CS-342 Computer Networks Lab

Assignment #4: Network Simulation using ns-3

App #3: Comparison of Tcp congestion control algorithms

Group #4: Aditya Patidar, 200101009

Advaita Mallik, 200101010

Akshat Mittal, 200101011

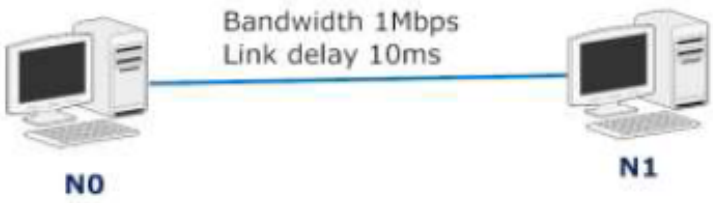
Aman Soni, 200101012

In this assignment, we have to simulate a computer network for the given application using the discrete event network simulator **ns-3**. We downloaded the software from the website https://www.nsnam.org. We will be using **version 3.36** for this assignment.

Application #3:

Create a topology of two nodes N0 and N1 connected by a link of bandwidth 1 Mbps and link delay 10 ms. Use a drop-tail queue at the link. Set the queue size according to bandwidth-delay product. Create a TCP agent (type of the agent specified below) and FTP traffic at N0 destined for N1. Create 5 CBR traffic agents of rate 300 Kbps each at N0 destined for N1. Make appropriate assumptions wherever necessary. The timing of the flows are as follows:

- FTP starts at 0 sec and continues till the end of simulation.
- CBR1 starts at 200 ms and continues till end.
- CBR2 starts at 400 ms and continues till end.
- CBR3 starts at 600 ms and stops at 1200 ms.
- CBR4 starts at 800 ms and stops at 1400 ms.
- CBR5 starts at 1000 ms and stops at 1600 ms.
- Simulation runs for 1800 ms.



Writing Source Code And Simulating:

- After successful installation of ns-3, we have to write the script. For this, we created the file ***source_code.cc*** inside the directory ***~/ns-allinone-3.36.1/ns-3.36.1/scratch*** and wrote the code in it.

- To run the simulation use ***./ns3 run "source_code.cc --tcp_algo=****tcp_algo_name****"*** inside ***~/ns-allinone-3.36.1/ns-3.36.1***, where *tcp_algo_name* is the name of congestion control algorithm we want to use for simulation. The five algorithms we will be comparing are:
  1. Tcp New Reno
  2. Tcp Hybla
  3. Tcp Westwood
  4. Tcp Scalable
  5. Tcp Vegas

  If we do not specify any algorithm, by default, the simulation will be using Tcp New Reno.

- We also used the **Flow Monitor Module** to collect and store the performance data of network protocols from the simulation. After the simulation using *tcp_algo*, the statistics are written in the file ***tcp_algo_stat.txt*** file and complete flow inside ***tcp_algo_stats.flowmon*** XML file.

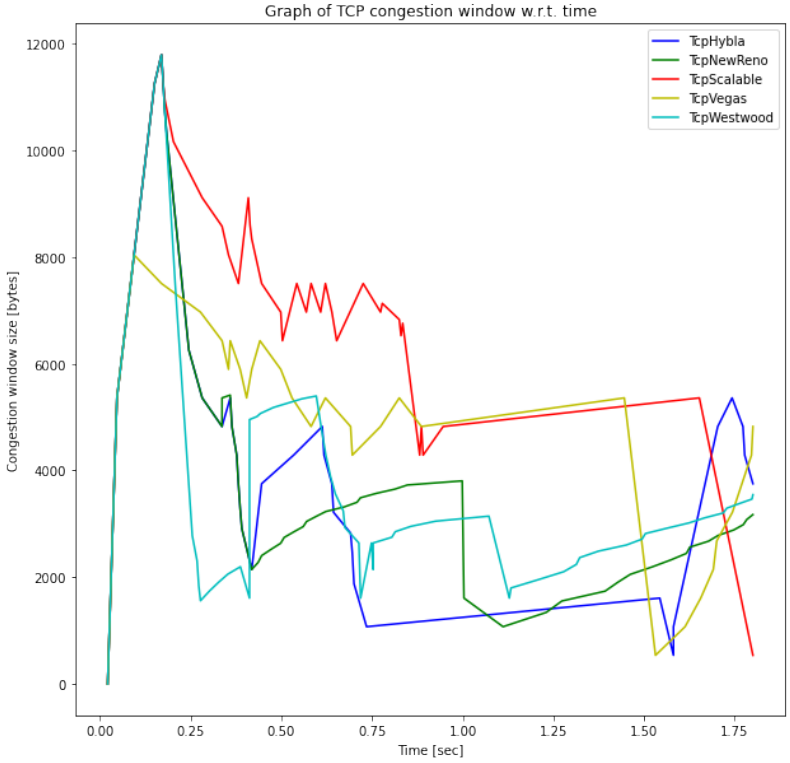
- Further, data regarding size of congestion window, cumulative packet drops and cumulative bytes transferred are also written inside the files ***tcp_algo_cong_wind.txt***, ***tcp_algo_dropped_packets.txt*** and ***tcp_algo_bytes_rcvd.txt*** respectively in the current working directory.

```
akshat@akshat:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run "source_code.cc --tcp_algo=TcpNewReno"

Using TcpNewReno TCP congestion control algorithm.
Flow monitor statistics are written in file TcpNewReno_stat.txt.
Complete flow monitor can be seen in XML file TcpNewReno_stats.flowmon.
akshat@akshat:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run "source_code.cc --tcp_algo=TcpHybla"

Using TcpHybla TCP congestion control algorithm.
Flow monitor statistics are written in file TcpHybla_stat.txt.
Complete flow monitor can be seen in XML file TcpHybla_stats.flowmon.
akshat@akshat:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run "source_code.cc --tcp_algo=TcpWestwood"

Using TcpWestwood TCP congestion control algorithm.
Flow monitor statistics are written in file TcpWestwood_stat.txt.
Complete flow monitor can be seen in XML file TcpWestwood_stats.flowmon.
akshat@akshat:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run "source_code.cc --tcp_algo=TcpScalable"

Using TcpScalable TCP congestion control algorithm.
Flow monitor statistics are written in file TcpScalable_stat.txt.
Complete flow monitor can be seen in XML file TcpScalable_stats.flowmon.
akshat@akshat:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run "source_code.cc --tcp_algo=TcpVegas"

Using TcpVegas TCP congestion control algorithm.
Flow monitor statistics are written in file TcpVegas_stat.txt.
Complete flow monitor can be seen in XML file TcpVegas_stats.flowmon.
```

- We run simulation for all five algorithms and collected data to plot graphs. The graphs are plotted using python code inside ***plot.py*** file using jupyter notebook.

Question 1:

Plot graph(s) of Tcp congestion window w.r.t. time for 5 Tcp congestion control algorithm implementations and describe their behavior.



- **Tcp New Reno (Green):** Congestion window size starts from 1 for every algorithm (i.e., slow start), increases exponentially with time and continues to increase until a packet drop occurs. When it reaches a threshold value, it changes to congestion avoidance phase. We can see the green line going near to 11000 and afterwards may be due to some packet drop, it reduces to half around 5000. Again on incurring packet drops, it halves the size of congestion window and then again increases as per the algorithm based on the way loss has occurred and continues.

- **Tcp Vegas (Yellow):** This started similar to Tcp New Reno but did not reach as high as others, going only till around 8000. Tcp Vegas works by analyzing RTT values to alter its congestion window size. If observed RTT becomes large, it recognizes that the network begins to be congested and throttles the window size. On the other hand, if observed RTT becomes small, the sender host of Tcp Vegas determines that the network is relieved from the congestion and increases the window size again. We can also see that the Tcp Vegas maintains better consistency in the congestion window size unlike others which show much variations.

- **Tcp Hybla (Blue):** It is very much identical to the Tcp New Reno upto around 0.35 sec. After incurring some drops, we can see that Tcp Hybla increases its congestion window size much faster than Tcp New Reno and also decreases suddenly. So we can say that Tcp Hybla increases window size by much higher rate, which shows a difference in behavior in the second half compared to Tcp New Reno.

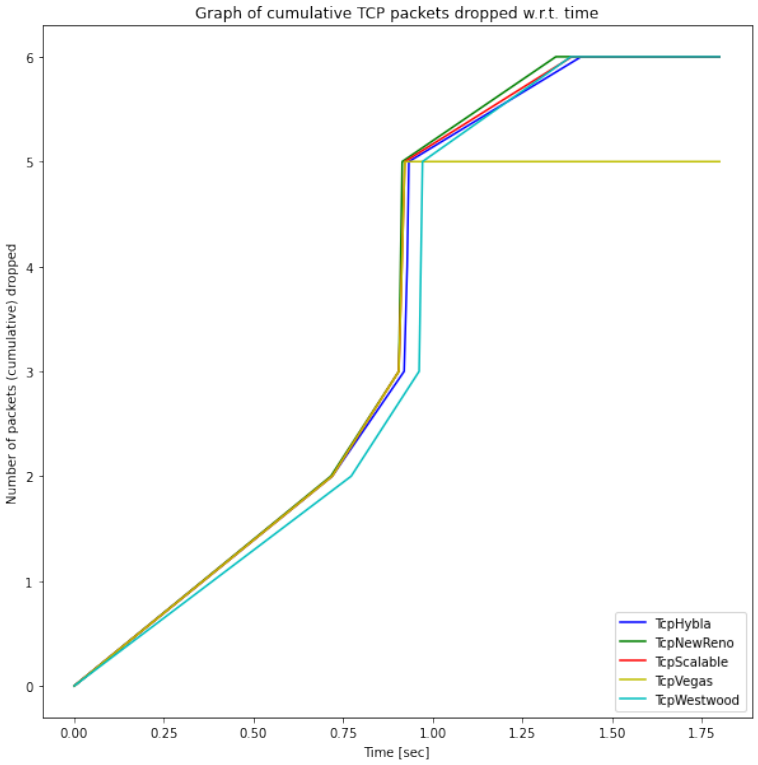
- **Tcp Scalable (Red):** It started like all others from 1 but maintains much higher congestion window size compared to all others and remains consistently on top. It does not reduce the window size

drastically but does it gradually. Its name “Scalable” itself signifies that it provides better scalability thus providing better throughput over large networks.

- **Tcp Westwood (Cyan)**: Initially, it is almost similar to Tcp New Reno but decreases the window size most drastically. Later, the window size goes up and down. Theoretically, Tcp Westwood is a modification of Tcp New Reno and hence the initial phase are similar. Tcp Westwood relies on mining the ACK stream for information on the sender side to help it better set the congestion control parameters like threshold etc.

Question 2:

Draw a graph showing cumulative Tcp packets dropped w.r.t. time comparing 5 Tcp congestion control algorithm implementations.



- **Tcp New Reno (Green):** Initially no packets are dropped until, around 0.25s, this is the slow start period. Around 0.4s there is a packet drop and we can see multiple drops as we proceed. Till the end, we can see a total of 6 packets are dropped.

- **Tcp Hybla (Blue):** It runs almost the same as Tcp New Reno with a slight variation in timings of packet drops. Again, we have total of 6 packet drops.

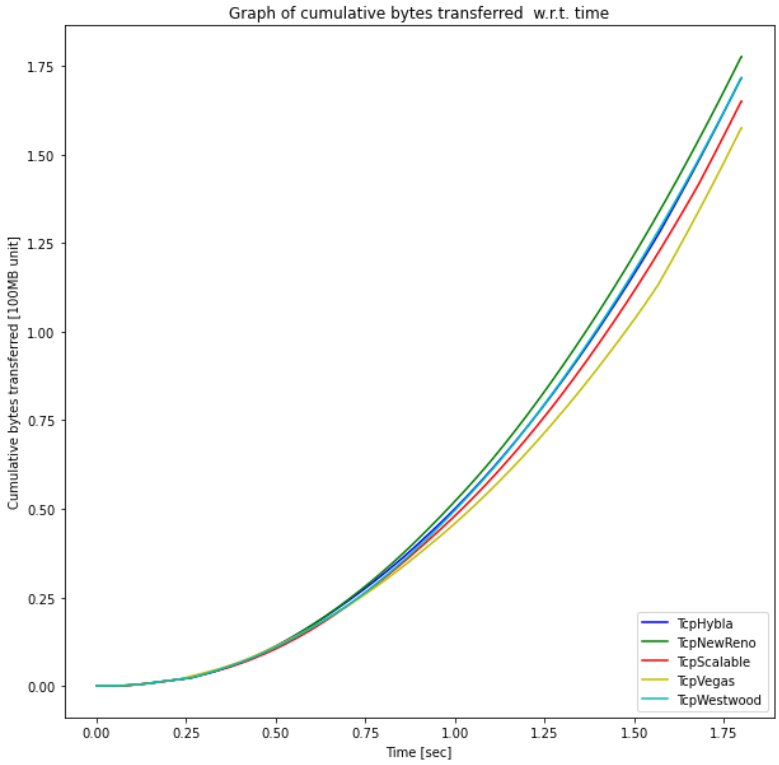
- **Tcp Vegas (Yellow):** This has lowest number of packet drops compared to others with a total of 5 till end. As we know it remains consistent and we can infer that it takes better control measure much before facing packet loss than others.

- **Tcp Scalable (Red):** As we already saw, here the congestion window size does not drop drastically but still we see similar variation of number of packets dropped.

- **Tcp Westwood (Cyan):** We can see that the packet drops most lately in this case compared to others. This gives us the impression that in the long run it will drop the least number of packets. But the behavior is almost the same as New Reno.

Question 3:

Draw a graph showing cumulative bytes transferred w.r.t. time comparing 5 Tcp congestion control algorithm implementations.



- Initially, all TCPs are starting similarly as window size increases gradually from 1. Whenever the congestion window size is increasing, the slope is increasing for all.

- We can see in our simulation that the Tcp New Reno (Green) reaches the highest peak compared to others.

- It can also be seen that Tcp Hybla (Blue) and Tcp Westwood (Cyan) are having almost the same slope throughout the course.

- Tcp Vegas (Yellow) has the least slope and thus smallest peak.

Submission:

The submitted zip folder *Group_4.zip* contains following:

1. Five folders containing the data files corresponding to each Tcp congestion control algorithm based on which the plots were constructed.
2. ***source_code.cc***: Script file for the simulation.
3. ***plot.py***: Python script used to construct the 3 plots.
4. The three graphs constructed (as image files).
5. ***Report.pdf***