

Lab1: NS-3 Simulation

Task 1.1

The data rate in between the client and the server was set to 2 Mbps, with a channel transmission delay of 10 ms. The client sent maximum 10 packets to the server, and wait one second between the packets. Data in outbound packets to set 1024 bytes.

Using given command line I was able to run and simulate the first.cc file which was the requirement of the this task.

`.\waf --run scratch/first > task-1-1.txt`

Content of task-1-1.txt

```
'build' finished successfully (26.827s)
Sent 1024 bytes to 10.1.1.2
Received 1024 bytes from 10.1.1.1
Received 1024 bytes from 10.1.1.2
```

Task 1.2

Data rate decreased to 0.5 Mbps , re-run the simulation and saved the output to 'task- 1-2.txt' using following command line.

`.\waf --run scratch/first > task-1-2.txt`

Content of task-1-2.txt

```
"build" finished successfully (1m15.063s)
Sent 1024 bytes to 10.1.1.2
Received 1024 bytes from 10.1.1.1
Received 1024 bytes from 10.1.1.2
```

Task 1.3

.pcap files were generated by adding following line of code to the simulated file.

```
pointToPoint.EnablePcapAll ("task-1-1");
pointToPoint.EnablePcapAll ("task-1-2");
```

Observation:

1-1-0.pcap

Between first and last two packets 9.028sec

average packets per second 2.215

1-1-1.pcap

Between first and last two packets 9.000sec

average packets per second 2.222

1-2-0.pcap

Between first and last two packets 9.054sec

average packets per second 2.209

1-2-1.pcap

Between first and last two packets 9.000sec

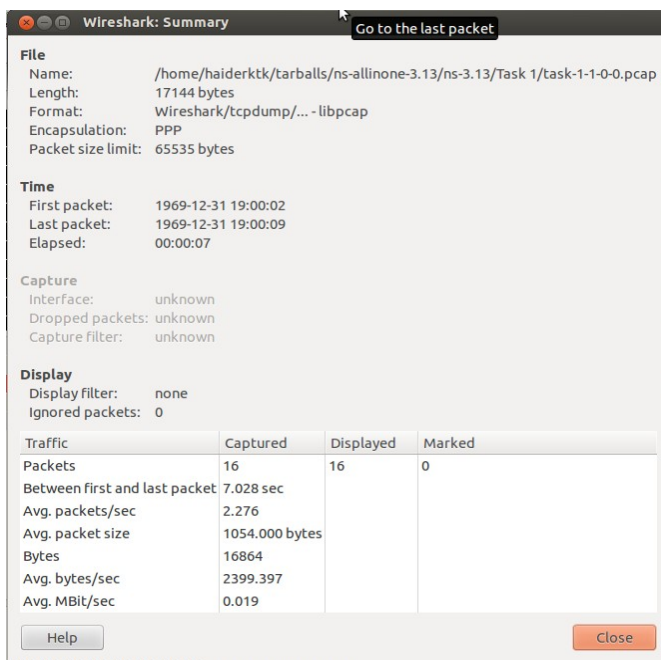
average packets per second 2.222

Reflection: With reducing the data-rate the time for file transfer increases from 9.028sec to 9.054sec.

following are the statistical summary of different communication links.

For data rate of 2 Mbps

Task-1-1-0-0 and Task-2-1-0-0



Wireshark: Summary

Go to the last packet

File
Name: /home/haiderkkt/tarballs/ns-allinone-3.13/ns-3.13/Task 1/task-1-1-0-0.pcap
Length: 17144 bytes
Format: Wireshark/tcpdump/... - libpcap
Encapsulation: PPP
Packet size limit: 65535 bytes

Time
First packet: 1969-12-31 19:00:02
Last packet: 1969-12-31 19:00:09
Elapsed: 00:00:07

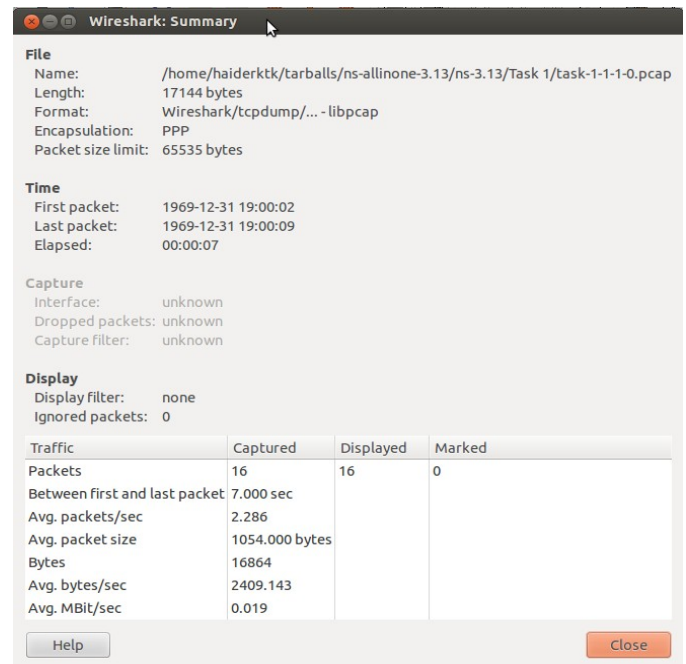
Capture
Interface: unknown
Dropped packets: unknown
Capture filter: unknown

Display
Display filter: none
Ignored packets: 0

Traffic	Captured	Displayed	Marked
Packets	16	16	0
Between first and last packet	7.028 sec		
Avg. packets/sec	2.276		
Avg. packet size	1054.000 bytes		
Bytes	16864		
Avg. bytes/sec	2399.397		
Avg. MBit/sec	0.019		

Help Close

Task-1-1-1-0 & Task-2-1-1-0



Wireshark: Summary

File
Name: /home/haiderkkt/tarballs/ns-allinone-3.13/ns-3.13/Task 1/task-1-1-1-0.pcap
Length: 17144 bytes
Format: Wireshark/tcpdump/... - libpcap
Encapsulation: PPP
Packet size limit: 65535 bytes

Time
First packet: 1969-12-31 19:00:02
Last packet: 1969-12-31 19:00:09
Elapsed: 00:00:07

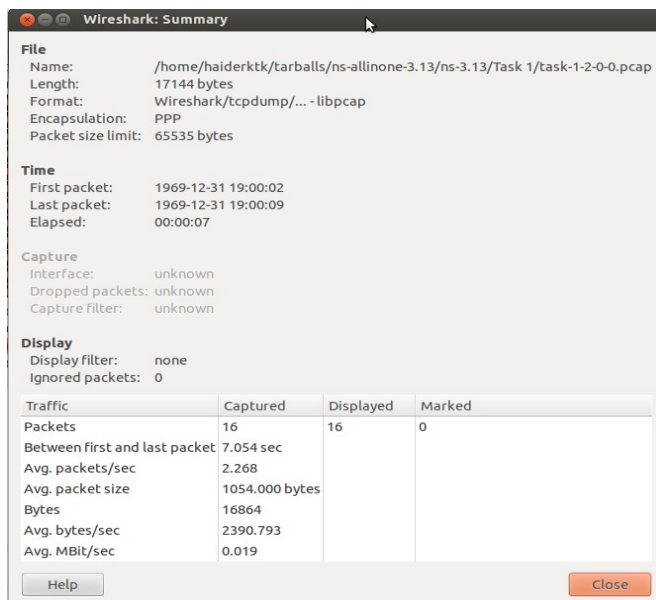
Capture
Interface: unknown
Dropped packets: unknown
Capture filter: unknown

Display
Display filter: none
Ignored packets: 0

Traffic	Captured	Displayed	Marked
Packets	16	16	0
Between first and last packet	7.000 sec		
Avg. packets/sec	2.286		
Avg. packet size	1054.000 bytes		
Bytes	16864		
Avg. bytes/sec	2409.143		
Avg. MBit/sec	0.019		

Help Close

For data rate to 0.5 Mbps



Wireshark: Summary

File
Name: /home/haiderkkt/tarballs/ns-allinone-3.13/ns-3.13/Task 1/task-1-2-0-0.pcap
Length: 17144 bytes
Format: Wireshark/tcpdump/... - libpcap
Encapsulation: PPP
Packet size limit: 65535 bytes

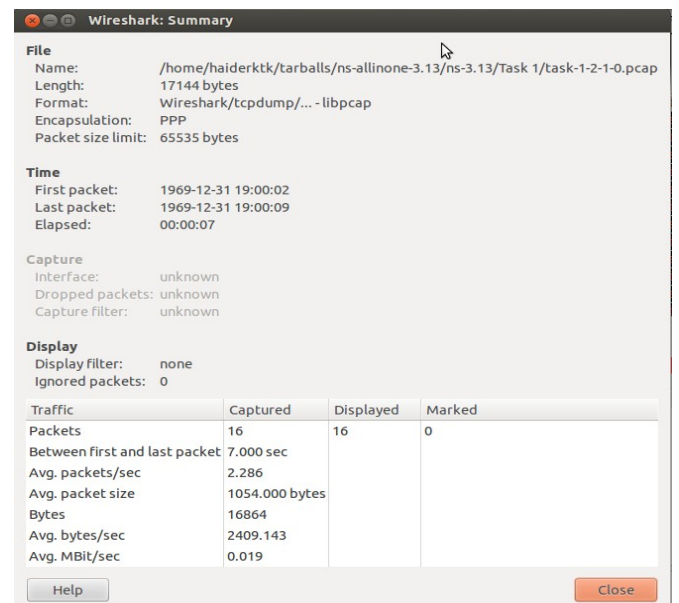
Time
First packet: 1969-12-31 19:00:02
Last packet: 1969-12-31 19:00:09
Elapsed: 00:00:07

Capture
Interface: unknown
Dropped packets: unknown
Capture filter: unknown

Display
Display filter: none
Ignored packets: 0

Traffic	Captured	Displayed	Marked
Packets	16	16	0
Between first and last packet	7.054 sec		
Avg. packets/sec	2.268		
Avg. packet size	1054.000 bytes		
Bytes	16864		
Avg. bytes/sec	2390.793		
Avg. MBit/sec	0.019		

Help Close



Wireshark: Summary

File
Name: /home/haiderkkt/tarballs/ns-allinone-3.13/ns-3.13/Task 1/task-1-2-1-0.pcap
Length: 17144 bytes
Format: Wireshark/tcpdump/... - libpcap
Encapsulation: PPP
Packet size limit: 65535 bytes

Time
First packet: 1969-12-31 19:00:02
Last packet: 1969-12-31 19:00:09
Elapsed: 00:00:07

Capture
Interface: unknown
Dropped packets: unknown
Capture filter: unknown

Display
Display filter: none
Ignored packets: 0

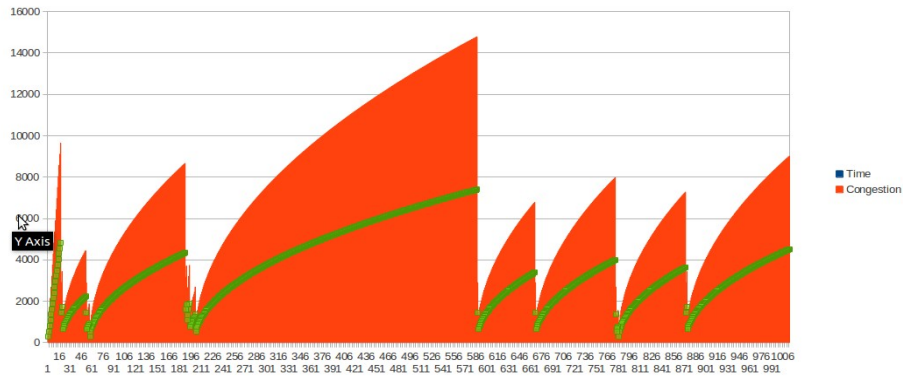
Traffic	Captured	Displayed	Marked
Packets	16	16	0
Between first and last packet	7.000 sec		
Avg. packets/sec	2.286		
Avg. packet size	1054.000 bytes		
Bytes	16864		
Avg. bytes/sec	2409.143		
Avg. MBit/sec	0.019		

Help Close

Task 2.1

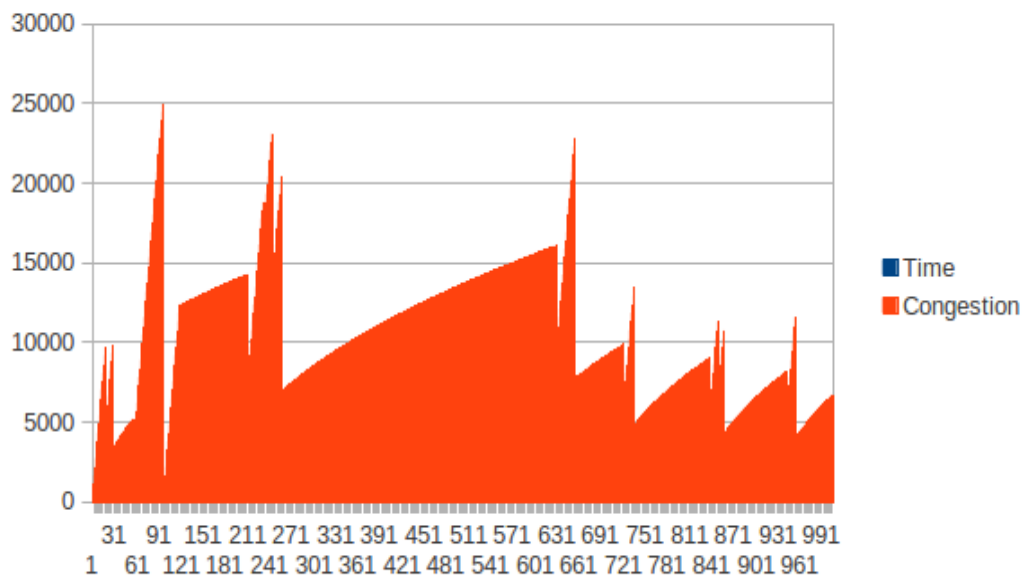
With data rate of communication link set to 5 Mbps plot is give below between time and congestion window.

Following graph shows the effect of data on congestion window.



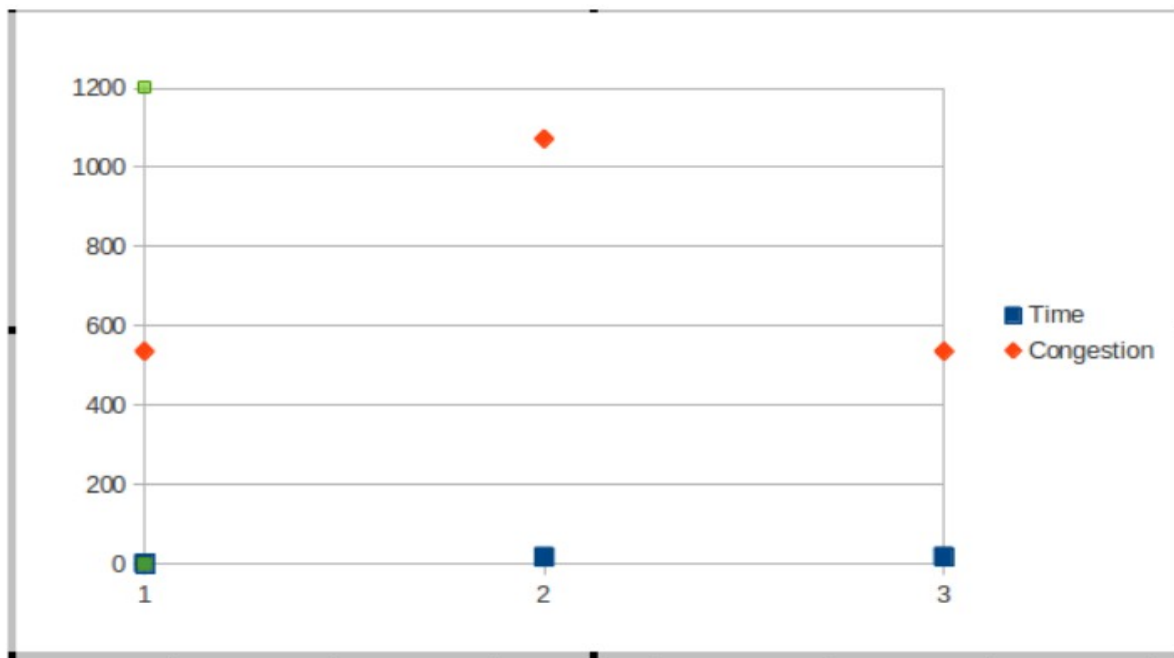
Task 2.2:

With a reduced data rate: 1 Mbps we get the following plot of time and congestion.



Task 2.3

with an increased error rate of 0.01

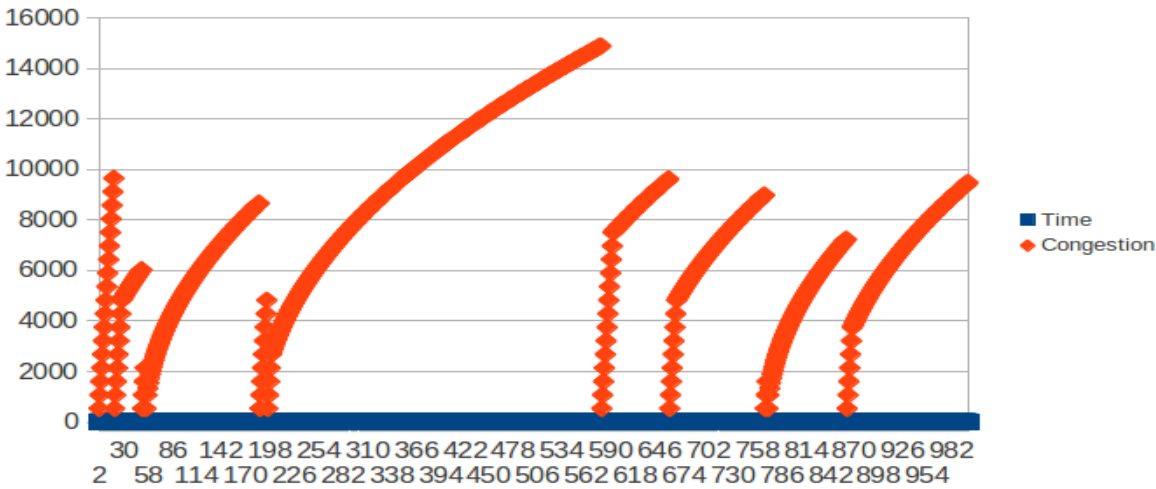


Task 2.4 :

Date Rate = 1 Mbps

Error rate = 0.00001

TCP Model = TCP Tahoe

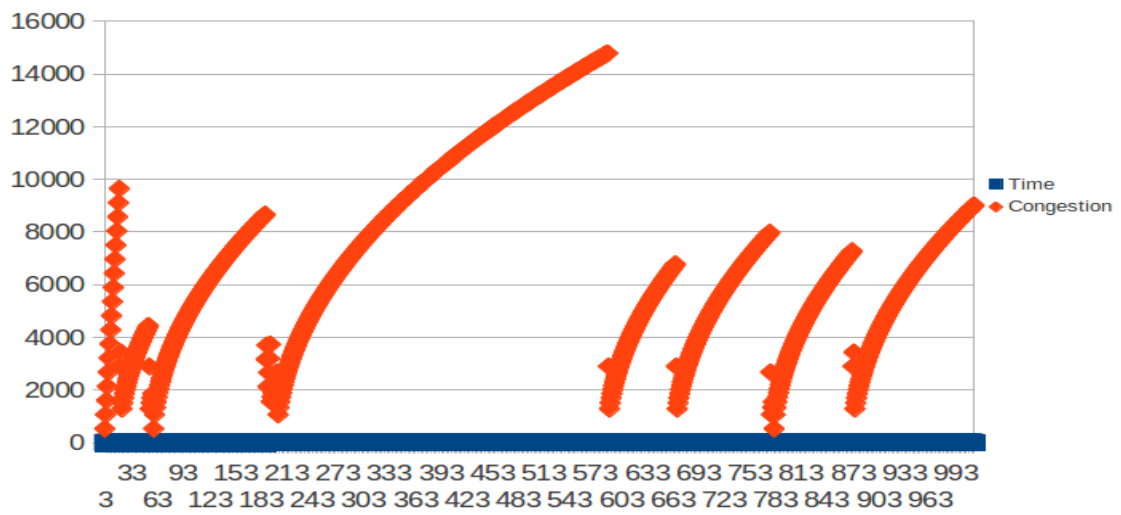


Task 2.5 :

Date Rate = 1 Mbps

Error rate = 0.00001

TCP Model = TCPNewReno



Reflection:

With a reduced data rate: 1 Mbps we got high values for congestion windows.

With an increased error rate of 0.01 we got very few transmission and most drops

Comparison between tcptahoe and NewReno TCP :

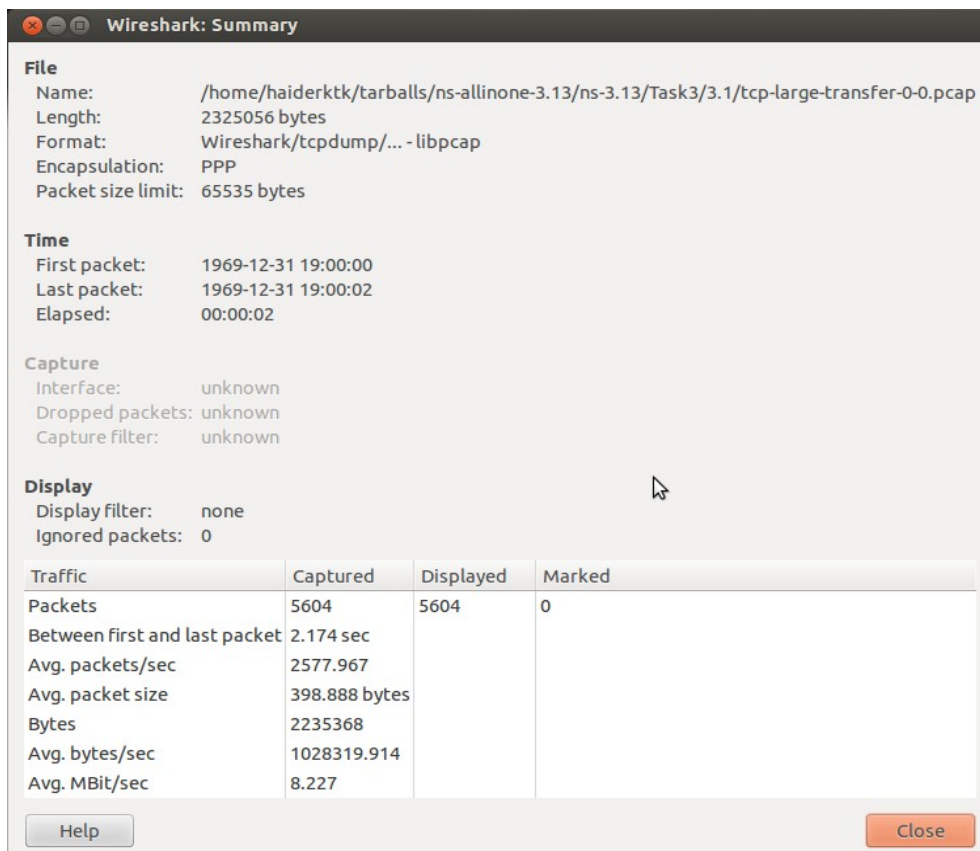
The congestion window for TCPTahoe increases immediately and gets almost half of its peak value. In addition when it gets drops during the transmission it touches zero almost all the times for the congestion window values. On the hand, for TCPNewReno congestion window peak value is much bigger than TCPTahoe. Congestion window for this model increases gradually and tries to be greater than 1/5 of the peak value most of the time during the transmission. It rarely touches the zero during this simulation in contrary to the TCPTahoe. Although, it has more fluctuations as compare to the TCPTahoe.

Task 3.1:

Study the simulation code examples/tcp/tcp-large-transfer.cc. Observe that the data rate in the communication channel is set to 10 Mb/s with a delay of 10ms on each packet. Note the initial output buffer. Once you run the simulation, it will create pcap files.

pcap files generated after running the following command line in the terminal.

`./waf --run scratch/ tcp-large-transfer.`



The image shows the 'Wireshark: Summary' window. It contains the following sections:

- File**
 - Name: /home/haiderkkt/tarballs/ns-allinone-3.13/ns-3.13/Task3/3.1/tcp-large-transfer-0-0.pcap
 - Length: 2325056 bytes
 - Format: Wireshark/tcpdump/... - libpcap
 - Encapsulation: PPP
 - Packet size limit: 65535 bytes
- Time**
 - First packet: 1969-12-31 19:00:00
 - Last packet: 1969-12-31 19:00:02
 - Elapsed: 00:00:02
- Capture**
 - Interface: unknown
 - Dropped packets: unknown
 - Capture filter: unknown
- Display**
 - Display filter: none
 - Ignored packets: 0

Traffic	Captured	Displayed	Marked
Packets	5604	5604	0
Between first and last packet	2.174 sec		
Avg. packets/sec	2577.967		
Avg. packet size	398.888 bytes		
Bytes	2235368		
Avg. bytes/sec	1028319.914		
Avg. MBit/sec	8.227		

At the bottom, there are 'Help' and 'Close' buttons.

While we had default output buffer size and data rate equal to 5Mbps large data was transferred between the nodes . In addition no packets were marked or lost during the transmission.

Task 3.2:

Change the size of the output buffer of the source node (n1) to 512 bytes, and then re- run the simulation. The pcap files should be named as "tcp-large-transfer-v2-x-y.pcap". Compare the transfer rate as well as the total duration it took to complete the file transfer based on the network traffic for the two versions of the pcap files.

Comparison:

v2-0-0

Packets : 7

Between First and last packet : 0.081 sec

Avg.packets/sec : 86.255

Avg.packet size : 115.143 bytes

Bytes : 806

Avg.bytes/sec : 9931.612

Avg.Mbit/sec : 0.079

v2-1-0

Packets : 7

Between First and last packet : 0.081 sec

Avg.packets/sec : 86.255

Avg.packet size : 115.143 bytes

Bytes : 806

Avg.bytes/sec : 9931.612

Avg.Mbit/sec : 0.079

v2-2-0

Packets : 7

Between First and last packet : 0.081 sec

Avg.packets/sec : 86.255

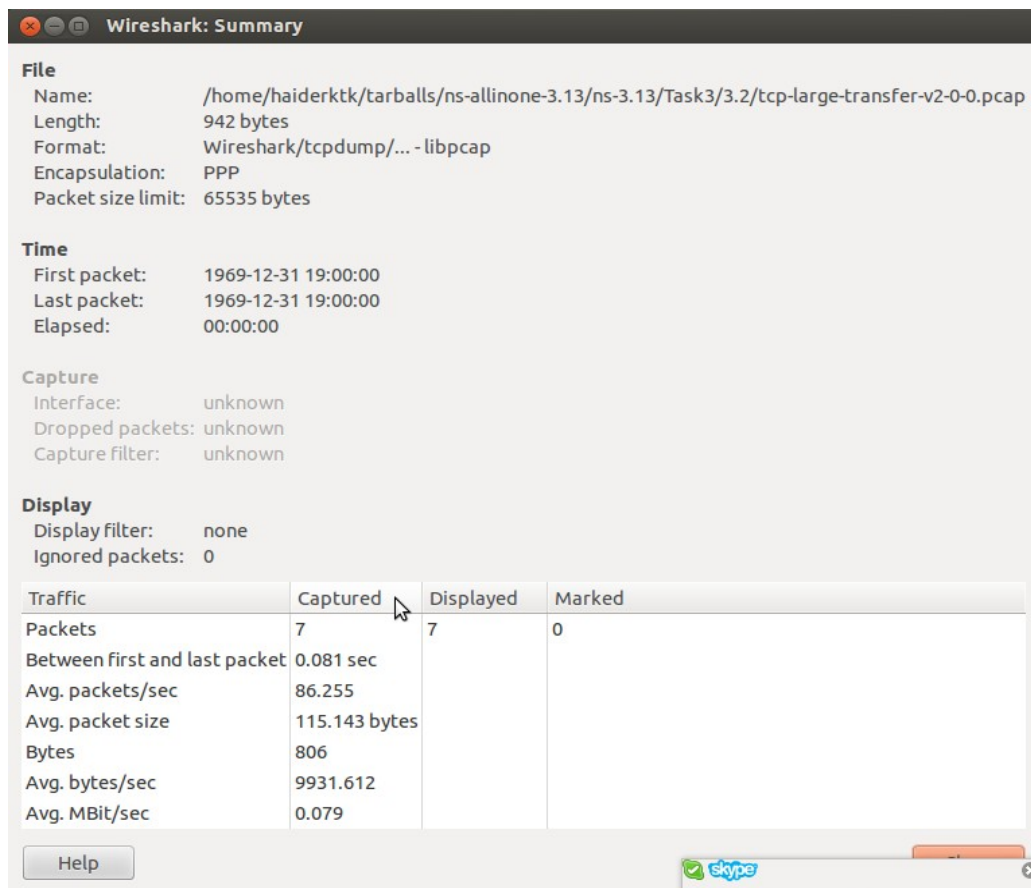
Avg.packet size : 115.143 bytes

Bytes : 806

Avg.bytes/sec : 9931.612

Avg.Mbit/sec : 0.079

Compare the transfer rate as well as the total duration it took to complete the file transfer based on the network traffic for the two versions of the pcap files.



when we reduced the output buffer size to 512, that traffic between the links was reduced much and only 7 packets having only 115.143 bytes average packet size were transmitted through links between node 1 and node 0. Same is the case for links between node 2 and node 0.

Task 3.3:

Change the output buffer back to 1024 bytes and this time add an error rate of 0.001 to the channel. Add a callback to the congestion window size so that the congestion window size changes will be recorded into a file. Re-run the simulation. Save the file as "tcp-large-transfer-v3.dat".

v3-0-0

Packets : 11

Between First and last packet : 0.738 sec

Avg.packets/sec : 14.915

Avg.packet size : 228.182 bytes

Bytes : 2510

Avg.bytes/sec : 3403.307

Avg.Mbit/sec : 0.027

By increasing the buffer size and adding error rate of 0.002 we see the packets have increased from 7 to 11 and the average packet per sec decreases from 86.255 to 14.915 and increased packet size from 115.143 bytes to 228.182 bytes time between first and last packet increases from 0.081 sec to 0.738 sec.

Task 4.1:

Open and study the dynamic-global-routing.cc file that is located under the examples/routing directory of NS-3. Please find the documentation of the code between lines 20-63.

```
// This script exercises global routing code in a mixed point-to-point
// and csma/cd environment. We bring up and down interfaces and observe
// the effect on global routing. We explicitly enable the attribute
// to respond to interface events, so that routes are recomputed
// automatically.
//
// Network topology
//
// n0
// \ p-p
// \ (shared csma/cd)
// n2 ----- n3
// / | |
// / p-p n4 n5 ----- n6
// n1 p-p
// | |
// -----
// p-p
//
// - at time 1 CBRUDP flow from n1 to n6's IP address on the n5/n6 link
// - at time 10, start similar flow from n1 to n6's address on the n1/n6 link
//
// Order of events
// At pre-simulation time, configure global routes. Shortest path from
// n1 to n6 is via the direct point-to-point link
// At time 1s, start CBR traffic flow from n1 to n6
// At time 2s, set the n1 point-to-point interface to down. Packets
// will be diverted to the n1-n2-n5-n6 path
// At time 4s, re-enable the n1/n6 interface to up. n1-n6 route restored.
// At time 6s, set the n6-n1 point-to-point Ipv4 interface to down (note, this
// keeps the point-to-point link "up" from n1's perspective). Traffic will
// flow through the path n1-n2-n5-n6
// At time 8s, bring the interface back up. Path n1-n6 is restored
// At time 10s, stop the first flow.
// At time 11s, start a new flow, but to n6's other IP address (the one
// on the n1/n6 p2p link)
// At time 12s, bring the n1 interface down between n1 and n6. Packets
// will be diverted to the alternate path
// At time 14s, re-enable the n1/n6 interface to up. This will change
// routing back to n1-n6 since the interface up notification will cause
// a new local interface route, at higher priority than global routing
```

Task 4.2:

Identify the shortest path between the source (n0) and the destination (n6) nodes.

The shortest path between n0 to n6 is through n0-n2-n1-n6.

Task 4.3:

Observe the traffic flow from source node (n0) to the destination node (n6). Draw the network routing scheme and save it as "task-4-1.png".

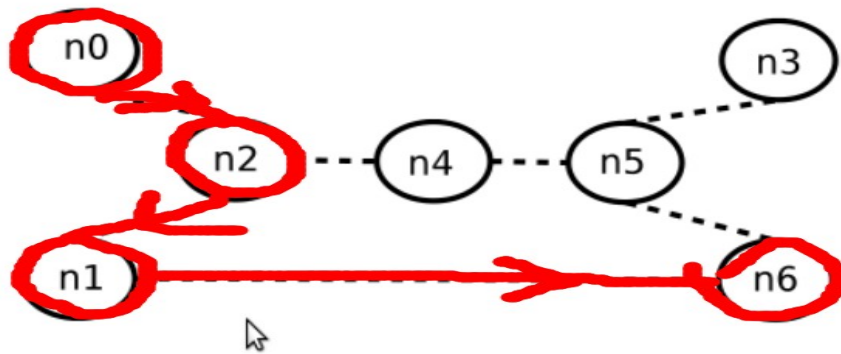


Figure 2: Network Topology of the Nodes.

Task 4.4:

Run the simulation again and then observe how the packets are diverted, when the link between n1 and n6 fails. Draw the network routing scheme and save it as "task-4-2.png"

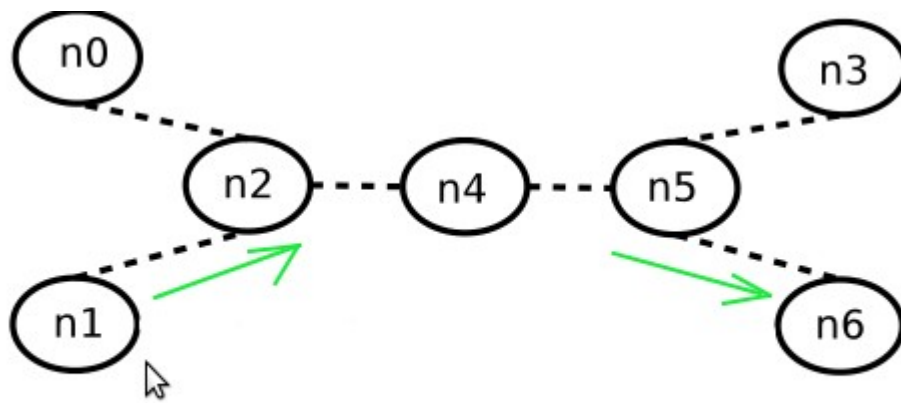
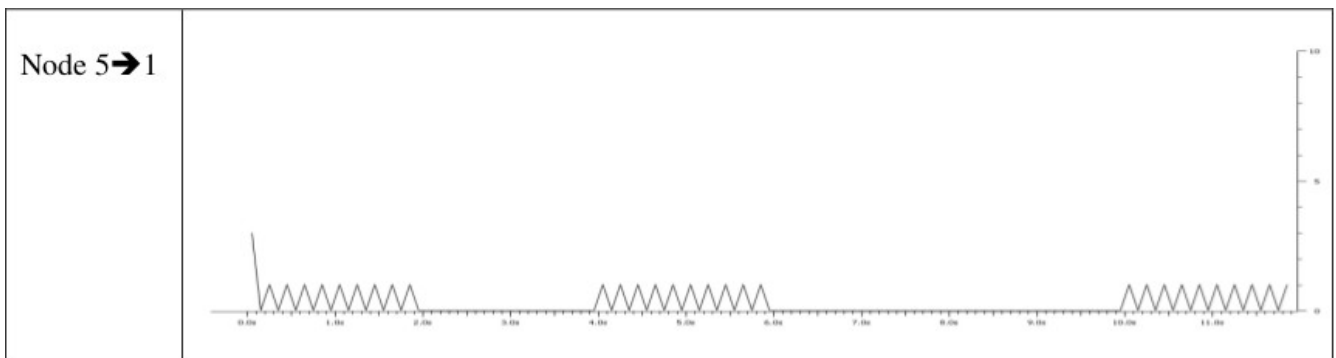
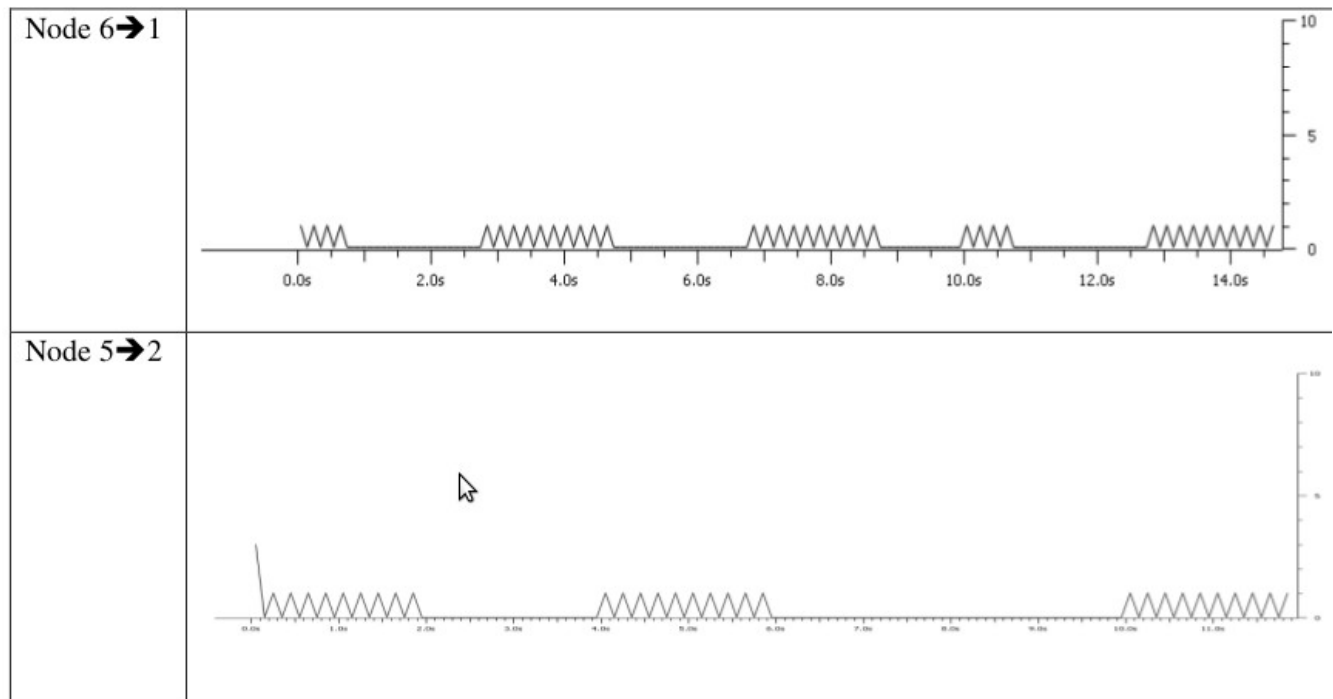


Figure 2: Network Topology of the Nodes.



Reflection:

when the link between n1 and n6 will fall the network routing scheme will change to n1-n2-n5-n6.

Task 5:

In this task, you will work on the same network as in Task-4. You will simulate a packet flow from n1 to n6. This time, you will add a cost parameter to the link between the n1 and n6. First, set the link cost to 1, and run the simulation. Record the pcap files on all interfaces of the nodes. Next, increase the link cost between the n1 and n6 to a relatively larger number, e.g., 4. Reflection: Please elaborate on the results. Explain how the packets are diverted, when the cost of a link at the shortest path changes.

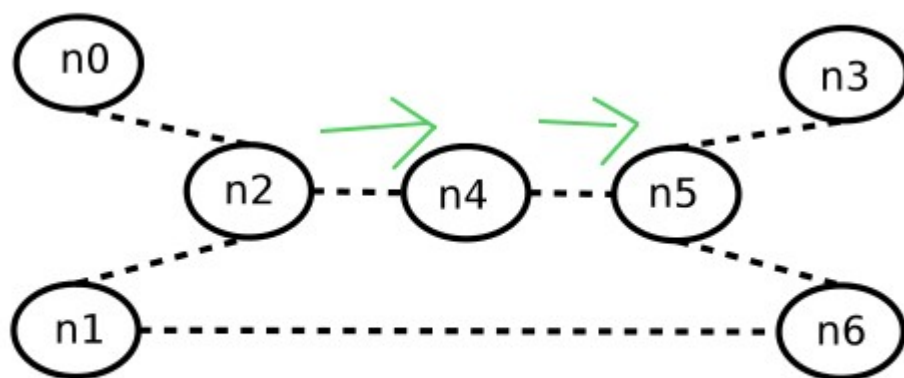


Figure 2: Network Topology of the Nodes.

In this task when the cost associated to a link (6→1) is increased the packets were redirected to other path (1 - 2 , 5 - 6). The following tables show that when we increased the cost of the link(6 - 1) packets traveling through this link are decreased. Same number of packets were added to the alternate path (1 - 2 - 5 - 6).

Link/ Cost

Cost of (6→1) = 1

Cost of (6→1) = 4

6→1

Display

Display filter: none

Ignored packets: 0 (0.000%)

	Captured	Displayed	Displayed %	Marked	Marked %
Packets	36	36	100.000%	0	0.000%
Between first and last packet 14.602 sec					
Avg. packets/sec	2.602				
Avg. packet size	80.000 bytes				
Bytes	3040	3040	100.000%	0	0.000%
Avg. bytes/sec	208.191				
Avg. MBK/sec	0.002				

Info OK Cancel

5→1

Display

Display filter: none

Ignored packets: 0 (0.000%)

	Captured	Displayed	Displayed %	Marked	Marked %
Packets	30	30	100.000%	0	0.000%
Between first and last packet 11.796 sec					
Avg. packets/sec	2.543				
Avg. packet size	80.000 bytes				
Bytes	2400	2400	100.000%	0	0.000%
Avg. bytes/sec	203.454				
Avg. MBK/sec	0.002				

Info OK Cancel

Display

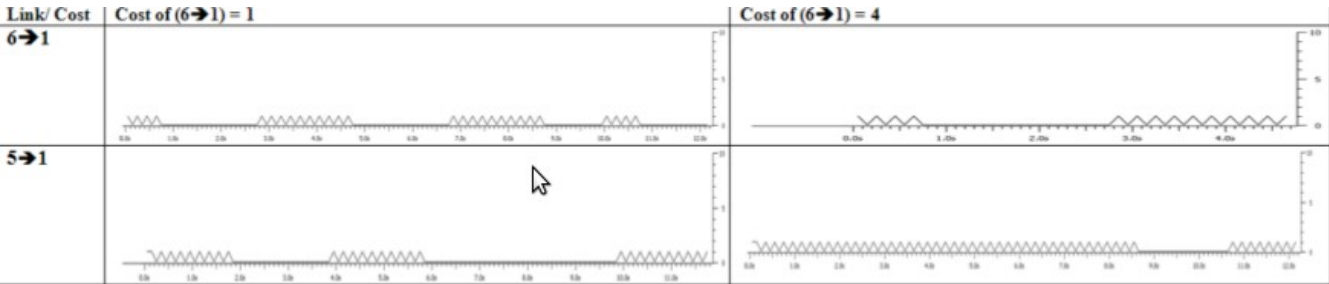
Display filter: none

Ignored packets: 0 (0.000%)

	Captured	Displayed	Displayed %	Marked	Marked %
Packets	54	54	100.000%	0	0.000%
Between first and last packet 12.597 sec					
Avg. packets/sec	4.287				
Avg. packet size	80.000 bytes				
Bytes	4320	4320	100.000%	0	0.000%
Avg. bytes/sec	342.944				
Avg. MBK/sec	0.003				

Info OK Cancel

The following graphs show the traffic and link position over the simulation time. We can see the when the link 6-1 had lower cost; it had more traffic during the time slots when link was up. On the other hand when this link had more cost; traffic was redirected to the alternate path



Conclusion:

By setting the cost value to 1 the packets over the link sent was 32 and after increasing the cost to 4 the packets on the link got down to 14 so that caused the deficit of 18 packets , and the diversion of those 18 packets got to link n2-n5.