Experiment No. 7

Aim: Implement ring topology in NS2 simulator

Experiment No: 7

Experiment Title: Implementation of Ring topology

Subject: CNE

Roll No.: 18IT1020

Name of Student: Bhavesh B. Shinde

Batch/Div: A/A2

Date of Performance:

Date of Submission:

Grade:

Signature:

DY PATIL
TECHNOLOGY

Experiment No. 7

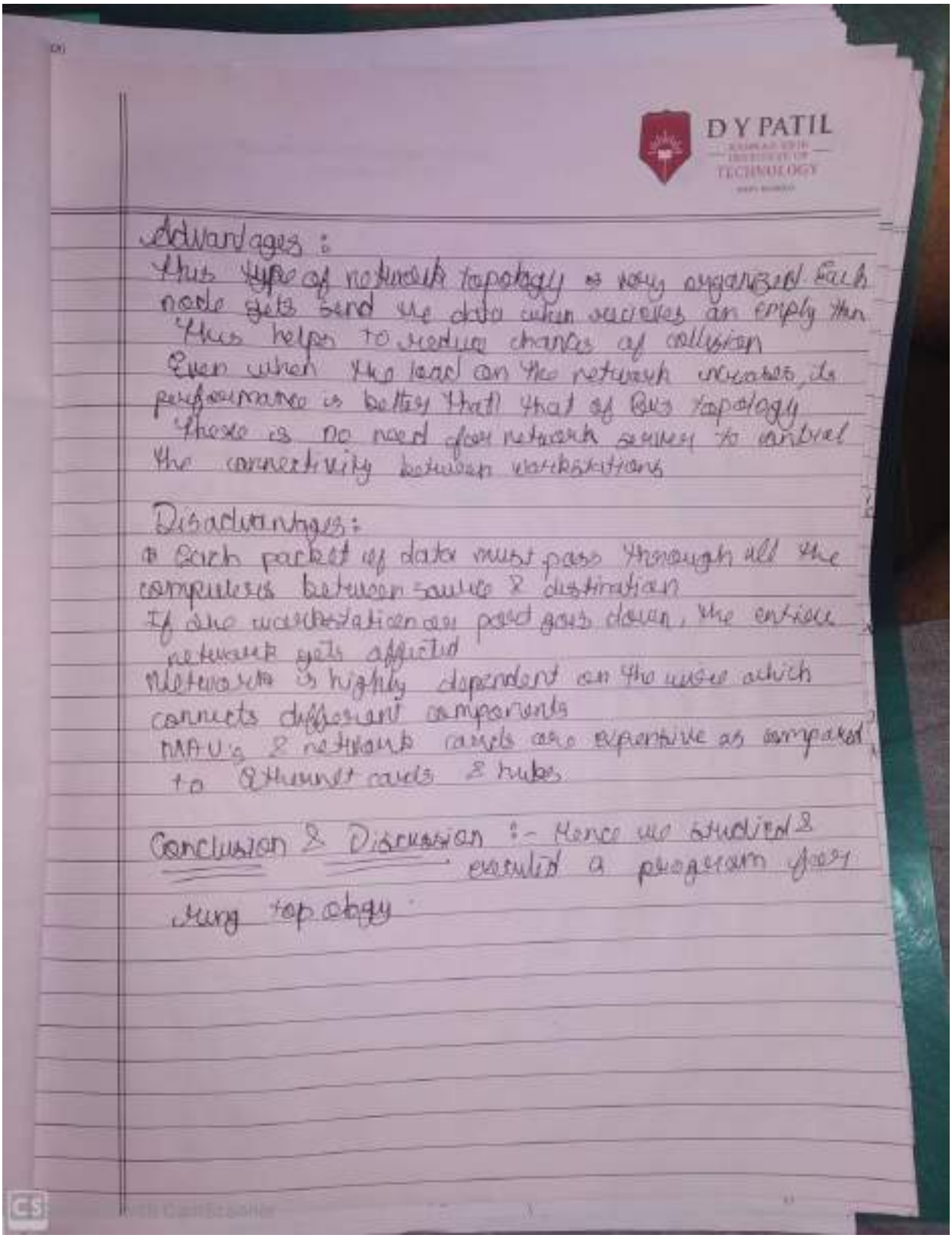
Aim : Implement ring topology in NS2 simulator

Software Required : - Ubuntu, NS2.34

Theory :- In local area networks where the ring topology is used each computer is connected to the network in 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 ring in one direction. By utilizing the scheme, only one machine can transmit on the network at a time. Data travels around the network in one direction. Sending and receiving of data takes place by the help of TOKEN

Token passing 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 the way with data to next node. The 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 usually found in offices, schools & small buildings.


D Y PATIL
TECHNOLOGY
Advantages :
This type of network topology is very organized. Each node gets send the data when receives an empty then
Thus helps to reduce chances of collision
Even when the load on the network increases, its performance is better than that of Bus topology
There is no need for network servers to control the connectivity between workstations
Disadvantages :
Each packet of data must pass through all the computers between source & destination
If one workstation or port goes down, the entire network gets affected
Network is highly dependent on the wire which connects different components
NIAU's & network cards are expensive as compared to Ethernet cards & hubs
Conclusion & Discussion :- Hence we studied & executed a program for ring topology.

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 &
exit0 }
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#Create links between the nodes
 $ns duplex-link $n0 $n1 1Mb 10ms DropTail
 $ns duplex-link $n1 $n2 1Mb 10ms DropTail
 $ns duplex-link $n2 $n3 1Mb 10ms DropTail
 $ns duplex-link $n3 $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
set tcp1 [new Agent/TCP]
$tcp1 set class_ 1
$ns attach-agent $n2 $tcp1
set tcp2 [new Agent/TCP]
$tcp2 set class_ 1
$ns attach-agent $n3 $tcp2
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink] $ns attach-agent $n1 $sink0
set sink1 [new Agent/TCPSink] $ns attach-agent $n2 $sink1
set sink2 [new Agent/TCPSink] $ns attach-agent $n3 $sink2
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
# 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
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500 $cbr1
set interval_ 0.01
```

```
$cbr1 attach-agent $tcp1

set cbr2 [new Application/Traffic/CBR]

 $cbr2 set packetSize_ 500 $cbr2

set interval_ 0.01

$cbr2 attach-agent $tcp2

#Schedule events for the CBR agents

$ns at 0.5 "$cbr0 start" $ns at 4.5 "$cbr0 stop"

$ns at 0.5 "$cbr1 start" $ns at 4.5 "$cbr1 stop"

$ns at 0.5 "$cbr2 start" $ns at 4.5 "$cbr2 stop"

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

#Run the simulation $ns run
```

Experiment No. 8

Aim: Study and analysis of routing protocol and Shortes path routing by DSDV

Experiment No: 8

Experiment Study and analysis of routing protocols

Subject: CNE

Roll No.: 18IT1020

Name of Student: Bhavesh B. Shinde

Batch/Div: A/A2

Date of Performance:

Date of Submission:

Grade:

Signature:

DY PATIL
INSTITUTE OF TECHNOLOGY

Experiment No. 8

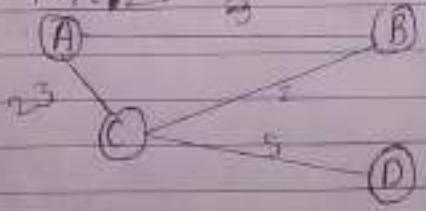
Aim :- Study & analyze routing protocol 2. Shortest Path routing by DSDV

Software Required : Ubuntu, NS 2.35

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.

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

A distance routing protocol requires that a router inform its neighbors of topology changes periodically & in some cases, when a change is detected in topology of a network. Compared to link state protocols which require a router to inform all nodes in a network of topology changes. In this network we have 4 routers A, B, C & D



We shall mark the current time in the algorithm with T & shall begin by creating distance matrices for each router to its immediate with neighbors. As we build the routing tables, the shortest path is highlighted with the color green & new shortest path is highlighted

At this point all routers have new (ie) their DV. They each broadcast this new DV. They each broadcast this new DV to all neighbours, they now recalculate the shortest path

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

Again all the routers have gained in the last iteration at (T=1) new shortest path, so they all broadcast their DVs to their neighbours this prompts each neighbour to recalculate 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 of 7. Since the current shortest path to B is shorter than the existing "shortest path" to D of length 28 (via C) so it becomes the new "shortest path" to D

At this time only routers A & D have new shortest paths from DVs. So they broadcast their new DVs to the neighbours. A broadcasts to B and C & D broadcasts to C. This causes each of the neighbours receiving the new DVs to re-calculate their shortest paths. However, since the information DVs doesn't yield any shorter paths than what they already have, there are no changes to routing tables

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 stops

Unicast Routing Configuration

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 10 Mb 10ms DropTail

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

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

Conclusion & Discussion: DSDV routing algorithms are analysis & studied to find out the shortest path in network.

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]
```

Experiment No. 9

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

D Y PATIL
RAMRAO ADIK INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

Experiment No: 9

Experiment Title: Analysis of network performance by configuring Queue size

Subject: CNE

Roll No.: 18IT1020

Name of Student: Bhavesh B. Shinde

Batch/Div: A/A2

Date of Performance:

Date of Submission:

Grade:

Signature:

DY PATIL

Experiment No. 01

Aim : Analysis of network performance by configuring the queue size & capacity of links for measuring QoS of generated traffic

Software Required :- Ubuntu, NS 2.34

Theory :- Definition of network of links & nodes
The way to define a node is
set n0 [$ ns node]
The node is created which is pointed by the variable n0. When we shall refer to the node in the script we shall use $n0.
$ ns duplex-link $n0 $n1 10Mb 10ms DropTail
which means that n0 & n1 are connected using a bi-directional link that has 10ms of propagation delay & a capacity of 10Mb per sec.
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. Using RED (Random Early Discard) mechanism we have to FQ (Fair Queuing) the DAR (Deficit Round Robin) the stochastic Fair Queuing (SFQ) & the CBQ.
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.

DY PATIL
INSTITUTE OF TECHNOLOGY

set Queue Size of link (n0-n2) to 20

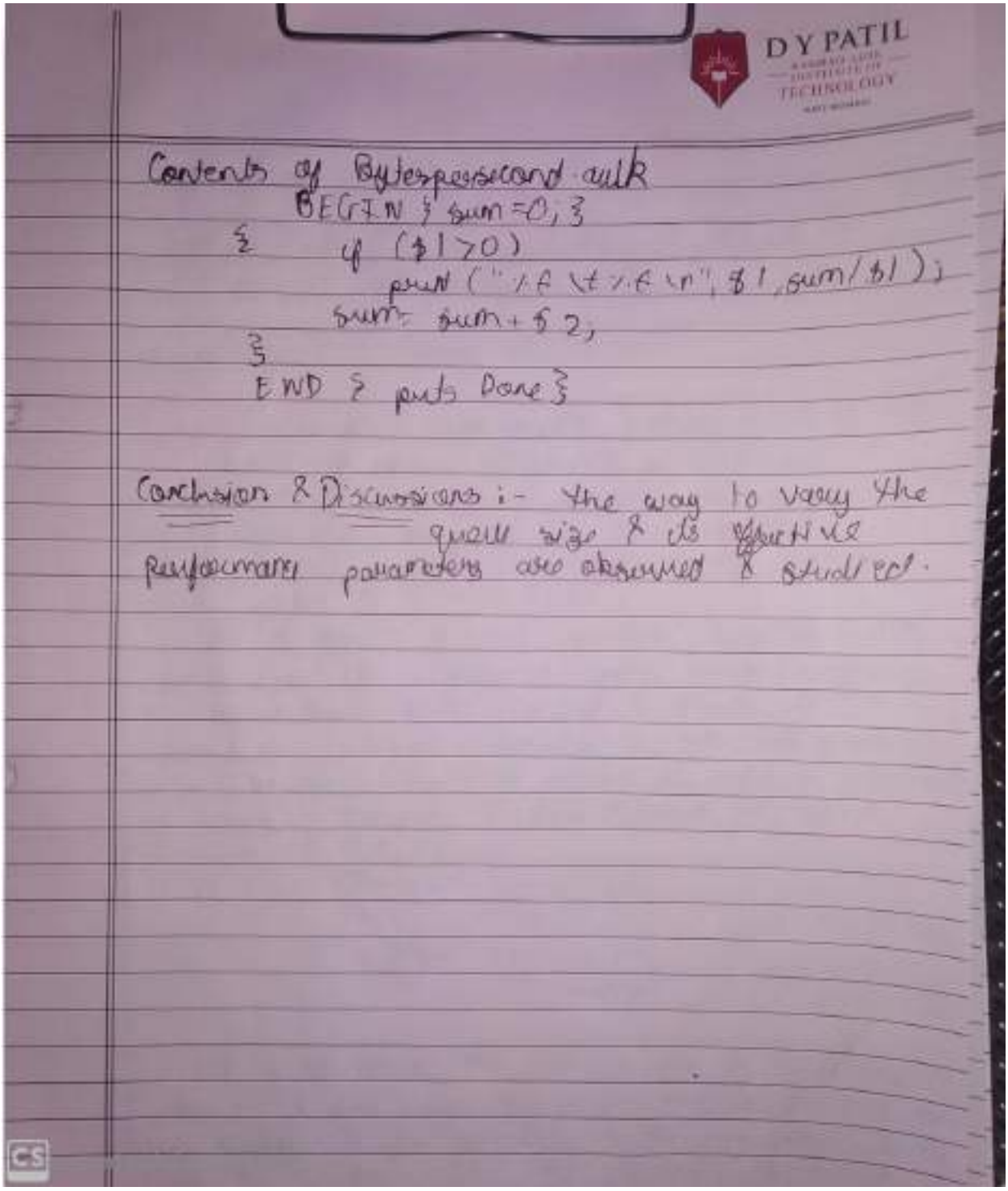
Structure of Trace Files

When tracing into an output ASCII file the trace is organized in 12 field as follows the meaning of fields are

Event Time From TO PKT PKT Flags Fid Src Dest Seq PKT
Node Node Type Size Addr Addr Num id

1. The first field is event type. It is given by r/d/+ possible with which corresponds respectively to receive enqueued & dropped
2. The second field gives the time at which event occurs
3. Gives input of link at which event occurs
4. Gives output of link at which event occurs
5. Gives packet type
6. Gives packet size
7. Some flags
10. This is the source address given in the same form
11. This is the destination address given in the same form
12. This is the network layer protocol's packet sequence number. Even though implementations in ns do not keep track of not use sequence numbers
13. The last field shows the unique id of the packet

Awk- Advantages: awk is a programmable pattern matching & processing tool available in UNIX. It works equally well with text & numbers



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

#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 trace cwnd_

$tcp trace rtt_

$ns at 625.0 "finish"

$ns ru
```

Experiment No. 10

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

Experiment No: 10

Experiment Title: Comparative QoS parameters using Xgraph for Different load conditions

Subject: CNE

Roll No.: 18IT1020

Name of Student: Bhavesh B. Shinde

Batch/Div: A/A2

Date of Performance:

Date of Submission:

Grade:

Signature:

DY PATIL INSTITUTE OF TECHNOLOGY

Experiment No. 10

Aim :- Comparing QoS parameter using Xgraph / gnuplot for different load conditions

Software Required :- Ubuntu, NS 2.34, Xgraph

Theory :- XGRAPH

The xgraph draws a graph on x-display given data and gives either click file or from standard input if no files are specified. It can display upto 64 independent data sets using different colors & line styles for each sets, grid lines & tick marks, grid labels & a legend.

Xgraph [options] file-name

options are listed here

/- bd <color> (Border)
This specifies border color of window

/- bg <color> (Background)
This specifies Background color of window

/- x <unit name> (X unit Text)
This the unit name for x-axis. Its default is "X"

/- y <unit name> (Y unit Text)
This is the unit name for y-axis. Its default is "Y"

/-t <string> (Title Text)
This string is entered at top of the graph

Analysis 1:

Simple AWK awk

Contents:

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

awk -f Simple_AWK.awk out.tr > Temp

Bytespersecond.awk

```
BEGIN {sum = 0;}
{
  if ($1>0)
  printf "%f %f \n", $1, sum/$1);
  sum= sum + $7;
}
END {puts Done}
```

awk -f Bytespersecond.awk Temp > bps.tr1

graph 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 terminal postscript
# set output "tcp.ps"
```

gnuplot - persist simple.gnu

Analysis 2:

awk awk

(contents of awk awk

(if ($6 == "awk") print $1, $7)

awk -f awk.awk awk.tr > awk

awk.gnu

(contents of awk.gnu

- extend.gt

. awk.gnu

set xrange [0:200]

set yrange [0:40]

set xlabel 'Time (sec)'

set ylabel 'awk (pkts)'

plot 'awk' title 'tcp' with lines

gnuplot - persist awk.gnu

Analysis 3

- thard script analysis.awk

BEGIN

{

seqno = -1;

droppedPackets = 0;

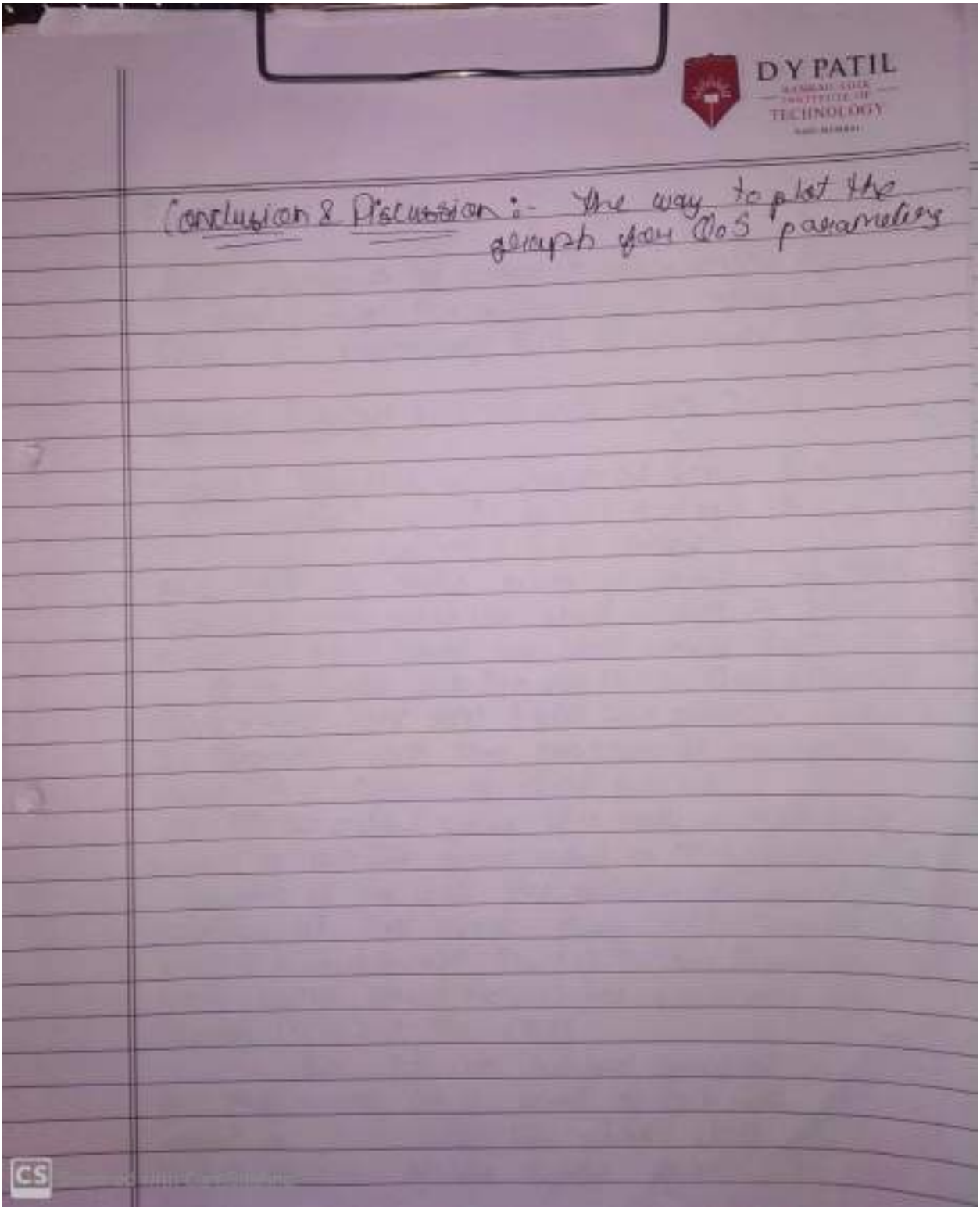
receivedPackets = 0

tcps = 0;

tcpA = 0;

tcpd = 0;

udps = 0;



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:

```
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 couting total no of send (tcp,udp and ack)
if (($1 == "+") && (seqno < $12))
{
seqno = $12;
}
#For couting 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 couting 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
80
Department of Information Technology
{
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";

}
```

Experiment No. 11

Aim: **Installation of Wire shark and Analysis of Packets**

Experiment No: 11

Experiment Title: Installation of wire sharks and analysis of packets

Subject: CNE

Roll No.: 18IT1020

Name of Student: Bhavesh B. Shinde

Batch/Div: A/A2

Date of Performance:

Date of Submission:

Grade:

Signature:

D Y PATIL
INSTITUTE OF TECHNOLOGY

Experiment No. 11

Aim: Installation of Wireshark & analysis of Packets

Hardware/Software Required: Wireshark, Ethereal & Tcpdump

Theory: Wireshark a network analysis tool formerly known as Ethereal, captures in real time & displays them in human readable format.
It includes filter, color-coding & other sorting features that let you dig deep into network & inspect packets

Applications:
- Network administrators use it to trouble shoot network problems
- Network security engineers use it to examine problems
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals besides these examples can be helpful in other situations too

Features:
The following are some of many features provided
- Available for UNIX & Windows
- Capture live packet data from a network interface
- Export some or all packets in a number of capture file formats
- Colorize packet display based on filters
- Create various statistics

DY PATIL INSTITUTE OF TECHNOLOGY

The most basic way to apply a filter is by typing it into the filter box at the top of the window & clicking Apply. For e.g. type "dns" & you'll see only DNS packets. When you start typing, Wireshark will help you add autocomplete your filter.

Another interesting thing you can do is right-click a packet & select follow TCP stream.

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

Close the window & 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 & you can dig down to view its details.

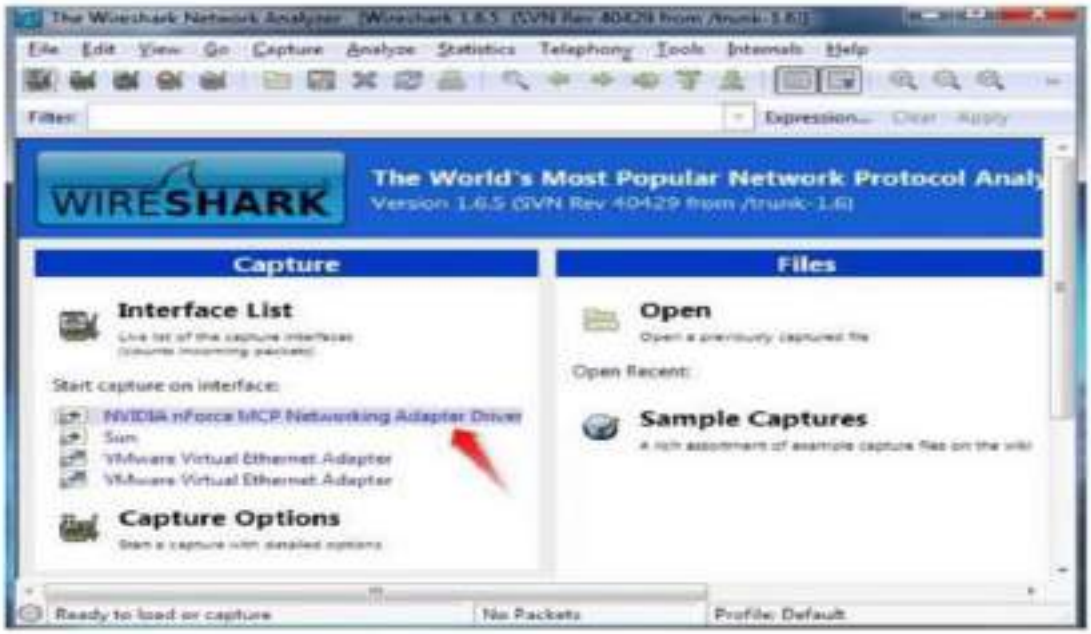
You can also create filters from here: just right-click one of the details & use the apply as Filter subtree to create a filter based on it.

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

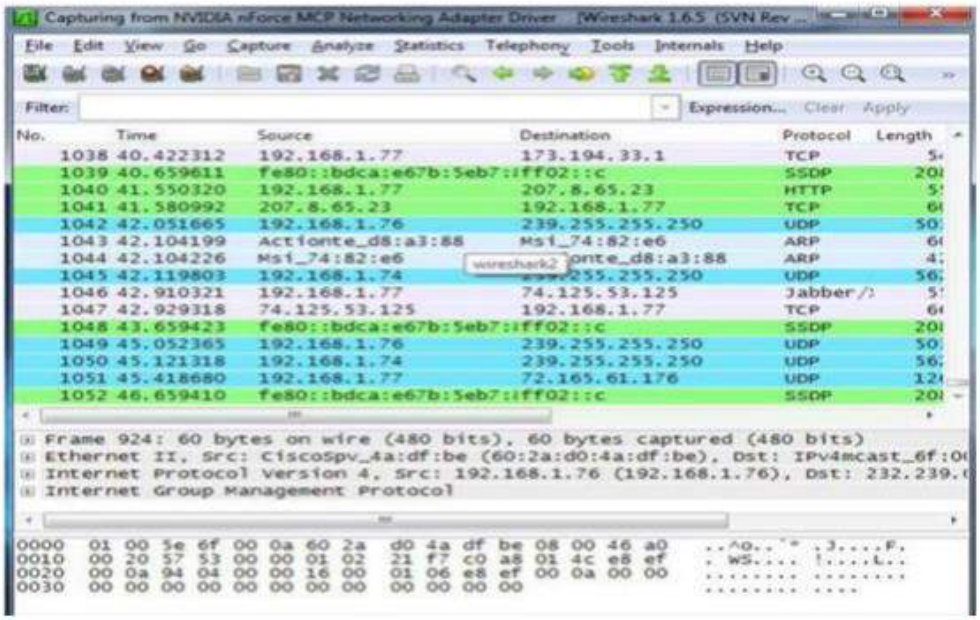
DY PATIL
TECHNOLOGY

Conclusion: In this experiment we analyze packet sniffing tool that monitors network traffic transmitted between legitimate users on the network. The packet sniffer is network monitoring tool. It is called for network monitoring, traffic analysis, troubleshooting. Packet sniffing, message, protocol analysis, penetration testing & many others purposes

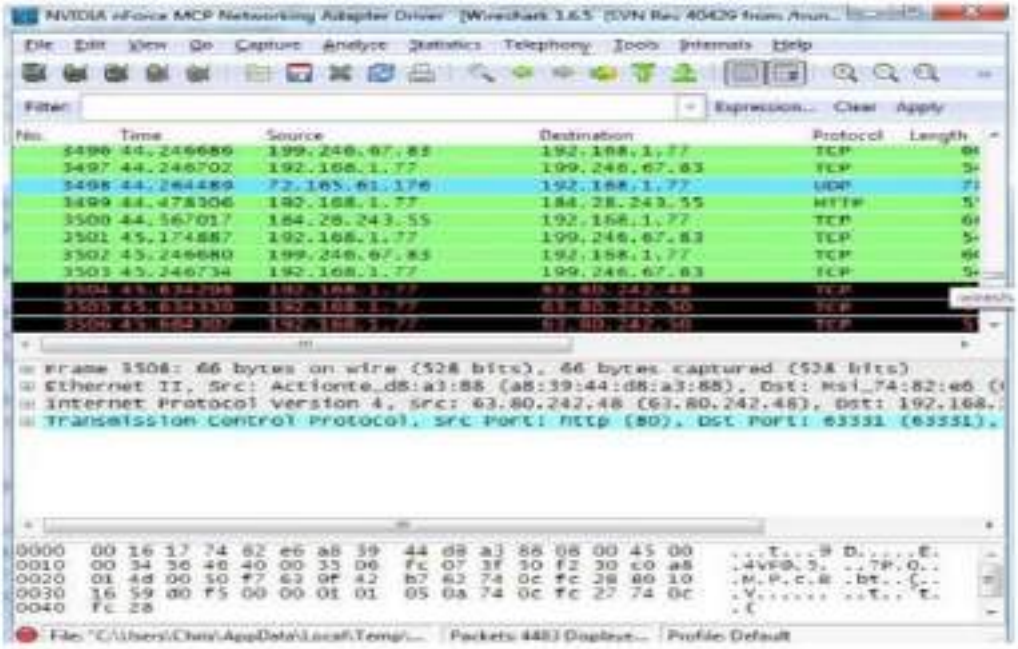
1.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply
WIRESHARK
The World's Most Popular Network Protocol Analy
Version 1.6.5 (SVN Rev 40429 from /trunk-1.6)
Capture
Files
Interface List
Start capture on interface:
NVIDIA nForce MCP Networking Adapter Driver
Sun
VMware Virtual Ethernet Adapter
VMware Virtual Ethernet Adapter
Capture Options
Open
Open Recent:
Sample Captures
Ready to load or capture
No Packets
Profile: Default

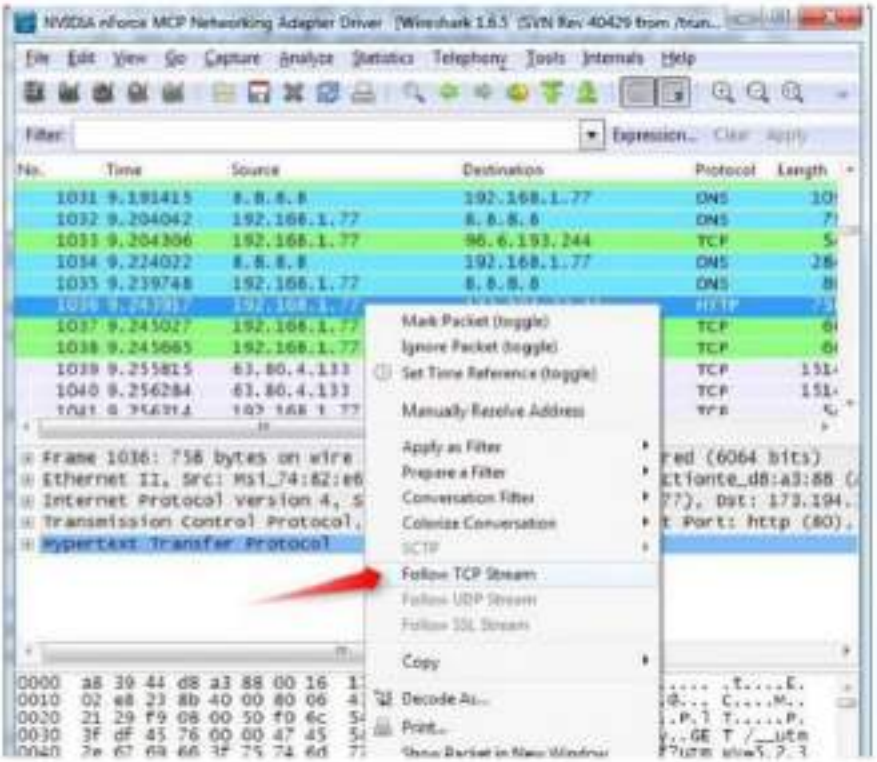
2.


Capturing from NVIDIA nForce MCP Networking Adapter Driver [Wireshark 1.6.5 (SVN Rev
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Length
1038 40.422312 192.168.1.77 173.194.33.1 TCP
1039 40.659611 fe80::bdca:e67b:5eb7:ff02::c SSDP
1040 41.550320 192.168.1.77 207.8.65.23 HTTP
1041 41.580992 207.8.65.23 192.168.1.77 TCP
1042 42.051665 192.168.1.76 239.255.255.250 UDP
1043 42.104199 Actionte_d8:a3:88 Msi_74:82:e6 ARP
1044 42.104226 Msi_74:82:e6 ARP
1045 42.119803 192.168.1.74 UDP
1046 42.910321 192.168.1.77 74.125.53.125 Jabber/
1047 42.929318 74.125.53.125 192.168.1.77 TCP
1048 43.659423 fe80::bdca:e67b:5eb7:ff02::c SSDP
1049 45.052365 192.168.1.76 239.255.255.250 UDP
1050 45.121318 192.168.1.74 239.255.255.250 UDP
1051 45.418680 192.168.1.77 72.165.61.176 UDP
1052 46.659410 fe80::bdca:e67b:5eb7:ff02::c SSDP
Frame 924: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
Ethernet II, Src: CiscoSpv_4a:df:be (60:2a:d0:4a:df:be), Dst: IPv4mcast_6f:0
Internet Protocol Version 4, Src: 192.168.1.76 (192.168.1.76), Dst: 232.239.
Internet Group Management Protocol
0000 01 00 5e 6f 00 0a 60 2a d0 4a df be 08 00 46 a0
0010 00 20 57 53 00 00 01 02 21 f7 c0 a8 01 4c e8 ef
0020 00 0a 94 04 00 00 16 00 01 06 e8 ef 00 0a 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00

3.

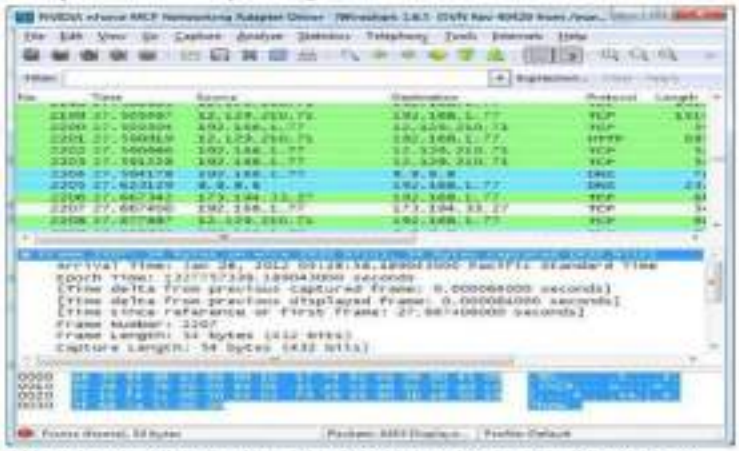

NVIDIA nForce MCP Networking Adapter Driver [Wireshark 1.6.5 (SVN Rev 40429 from /trun...
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Length
3496 44.246686 199.246.67.83 192.168.1.77 TCP
3497 44.246702 192.168.1.77 199.246.67.83 TCP
3498 44.264489 72.165.61.176 192.168.1.77 UDP
3499 44.478306 192.168.1.77 184.28.243.55 HTTP
3500 44.567017 184.28.243.55 192.168.1.77 TCP
3501 45.174887 192.168.1.77 199.246.67.83 TCP
3502 45.246680 199.246.67.83 192.168.1.77 TCP
3503 45.246734 192.168.1.77 199.246.67.83 TCP
Frame 3504: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: Actionte_d8:a3:88 (a8:39:44:d8:a3:88), Dst: Msi_74:82:e6
Internet Protocol Version 4, Src: 63.80.242.48 (63.80.242.48), Dst: 192.168.
Transmission Control Protocol, Src Port: http (80), Dst Port: 63331 (63331),
0000 00 16 17 74 62 e6 a8 39 44 d8 a3 88 08 00 45 00
0010 00 34 56 46 40 00 35 06 fc 07 3f 50 f2 30 c0 a8
0020 01 4d 00 50 f7 63 9f 42 b7 62 74 0c fc 28 80 10
0030 16 59 d0 f5 00 00 01 01 05 0a 74 0c fc 27 74 0c
0040 fc 28
Packets: 4483 Displaye... Profile: Default

4.


NVIDIA nForce MCP Networking Adapter Driver [Wireshark 1.6.5 (SVN Rev 40429 from /trun...
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Length
1031 9.191415 8.8.8.8 192.168.1.77 DNS
1032 9.204042 192.168.1.77 8.8.8.8 DNS
1033 9.204306 192.168.1.77 96.6.193.244 TCP
1034 9.224022 8.8.8.8 192.168.1.77 DNS
1035 9.239748 192.168.1.77 8.8.8.8 DNS
1036 9.241307 192.168.1.77 HTTP
1037 9.245027 192.168.1.77 TCP
1038 9.245065 192.168.1.77 TCP
1039 9.255815 63.80.4.133 TCP
1040 9.256284 63.80.4.133 TCP
Frame 1036: 758 bytes on wire
Ethernet II, Src: Msi_74:82:e6
Internet Protocol Version 4, S
Transmission Control Protocol,
Hypertext Transfer Protocol
Mark Packet (toggle)
Ignore Packet (toggle)
Set Time Reference (toggle)
Manually Resolve Address
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
SCTP
Follow TCP Stream
Follow UDP Stream
Follow SSL Stream
Copy
Decode As...
Print...
0000 a8 39 44 d8 a3 88 00 16
0010 02 e8 23 8b 40 00 80 06
0020 21 29 f9 08 00 50 f0 6c
0030 3f df 45 76 00 00 47 45
0040 2e 67 69 66 3f 75 74 6d

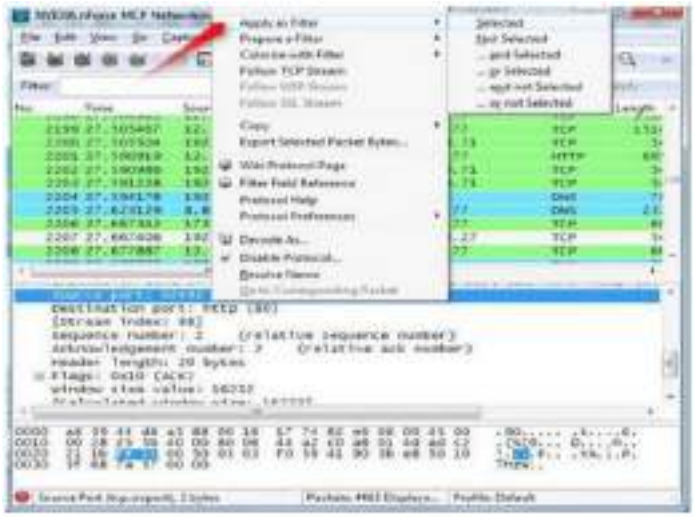
5.

6.



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

7.



Experiment No. 12

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.

Experiment No: 12

Experiment Title: Socket programming with client server model

Subject: CNE

Roll No.: 18IT1020

Name of Student: Bhavesh B. Shinde

Batch/Div: A/A2

Date of Performance:

Date of Submission:

Grade:

Signature:

D Y PATIL

Experiment No. 12

Aim : Using TCP/IP Sockets, write a client server program to make client sending the file name & the server to send back the contents of the requested file if present.

Software Required : Ubuntu, NS 2.34

Theory : Socket Characteristics

Sockets are characteristics

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

AF_UNIX : address format is UNIX pathname

AF_INET : address format is host & port number

Each Socket type has one or more protocol 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

Socket API's

socket : creates a socket of a given domain, type, protocol (buy a phone)

bind : assigns a name to the socket

accept : server accepts a connection with specific client

send, recv : stream-based equivalents of read & write

shutdown : end reading or writing

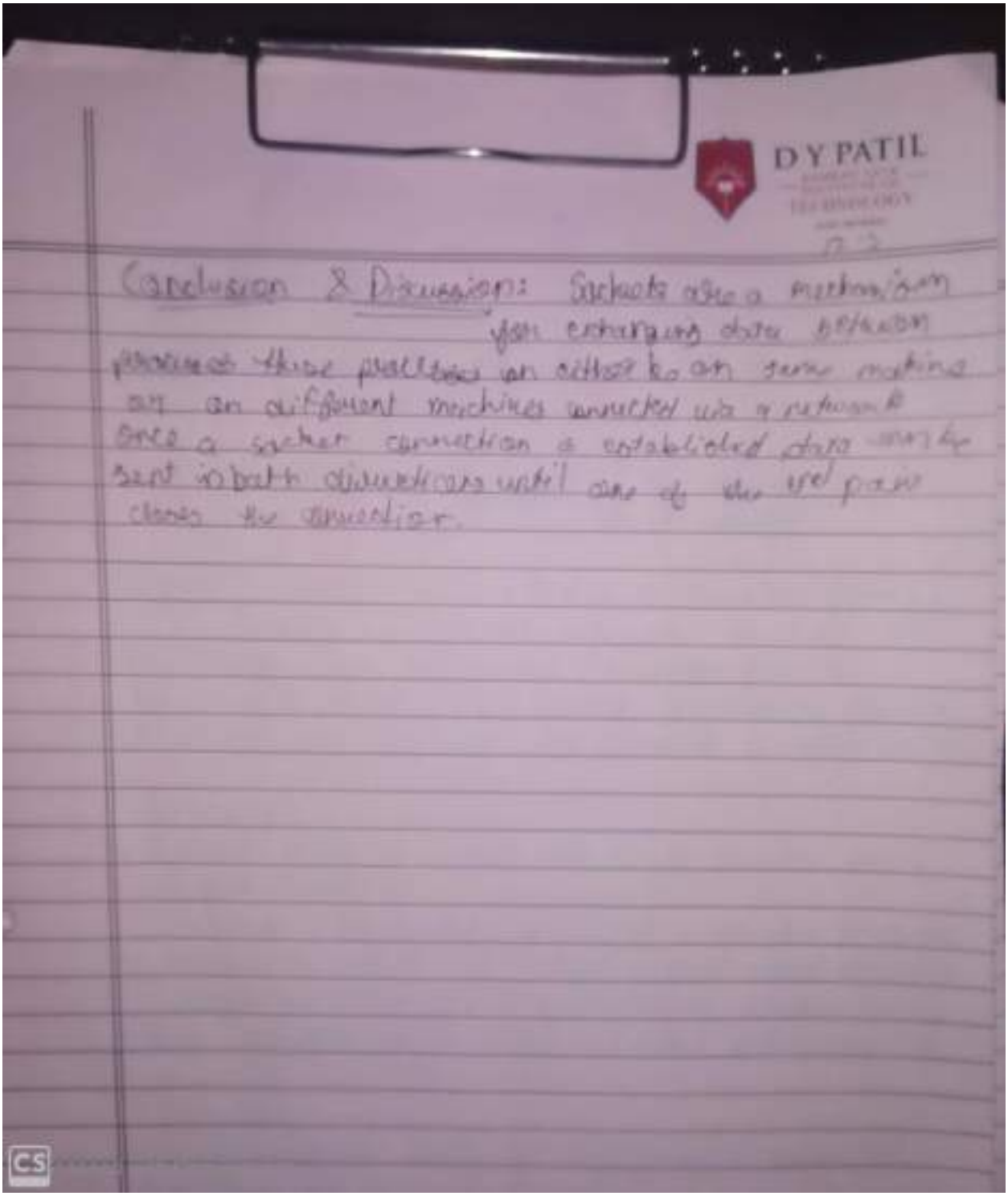
Client performs the following actions

socket : create the socket

send
#include <sys/types.h>
#include <sys/socket.h>
int send (int sid, const char *buffer_ptr, int len)
Send a message. Returns the no. of bytes sent
or -1 if failure
- 0 : default
- MSG_OOB : Out of band high priority communication

recv
#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). Flags can be either
- 0 : default
- MSG_OOB : Out of band message without
- 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.



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)

} }
```