COL334 ASSIGNMENT 3 REPORT

By Khyateeswar Naidu Nalla

2019CS10376

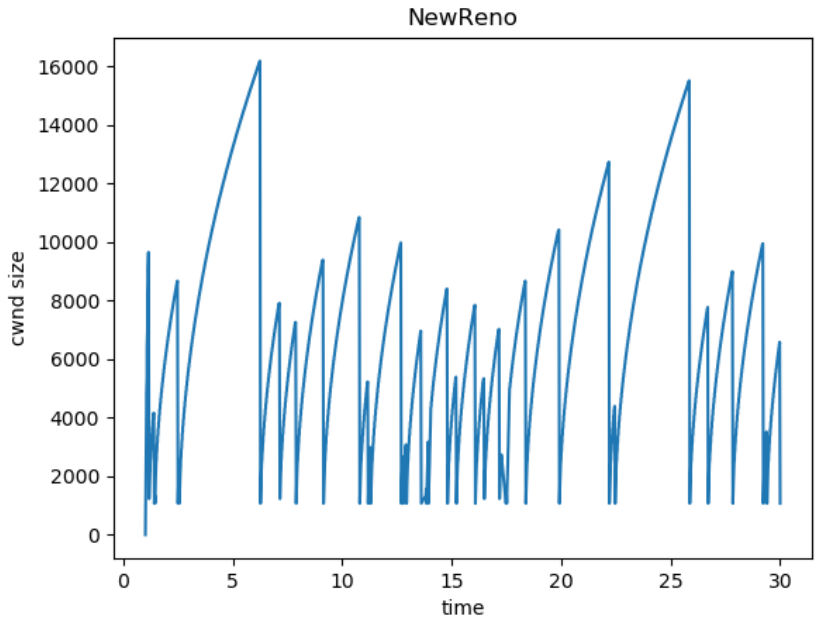
Part 1

Just changing the protocol name in the given line each code we get to use the protocol

Config::SetDefault ("ns3::TcpL4Protocol::SocketType", StringValue ("ns3::TcpNewReno"));

I plotted the graph using matplotlib and the code is in “plot.py”

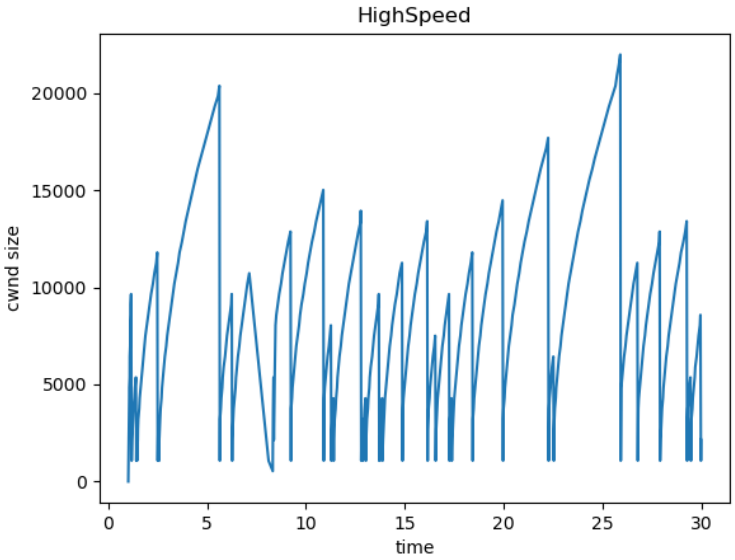
a) NewReno



No of Dropped packets = 38

The code for this case is present in the First_1.cc

b) HighSpeed

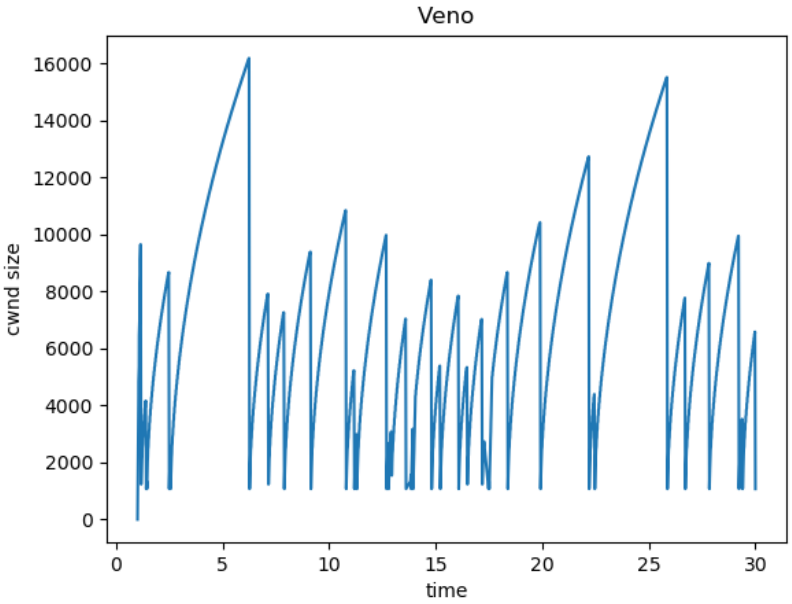


No of Dropped packets = 38

The code of this case is present in First_2.cc

Instead of TcpNewreno we write TcpHighSpeed in the equation on the top

c) Veno

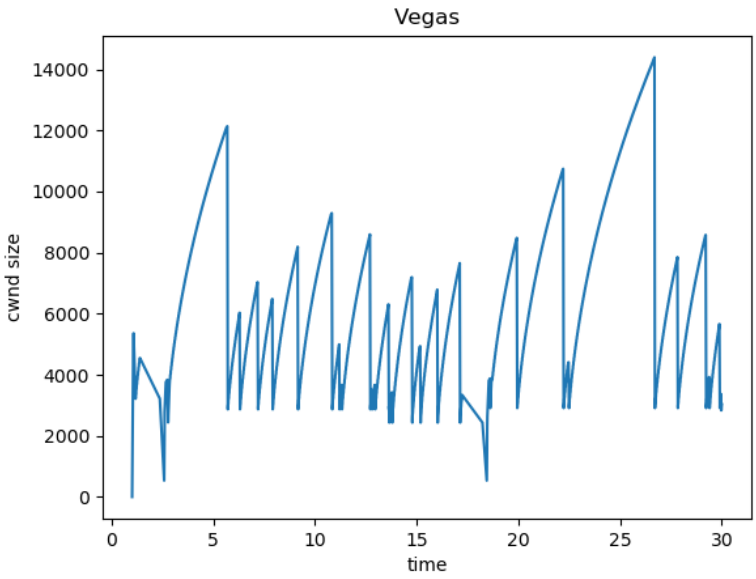


No of Dropped packets = 38

The code of this case is present in First_3.cc

Instead of TcpNewreno we write TcpVeno in the equation on the top

d) Vegas



No of Dropped packets = 39

The code of this case is present in First_4.cc

Instead of TcpNewreno we write TcpVegas in the equation on the top

Trends Observed:

1.Having same Application datarate , Channel Datarate , Propagation Delay ,RTT and ErrorRateModel we get similar graphs because they also have similar congestion avoidance models so they have similar peaks and similar thresholds.

2.Therefore they have packet loss due to error rate of 0.00001 as they have similar cwnd size they have same loss of packets in all cases.

3.Highspeed have a higher peak among vegas,veno and newreno because it has a faster congestion avoidance model.

Part 2

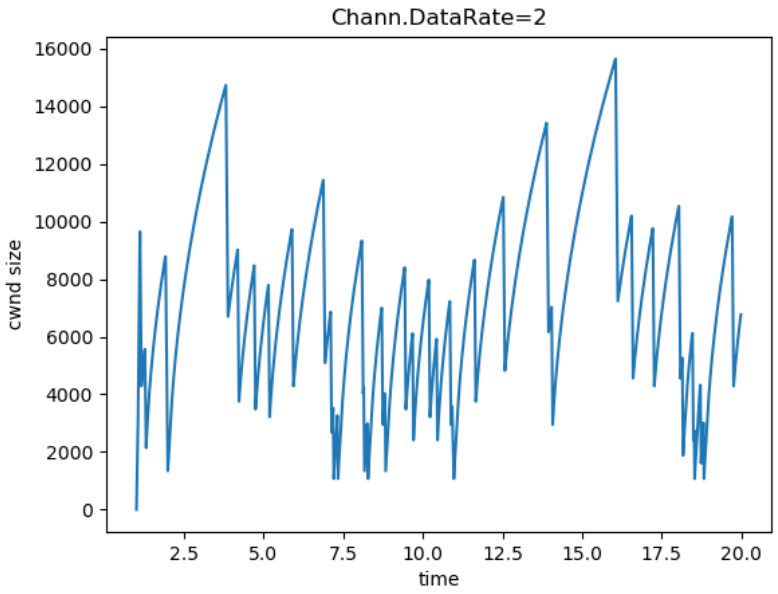
app->Setup (ns3TcpSocket, sinkAddress, 3000, 100000, DataRate ("2Mbps"));//To change Application DataRate

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("2Mbps"));

//To change Channel DataRate

Application rate = 2Mbps & Varying Channel Bitrate

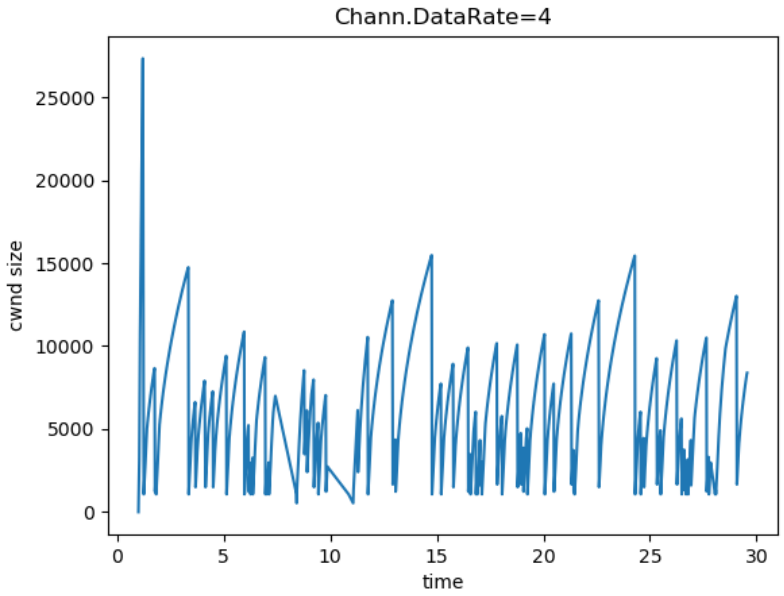
a) Channel Bitrate = 2



The code for this case is present in Second_1.cc

We just change the Channel datarate to 2 while fixing the Application datarate to 2

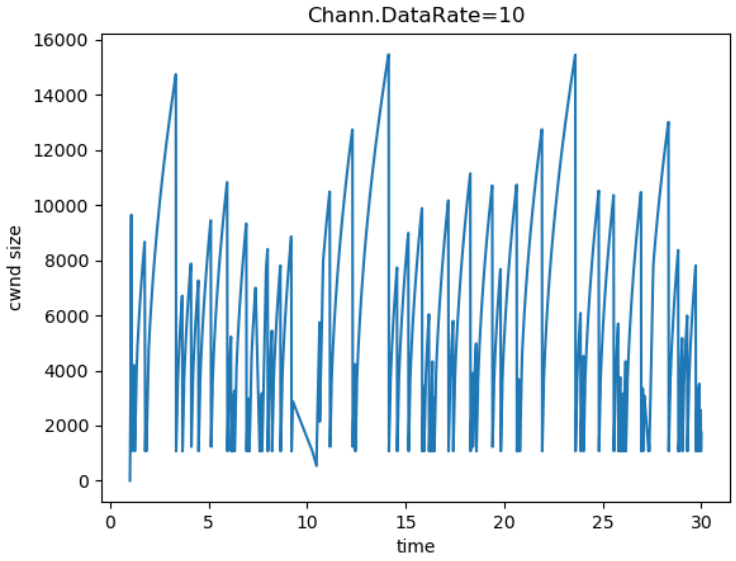
b) Channel Bitrate = 4



The code for this case is present in Second_2.cc

We just change the Channel datarate to 4 while fixing the Application datarate to 2

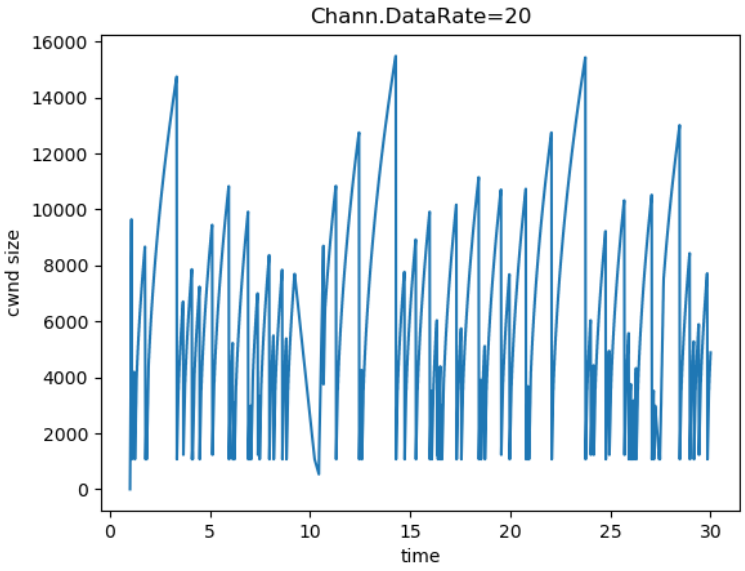
c) Channel Bitrate = 10



The code for this case is present in Second_3.cc

We just change the Channel datarate to 10 while fixing the Application datarate to 2

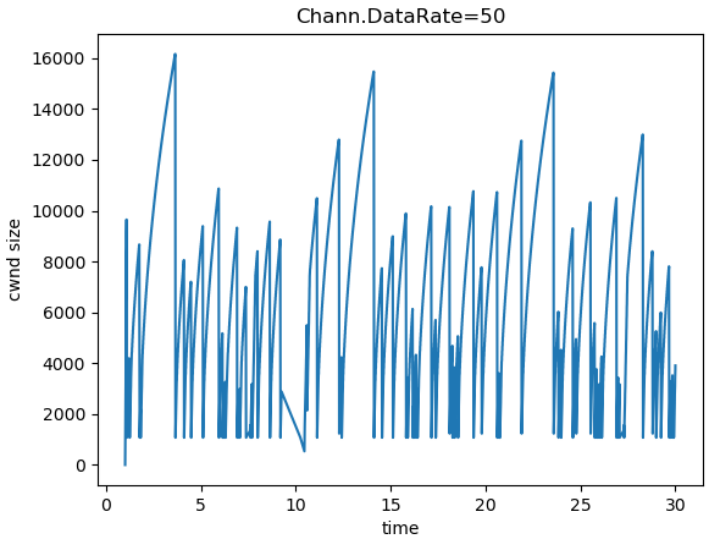
d) Channel Bitrate = 20



The code for this case is present in Second_4.cc

We just change the Channel datarate to 20 while fixing the Application datarate to 2

e) Channel Bitrate = 50



The code for this case is present in Second_5.cc
We just change the Channel datarate to 50 while fixing the Application datarate to 2

Trends Observed:

1.Increasing Channel data rate means decreasing RTT time therefore as cwnd increases every RTT therefore more change in cwnd with increase in channel data rate

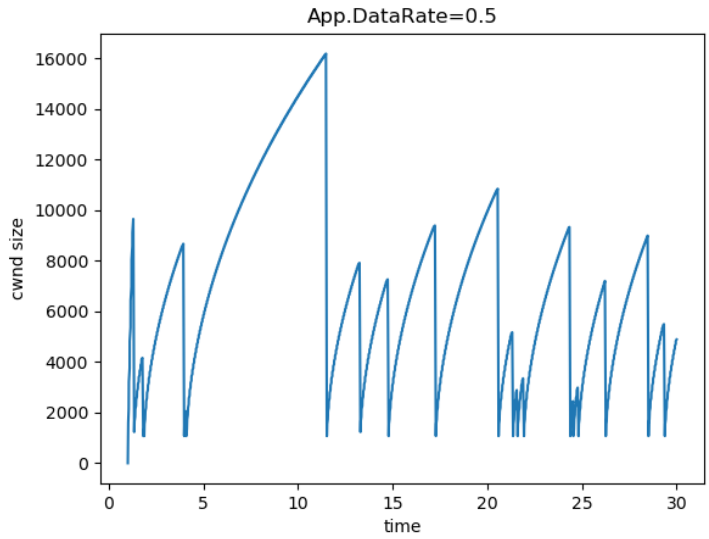
2. We know that having higher no of packets transmitted higher the chance of packet loss from the RateErrorModel so having higher number channel rate means higher packet loss so have a dense graph due to faster increasing cwnd and higher packet losses.

3.Ssthreshold of Newreno = max(2*536,bytes in flight/2)

Therefore as bytes in flight proportional to Application data Rate and Inversely proportional to Channel rate so having low Channel Data Rate have higher Ssthreshold otherwise it is 1072

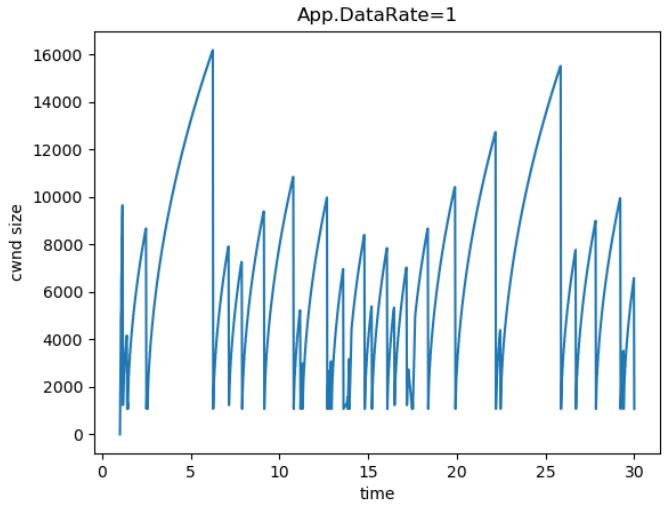
Channel BitRate = 6Mbps & varying Application Datarate

a) Application Bitrate = 0.5



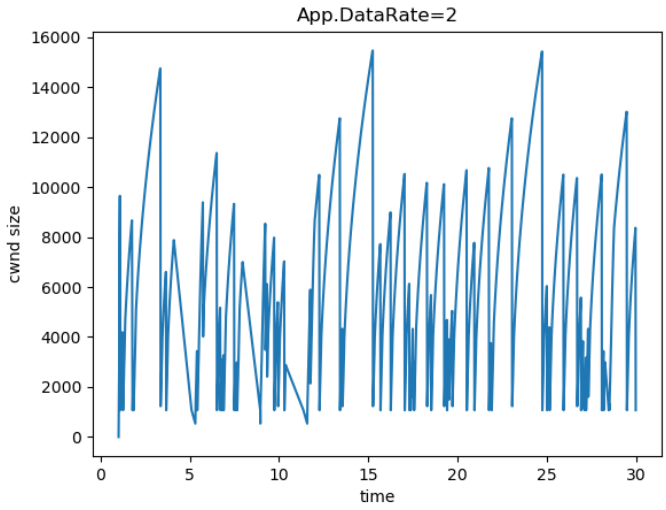
The code for this case is present in Second_6.cc
We just change the Application datarate to 0.5 while fixing the Channel datarate to 6

b) Application Bitrate = 1



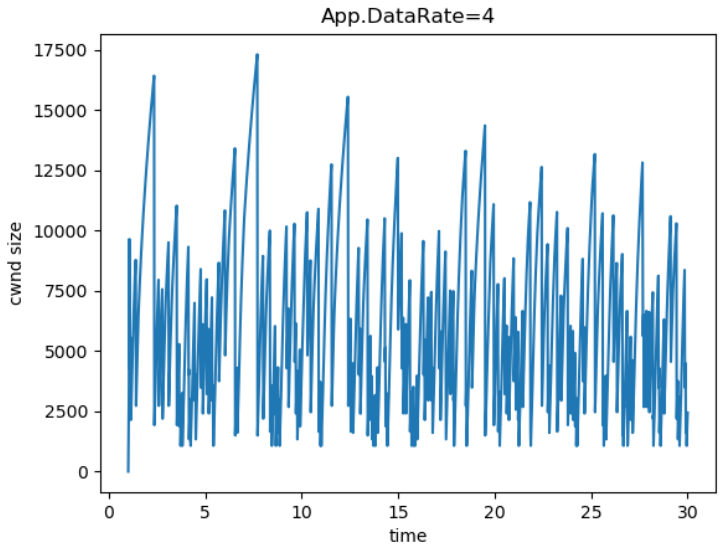
The code for this case is present in Second_7.cc
We just change the Application datarate to 1 while fixing the Channel datarate to 6

c) Application Bitrate = 2



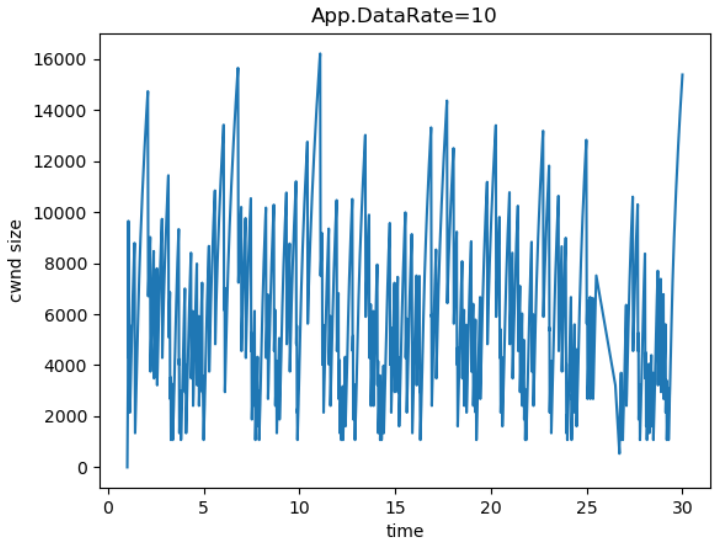
The code for this case is present in Second_8.cc
We just change the Application datarate to 2 while fixing the Channel datarate to 6

d) Application Bitrate = 4



The code for this case is present in Second_9.cc
We just change the Application datarate to 4 while fixing the Channel datarate to 6

e) Application Bitrate = 10



The code for this case is present in Second_10.cc
We just change the Application datarate to 10 while fixing the Channel datarate to 6

Trends Observed:

1.Similarly increases application Rate increases the speed of the packet transfer so RTT decreases so with increasing in Application Data rate So having low RTT means fast increasing cwnd so higher number of packets transferred so dense graph and also higher chances of packet loss so higher no of packet drops.

2. Ssthreshold in newreno = max(1072,bytes in flight/2)

We know that bytes in flight is proportional to application data rate so having higher Application Data Rate means have higher Ssthreshold so having higher cwnd in case of packet drops in higher Application Data Rate

Part 3

We construct new congestion protocol similar to Tcpvegas i.e importing the parent class tcp_congestion_ops from that file and then override slow start , congestion avoidance and some other functions.

```
uint32_t TcpNewRenoCSE::SlowStart (Ptr<TcpSocketState> tcb, uint32_t segmentsAcked)
{
  NS_LOG_FUNCTION (this << tcb << segmentsAcked);

  if (segmentsAcked >= 1)
    {
      double adder = (std::pow(static_cast<double>(tcb->m_segmentSize),1.9))/(static_cast<double>(tcb->m_cWnd));
      adder = std::max (1.0, adder);
      tcb->m_cWnd += static_cast<uint32_t>(adder);
      NS_LOG_INFO ("In SlowStart, updated to cwnd " << tcb->m_cWnd << " ssthresh " << tcb->m_ssThresh);
      return segmentsAcked - 1;
    }

  return 0;
}
```

```
void
TcpNewRenoCSE::CongestionAvoidance (Ptr<TcpSocketState> tcb, uint32_t segmentsAcked)
{
  NS_LOG_FUNCTION (this << tcb << segmentsAcked);

  if (segmentsAcked > 0)
    {
      double adder = 0.5*(static_cast<double> (tcb->m_segmentSize)) ;
      adder = std::max (1.0, adder);
      tcb->m_cWnd += static_cast<uint32_t> (adder);
      NS_LOG_INFO ("In CongAvoid, updated to cwnd " << tcb->m_cWnd <<
                   " ssthresh " << tcb->m_ssThresh);
    }
}
```

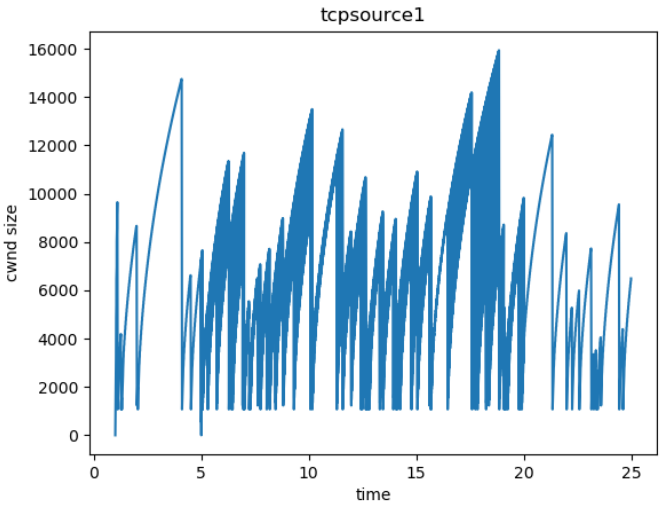
We add the header and cc files in the wscript and we compile the build and then run the new congestion protocol.

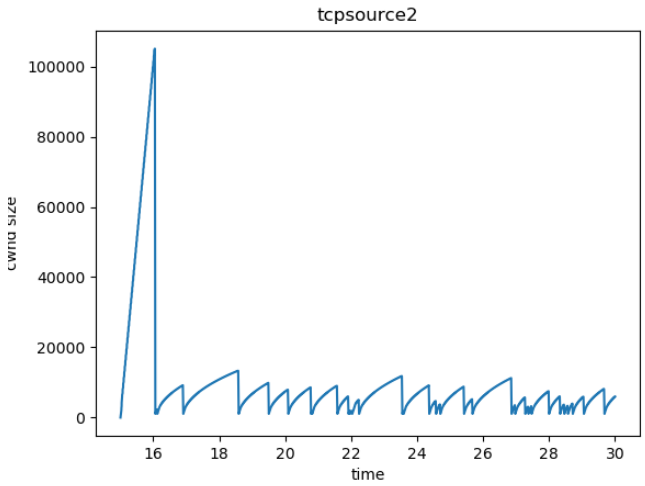
Next we then create 3 nodes and use these to assign devices , ipv4addresses,pointtopoint connections and ErrorRateModels.

Then we create 3 sender sockets of different congestion protocol , 3 reciever sockets and 3 applications for running the connection.

We then assign different congestion protocols for different configurstions and run the cases

Configuration 1



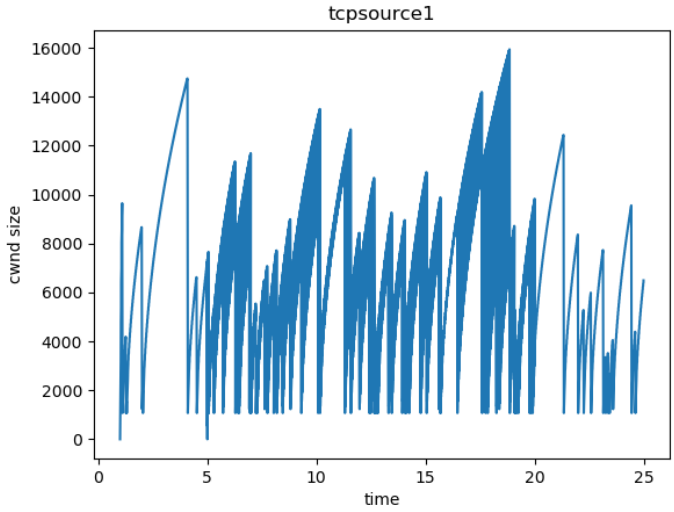


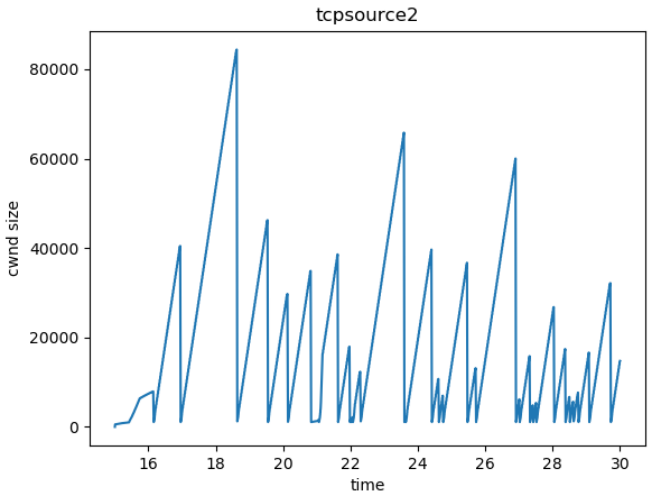
No of Dropped Packets = 113

The code of this configuration is present in Third_1.cc

All the three sender sockets are set to the same congestion protocol Newreno.

Configuration 2



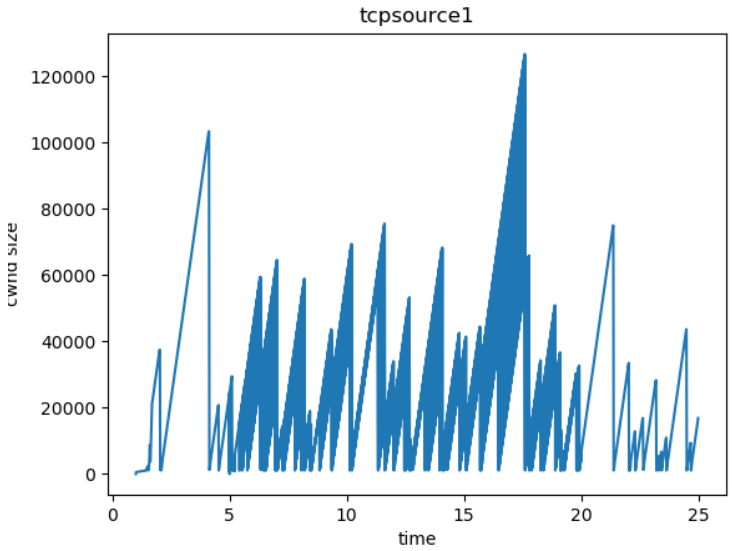


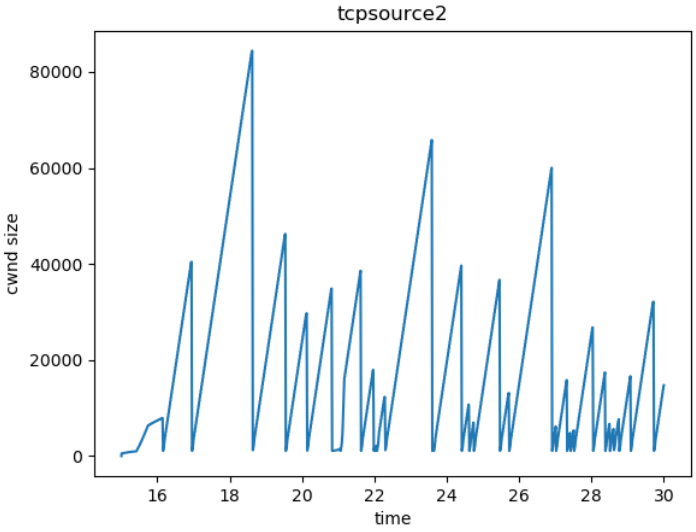
No of Dropped packets = 112

The code of this configuration is present in Third_2.cc

Senders of Connection 1&2 are set to Newreno but the sender of Connection 3 is set to Newrenocse Congestion protocol

Configuration 3





No of Dropped packets = 110

The code of this configuration is present in Third_3.cc

All the three sender sockets are set to the same congestion protocol Newrenocse

Trends Observed:

1.Congestion avoidance functions below

Newreno: cwnd += (536^2)/cwnd

Newrenocse: cwnd+= 268

2.Newreno and Newrenocse have different congestion avoidance functions but same RTT value because of same Application Data rate , Channel data rate and propagation delay . Newreno saturates while increasing but newrenocse have a linear increase so newrenocse can get to high peaks compared to newreno because it takes less packets to be transferred to attain the peak before a packet drop happens.

3.Newrenocse have slightly more packet drops because it attains peaks easily therefore more packets transferred so high probability of packet drops compared to newreno but they are only slightly more because it depends on the area of the graph not only the peaks to get the no of packets transmitted.