ECE 545 – Advanced Computer Networks

Project Report

In-Depth Investigation of TCP via NS3 Simulation

Course Instructor: Prof. Yu Cheng

Submitted by,

Anand Baraguru Venkateshmurthy

A20429060

INTRODUCTION

NS is discrete event driven network simulator that simulates variety of networks and used in research and education. Programming language used is python or C++. Here ns3 is used for in-depth investigation of TCP and UDP protocols. Transmission Control Protocol provides connection oriented service by host to host connectivity. TCP uses handshaking to establish a connection. Whereas, User Datagram Protocol is connectionless and it is unreliable.

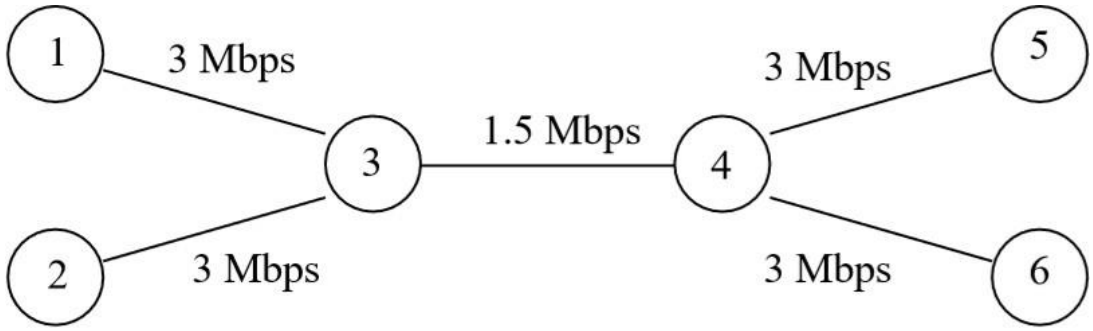


Fig.1 Network Topology

Number of nodes is six and five links. Node 3 and 4 is connected by a bottleneck link with speed of 1.5 Mbps. Default values are used throughout the project and TCP Reno is used by default unless specified.

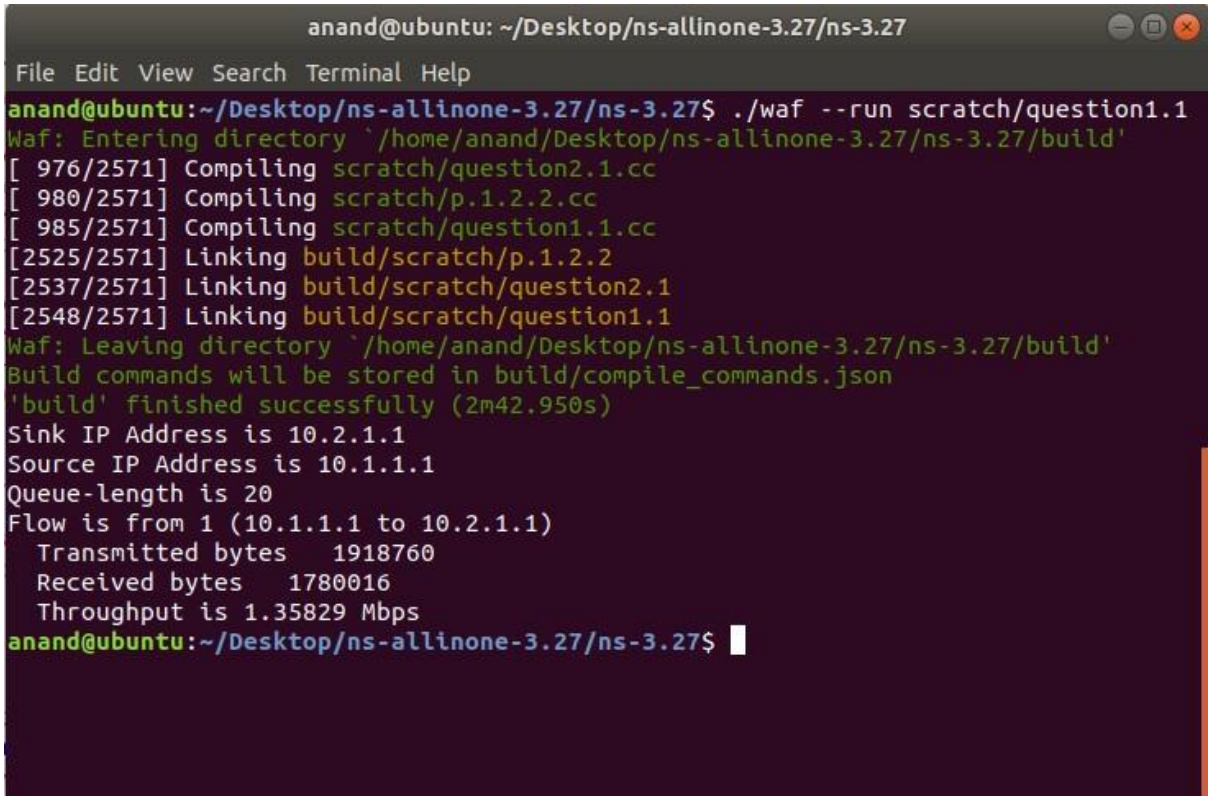
IP and subnet mask configuration is as shown below:

Connection	IP Address	Subnet Mask
Node 1 to Node 3	10.1.0.0	255.255.255.0
Node 2 to Node 3	10.2.0.0	255.255.255.0
Node 3 to Node 4	10.3.0.0	255.255.255.0
Node 4 to Node 5	10.4.0.0	255.255.255.0
Node 5 to Node 6	10.5.0.0	255.255.255.0

Part 1: Efficiency of the TCP congestion control, flow control, and reliable data transfer protocols.

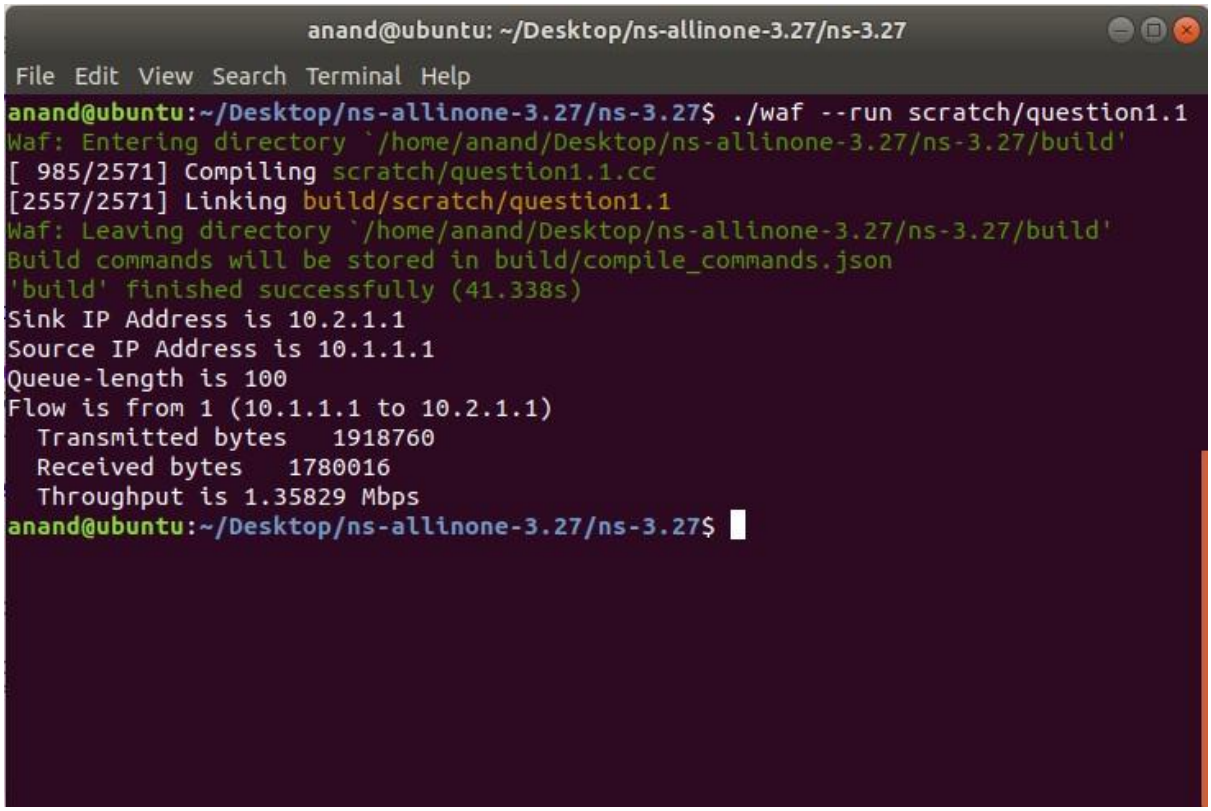
1).Use the default parameters, and run your simulation from 0 to 10 seconds. Analyze your simulation results and answer the following questions. What is the total number of segments and the number of bytes successfully transmitted during the 10 seconds? What is the average throughput achieved? How does this compare with the bottleneck bandwidth in your topology? Then, change the queue size used at node 3 to a value much smaller than the default value in one case, and to a value much larger than the default value in the other case. Run simulations to obtain the average throughput in these two cases, respectively. Do you get different average throughputs compared to that under the default queue size? Explain your observations?

Soln: By using default parameters and running simulation from 0 to 10 seconds, the output is shown below.



According to output throughput obtained is 1.35829 Mbps. Assuming MTU of 536 bytes, the total number of segments transmitted is 3579.776 ≈ 3580. The number of bytes successfully transmitted during 10 seconds is 1918760 bytes. The average throughput achieved is 1.35829 Mbps. Because of bottleneck link between nodes 3 and 4 with link speed of 1.5 Mbps, the throughput achieved (1.35829 Mbps) cannot be more than 1.5 Mbps.

When queue is much large than default value say 100 packets. Then the output is :



Because of the bottleneck link of 1.5 Mbps speed, the throughput achieved cannot be more than 1.5 Mbps even though the queue is increased to 100 packets.

When the queue is decreased than the default length (say 5 packets):

The throughput achieved is 1.15829 Mbps which is less than the bottleneck link speed of 1.5 Mbps because small queue packets may get lost.

2). Set the queue size in the bottleneck link to a proper finite value, so that retransmission and congestion control due to packet dropping and timeout can be observed. Set the receive window large enough to bypass the flow control. You need to write appropriate codes to trace the congestion window, RTT, EstimatedRTT and TimeoutInterval. Run your simulation for long enough time, so that you can capture some packet loss events and timeout events; using multiple simulation runs to capture such events is also fine.

i). Plot the congestion window (cwnd) as a function of time. Use the graphs generated from one or multiple simulation runs to demonstrate the options of slow start, congestion avoidance, the reaction to triple duplicate ACK, and the reaction to timeout.

Soln: I have generated a cwnd out file and then plotted the values of cwnd vs time. The graph obtained is shown below. GNU plot was not working, so I have plotted the graph using Excel.

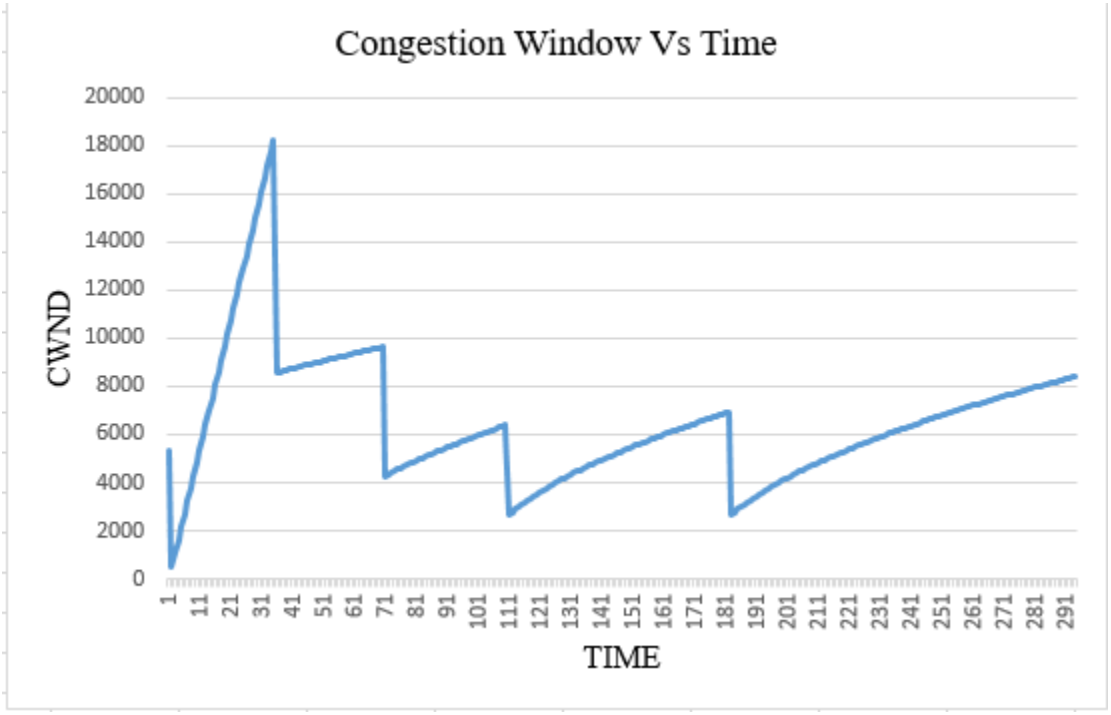


Fig. 2. Congestion window Vs Time

From the above graph we can infer packet loss and timeout events as follows:

Slow Start: - In this phase, the congestion window grows exponentially i.e. for every RTT the congestion window gets doubled. Mathematical formula is CongWin=CongWin + MSS. The slow start is observed in the interval [1,31].

Congestion Avoidance: - In this phase, the congestion window grows linearly i.e. for every RTT the congestion window is increased by 1MSS. Mathematical formula is CongWin = CongWin + MSS*(MSS/CongWin). The congestion Avoidance is onserved in the interval [31, 70], [71, 110], [111, 180] and [181, 291].

When packet loss occurs due to triple-duplicate ACK then congestion window is set to half of the current value and it goes to the congestion avoidance phase. The packet loss due to triple-duplicate ACK is observed after transmission round 31, 71, 111 and 181.

When packet loss occurs due to time-out then congestion window is set to 1 and goes to slow start phase. In the graph shown packet loss due to timeout has not occurred.

ii). Plot the RTT and EstimatedRTT as a function of time according to one of your trace files, to obtain a similar graph to Fig. 3.32 in our textbook.

Soln: I have generated an out file and then plotted a graph in Excel using values from that file.

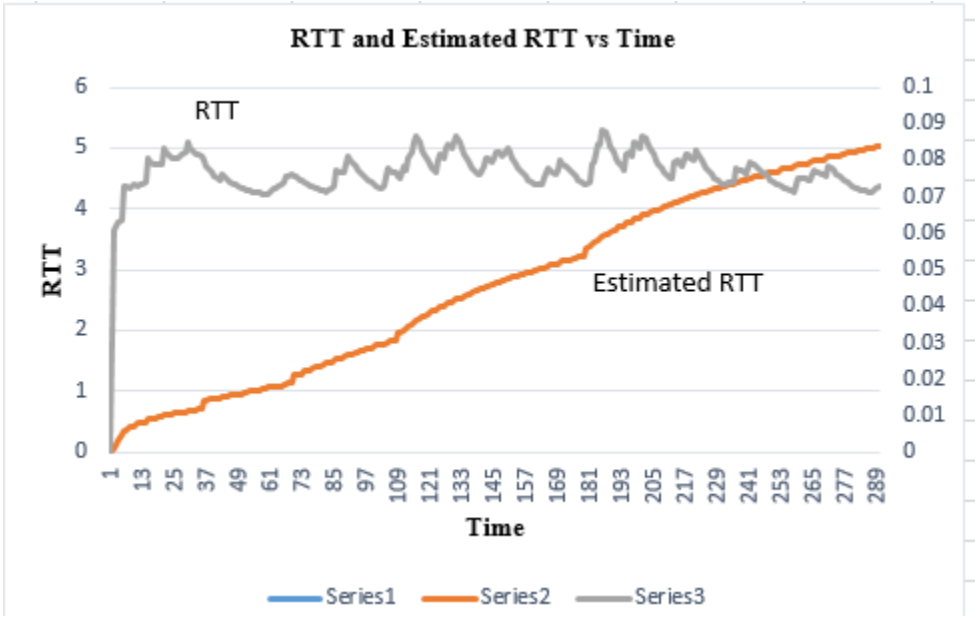
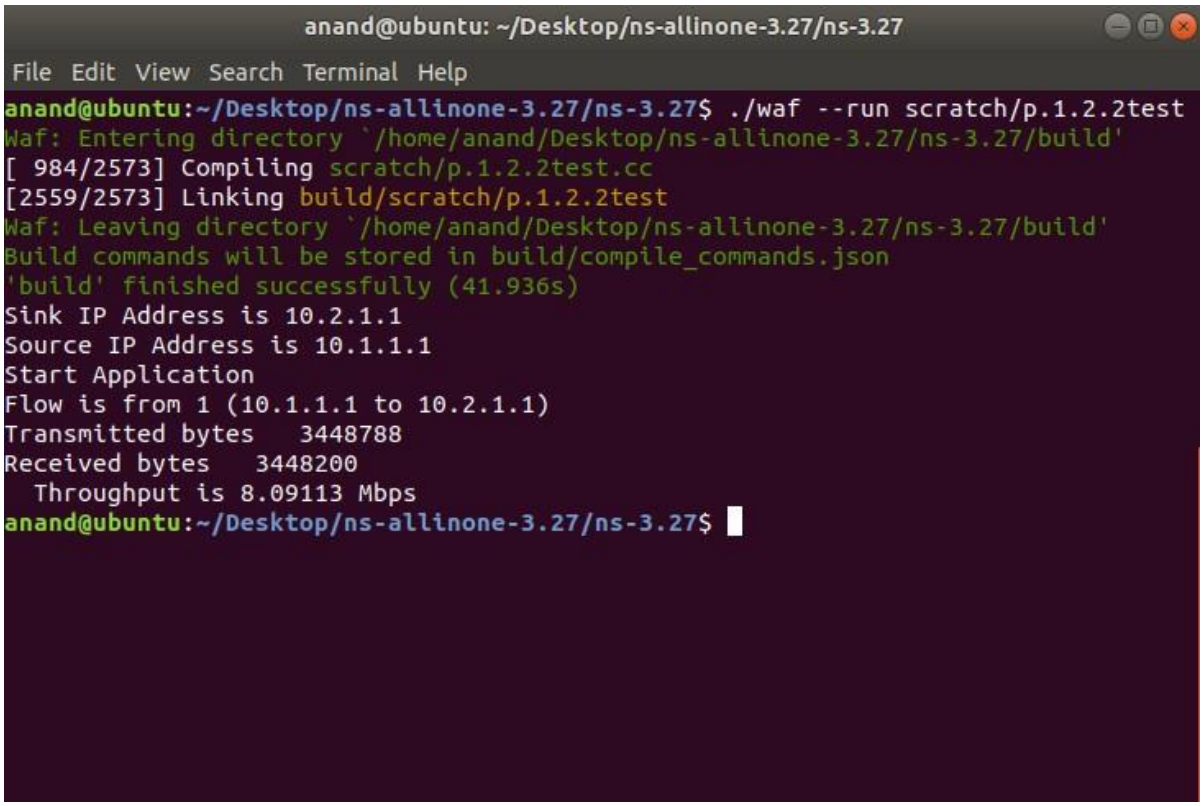


Fig.3 RTT Vs Time

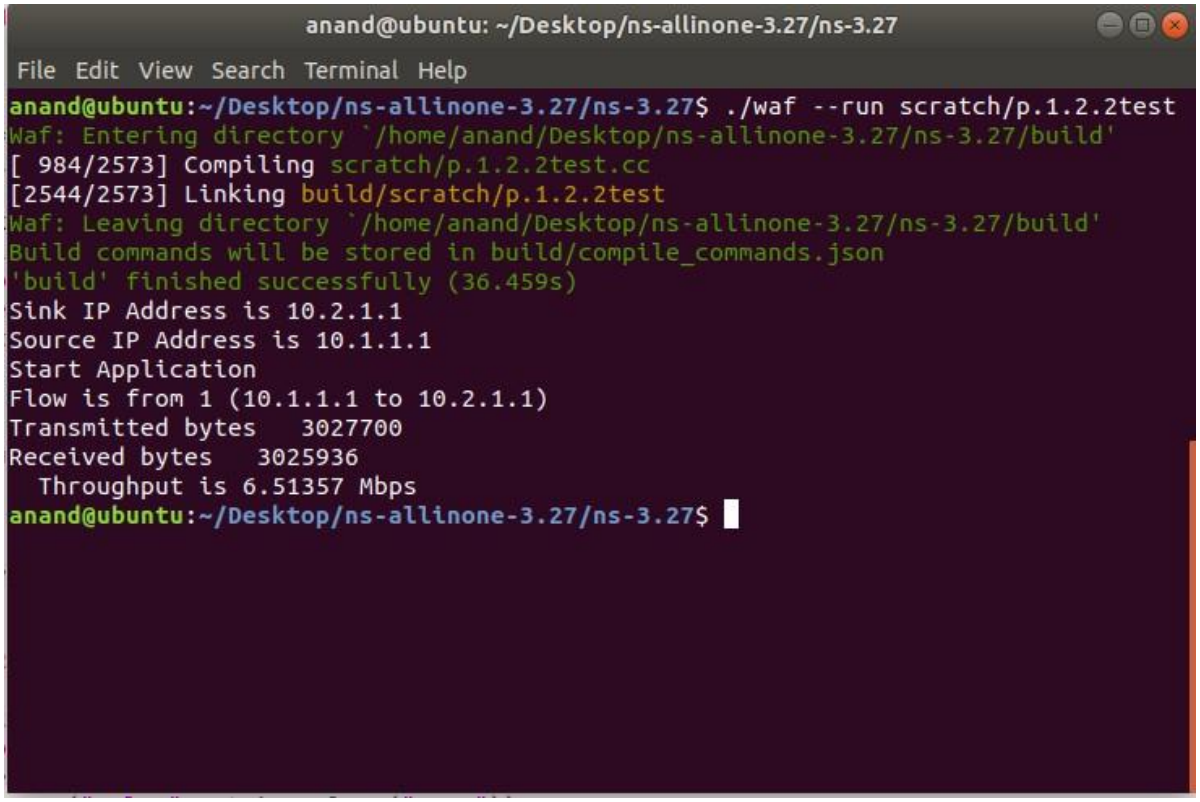
Due to congestion, Round Trip Time (RTT) (represented by silver line) values fluctuates from packet to packet. Whereas, the Estimated RTT gives a smooth curve because it is calculated on weighted averages of RTT value and more weight is given to recently calculated samples.

iv). Average throughput output when packet loss rate 10^{-4}, 10^{-3}, 10^{-2} and 10^{-1} respectively are:

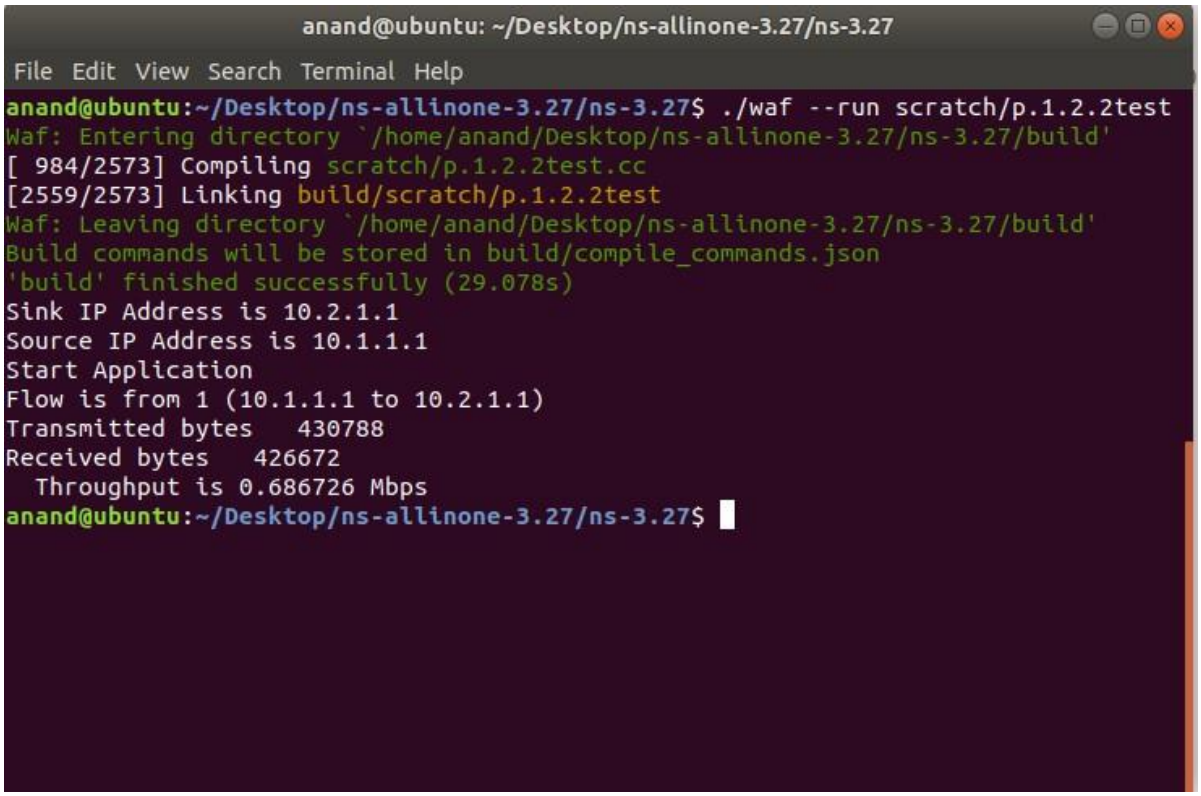
When packet loss rate is 10^{-4}:



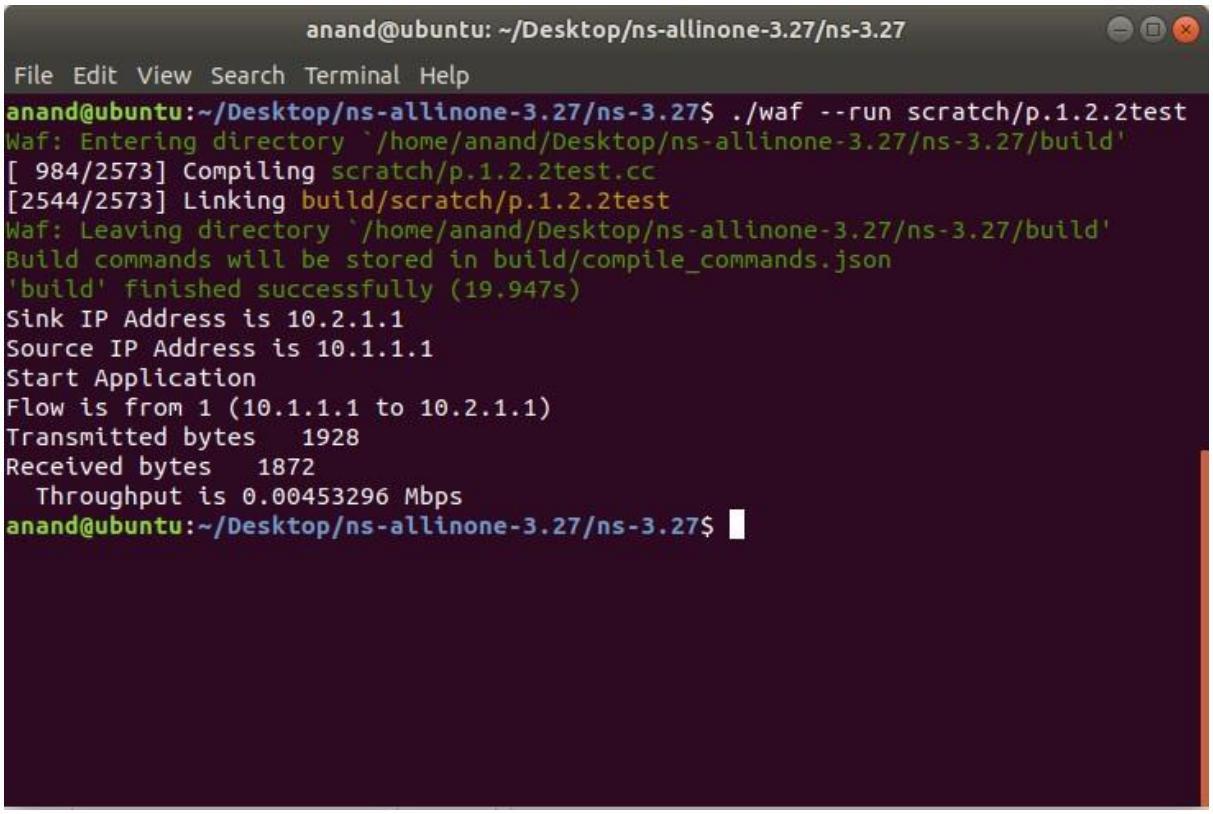
When packet loss rate is 10^{-3}:



When packet loss rate is 10^{-2}:



When packet loss rate is 10^{-1}:



By Simulation:

Packet Loss rate	Throughput
10^{-4}	8091130
10^{-3}	6513570
10^{-2}	686726
10^{-1}	4532.96

Average Throughput $= \frac{1.22*MSS}{RTT*\sqrt{L}}$

From experiments, we know that the RTT varies from 0.060 to 0.070. Hence, taking RTT as 0.065 and Maximum Segment Size (MSS) =1040.

By Formula:

Packet Loss rate	Throughput
10^{-4}	1952000
10^{-3}	617276
10^{-2}	195200
10^{-1}	61727

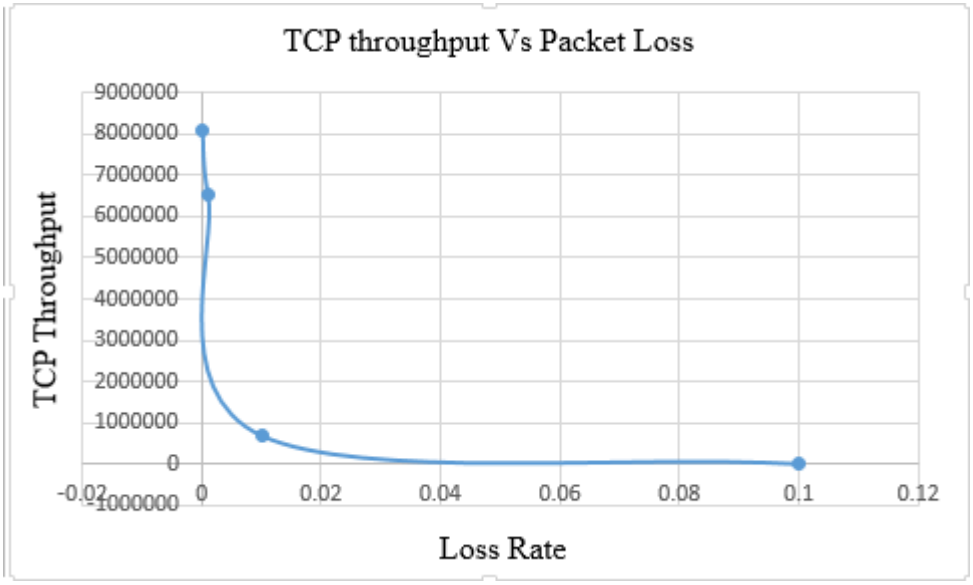


Fig.4 By Simulation: TCP throughput Vs Packet loss Rate

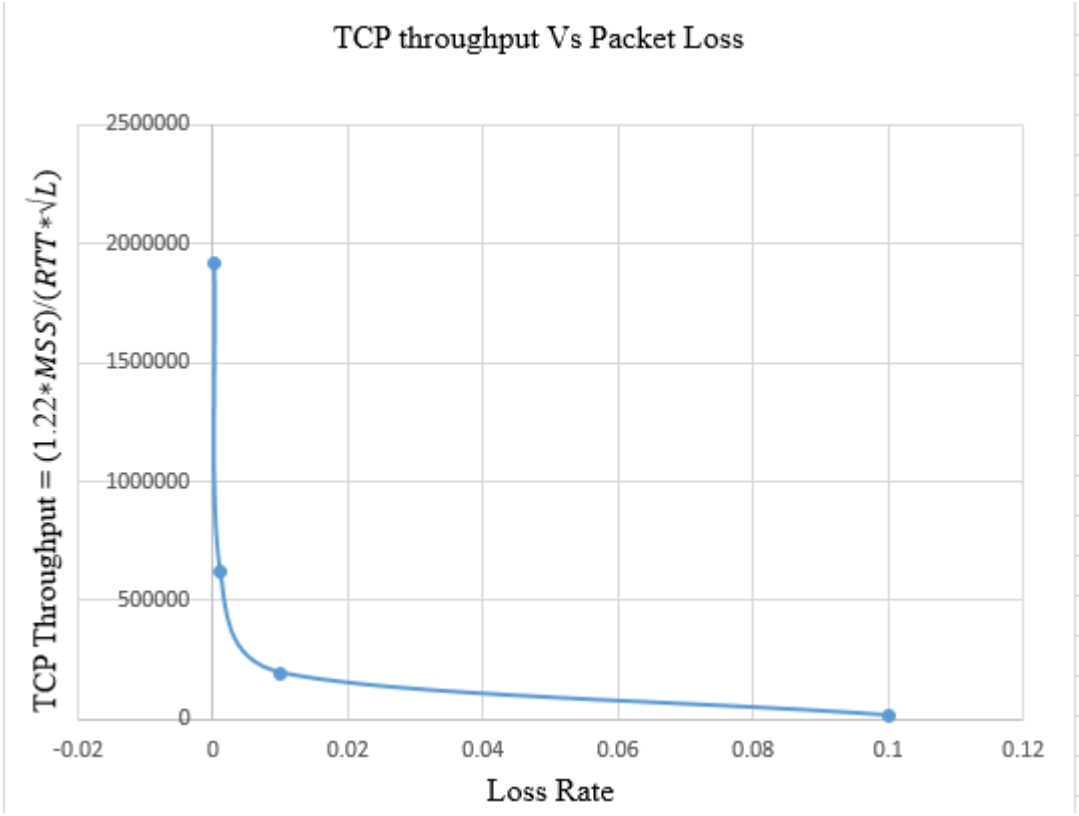


Fig.5 By Formula: TCP throughput Vs Packet loss Rate

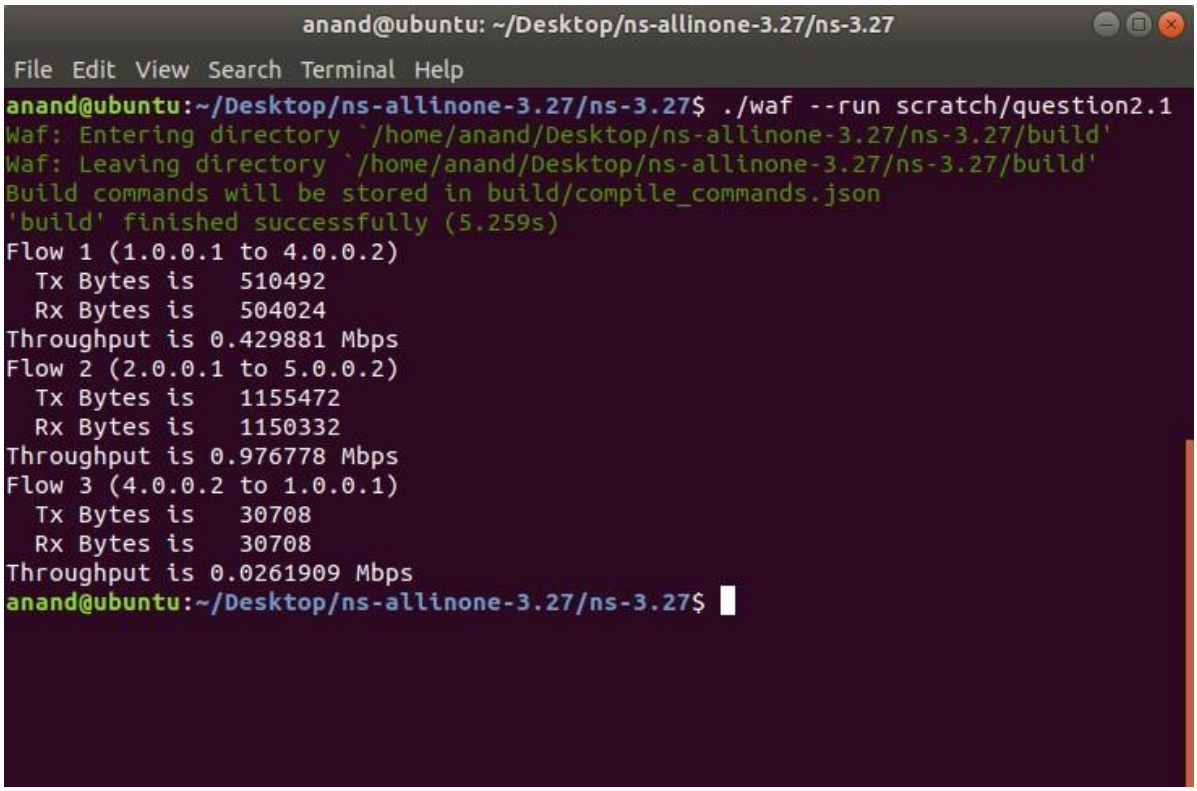
The two curves are similar and are almost close to each other.

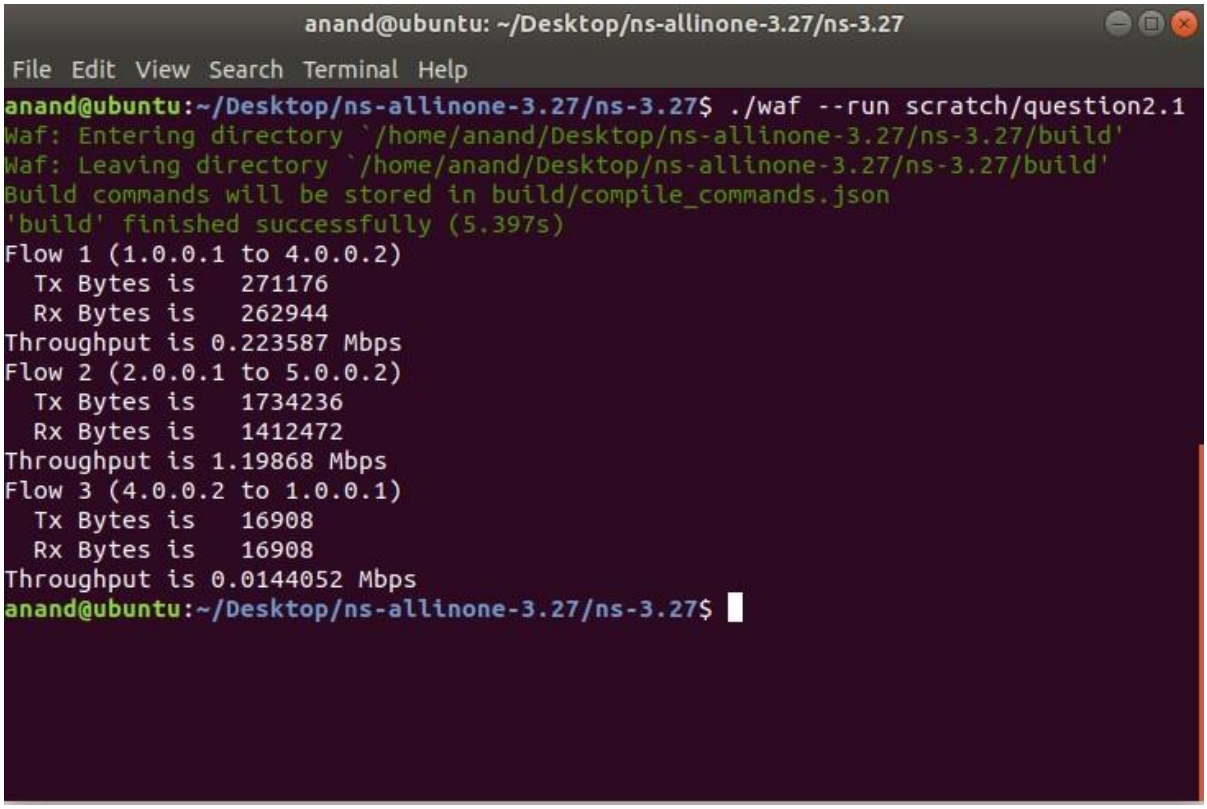
Part 2: Resource sharing under the transport layer protocol.

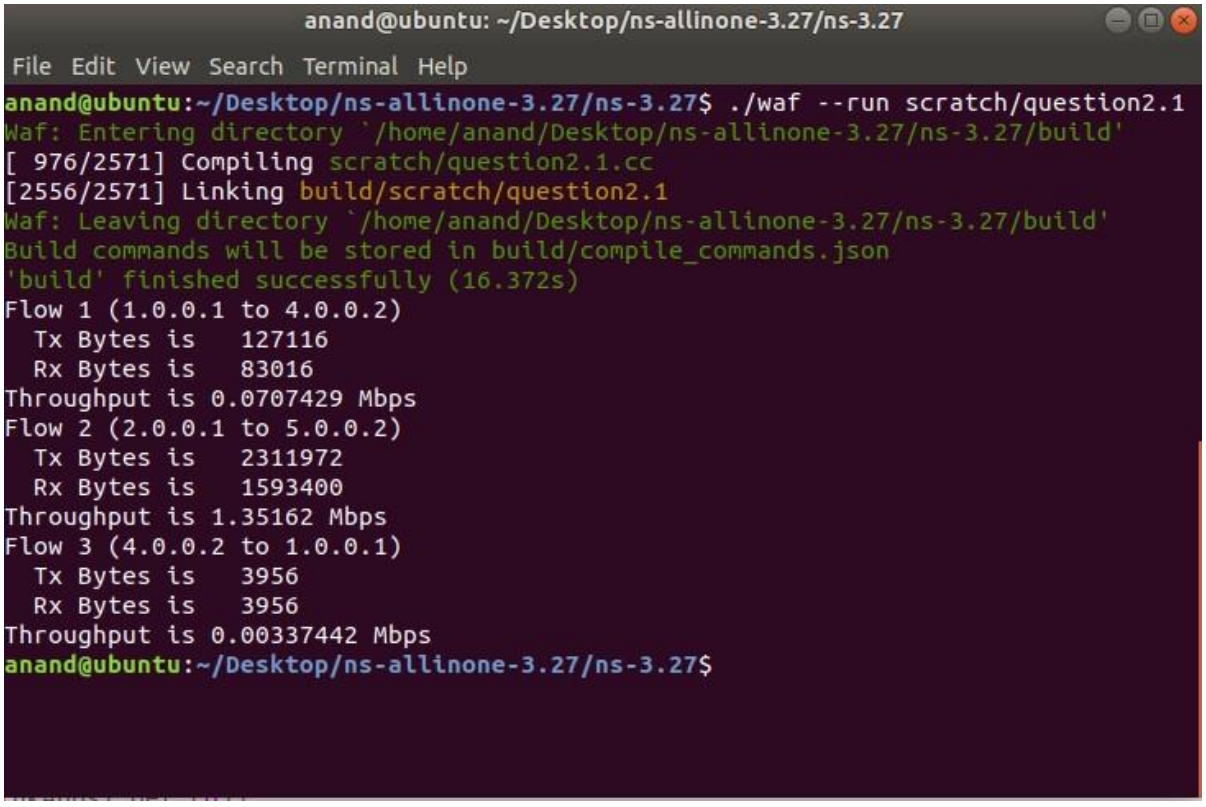
Question 2.1: I have configured 2 flows. One flow is UDP and one flow is TCP. Flow-1 is between node-1 (source) and node-5 (destination). Flow-2 is between node-2 (source) and node-6 (destination). Below figure shows the output of both the flows.

Output after running the experiment multiple times by modifying the traffic generation rate of the UDP flow is

Here I have shown only three iterations:




anand@ubuntu: ~/Desktop/ns-allinone-3.27/ns-3.27
File Edit View Search Terminal Help
anand@ubuntu:~/Desktop/ns-allinone-3.27/ns-3.27$ ./waf --run scratch/question2.1
Waf: Entering directory `/home/anand/Desktop/ns-allinone-3.27/ns-3.27/build'
Waf: Leaving directory `/home/anand/Desktop/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (5.397s)
Flow 1 (1.0.0.1 to 4.0.0.2)
Tx Bytes is 271176
Rx Bytes is 262944
Throughput is 0.223587 Mbps
Flow 2 (2.0.0.1 to 5.0.0.2)
Tx Bytes is 1734236
Rx Bytes is 1412472
Throughput is 1.19868 Mbps
Flow 3 (4.0.0.2 to 1.0.0.1)
Tx Bytes is 16908
Rx Bytes is 16908
Throughput is 0.0144052 Mbps
anand@ubuntu:~/Desktop/ns-allinone-3.27/ns-3.27$


anand@ubuntu: ~/Desktop/ns-allinone-3.27/ns-3.27
File Edit View Search Terminal Help
anand@ubuntu:~/Desktop/ns-allinone-3.27/ns-3.27$ ./waf --run scratch/question2.1
Waf: Entering directory `/home/anand/Desktop/ns-allinone-3.27/ns-3.27/build'
[976/2571] Compiling scratch/question2.1.cc
[2556/2571] Linking build/scratch/question2.1
Waf: Leaving directory `/home/anand/Desktop/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (16.372s)
Flow 1 (1.0.0.1 to 4.0.0.2)
Tx Bytes is 127116
Rx Bytes is 83016
Throughput is 0.0707429 Mbps
Flow 2 (2.0.0.1 to 5.0.0.2)
Tx Bytes is 2311972
Rx Bytes is 1593400
Throughput is 1.35162 Mbps
Flow 3 (4.0.0.2 to 1.0.0.1)
Tx Bytes is 3956
Rx Bytes is 3956
Throughput is 0.00337442 Mbps
anand@ubuntu:~/Desktop/ns-allinone-3.27/ns-3.27$

Graph of UDP throughput on x-axis and TCP throughput on y-axis:

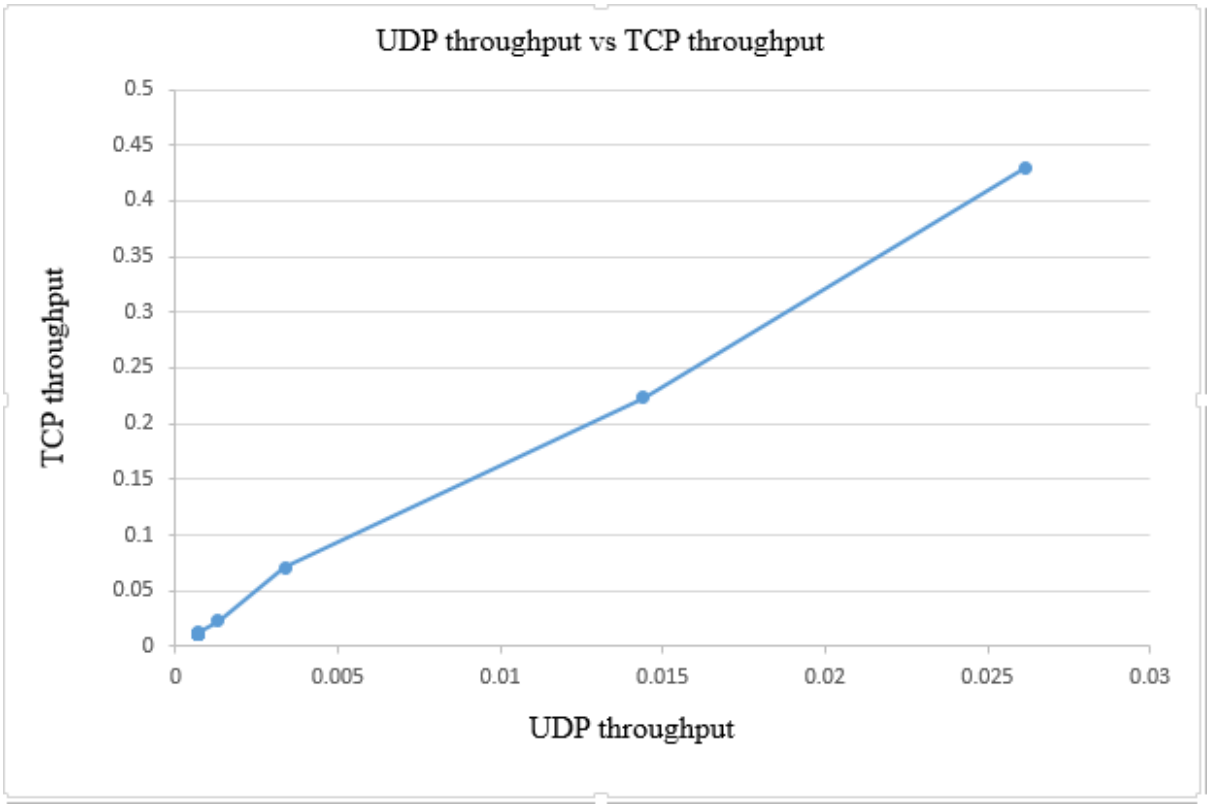


Fig. 6 UDP throughput Vs TCP throughput

UDP throughput at which two flows achieve a fair share of a link is 0.015 Mbps.

Calculating loss rate of TCP connection:

```
anand@ubuntu: ~/Desktop/ns-allinone-3.27/ns-3.27
File  Edit  View  Search  Terminal  Help
anand@ubuntu:~/Desktop/ns-allinone-3.27/ns-3.27$ ./waf --run scratch/question2.1
new
Waf: Entering directory `/home/anand/Desktop/ns-allinone-3.27/ns-3.27/build'
[ 979/2575] Compiling scratch/p.1.2.4.cc
[ 990/2575] Compiling scratch/question2.1new.cc
[2545/2575] Linking build/scratch/question2.1new
[2557/2575] Linking build/scratch/p.1.2.4
Waf: Leaving directory `/home/anand/Desktop/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (32.805s)
Flow 1 (1.0.0.1 -> 4.0.0.2)
  Tx Bytes:   991476
  Rx Bytes:   983244
Throughput0.833606 Mbps
Flow 2 (4.0.0.2 -> 1.0.0.1)
  Tx Bytes:   53208
  Rx Bytes:   52948
Throughput0.0450595 Mbps
Flow 3 (2.0.0.1 -> 5.0.0.2)
  Tx Bytes:   16448
  Rx Bytes:   16448
Throughput0.0155951 Mbps
anand@ubuntu:~/Desktop/ns-allinone-3.27/ns-3.27$
```

For TCP connection:

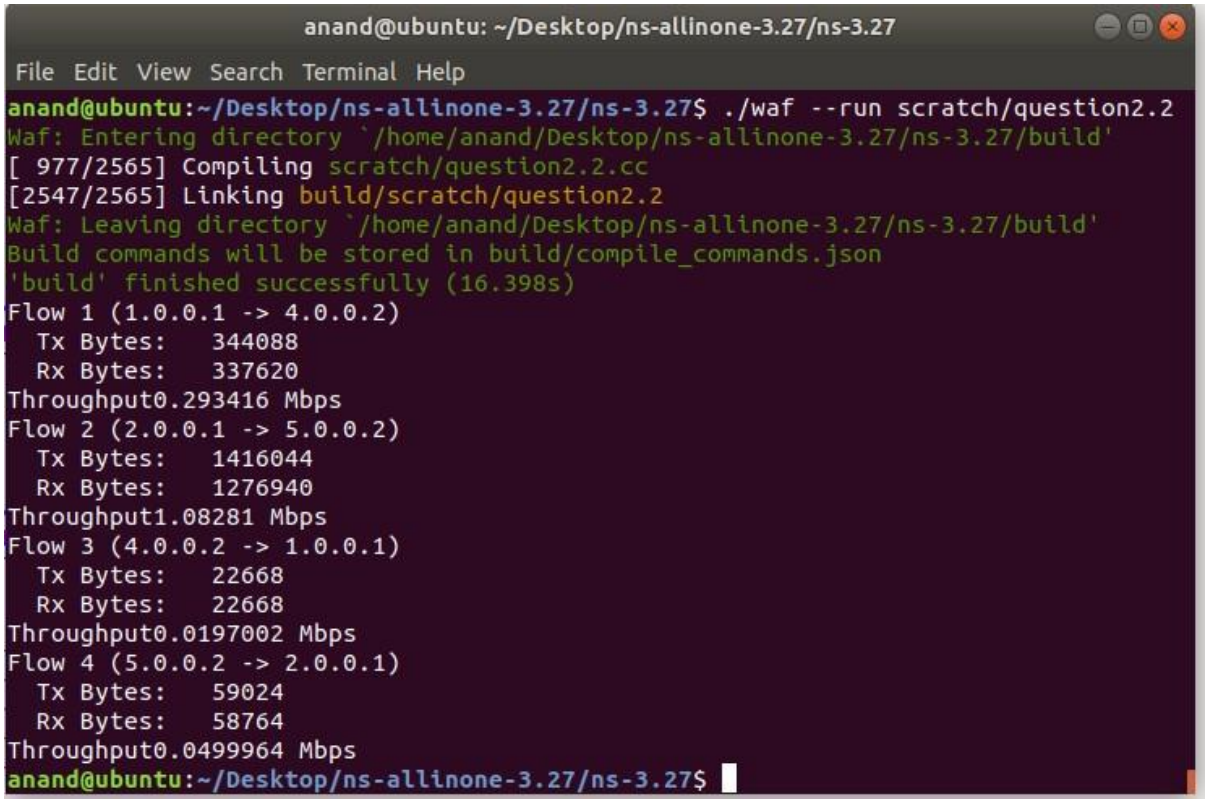
Transmitted bytes is 991476.

Received bytes is 983244

Loss rate is 0.008

Question 2.2: Both flows are TCP, with a maximum window size (MWS) of 30 packets for flow-1 and 6 packets for flow-2, respectively.

Output:



The flows do not get fair share of link. Because flow-1 path has 30 packets and flow-2 has 6 packets to transmit, flow-1 takes most of the available bandwidth. So the throughput of flow-1 is more than that of flow-2.

Question 2.3: For flow-1 packet size is 1000 bytes and for flow-2 packet size is 500 bytes and two flows have same MWS. The packet size of flow-1 is more than that of flow-2 hence flow-1 transmits more packets than flow-2 so the throughput of flow-1 will be more than that of flow-2. Thus, two flows don't get fair share of link.

Question 2.4: I was not able to implement practically but based on my theoretical knowledge I can say that the throughput of TCP Westwood is higher than that of TCP Reno.