INDIAN INSTITUTE OF TECHNOLOGY, DELHI

ASSIGNMENT REPORT

COL334 : Assignment 3

APAR AHUJA | ENTRY NO. 2019CS10465

Course - COL334 | Prof. Abhijnan Chakraborty

Compiled on October 26, 2021

Contents

Chapter 1

Part 1

1.1 Running Code

```
1. Place the First.cc file inside the directory ns-allinone-3.29/ns-3.29/scratch
2. Commands to analyze different protocols -
    a. TcpNewReno
        ./waf --run "scratch/First -useProtocol=1"
    b. TcpHighSpeed
        ./waf --run "scratch/First -useProtocol=2"
    c. TcpVeno
        ./waf --run "scratch/First -useProtocol=3"
    d. TcpVegas
        ./waf --run "scratch/First -useProtocol=4"
3. Command to plot different protocols -
    gnuplot plot1.plt
```

1.2 Congestion Window

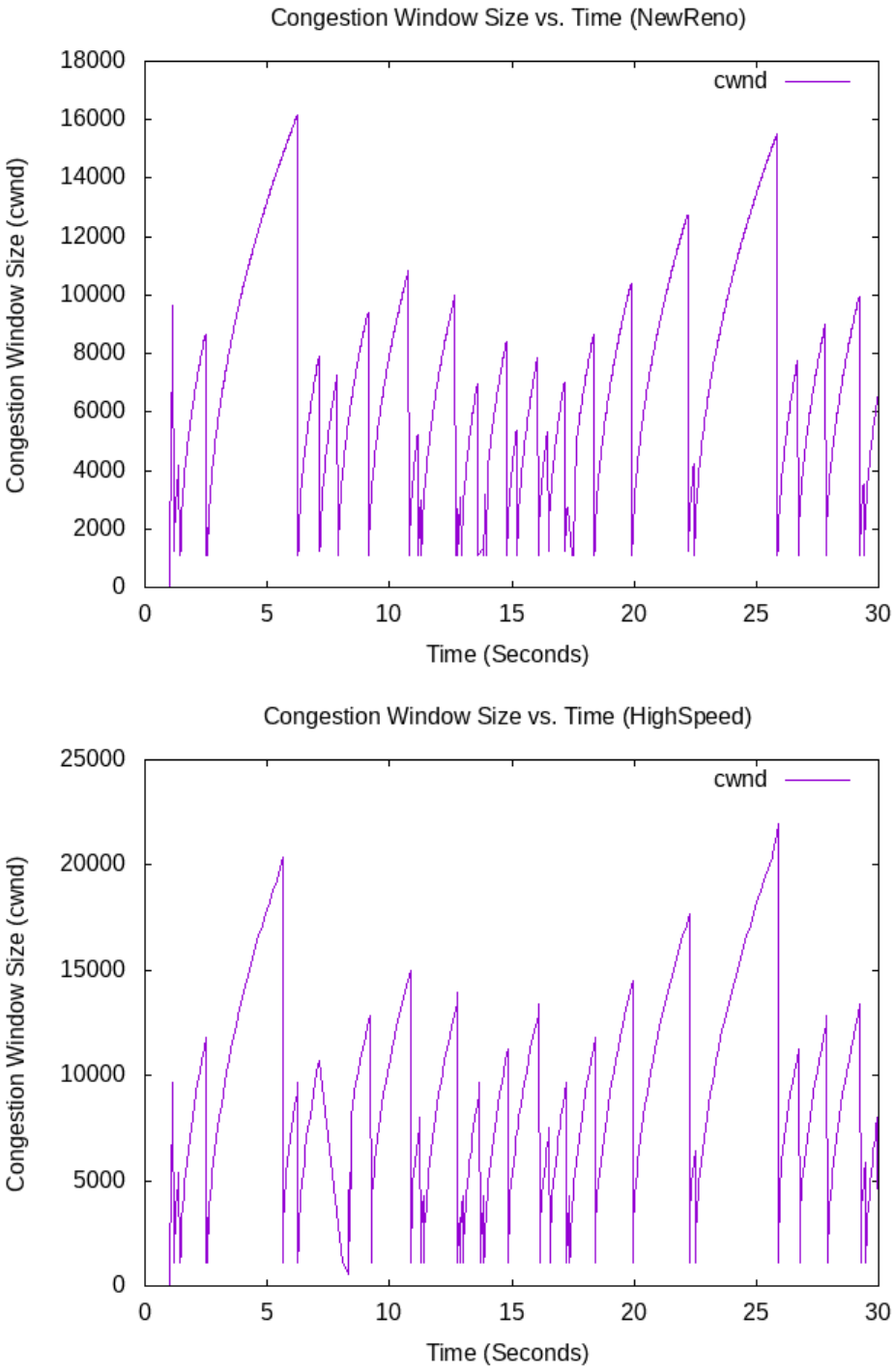

Figure 1.1: Plots for TcpNewReno and TcpHighSpeed

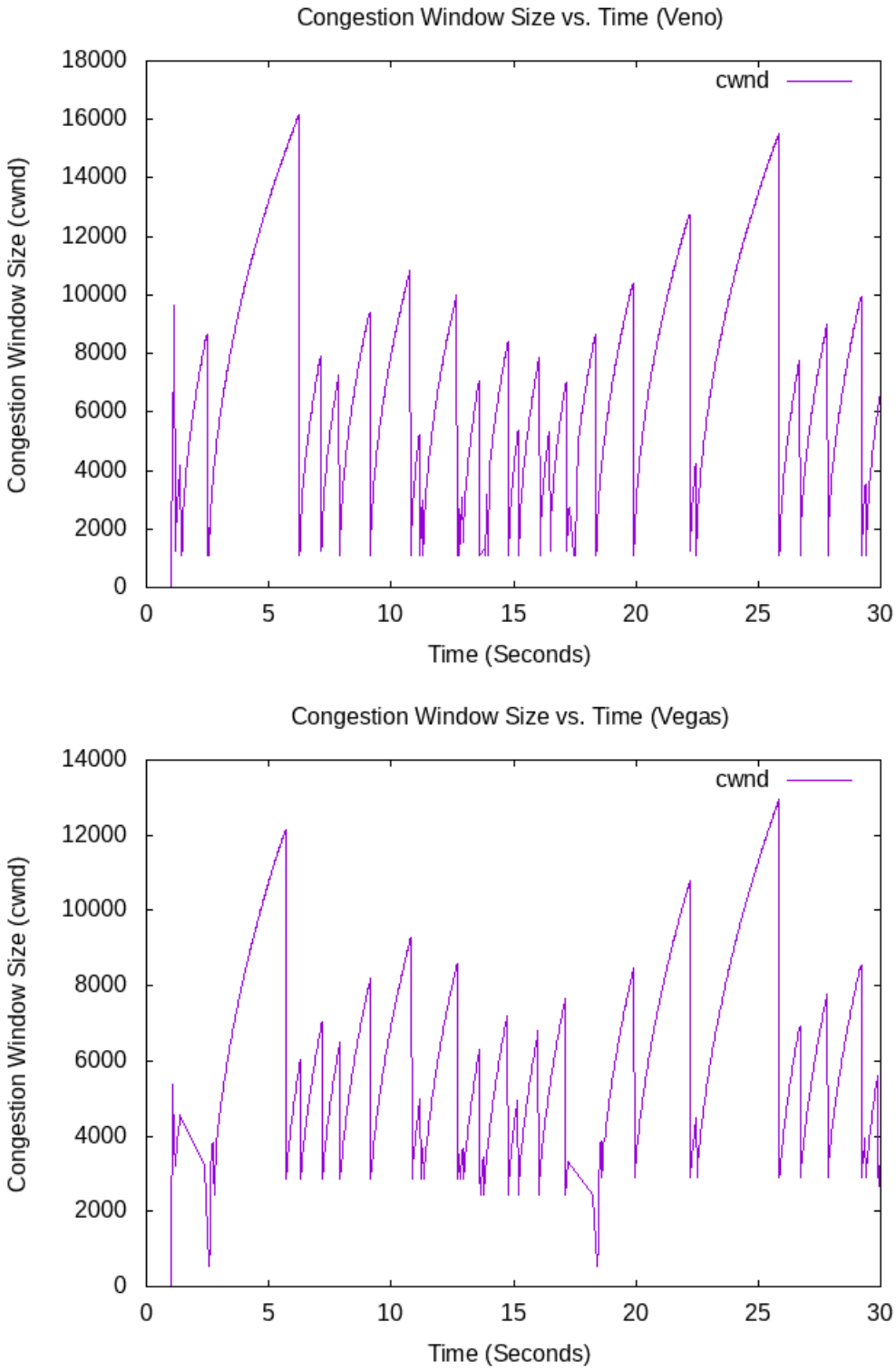

Figure 1.2: Plots for TcpVeno and TcpVegas

1.3 Packet Drops

a. TcpNewReno
Number of packet drops = 38
Number of packets sent by application = 1209
b. TcpHighSpeed
Number of packet drops = 38
Number of packets sent by application = 1209
c. TcpVeno
Number of packet drops = 38
Number of packets sent by application = 1209
d. TcpVegas
Number of packet drops = 39
Number of packets sent by application = 1209

Average percent packet drops assuming all packets are sent within the simulation period $\approx 3.2\%$
TcpNewReno, TcpHighSpeed and Tcp Veno have the same number of packet drops. The 3 protocols work equally well in this case. TcpVegas gives poor performance with 1 extra packet drop as compared to the other protocols.

1.4 Analysis

a. TcpNewReno
Congestion Avoidance: cwnd + = max(1, (segment-size)^2/cwnd)
Slow Start: cwnd + = segment-size
Incase of congestion event: cwnd = 1/2 * cwnd
b. TcpHighSpeed
Congestion Avoidance: cwnd + = a(cwnd) / cwnd
Slow Start: cwnd + = segment-size
Incase of congestion event: cwnd = (1 − b(cwnd)) * cwnd
HighSpeed makes the cWnd grow faster during the probing phases and accelerates the cWnd recovery from losses. This behavior is executed only when the window grows beyond a certain threshold. For small congestion window it behaves like standard TCP with a() = 1 and b() = 1/2
c. TcpVeno
N = Number of packets accumulated at the bottleneck queue
N = diff * BaseRTT (diff is defined in Vegas below.)
Veno makes decision on cwnd modification based on the calculated N and its threshold beta.
d. TcpVegas
actual throughput = cwnd / RTT
expected throughput = cwnd / BaseRTT
diff = expected − actual
To avoid congestion, Vegas linearly increases/decreases its congestion window to ensure the diff value falls between the two thresholds, alpha and beta. diff and another threshold, gamma, are used to determine when Vegas should change from its slow-start mode to linear increase/decrease.

Observations:
1. TcpHighSpeed has highest cwnd peak when compared to other protocols. It's average cwnd is bigger than other protocols. Even the rate of increase of cwnd is high when compared to other protocols. Hence, it is useful for connections that have large congestion windows.
2. TcpVegas has the smallest cwnd peak and a low average cwnd size as compared to other variants. Thus, it tries to keep the cwnd to a minimum.
3. The graphs for TcpVeno and TcpNewReno are identical. Now, TcpVeno enchaces TcpReno to handle random packet losses in wireless access networks. This assignment however has a direct wired network with no router and hence the graph is same.
4. Timeout in TcpHighSpeed occurs at around 8 secs and it enters slow start phase. Similarly timeout occurs in TcpVegas at around 3 secs and 18 secs where the cwnd drops to 1 and the protocol enters slow start phase.
5. The first slow start phase peaks at around 10,000 for the first three protocols whereas it peaks at approximately 6000 in TcpVegas.

Chapter 2

Part 2

2.1 Running Code

```
1. Place the Second.cc file inside the directory ns-allinone-3.29/ns-3.29/scratch
2. Commands to analyze different rates -
    a. Channel Data Rate = 2 Mbps
        ./waf --run "scratch/Second -useRate=1"
    b. Channel Data Rate = 4 Mbps
        ./waf --run "scratch/Second -useRate=2"
    c. Channel Data Rate = 10 Mbps
        ./waf --run "scratch/Second -useRate=3"
    d. Channel Data Rate = 20 Mbps
        ./waf --run "scratch/Second -useRate=4"
    e. Channel Data Rate = 50 Mbps
        ./waf --run "scratch/Second -useRate=5"
    f. Application Data Rate = 0.5 Mbps
        ./waf --run "scratch/Second -useRate=6"
    g. Application Data Rate = 1 Mbps
        ./waf --run "scratch/Second -useRate=7"
    h. Application Data Rate = 2 Mbps
        ./waf --run "scratch/Second -useRate=8"
    i. Application Data Rate = 4 Mbps
        ./waf --run "scratch/Second -useRate=9"
    j. Application Data Rate = 10 Mbps
        ./waf --run "scratch/Second -useRate=10"
3. Command to plot different protocols -
    gnuplot plot2.plt
```

2.2 Congestion Window

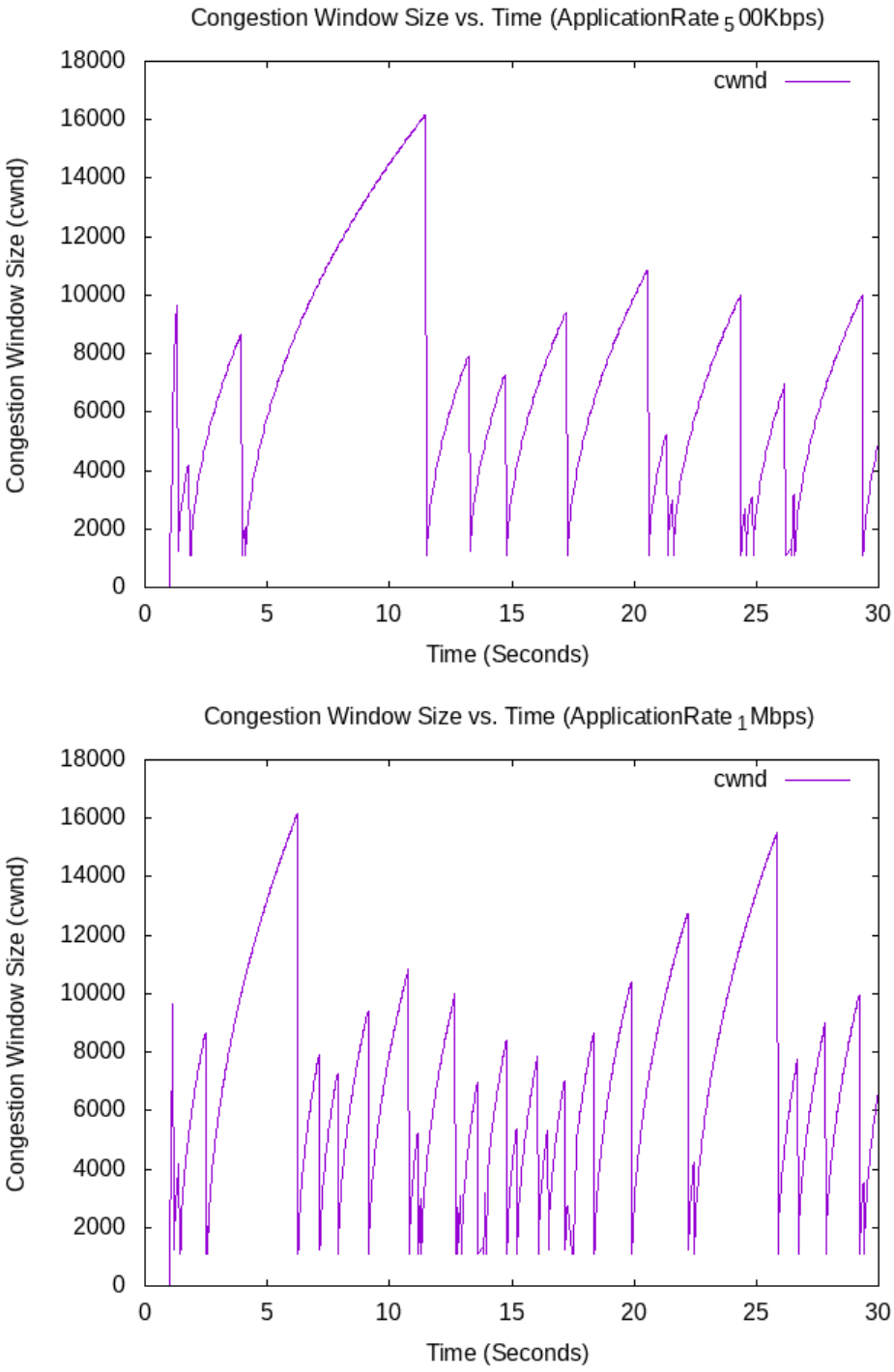

Figure 2.1: Plots for different Application Rates

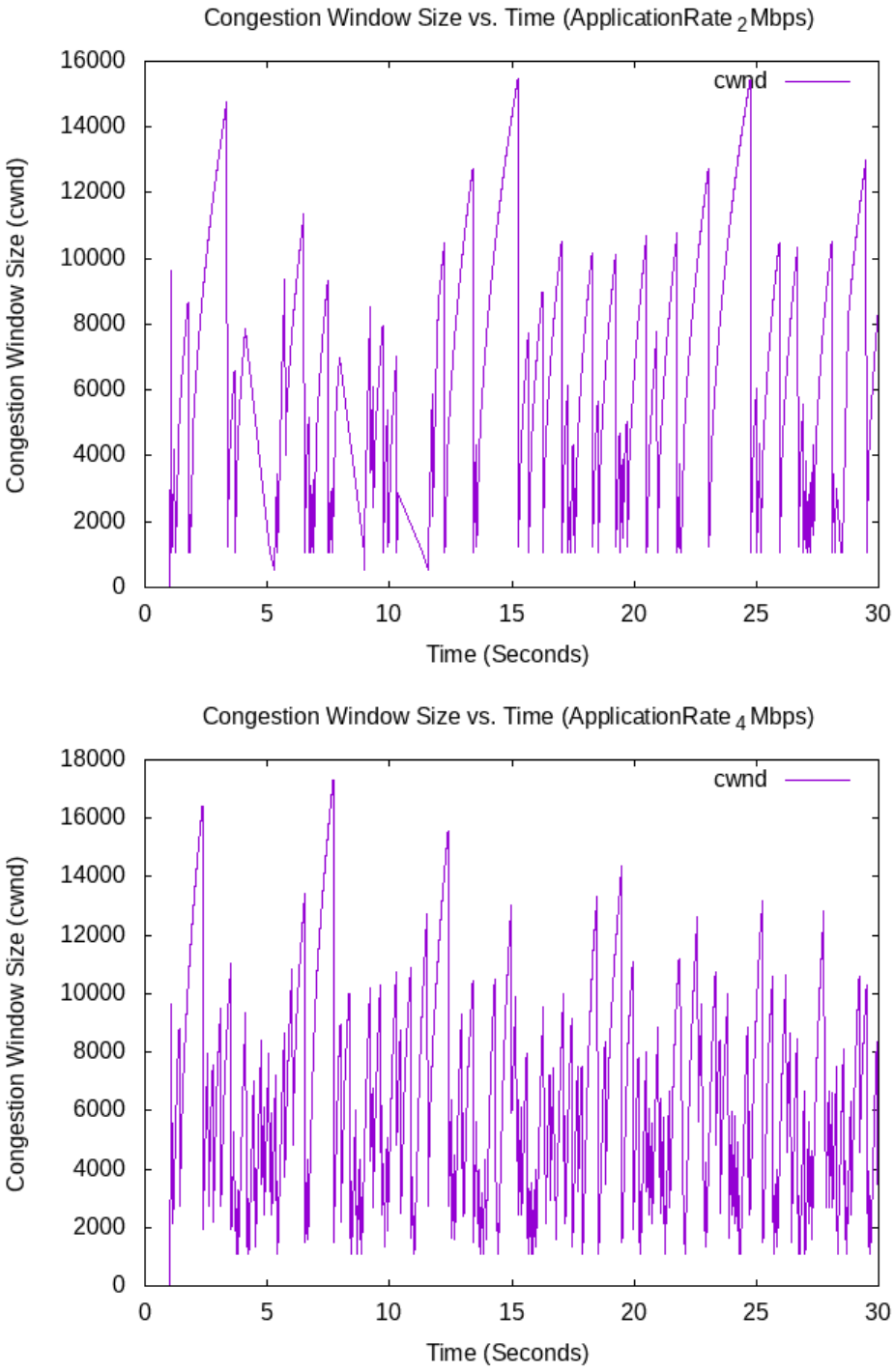

Figure 2.2: Plots for different Application Rates

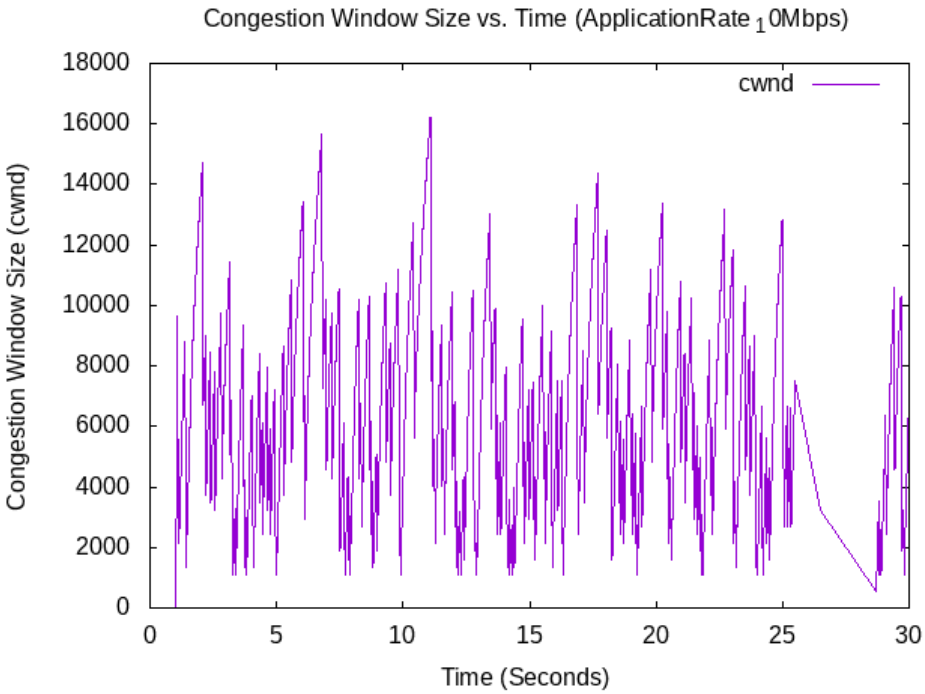

Figure 2.3: Plots for different Application Rates

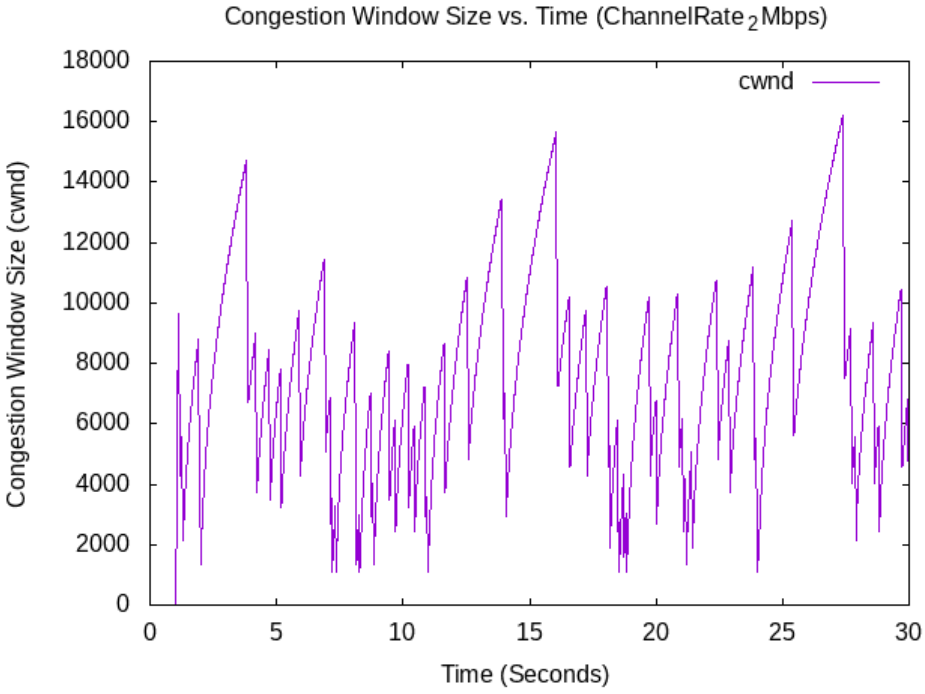

Figure 2.4: Plots for different Channel Rates

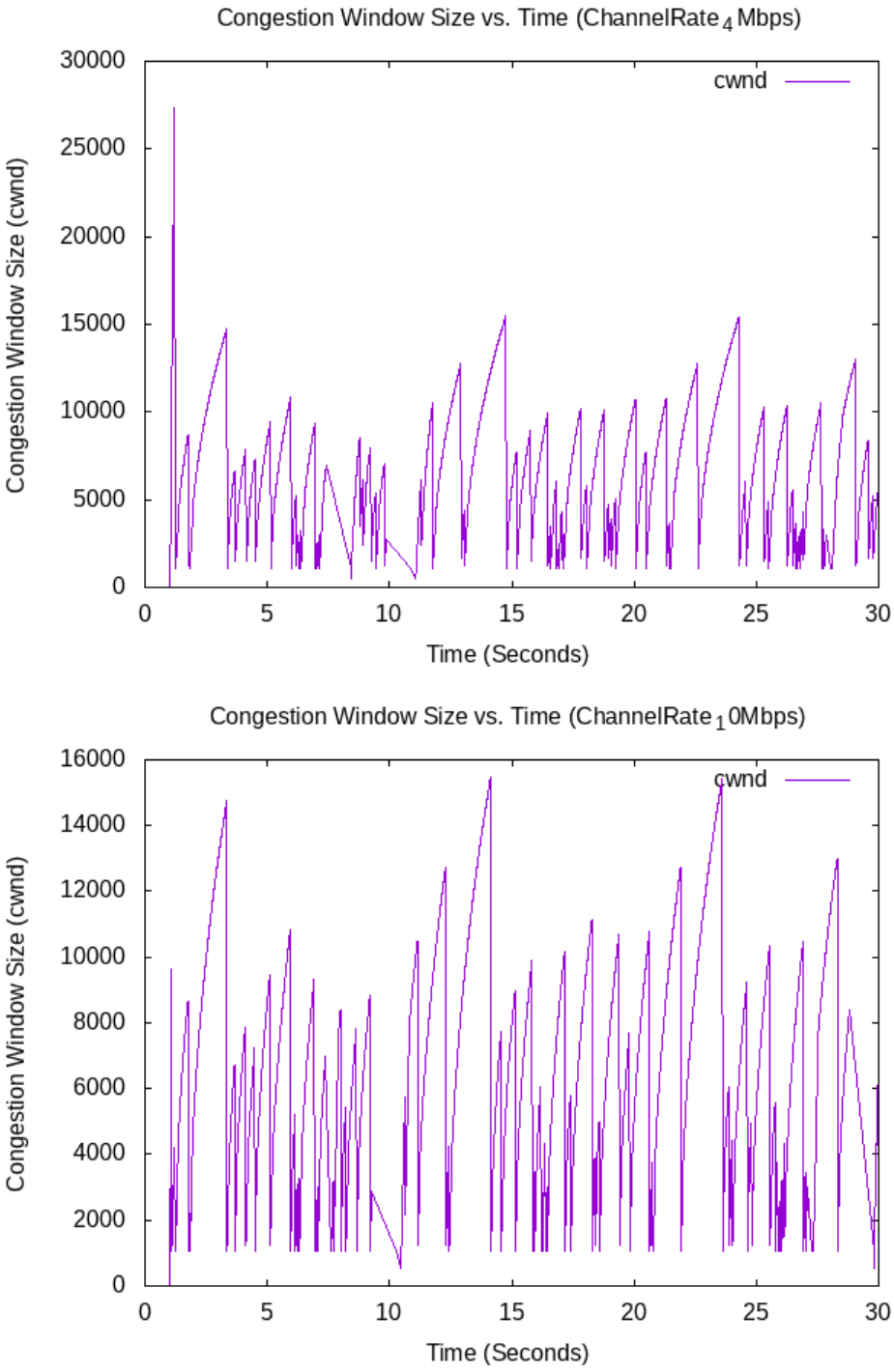

Figure 2.5: Plots for different Channel Rates

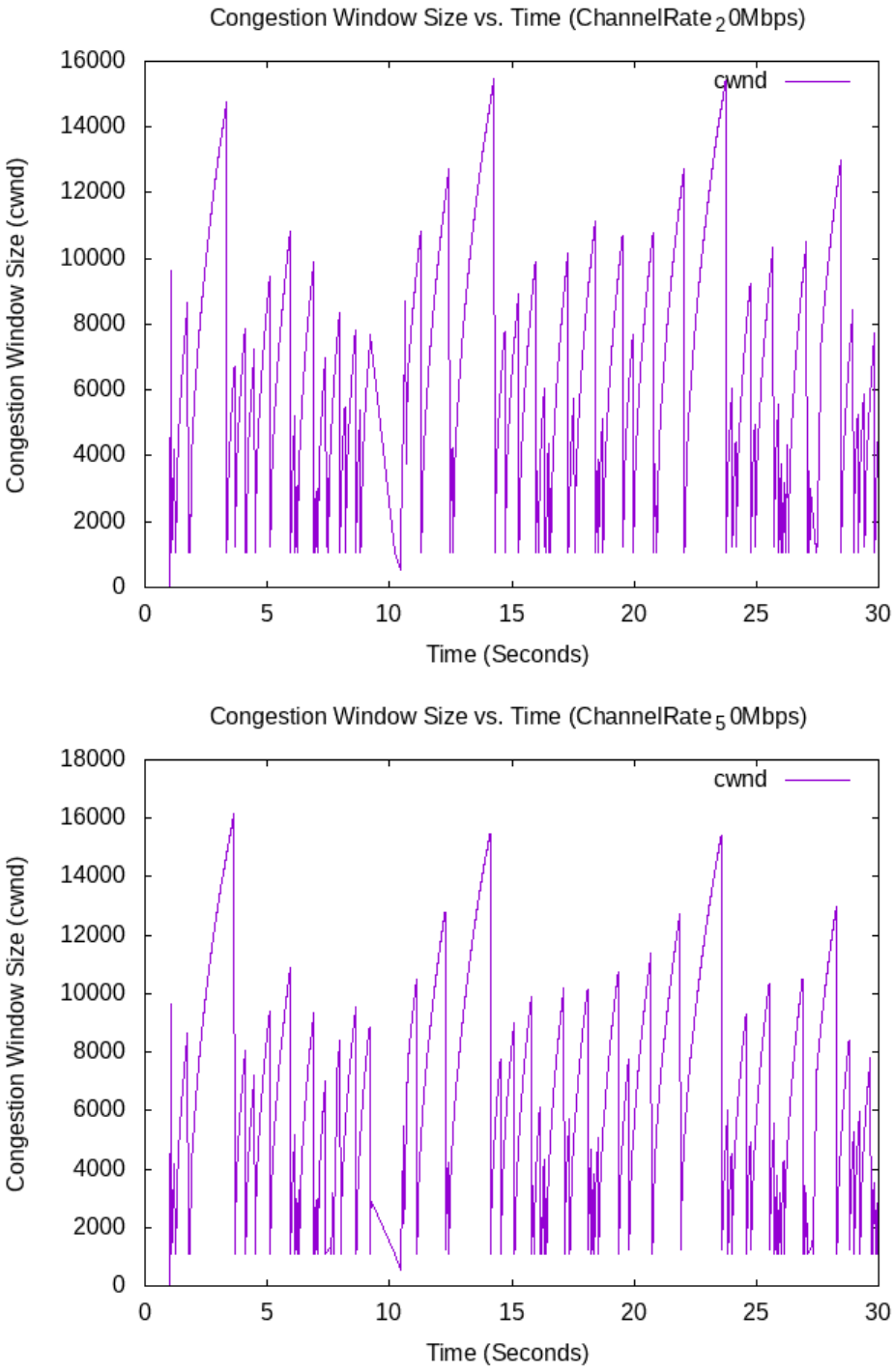

Figure 2.6: Plots for different Channel Rates

2.3 Packet Drops

a. Channel Data Rate = 2 Mbps
Number of packet drops = 62
Number of packets sent by application = 2417
Packet drop percentage [1] = 2.56%

b. Channel Data Rate = 4 Mbps
Number of packet drops = 72
Number of packets sent by application = 2417
Packet drop percentage = 2.97%

c. Channel Data Rate = 10 Mbps
Number of packet drops = 73
Number of packets sent by application = 2417
Packet drop percentage = 3.02%

d. Channel Data Rate = 20 Mbps
Number of packet drops = 74
Number of packets sent by application = 2417
Packet drop percentage = 3.06%

e. Channel Data Rate = 50 Mbps
Number of packet drops = 75
Number of packets sent by application = 2417
Packet drop percentage = 3.10%

f. Application Data Rate = 0.5 Mbps
Number of packet drops = 22
Number of packets sent by application = 605
Packet drop percentage = 3.63%

g. Application Data Rate = 1 Mbps
Number of packet drops = 38
Number of packets sent by application = 1209
Packet drop percentage = 3.14%

h. Application Data Rate = 2 Mbps
Number of packet drops = 71
Number of packets sent by application = 2417
Packet drop percentage = 2.93%

i. Application Data Rate = 4 Mbps
Number of packet drops = 156
Number of packets sent by application = 4834
Packet drop percentage = 3.22%

j. Application Data Rate = 10 Mbps
Number of packet drops = 156
Number of packets sent by application = 12084
Packet drop percentage = 1.29%

[1] Packet drop percentage assumes that all the packets were sent within the simulation period. Note that this might not be the case if some packets are still in the sender queue after t = 30 secs

2.4 Analysis

1. As the application rate increases the rate at which cwnd changes increases. The increase gets steeper with higher application data rates. Congestion events occur early and more often and hence cwnd size drops are more frequent. The peak cwnd size remains nearly the same for all cases.
2. The total number of packets sent increase nearly linearly with increase in application data rate. The number of packet drops increase as well. But the drop trend is non-linear. Further, total packets sent increase but the number of dropped packets don't change when the application data rate is increased from 4 Mbps to 10 Mbps.
3. Number of packet drops increase with increase in channel data rate. As the channel rate is higher more packets are being sent in the same time leading to more drops. The number of packets sent by the application remain constant due to same time and application data speed.
4. All the cwnd v/s time plots look alike and match the TcpNewReno graph. Further, the protocol enters the slow start phase at time t $\approx$ 10 seconds due to a triple ACK possibly.
5. Lowest packet drop percentage is observed at Channel rate of 2Mbps keeping the application rate fixed at 2Mbps.
6. Lowest packet drop percentage is observed at Application rate of 10Mbps keeping the channel rate fixed at 6Mbps.

Chapter 3

Part 3

3.1 Running Code

```
1. Place the Third.cc file inside the directory ns-allinone-3.29/ns-3.29/scratch
2. Commands to analyze different configurations -
    a. Configuration 1
        ./waf --run "scratch/Third -config=1"
    b. Configuration 2
        ./waf --run "scratch/Third -config=2"
    c. Configuration 3
        ./waf --run "scratch/Third -config=3"
3. Command to plot different protocols -
    gnuplot plot3.plt
```

3.2 Congestion Window

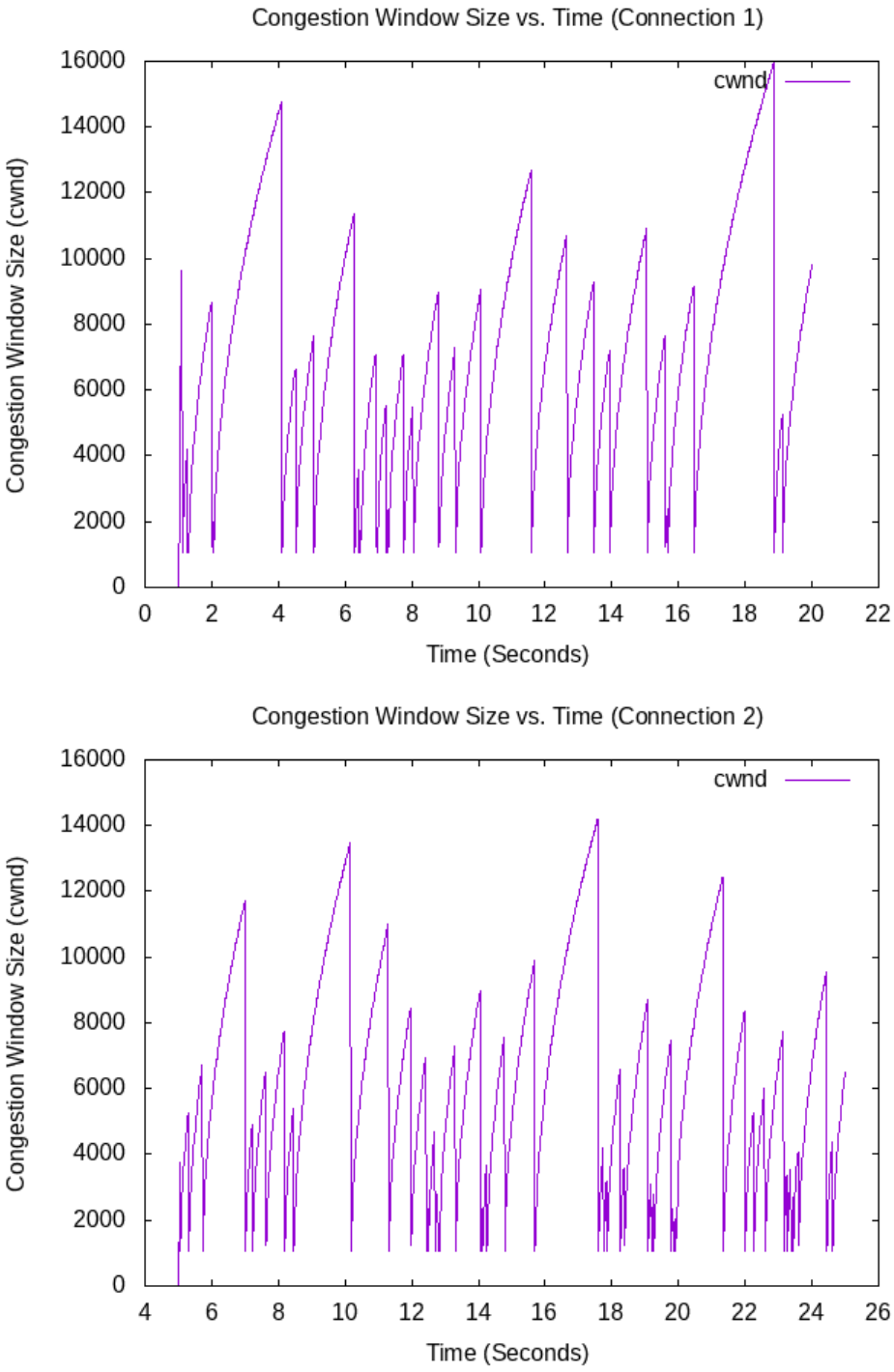

Figure 3.1: Plots for Configuration 1

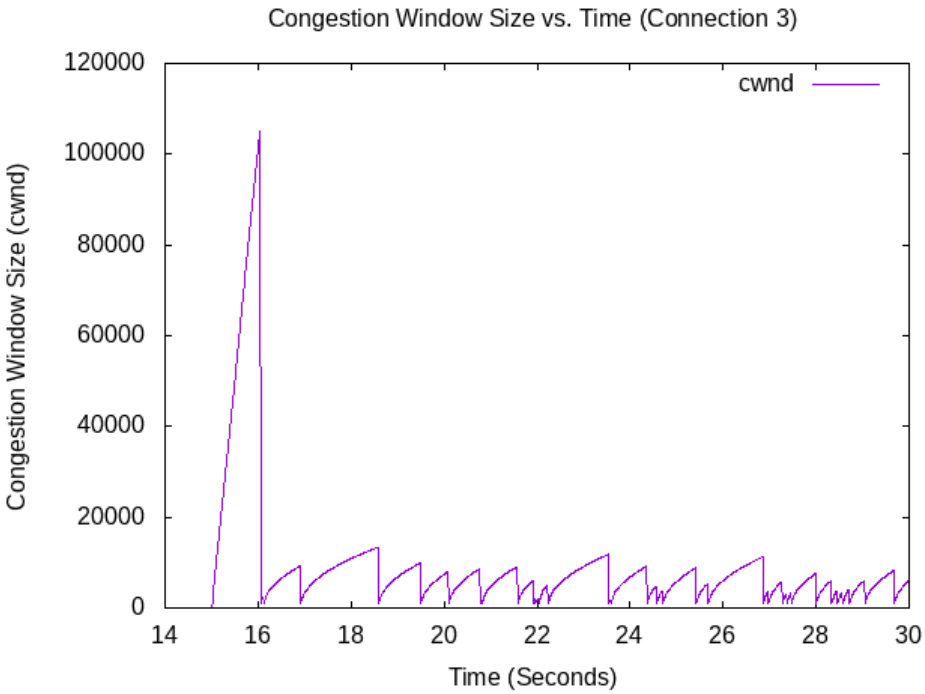

Figure 3.2: Plots for Configuration 1

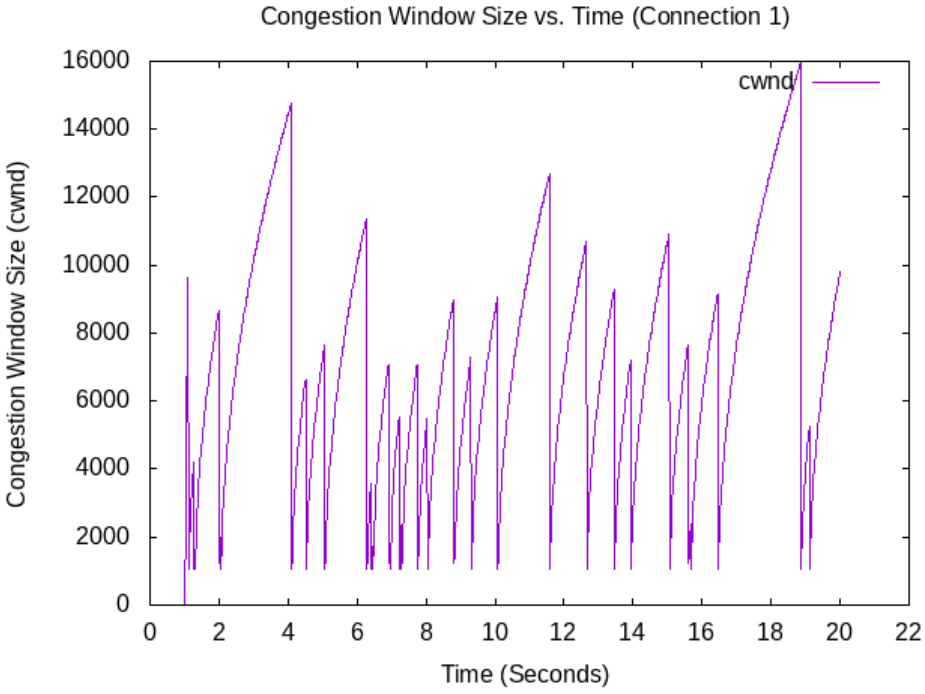

Figure 3.3: Plots for Configuration 2

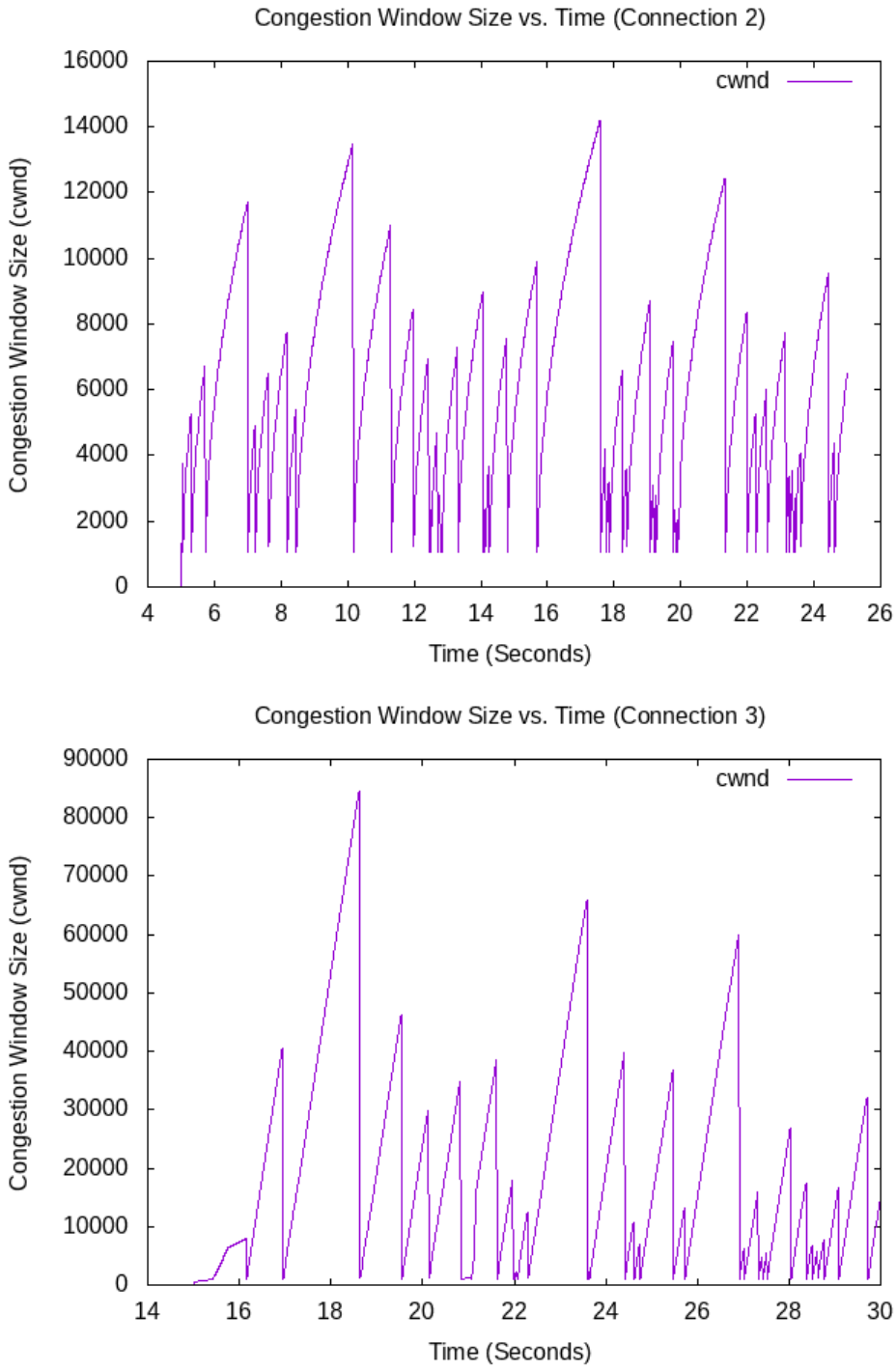

Figure 3.4: Plots for Configuration 2

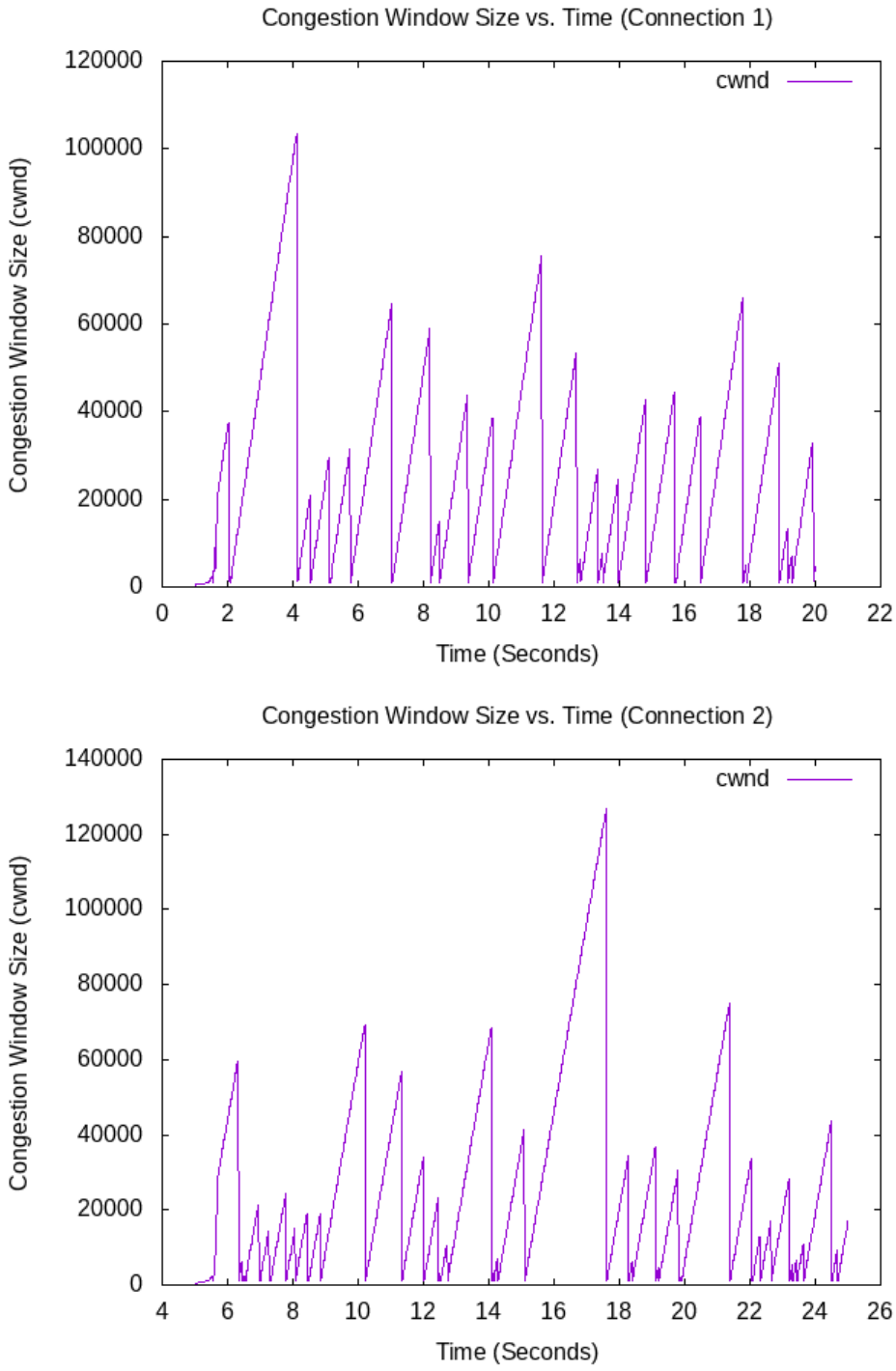

Figure 3.5: Plots for Configuration 3

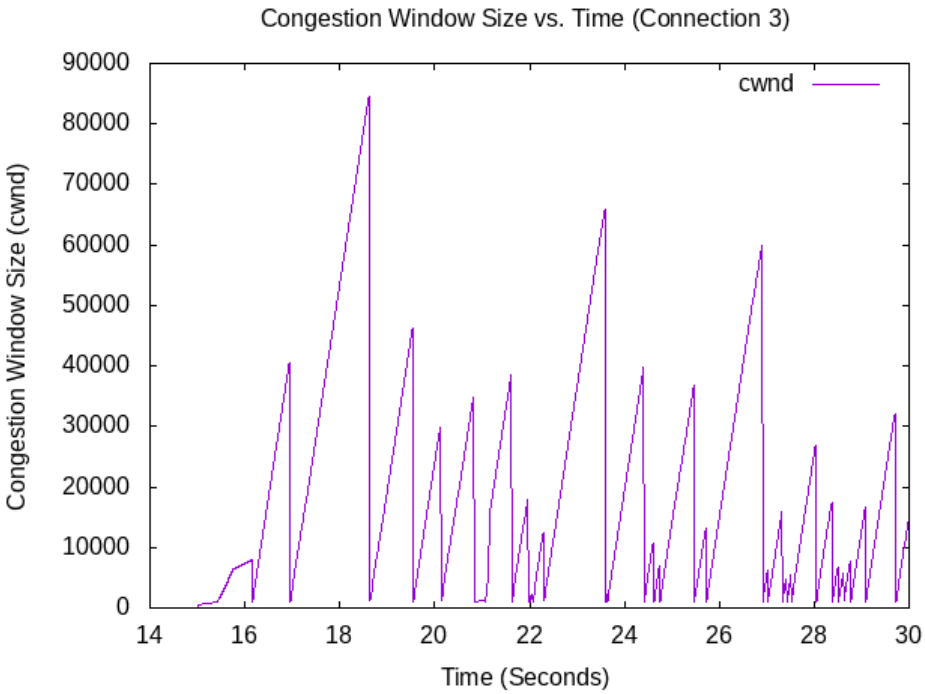

Figure 3.6: Plots for Configuration 3

3.3 Packet Drops

a. Configuration 1

Total Number of Packets sent = 3376
Total Number of Packets dropped = 113
Number of Packets dropped by connection 1 = 31
Number of Packets dropped by connection 2 = 48
Number of Packets dropped by connection 3 = 34

b. Configuration 2

Total Number of Packets sent = 3376
Total Number of Packets dropped = 112
Number of Packets dropped by connection 1 = 31
Number of Packets dropped by connection 2 = 48
Number of Packets dropped by connection 3 = 33

c. Configuration 3

Total Number of Packets sent = 3376
Total Number of Packets dropped = 110
Number of Packets dropped by connection 1 = 33
Number of Packets dropped by connection 2 = 44
Number of Packets dropped by connection 3 = 33

3.4 Analysis

1. Replacing the protocols doesn't affect the total number of packets to be sent = 3376 in all configurations. However, Packet drops are dependent on the configurations.
2. Replacing TcpNewReno with TcpNewRenoCSE in connection 3 leads to one less packet loss for that connection. The rest of the network parameters remains unchanged. Thus, the network outside connection 3 is unaffected.
3. Replacing all protocols with TcpNewRenoCSE lowers the packet drop for connection 2 and 3 by four and one packet respectively whereas packet drops in connection 1 is increased by 2. Thus, there are overall 3 less packet drops across the network.
4. For connection 3, in configuration 1 the peak of congestion control is very low ($\approx$ 20,000) as compared to configuration 2 where the congestion control peaks very high ($\approx$ 80,000). In contrast, the slow start phase peaks to 100,000+ in configuration 1 whereas it peaks at 10,000 in the configuration 2. The other plots are relatively unchanged. Thus, adding a single TcpNewRenoCSE at the connection 3 didn't affect the network much. Even the packet drop is barely affected.
5. Adding TcpNewRenoCSE to all connections increases the peak cwnd for all protocols to nearly 100,000. Further, the graphs have sharper congestion control peaks as compared to the standard TcpNewReno.
6. The congestion avoidance phase is very conservative in the orignal TcpNewReno whereas it reaches higher cwnd size in our new TcpNewRenoCSE protocol.