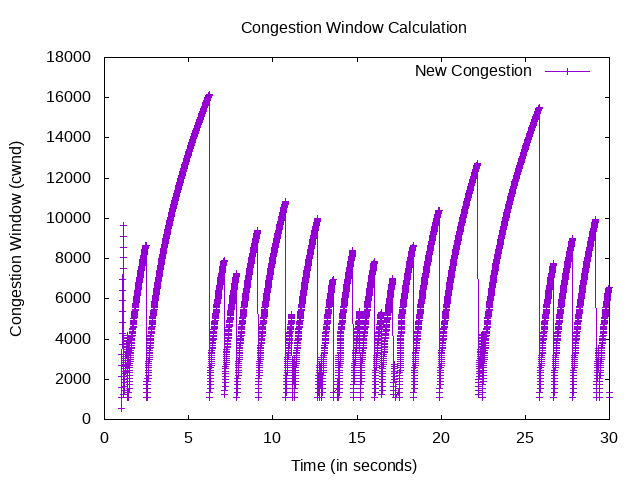
# Q1

* **NewReno :**

We see a sharper rise in the congestion window size as New Reno doesn’t exit the fast recovery phase until it receives the full ACK.

cwnd += min(N, SMSS), N = last un akn bytes, SMSS = threshold bytes

Dropped packet count = 38



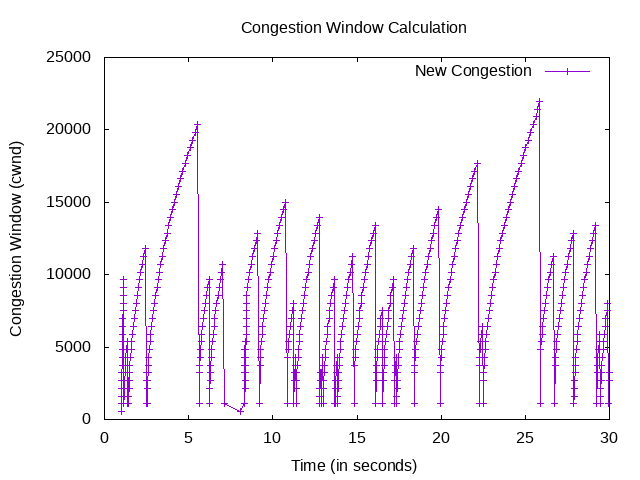
* **HighSpeed :**

It is designed for high-capacity channels/large congestion windows.

cWnd grow faster during probing phases

Cwnd = cwnd + a(cwnd)/cwnd, a() fn from looup table

Dropped packet count = 38



* **Veno :**

Veno enhances Reno algorithm for random loss in wireless network.

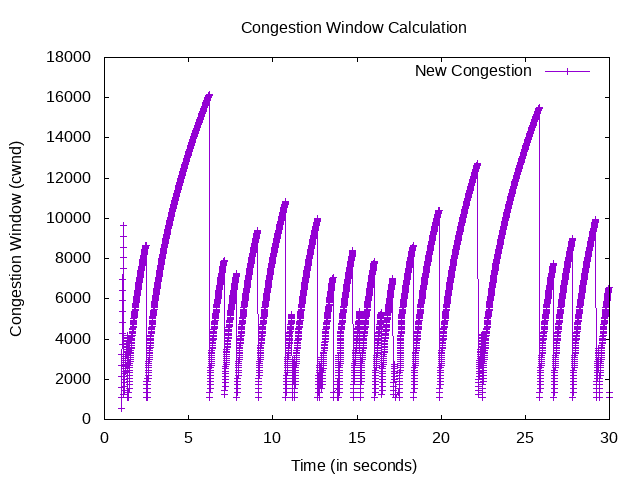
N = Actual \* (RTT - BaseRTT) = Diff \* BaseRTT

Diff = Expected - Actual = cwnd/BaseRTT - cwnd/RTT

cWnd modification based on N

cwnd += 1/cwnd

Dropped packet count = 38



* **Vegas**

Purely delay based congestion control algorithm.

Prevents packet drop by maintaining a small backlog at the bottleneck queue.

actual = cWnd / RTT

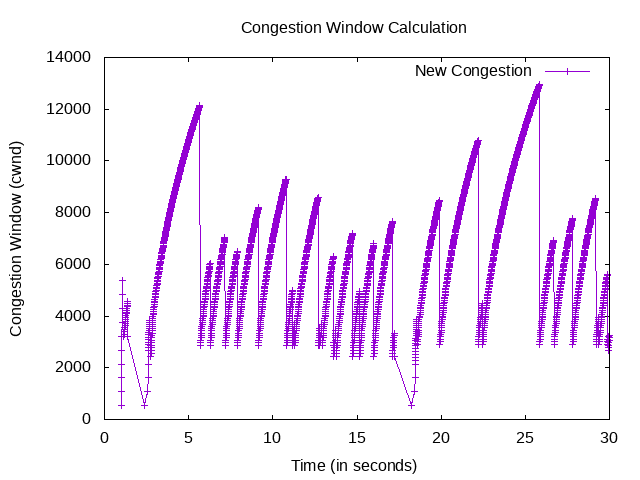
expected = cWnd / BaseRTT

diff = expected – actual

diff -> amount of extra packet queued at the bottleneck

To avoid congestion it linearly increases/decreases it’s congestion window size.

Dropped packet count = 39



**Observation:**

Higher peaks at time ~5, ~25

In highspeed Peak point is high (~20000) as compared to others.

In highspeed Less dense packet throw.

In vegas starting window size is high.

Graph of NewReno and Veno is similar (as veno enhances reno)

Similar packet drop.

(B) Packet drop happen nearly at same time when there is congestion and all are transferring same amount of data over same data rate, hence all have nearly same drop count.

***Three phases of congestion***:

Slow start phase: In this phase after every RTT the congestion window size increments exponentially

Congestion avoidance phase: This phase starts after the threshold value also denoted as ssthresh. And cwnd increase wrt to given algorithm.

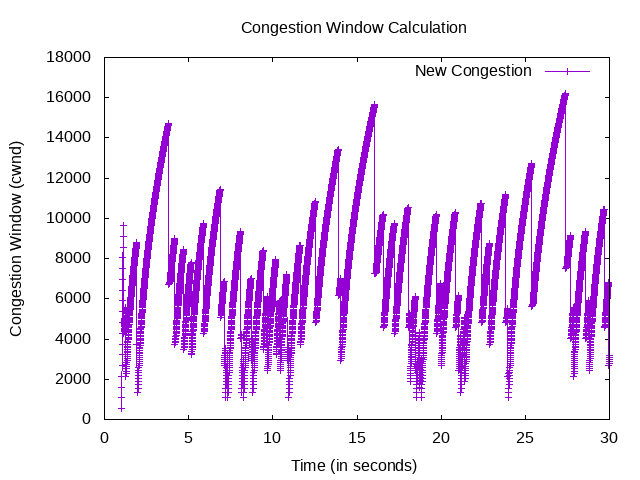
Congestion detection phase: Multiplicative decrease, The only way a sender can guess that congestion has occurred is the need to retransmit a segment.

# Q2

## (A)

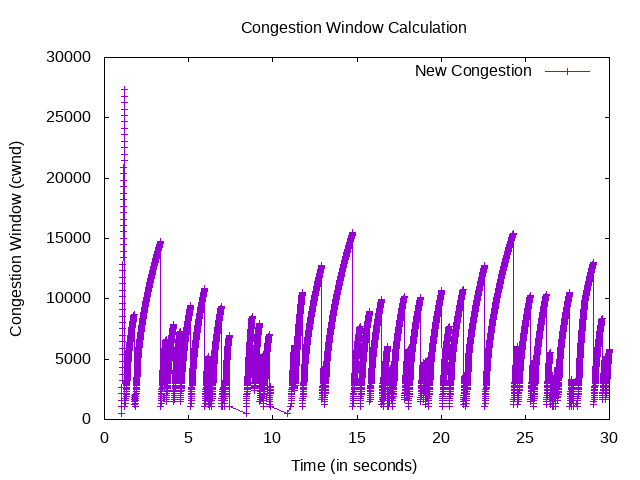
* Channel Data rate = 2

Dropped packet count = 62



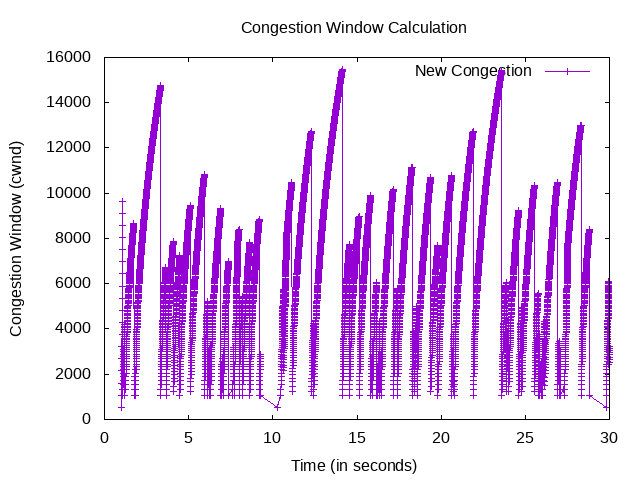
* Channel Data rate = 4

Dropped packet count = 72



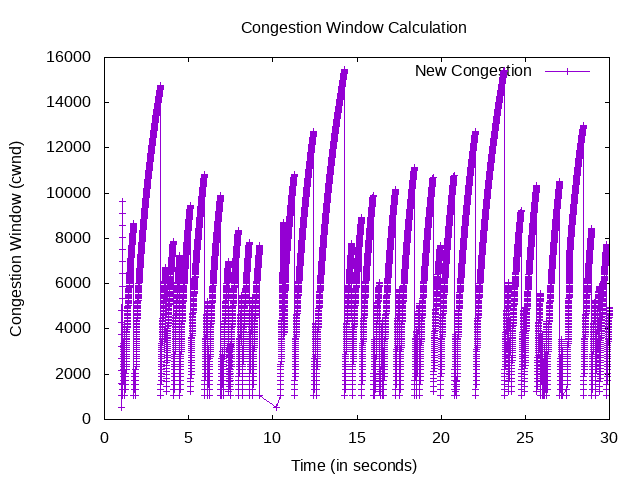
* Channel Data rate = 10

Dropped packet count = 73



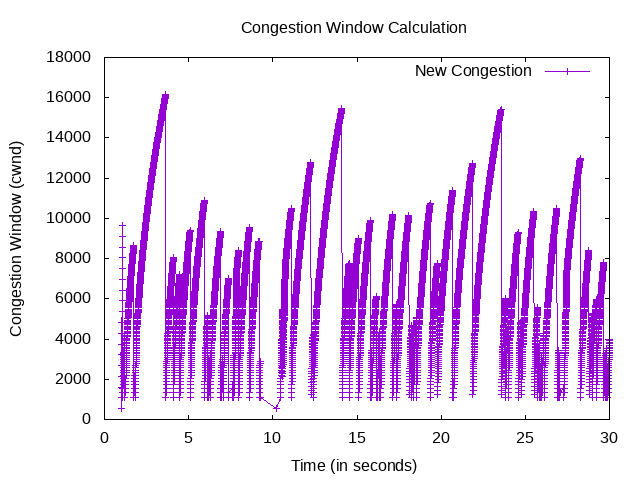
* Channel Data rate = 20

Dropped packet count = 74



* Channel Data rate = 50

Dropped packet count = 75



**Observation:**

We can see that dropped packet count increases wit increase in channel data rate (very slow increase as drop mainly depend on application data rate).

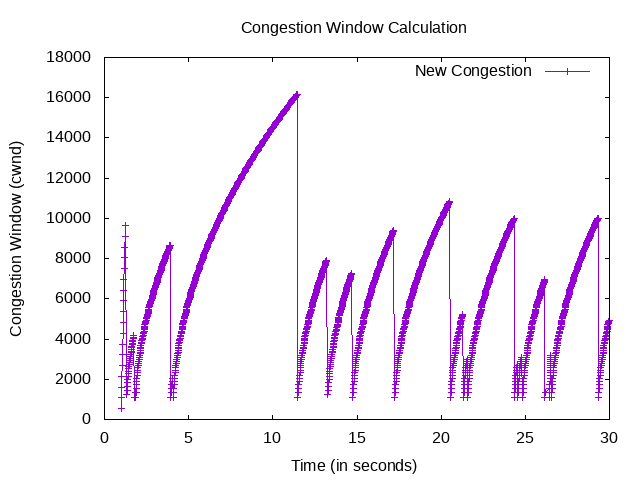
With increase in application data rate we can see frequency of reaching threshold has increased.

For data rate = 2, we can see there is multiplicative decrease (1/2).

## (B)

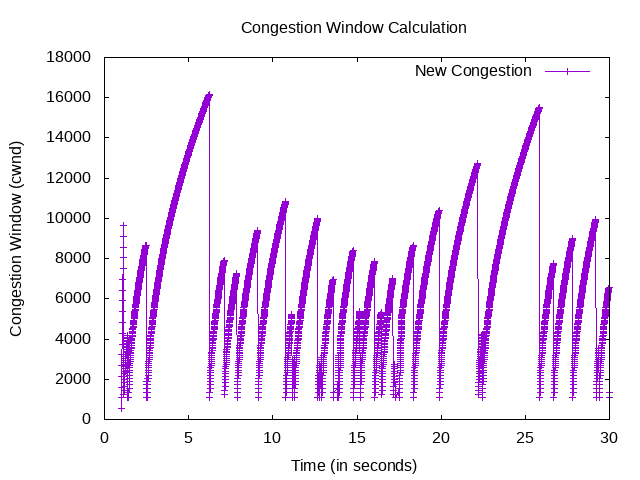
* Application Data rate = 0.5

Dropped packet count = 22



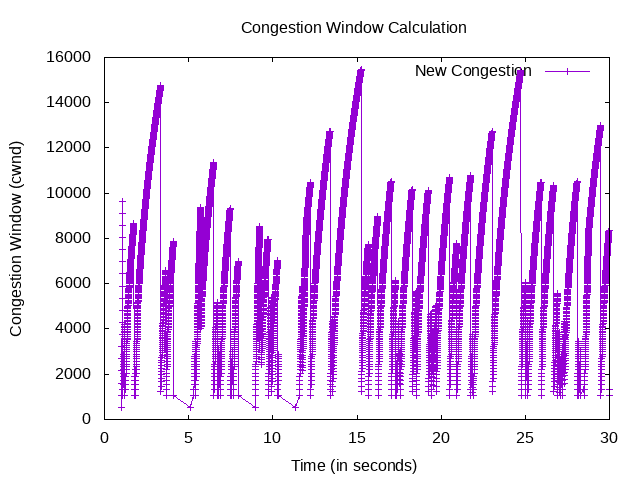
* Application Data rate = 1

Dropped packet count = 38



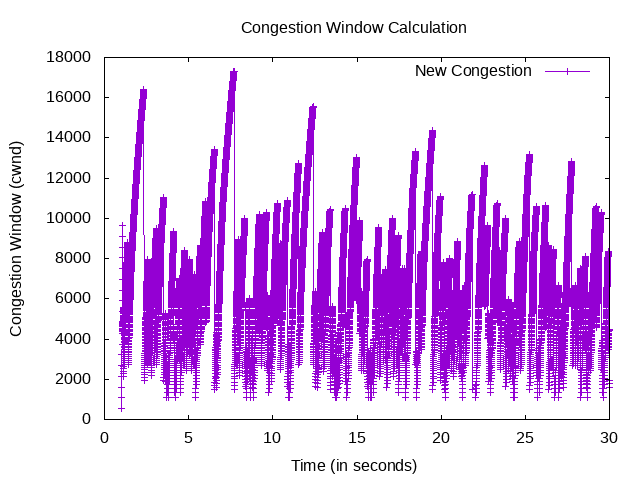
* Application Data rate = 2

Dropped packet count = 71



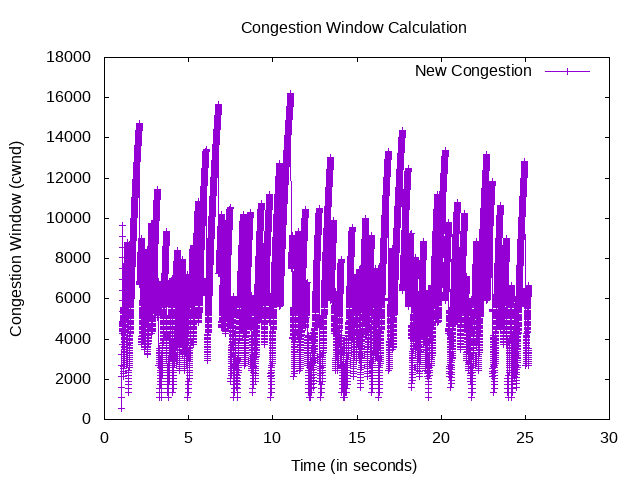
* Application Data rate = 4

Dropped packet count = 156



* Application Data rate = 10

Dropped packet count = 144 (Decrease because upto 25 sec only)



**Observation:**

Significant increase in dropped packet count wrt Application data rate (how fast we send data independent of channel data rate).

With increase in application data rate we can see frequency of reaching threshold has increased.

With observation we see all graphs are similar but because of different rate lower rate transfer data less in 30 sec and when we increase data rate similar graph gets more compressed (increase frequency). And we can see it that 16000 peak for .5 came at around 10 sec, 1 at 5 sec, 2 at 3 sec, 4 at 1, and 10 at ~0.

Base code taken from sixth.cc of ns-3