Program to simulate UDP server client

Procedure:

- i. Create a new project = "UDPserverclient"
- ii. Create Package = "UDP"
- iii. Create two files under package as "UDPserver.java" and "UDPclient.java"
- iv. Write the programs in respective file to send data from server to client.
- v. For transfer of a file, create Datagram socket in both with same number. (Socket numbers can be given from 1024 to 65532.
- vi. Also define read from beginning to end of file in both programs.
- ii. First run "UDPserver.java" then run "UDPclient.java".
- iii. Output will be displayed in console window.

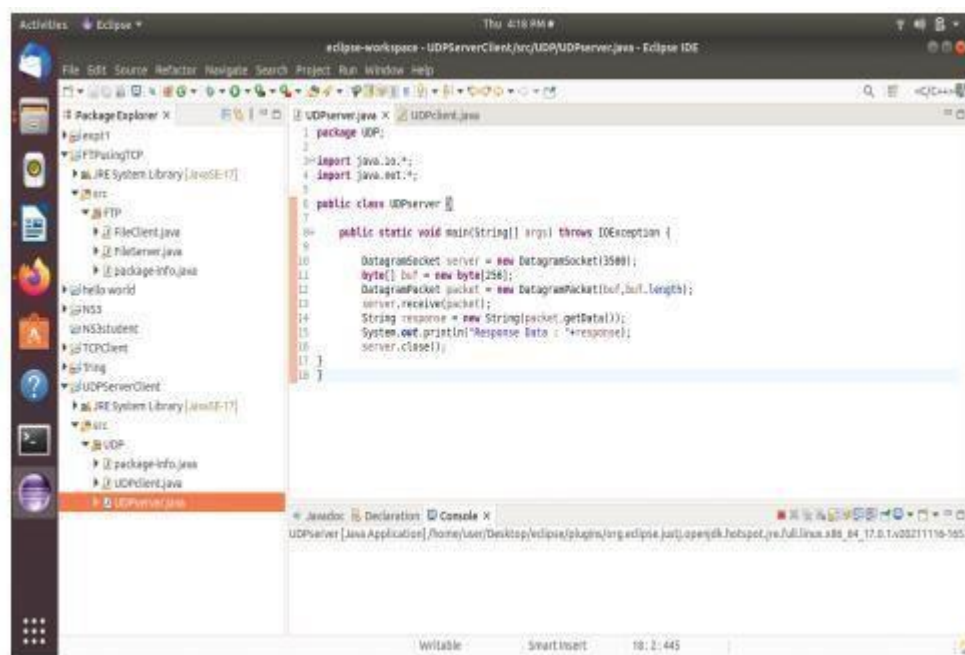
Program for UDP Server :

```
package UDP;

import java.io.*;
import java.net.*;

public class UDPserver {
    public static void main(String[] args) throws IOException {
        DatagramSocket server = new DatagramSocket(3500);
        byte[] buf = new byte[256];
        DatagramPacket packet = new
        DatagramPacket(buf,buf.length);
```

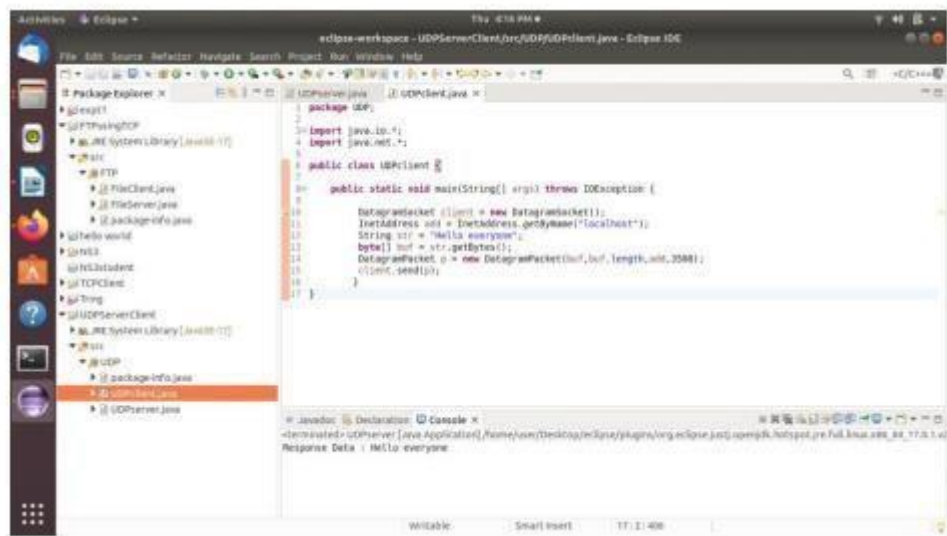
```
server.receive(packet);  
  
String response = new String(packet.getData());  
  
System.out.println("Response Data : "+response);  
  
server.close();  
  
}  
  
}
```



Program for UDP Client :

```
package UDP;  
  
import java.io.*;  
import java.net.*;  
  
public class UDPClient {  
  
    public static void main(String[] args) throws IOException {  
  
        DatagramSocket client = new DatagramSocket();  
  
        InetAddress add = InetAddress.getByName("localhost");  
  
        String str = "Hello everyone";
```

```
byte[] buf = str.getBytes();  
DatagramPacket p = new  
DatagramPacket(buf,buf.length,add,3500);  
client.send(p);  
}  
}
```



Practical No.05

Aim: Program to simulate FTP using TCP protocol

Procedure :

- i. Create a new project = "FTPusingTCP"
- ii. Create Package = "FTP"
- iii. Create two files under package as "FileServer.java" and "FileClient.java"
- iv. Write the programs in respective file to transfer a file from server to client.
- v. Give the path which file content need to transfer with new file name as destination.

Example = "Test.txt" to "cubic.txt"

- vi. For transfer of a file, create TCP socket in both with same number.

(Socket numbers can be given from 1024 to 65532.

- vii. Also define path, size of the file, read from beginning to end of file in both programs.

Vii. First run "FileServer.java" then run "FileClient.java".

Viii. Hereafter you can verify the file is transfered with new name succesfully by FTP using TCP protocol.

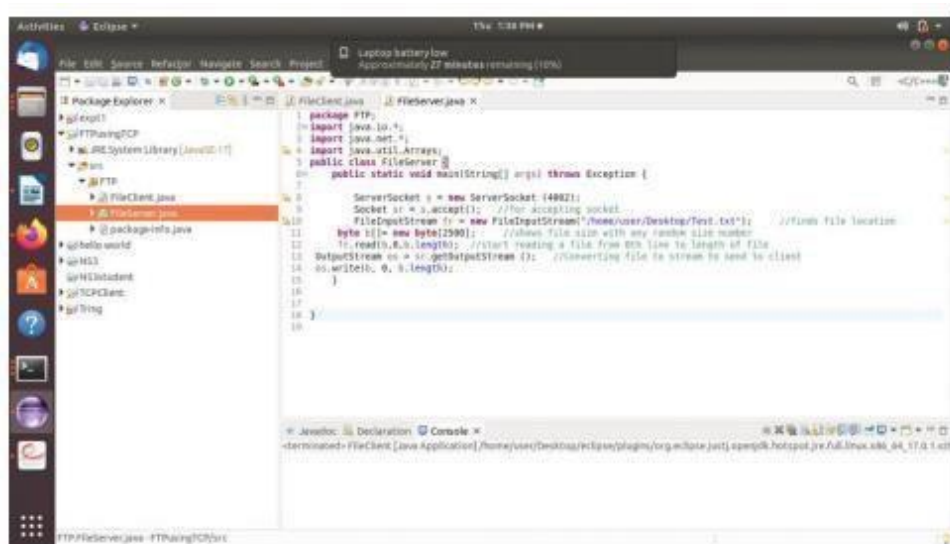
Program for TCP Server :

```
package FTP;

import java.io.*;
import java.net.*;
import java.util.Arrays;
```



```
public class FileServer {  
    public static void main(String[] args) throws Exception {  
        ServerSocket s = new ServerSocket (4002);  
        Socket sr = s.accept(); //for accepting socket  
        FileInputStream fr = new  
        FileInputStream("/home/user/Desktop/Test.txt"); //finds file location  
        byte b[]= new byte[2500]; //shows file size with any random size  
        number  
        fr.read(b,0,b.length); //start reading a file from 0th line to length of  
        file  
        OutputStream os = sr.getOutputStream (); //Converting file to stream to  
        send to client  
        os.write(b, 0, b.length);  
    }  
}
```



Program for Client :

```
package FTP;

import java.io.*;

import java.net.*;

import java.util.Arrays;

public class FileClient {

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

        byte []b= new byte[25004];

        Socket sr = new Socket("localhost", 4002);

        InputStream is=sr.getInputStream();

        FileOutputStream fr=new

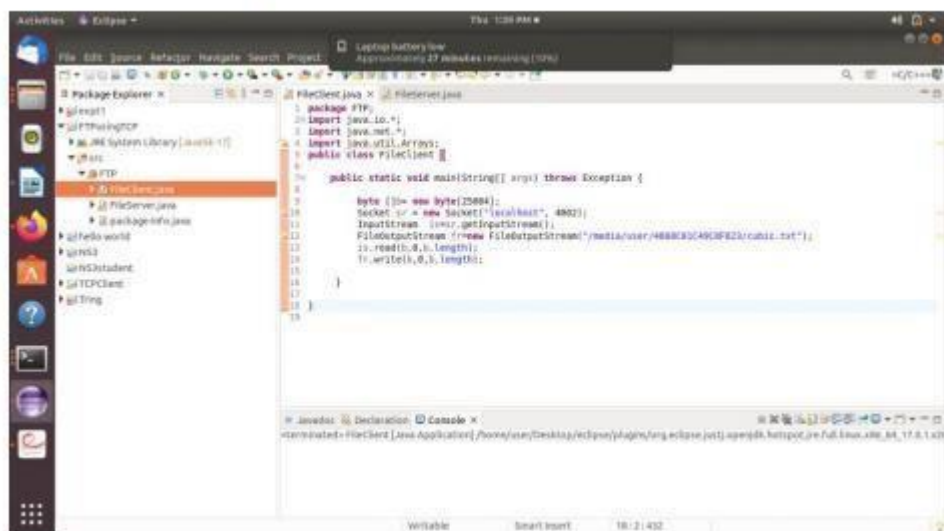
        FileOutputStream("/media/user/4668C01C49C8F823/cubic.txt");

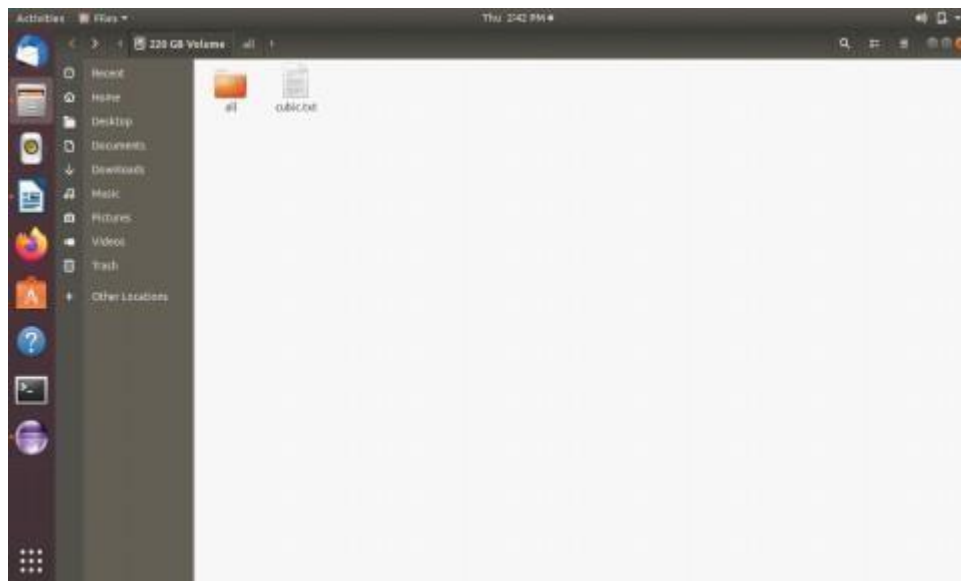
        is.read(b,0,b.length);

        fr.write(b,0,b.length);

    }

}
```





Practical No. 06

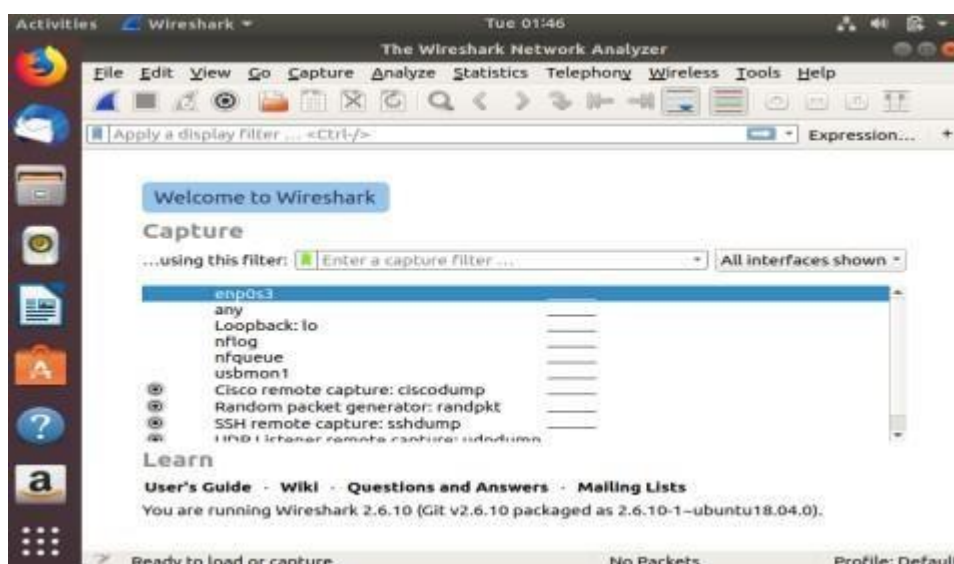
Aim: Analyze the network traffic using Wireshark

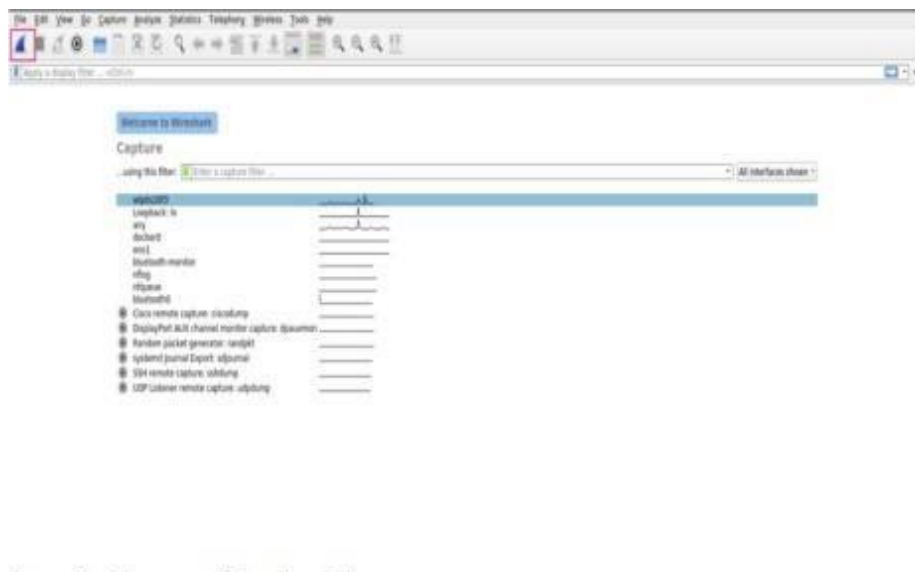
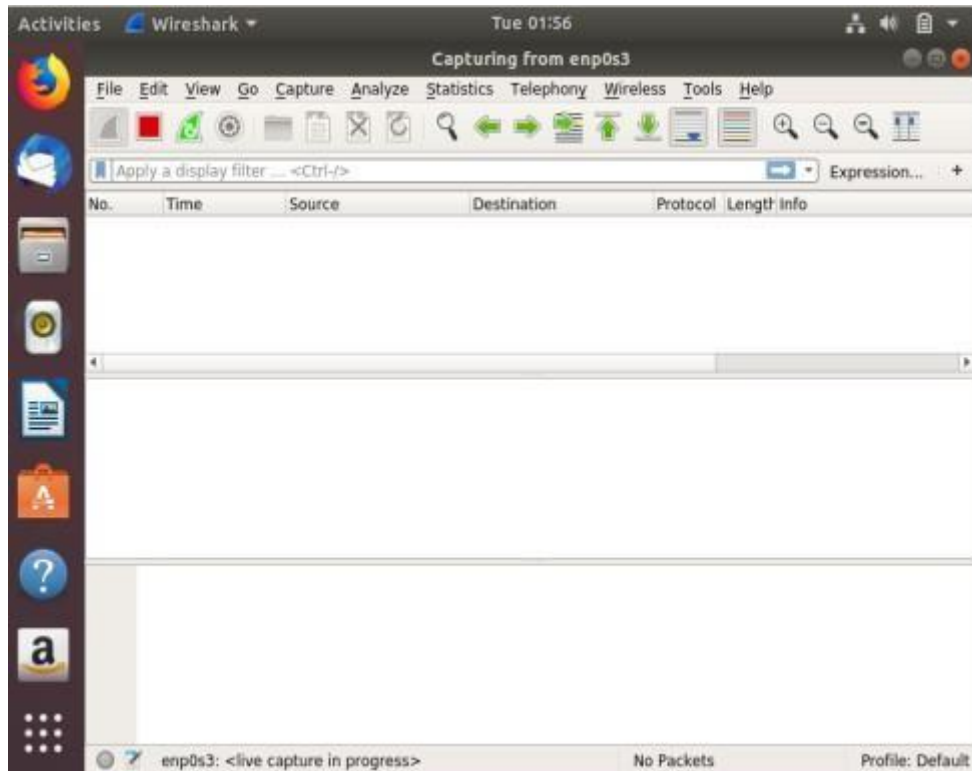
Wireshark is the world's foremost and widely-used network protocol analyzer. It lets us to know what's happening on our network at a microscopic level and is the de facto standard across many commercial and non-profit enterprises, government agencies, and educational institutions.

Network traffic analysis is the routine task of various job roles, such as network administrator, network defenders, incident responders and others. Capture filters with protocol header values Wireshark comes with several capture and display filters. But a user can create display filters using protocol header values as well. Use this technique to analyze traffic efficiently.

`proto[offset:size(optional)]=value` Following the above syntax, it is easy to create a dynamic capture filter, where:

- proto = desired protocol
- offset = header value
- size = data length
- value = data you want to find





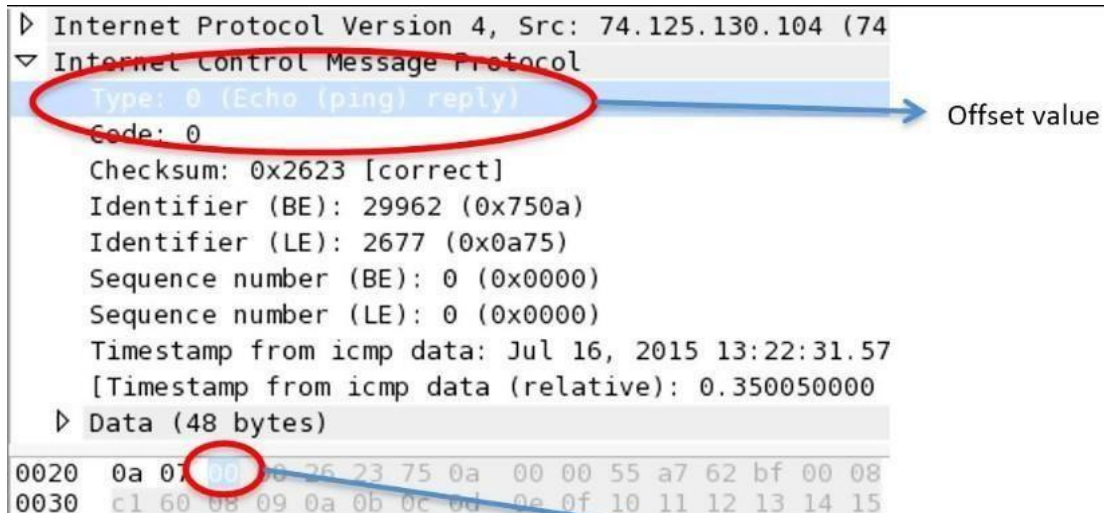


Figure 9 : ICMP reply

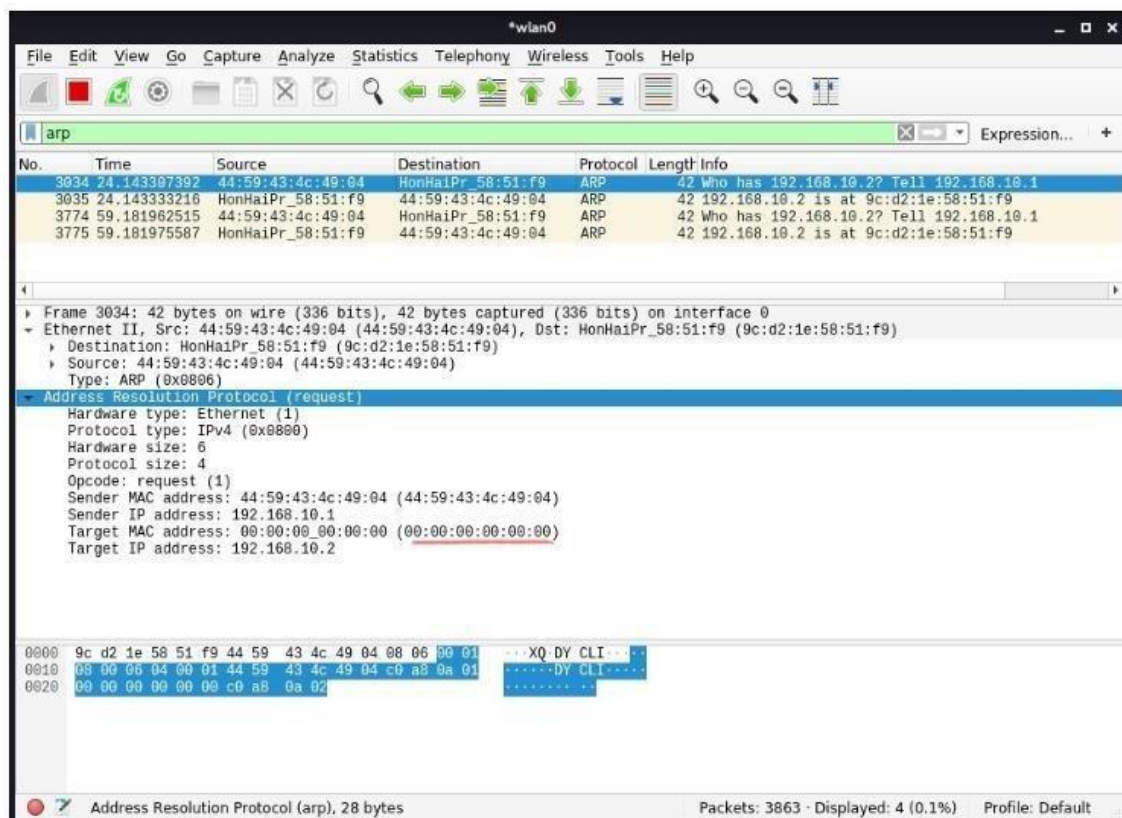
Position and Size

Analyze an ARP Request

Using the 'arp' filter, analyze the captured traffic in Wireshark.

Observe the packet request details from Ethernet and ARP; observe the source and destination IP and sender MAC and IP address.

Monitor the victim's MAC address. Since the destination MAC address is unavailable at the request packet stage, the victim's MAC address is zero, and the destination IP is the local system IP address.

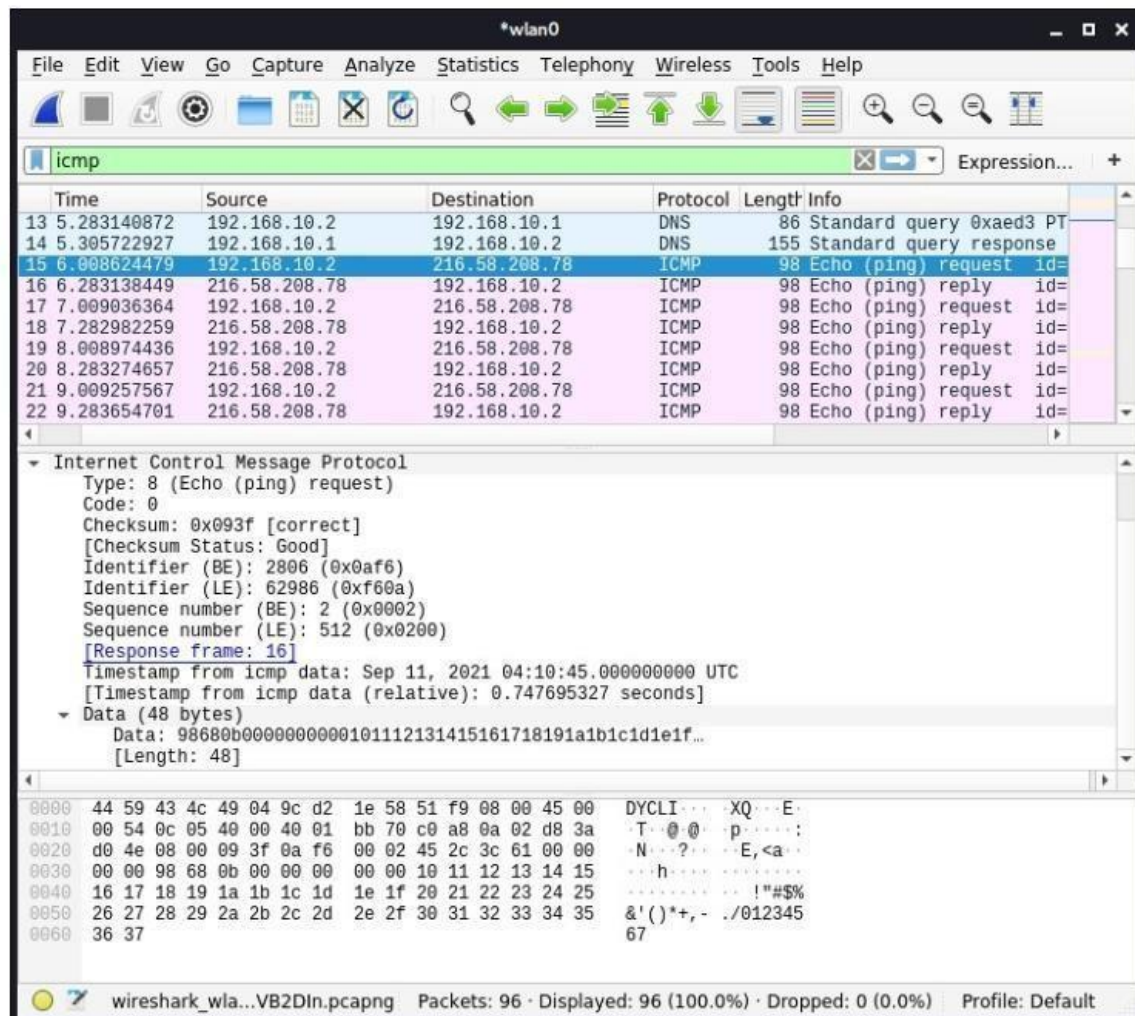


CMP traffic analysis

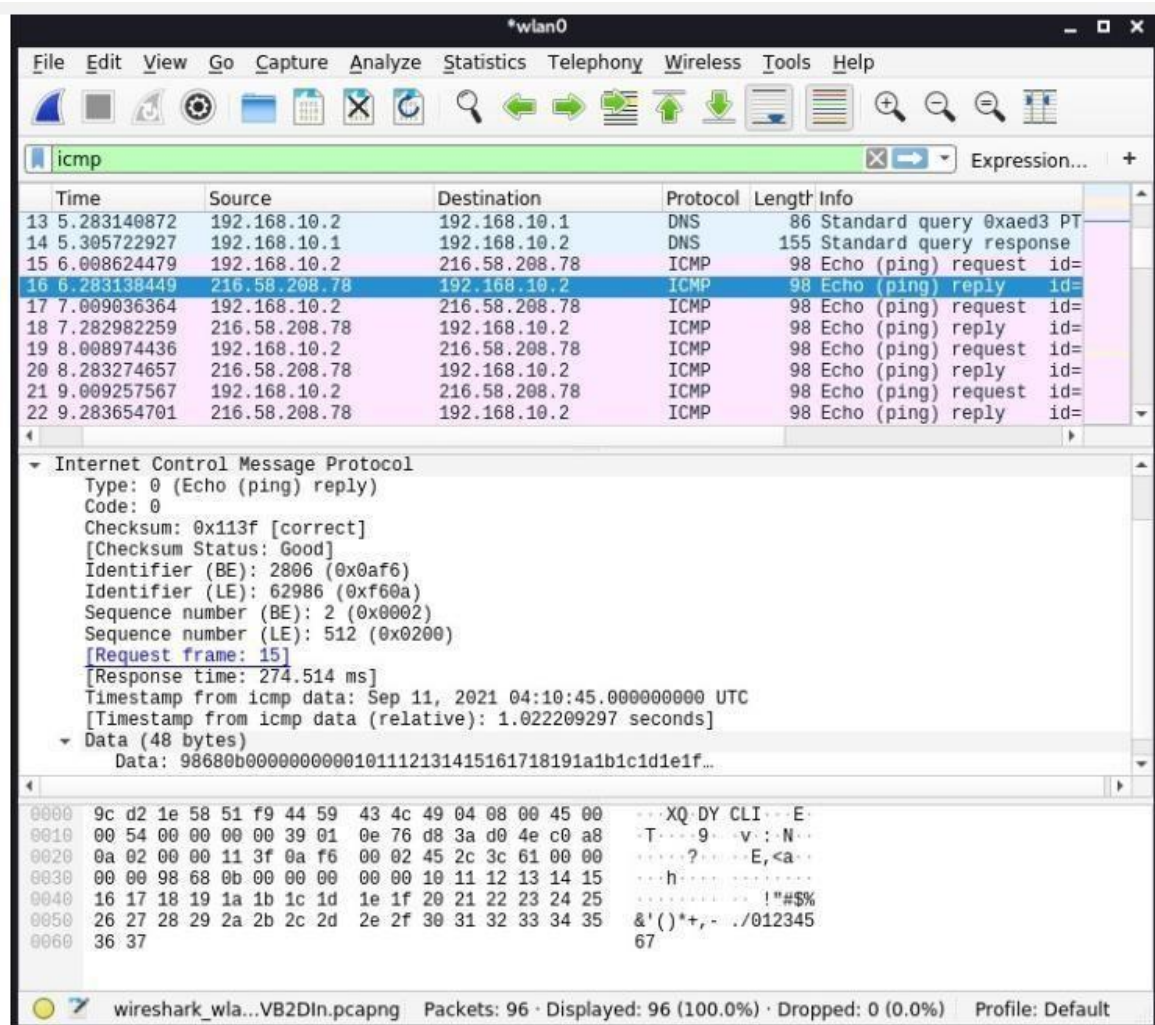
ICMP is used for error alerting and monitoring to verify whether data arrives in a timely basis at its desired destination.

To capture ICMP traffic, ping Google.com. Use the '*ICMP*' filter to see ICMP traffic. Click the ICMP echo-request packet from the Wireshark capture window and start observing the information.

In the **request packet**, the source IP is your (requestor) IP address. Whereas the destination IP is that of Google. You can also analyze the ICMP details like Checksum, Identifier Number, Sequence Number, etc.



In the **response packet**, observe the swapping of IPs between source and destination. You can also compare both request and response details, as they are similar.

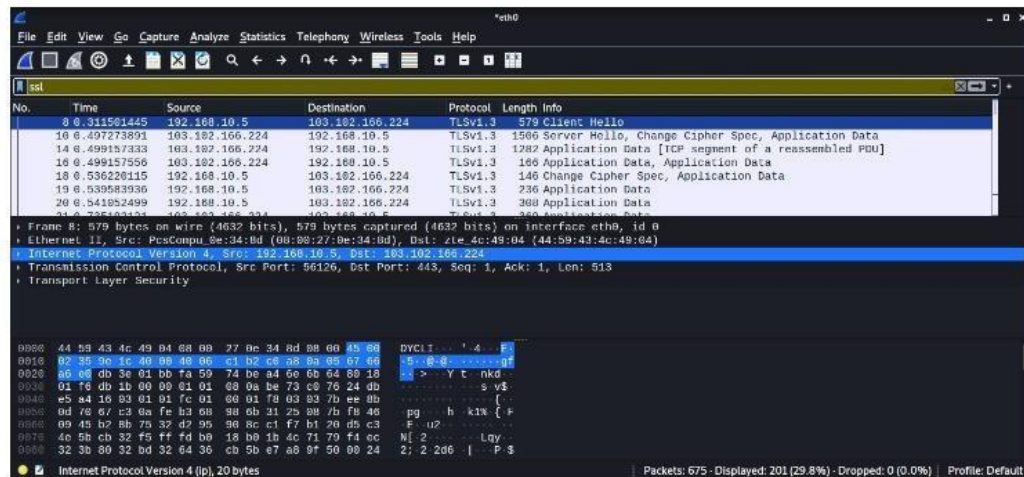


HTTPS traffic analysis

The Hypertext Transfer Application Layer Protocol (HTTP) utilizes the internet to establish protocols whenever the HTTP client/server transmits/receives HTTP requests.

Start a Wireshark capture -> Open a web browser -> Navigate to any HTTPS-based website -> Stop the Wireshark capture.

Input 'ssl' in the filter box to monitor only HTTPS traffic -> Observe the first TLS packet -> The destination IP would be the target IP (server).



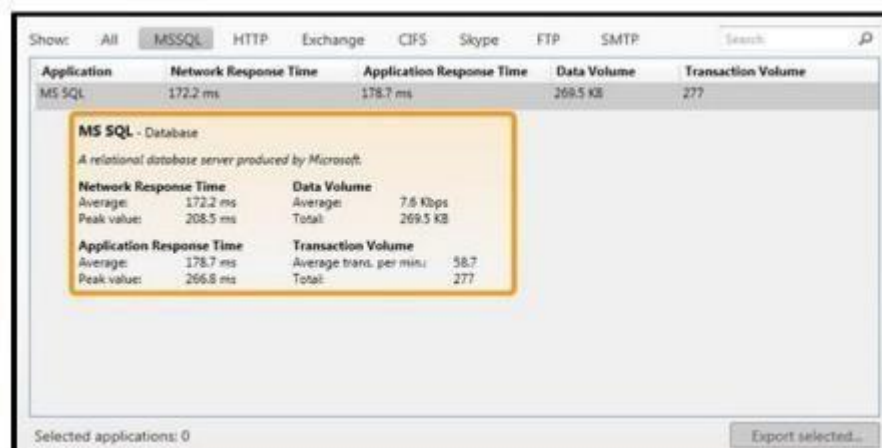
Practical No.07

Aim : Analyze the performance parameters of network using Wire Shark

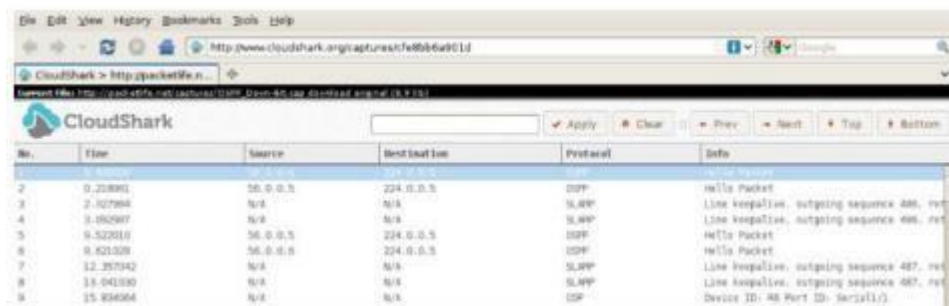
SolarWinds Network Performance Monitor



SolarWinds Response Time Viewer for Wireshark



Cloudshark is a platform designed to display network capture files directly in the browser without the need for desktop applications or tools. Simply upload, email, or link the captures files and get the results. It helps to resolve network issues faster and flawlessly. Besides, it includes features such as drag-and-drop capture files, drop-box-like activity, allows sharing of links with co-workers, and offers advanced analysis.



No.	Time	Source	Destination	Protocol	Info
2	0.218901	56.0.0.5	224.0.0.5	IGMP	Hello Packet
3	2.107994	N/A	N/A	SLAPP	Line keepalive, outgoing sequence 486, ret
4	3.490947	N/A	N/A	SLAPP	Line keepalive, outgoing sequence 486, ret
5	9.522010	56.0.0.5	224.0.0.5	IGMP	Hello Packet
6	9.421309	56.0.0.5	224.0.0.5	IGMP	Hello Packet
7	12.257942	N/A	N/A	SLAPP	Line keepalive, outgoing sequence 487, ret
8	13.041000	N/A	N/A	SLAPP	Line keepalive, outgoing sequence 487, ret
9	15.834364	N/A	N/A	IGMP	Device ID: 48 Port ID: Serial1/0

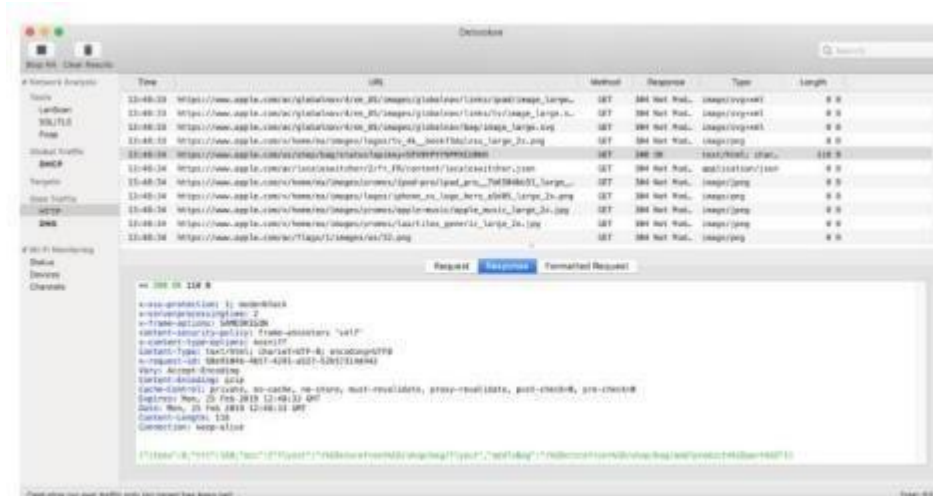
Sysdig Sysdig is a powerful, cross-platform, and open-source flexible network monitoring system designed specifically for Linux. It works with others but with limited functionalities. Sysdig uses various monitoring and troubleshooting tools for Linux system performance, such as:

- Strace (for discovering system calls)
- htop (for real-time process monitoring)
- iftop (for real-time bandwidth monitoring)
- netstat (for network connection monitoring)
- tcpdump (for raw network traffic monitoring)
- lsof (to know the process used to view the opened files)



Debooke Debooke is one of the simplest and most powerful network monitoring and traffic analyzer tools for macOS. It allows users to intercept and monitor the network traffic of any device in the same subnet. It helps to capture data network packet data from mobile, printer, Mac, TV. Features of Debooke:

- Keep track of Wi-Fi bandwidth use and consumption
- Help with network monitoring and in-depth analysis
- Display the Wi-Fi clients and the APIs to which they're associated
- Scan IP, LAN to keep track of all the active and connected devices



Advanced Statistical Tools The captured traffic is then analyzed using Wireshark basic statistical tools and advanced tools for various performance parameters.

Graphs

Wireshark offers graphing capabilities which can present captured packets in an interesting format

that makes the analysis process much more effective and easy to adapt [14] [15]. There are multiple

types of graphs available

