Ns2 Familiarization Report

Team No. 23

1. Kranthi Kumar Katam (UIN: 225009204)
2. Mathew Roe (UIN: 321007055)
3. Jifang Mu (UIN: 821005313)

**Description:**

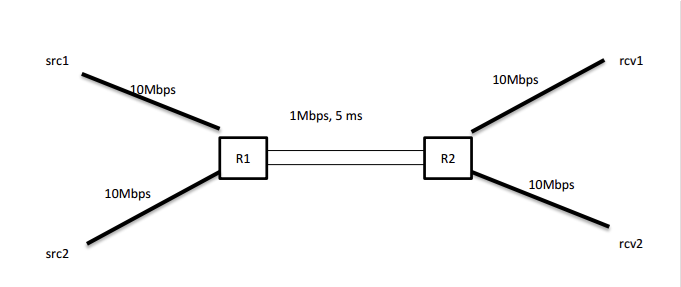
In this assignment, we use NS-2 simulator to simulate the following topology and configuration:

• Two routers (R1, R2) connected with a 1 Mbps link and 5ms of latency

• Two senders (src1, src2) connected to R1 with 10 Mbps links

• Two receivers (rcv1, rcv2) connected to R2 with 10 Mbps links

• Application sender is FTP over TCP



We have implemented two versions of TCP, TCP Vegas and TCP Sack. For these two versions, there are 3 cases which will differ in delay time for souce2-router1 and router2-receiver2. The cases are shown as below:

1. Case 1:

* Src1-R1 && R2-Rcv1, end to end delay =5ms
* Src2-R1 && R2-Rcv2, end to end delay =12.5ms

1. Case 2:

* Src1-R1 && R2-Rcv1, end to end delay =5ms
* Src2-R1 && R2-Rcv2, end to end delay =20ms

1. Case 3:

* Src1-R1 && R2-Rcv1, end to end delay =5ms
* Src2-R1 && R2-Rcv2, end to end delay =27.5ms

The end to end RTTs of both the sources is in the ratio of 1:2, 1:3, 1:4 for the cases 1,2,3 respectively.

**Procedure:**

1. Create a new simulator, and then trace all the data to tcp.tr file.

2. The network topology is designed as in the given assignment.

3. Take input from the command line such as TCP flavor and the case number.

4. As given in the assignment change the delay values as per the case number.

5. Set the TCP flavor as defined by the user and connect it to source1 and similarly to the other source also.

6. Set up TCP sinks to receive data and connect them to receivers at the other end.

7. Set up FTP application to send data over TCP.

8. Create a function proc record to calculate throughput of the sources at 0.5ms time intervals and finally after 400ms find the average throughput of the sources and their ratios.

9. Create a function proc finish to execute the NAM file to analyze the data flow in the topology.

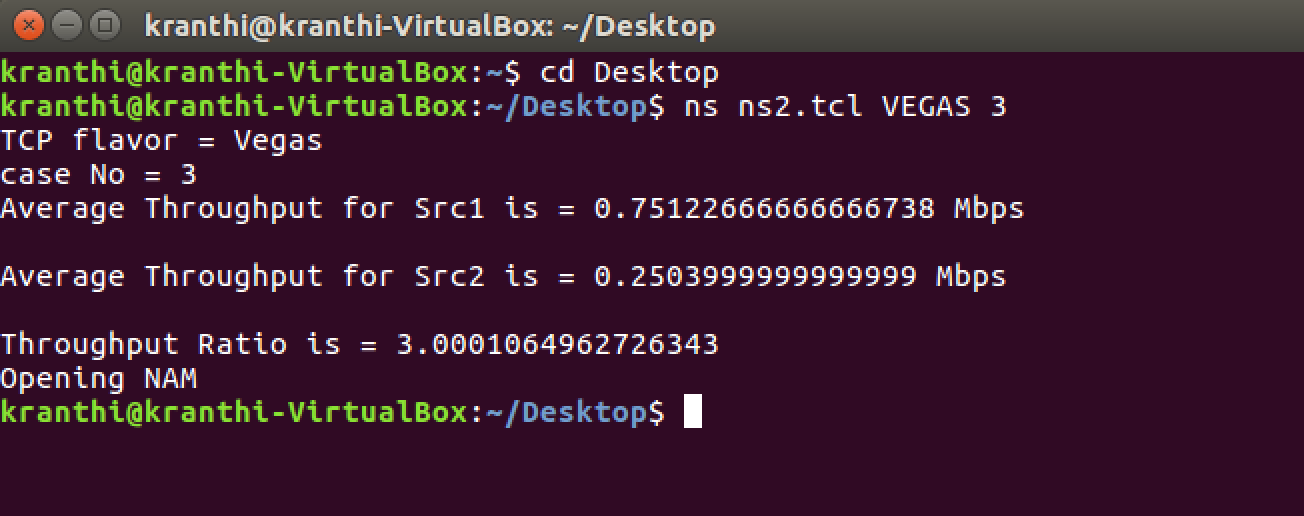
10. Both sources start sending at data at time 0 and ends at 400ms time.

After the implementation of code, then debug the code.

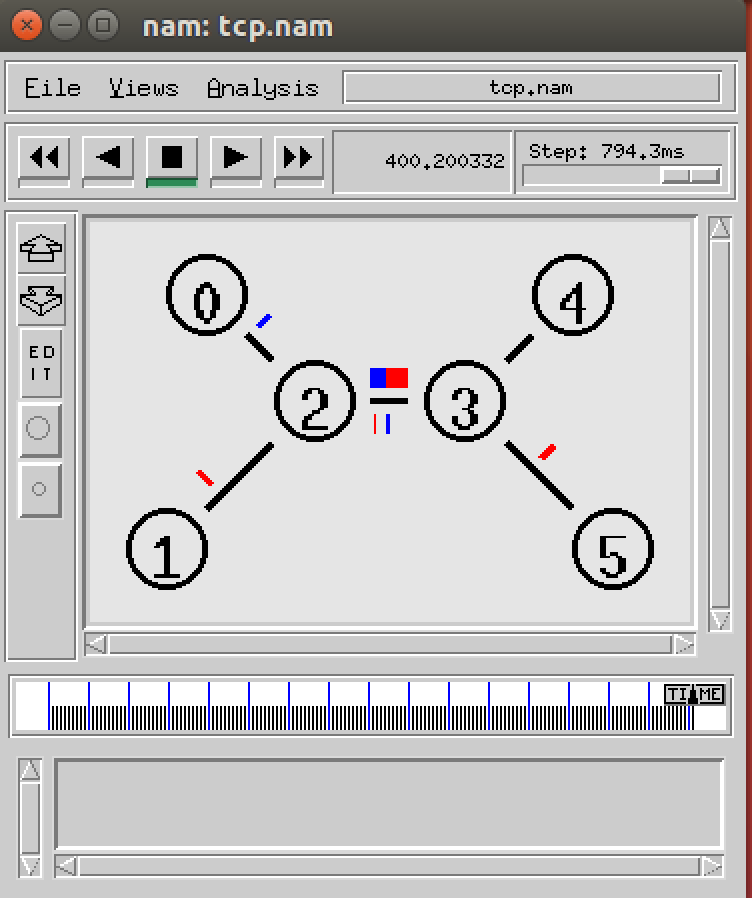
Use the command line as shown to debug the code

**ns ns2.tcl <TCP Flavor(all Caps)> <case number(1 or 2 or 3)>**

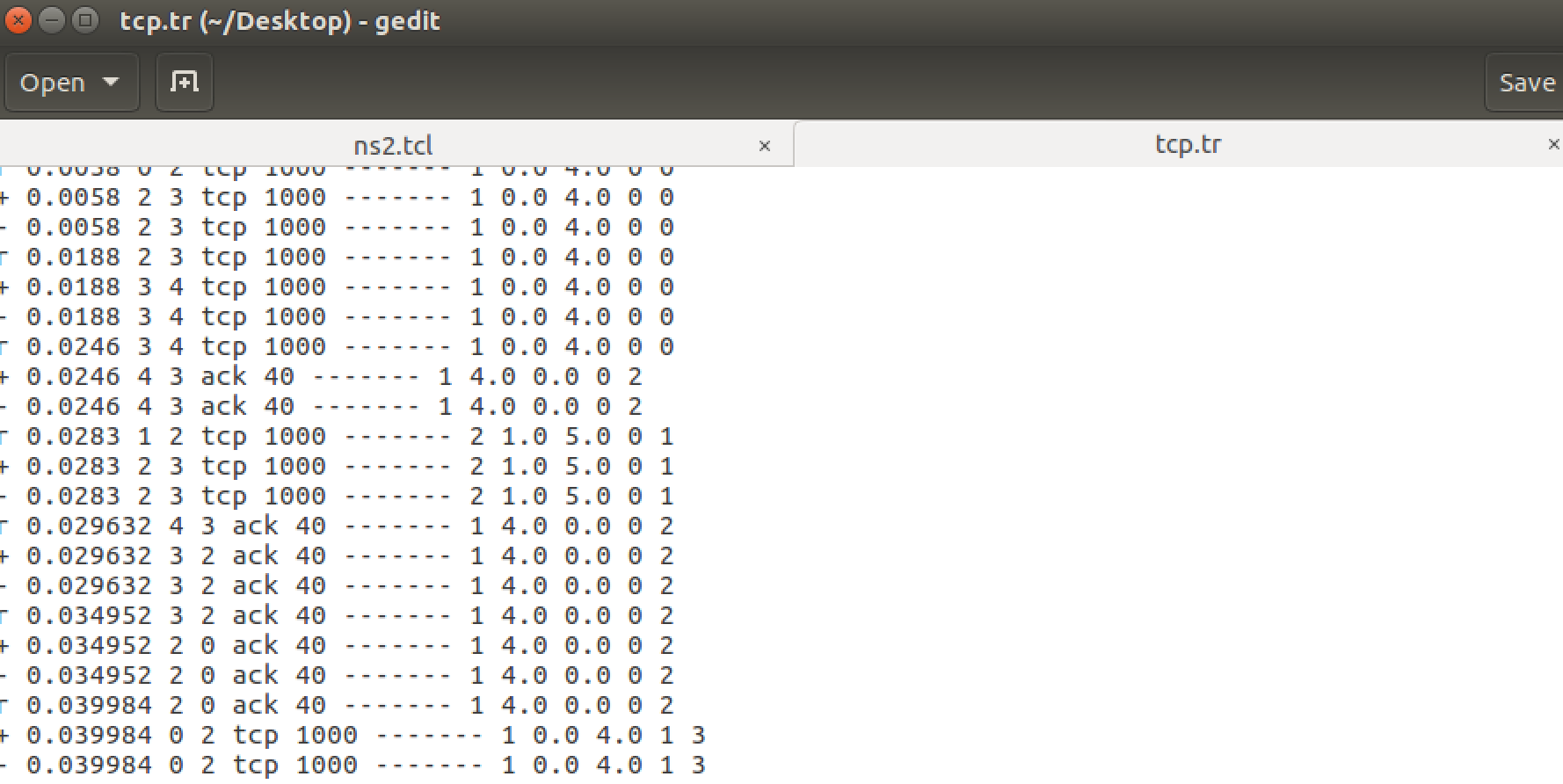
After simulating the code, we get the average throughput for sources and the ratio of their throughputs.



Then a NAM, a network animator tool will appear. This tool shows the physical topology of the network and we can go through the simulation process.

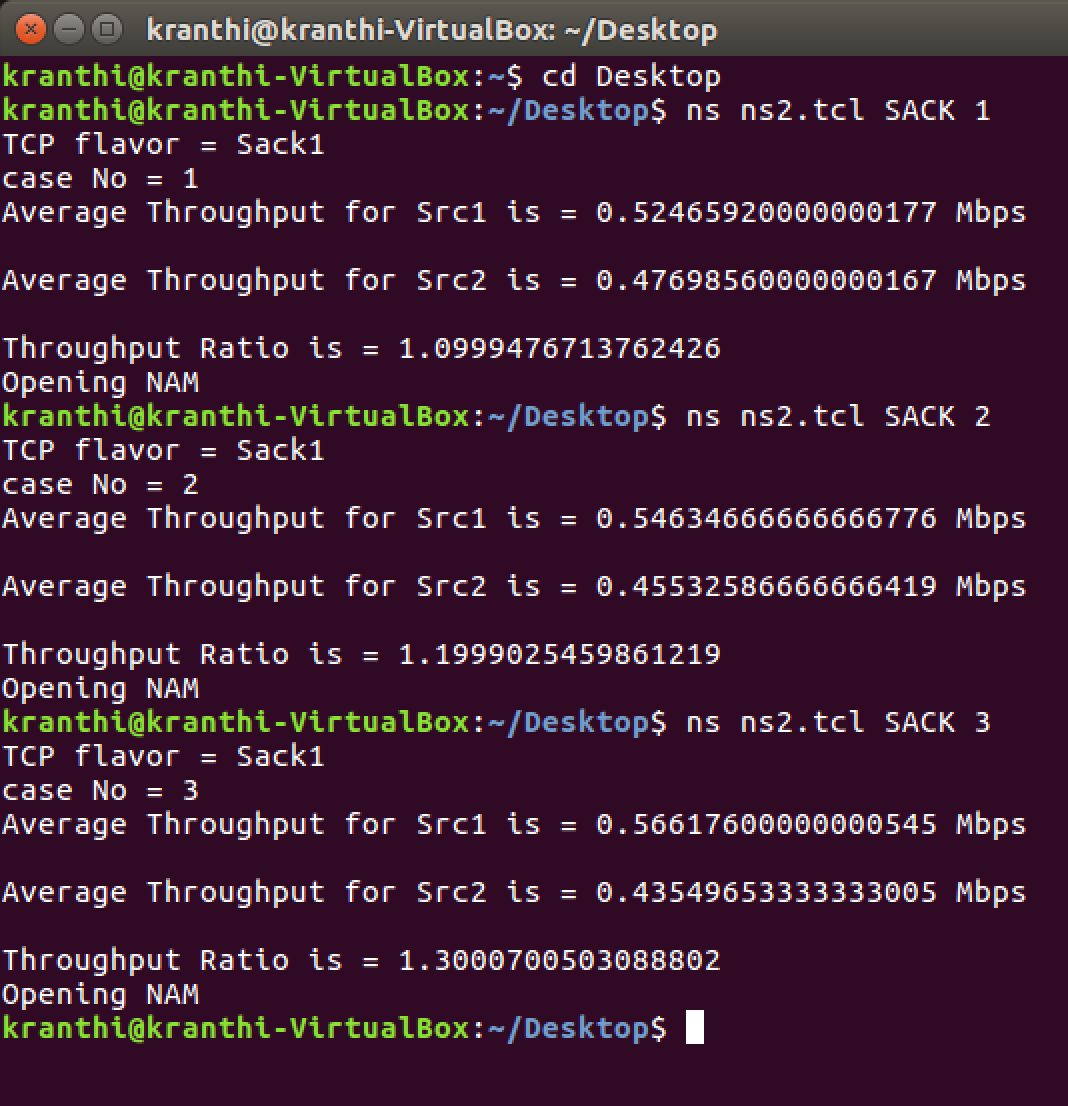


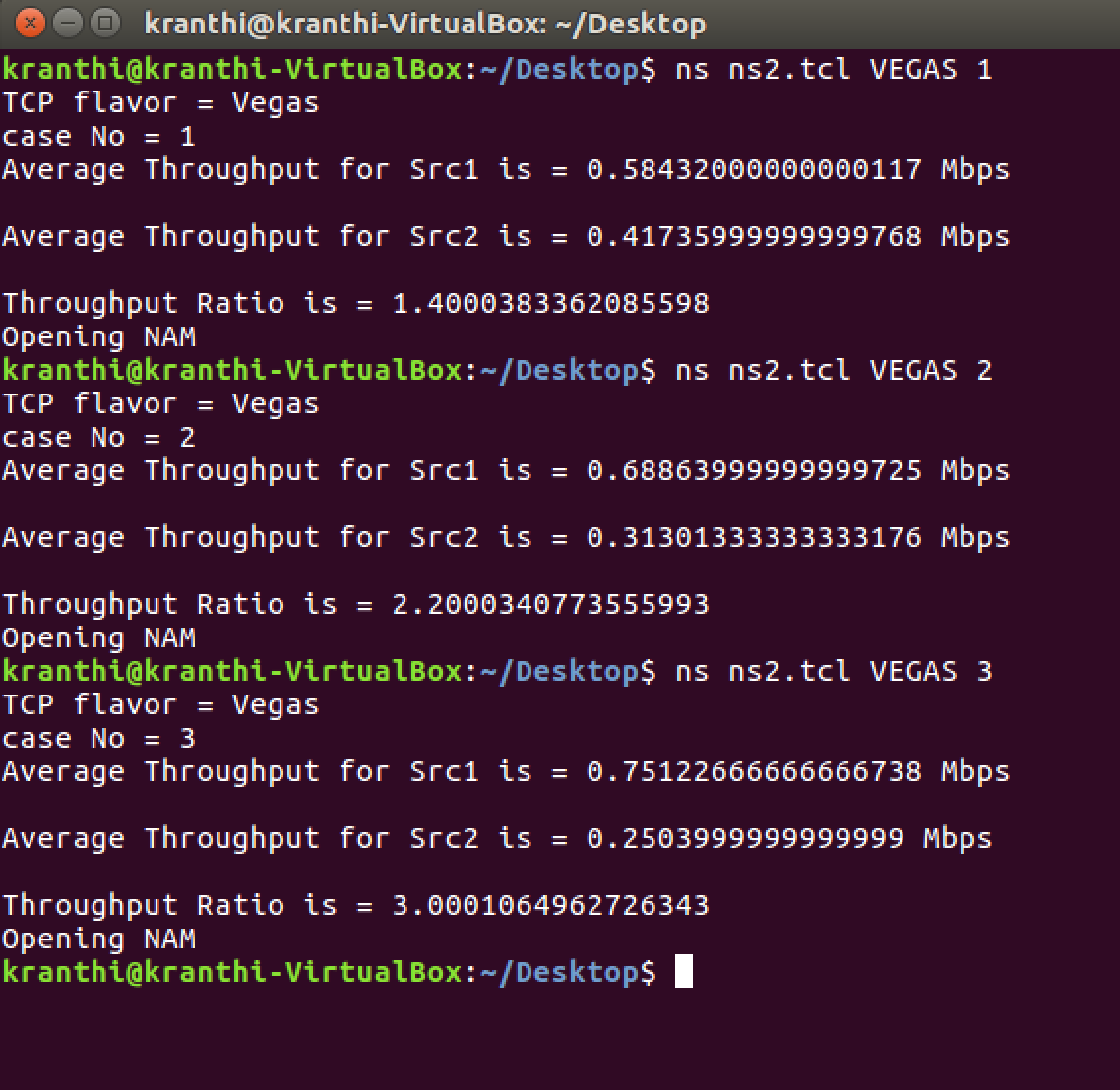
And we can look through details of the data in the trace file.



**Results:**

After execution of the code we get the results shown in the command line for each of the cases as mentioned above. Run the six cases individually.





The throughputs and their ratios are given in the following table (in Mbps):

1. TCP version SACK

|  |  |  |  |
| --- | --- | --- | --- |
| Ratio of RTT | Throughput for TCP SACK | | |
| Src1 | Src2 | Src1/Src2 |
| 1:2 | 0.5246 | 0.4769 | 1.0999 |
| 1:3 | 0.5463 | 0.4553 | 1.1999 |
| 1:4 | 0.5662 | 0.4355 | 1.3000 |

1. TCP version Vegas

|  |  |  |  |
| --- | --- | --- | --- |
| Ratio of RTT | Throughput for TCP Vegas | | |
| Src1 | Src2 | Src1/Src2 |
| 1:2 | 0.5843 | 0.4173 | 1.4000 |
| 1:3 | 0.6886 | 0.3130 | 2.2003 |
| 1:4 | 0.7512 | 0.2504 | 3.0000 |

**Analysis:**

RTT is typically a function of the traffic, congestion, and reliability of the network. A longer RTT means that it takes more time for the packet to reach its destination and for the corresponding ACK to be received. This is the foundation of most congestion control, RTT is easily measured and is used to tell if the network is experiencing heavy traffic, reliability issues, or other problems. Congestion control uses this measurement to determine if the source should keep allowing traffic or to limit the amount being sent. This explains why both TCP flavors have lower throughputs for source 2 as the RTT is increased.

TCP SACK does not react as assertively as TCP Vegas to higher RTT. Notice that TCP SACK at 1:2 has a ratio ~1.1 while TCP Vegas has a ratio of 1.4. This phenomenon is more pronounced at 1:4 in which TCP SACK has a 1.3 ratio as compared to TCP Vegas which is 3. This means that TCP Vegas is allowing 3x the throughput for the lower RTT source. This shows that TCP Vegas will give a higher overall throughput as source 1 will be able to send more packets which take less time to deliver, limiting the traffic from source 2 which will take 4 times as long to deliver each packet. TCP SACK does not discriminate as heavily to longer RTT which may be preferred if one is trying to give equal throughput to all traffic regardless of RTT. Overall TCP Vegas will give a higher throughput by reacting more aggressively to increased RTT.