# 6.5820 Problem Set 1

joekim02
https://github.mit.edu/joekim02/6.5820-pset1
latest commit hash: a1382b6a26c5650ccbeb1255ef11073af610d380

# Theoretical Questions

## 1. Congestion Control (Lecture 2)

1. This is a linear control algorithm, as the only changes made to the window size are additive increases or decreases, which are both linear.
2. **(a)** After a window is submitted, the new window sizes $w_1', w_2'$
   $w_1' = w_1 + \frac{2}{w_1}$ per packet, of which there are $w_1$, so
   $w_1' = w_1 + 2 = 2w_2$, and
   $w_2' = w_2 + 2$
   **(b)** Figure (i) represents the trajectories of w1 and w2. The other figures contain some amount of multiplicative change that is not present in this algorithm.
3. iv. None of the above. The lack of a multiplicative decrease means the system cannot move towards 100% fairness, so the system will never approach fairness unless the initial conditions were already at 100% fairness.

## 2. End-to-end Congestion Control (Lecture 3)

A. The bandwidth-delay product of the path is

$$\frac{10,000\text{packets}}{\text{sec}} \cdot 0.02\,\text{sec} = 200\text{packets}$$

B. True. To achieve 100% utilization in the long-run, the size of the buffer must be at least the value of the BDP. Given the answer in part(c) regarding maximum queue length, the buffer is of length 600 (packets), which is larger than the 200 packet BDP. Therefore, this flow will always achieve 100% throughput.

C. The maximum length queue length at the bottleneck is equal to bottleneck rate *time to drain, so its value is*
*$10,000* 0.06 = 600$ packets.

D. Given that we have the maximum length queue and this path maintains 100% utilization, we can determine the minimum window size (which is 1/2 the maximum) and extrapolate the minimum queue length from there:

$$w_{max} = BDP + B_{max} = 200 + 600 = 800$$

$$w_{min} = \frac{1}{2} w_{max} = 400$$

$$w_{min} = BDP + B_{min} \to B_{min} = w_{min} - BDP = 400 - 200 = 200$$

The minimum queue length is 200 packets.

E. To achieve 100% utilization in the long run, the size of the buffer must be at least the value of the BDP, which is equal to the bandwidth rate times the RTTmin. Therefore,

$$B_{size} \geq BDP$$

$$600 \geq 10000 \cdot RTT_{min}$$

$$0.06 \geq RTT_{min}$$

100% utilization occurs when $RTT_{min}$ is at most 60 ms.

## 3. PIE (Lecture 4)

1. The drop probability is maximized at p=22, since the delay drops at the next time step when p=10. The decrease in drop probability at that time step indicates the maximum drop probability during it.
2. The fully expanded calculation for p after the 10 ms delay sample is

$$0 + \frac{1}{400}(0 + 2 + 4 + 6 + 10 - 2) + \frac{1}{1000}(12 + 2 + 2 + 2 + 4 - 12)$$
$$= 0.06$$

3. $BBR > CUBIC > Reno$
   BBR estimates the BDP and keeps as close to that many packets in transit as possible, keeping the queue empty and keeping latency mostly consistent. This property does not change with different minimum RTT, so the throughputs are roughly equal.

CUBIC has a higher throughput than Reno because upon after dropping a packet from congestion, it scales its window size much more aggressively than Reno.

## 4. XCP (Lecture 4)

1. True. A flow with a lower RTT can send more packets than a flow with higher RTT, which means it would be able to get more of the required change than the other flows, giving it a higher throughput.
2. True. No change occurs to the efficiency controller, so the total amount of change at each time is still the same. The only difference between the assumed change and the normal method is the allocation of the total change across all flows.

3. False. In a given time frame, a flow with larger packets will end up sending less total packets, and thus the total change it receives is equal to a flow with smaller packets since the change across each packet is still the same.

## 5. Fair Queueing (Lecture 5)

1. The rate of each flow is $F_1 = 3, F_2 = 7, F_3 = 4, F_4 = 3$
2. Removing $F_4$ would increase $F_1$'s rate to increase because they are both bottlenecked at $L_3$ .
3. The rate of each flow when $F_3$ has weight 3 is $F_1 = 2, F_2 = 8, F_3 = 6, F_4 = 2$

## 6. BGP (Lecture 6)

1. AS1 will only announce it's self-origin route: [AS1]
2. AS3 will announce all its routes to its customer AS1: [AS4,AS5,AS3]
3. AS3 will only announce it's customers routes and self-origin to the peer AS5: [AS4,AS3,AS1]
4. Packets from AS1 to AS4 can traverse the autonomous systems AS1, AS2, AS3, and AS4.
5. There is no route from AS2 to AS5 because AS4 is in a peering relation with both of them.

# Practical Questions

## (A)

1. What are the slopes of the curves for Throughput vs RTT for Reno, CUBIC, and BBR? Does this match the expected theoretical result?

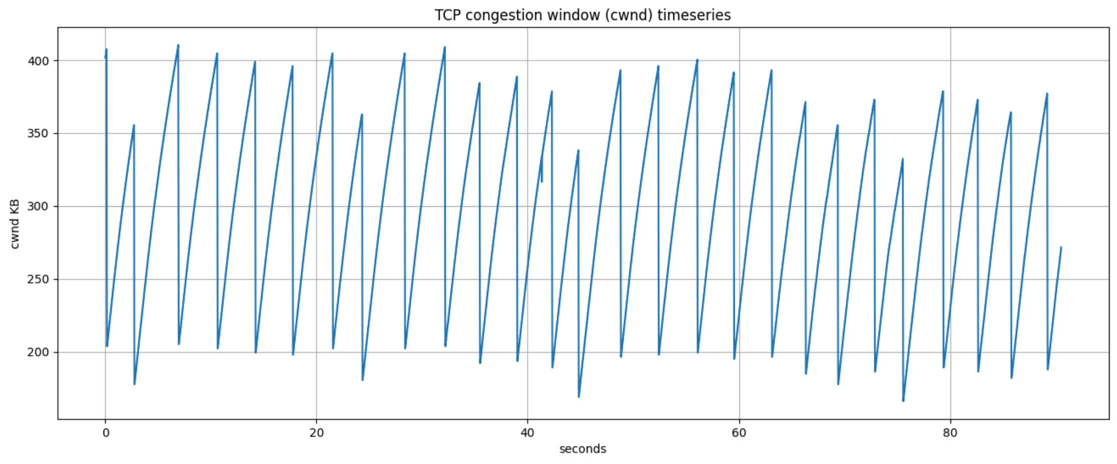
Slope of the function log(tput) vs log(delay) for Reno = -0.9101732868104401

Slope of the function log(tput) vs log(delay) for Cubic = -0.19290512781576855

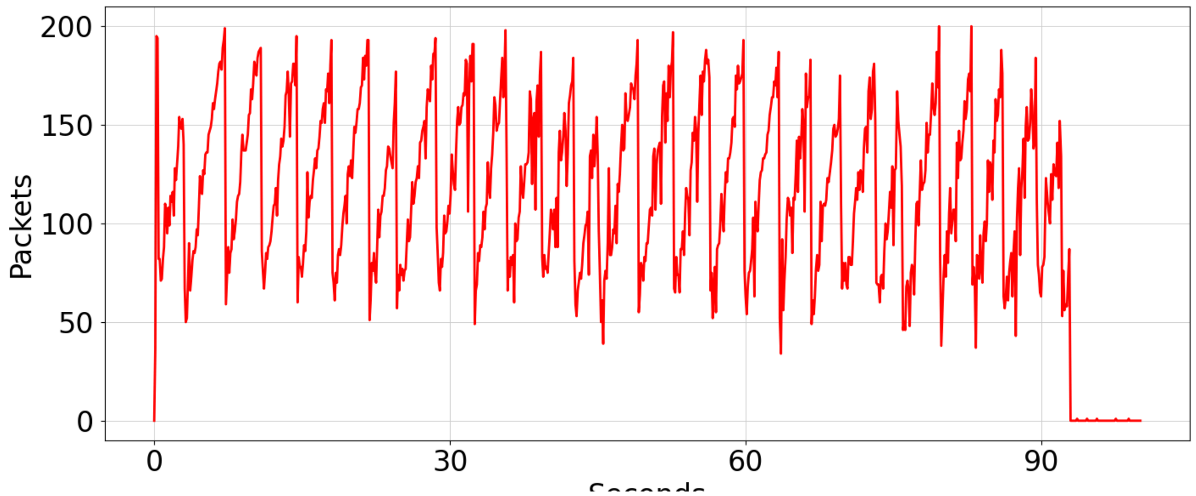
Slope of the function log(tput) vs log(delay) for BBR = -0.3024964460525641

It isn't a perfect fit to the expected theoretical result, but it does encapsulate the relationship that BBR has better throughputs than Cubic which is better than Reno for a given RTT_min

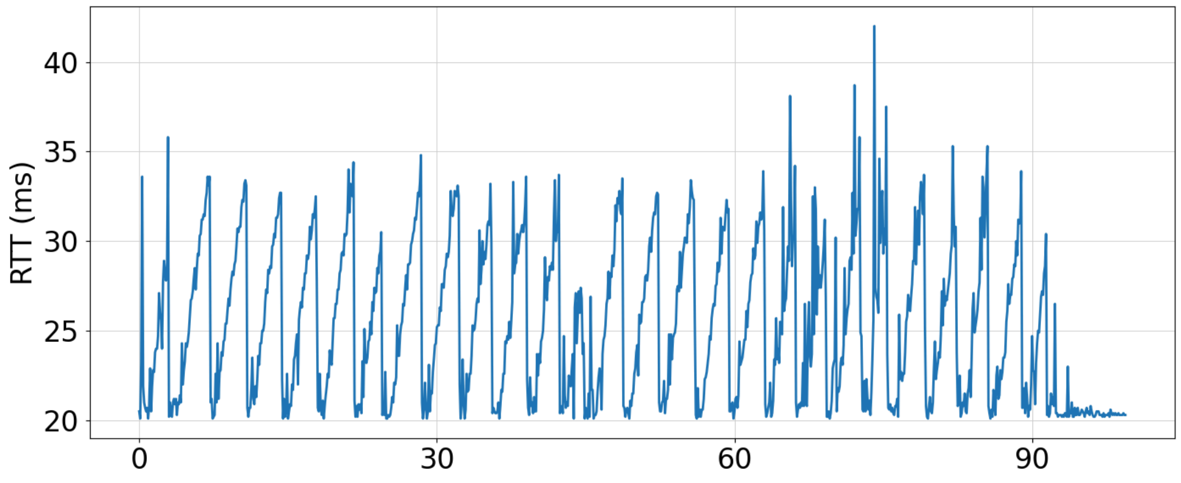
## A-cwnd



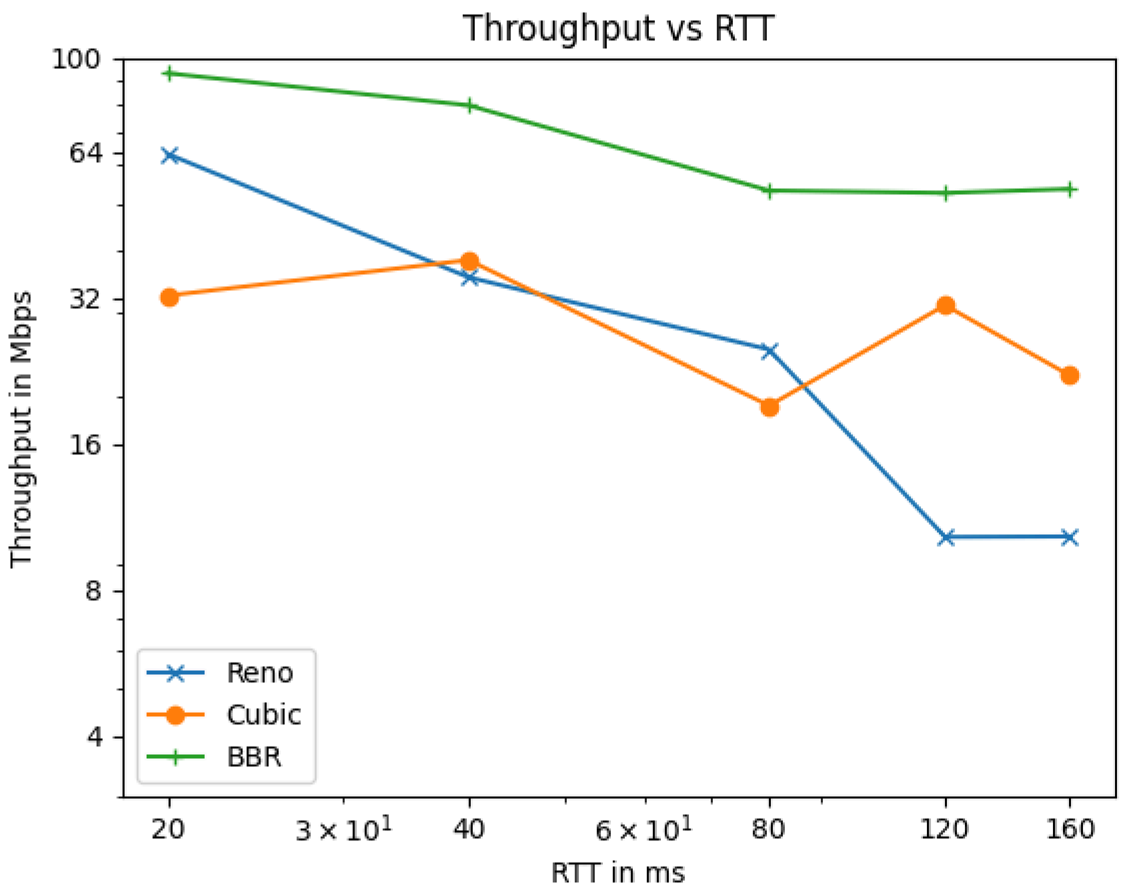
## A-queue



## A-rtt



**A-tput_rtt.png**



## (B)

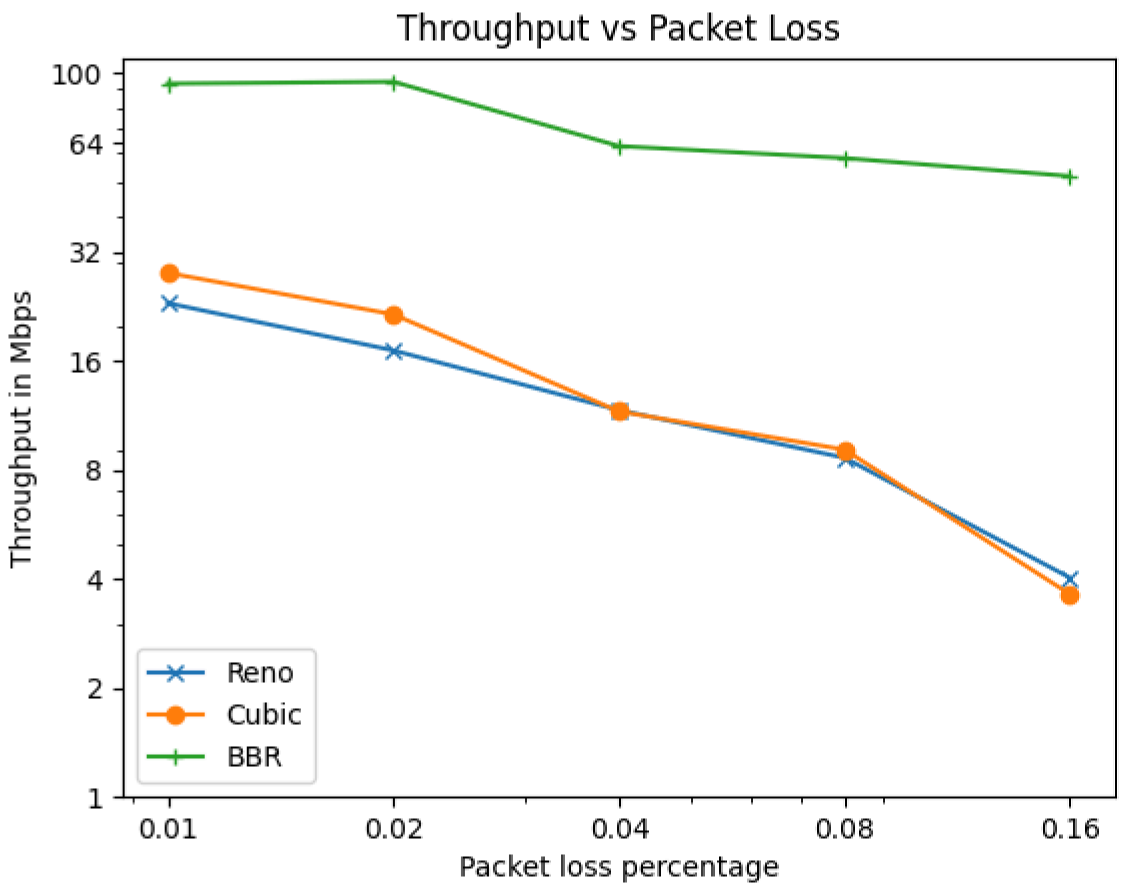
1. What is the slope of the curve for Throughput vs Loss for Reno, Cubic and BBR? Does this match the expected theoretical result?

Slope of the function log(tput) vs log(loss) for Reno = -0.6022975323160694

Slope of the function log(tput) vs log(loss) for Cubic = -0.7135168957063925

Slope of the function log(tput) vs log(loss) for BBR = -0.23980986622877204

B-tput_loss



## (C)

Reno v Reno - Short Flow average throughput: 7.34

Reno v Reno - Long Flow average throughput: 2.5700000000000003

BBR v BBR - Short Flow average throughput: 1.6600000000000001

BBR v BBR - Long Flow average throughput: 7.3

## Question

1. Compute the expected theoretical throughput between the two Reno flows in the first experiment. Do this match your experimental results?
   The RTT for h1 is 3x as fast as it is for h2. In theory, h1 should have 3x the throughput of h2, so the theoretical throughputs of the two Reno flows is $r_1 = 7.5 Mbps, r_2 = 2.5 Mbps$. This is roughly consistent with the measured flows.
2. Now look at the second experiment. Let `r1` and `r2` be the rates of the flows originating respectively from `h1` and `h2`, and `q` be the queueing delay experienced by both flows at the bottleneck switch. Assume that both senders correctly estimate the min RTT for their

path (40ms for `h1` and 160ms for `h2`), and estimate the bandwidth of the bottleneck link to 10Mbps.

```
1) Write the two equations for rates `r1` and `r_2` as functions of the
queueing delay `q`, the bottleneck link's bandwidth estimate, and each
flow's min RTT. Remember that BBR sets its congestion window to the
product of the estimated bottleneck link bandwidth and minimum RTT, and
that the flows' rates are given by `cwnd/RTT`.
```

$$r_1 = \frac{0.4}{0.04 + q}$$

$$r_2 = \frac{1.6}{0.16 + q}$$

2) Along with the fact that `r_1` and `r_2` should sum up to 10Mbps, this should give you three equations, with three unknown (`q`, `r_1` and `r_2`). Solve them.
Solving the system of equations gives the following answers of
$r_1 = \frac{4}{3}Mb/s, r_2 = \frac{8}{3}Mb/s, q = 80ms.$
3) Does this match your experimental results? If not, what do you think is wrong with the analysis? Do you think the estimate of the bottleneck's link bandwidth by each flow is accurate, below, or above 10Mbps? Why that might be the case?
The general trends match, but not exactly. I think there are some subtleties to real life networks that aren't captured in the theoretical analysis, such as some physics phenomena. The estimate of the bottleneck's link seems to be under 10Mbps.

## (D)

1 Flow without PIE
Average of fetch times: 0.2582
Standard deviation of fetch times: 0.0488

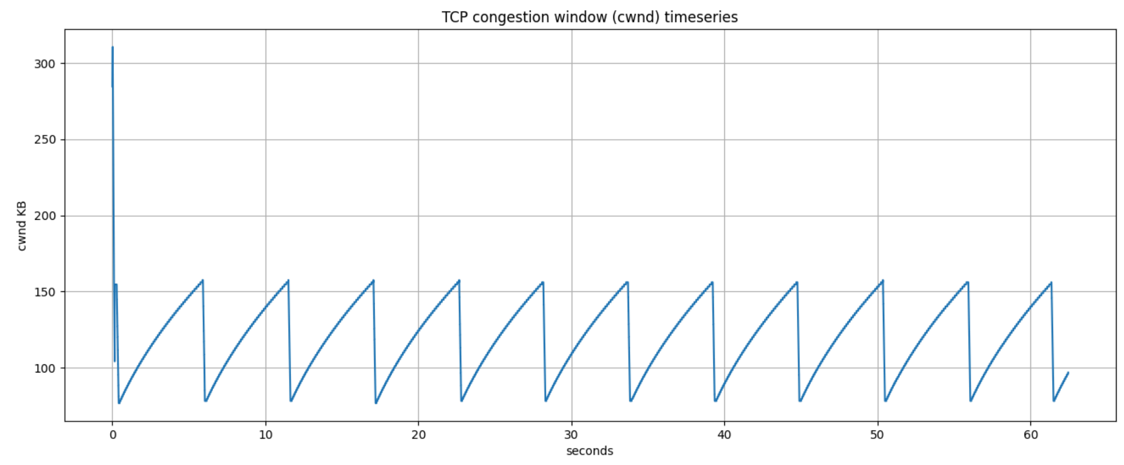
1 Flow with PIE
Average of fetch times: 0.2308
Standard deviation of fetch times: 0.0091

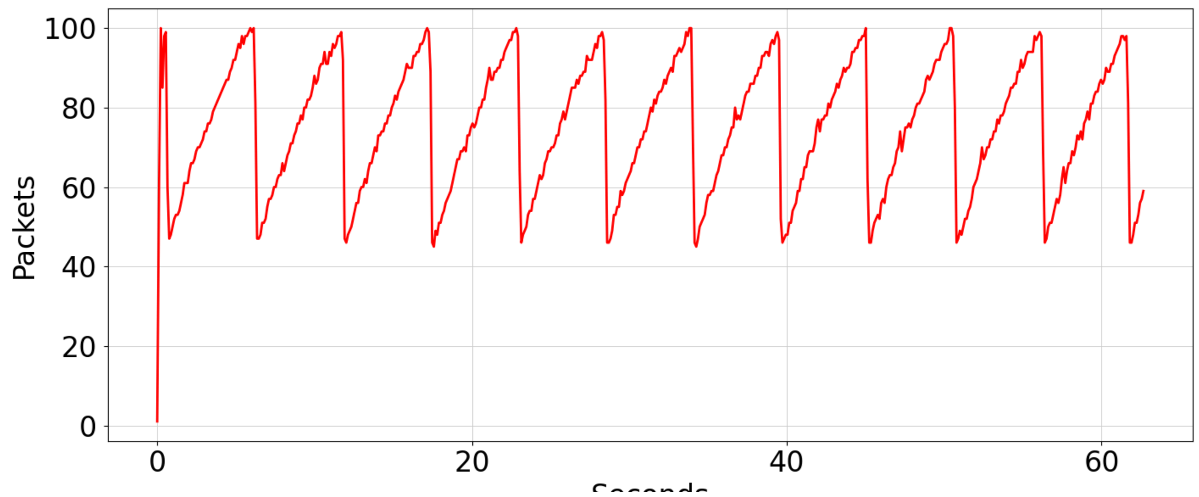
10 flows with PIE
Average of fetch times: 0.3351
Standard deviation of fetch times: 0.0946

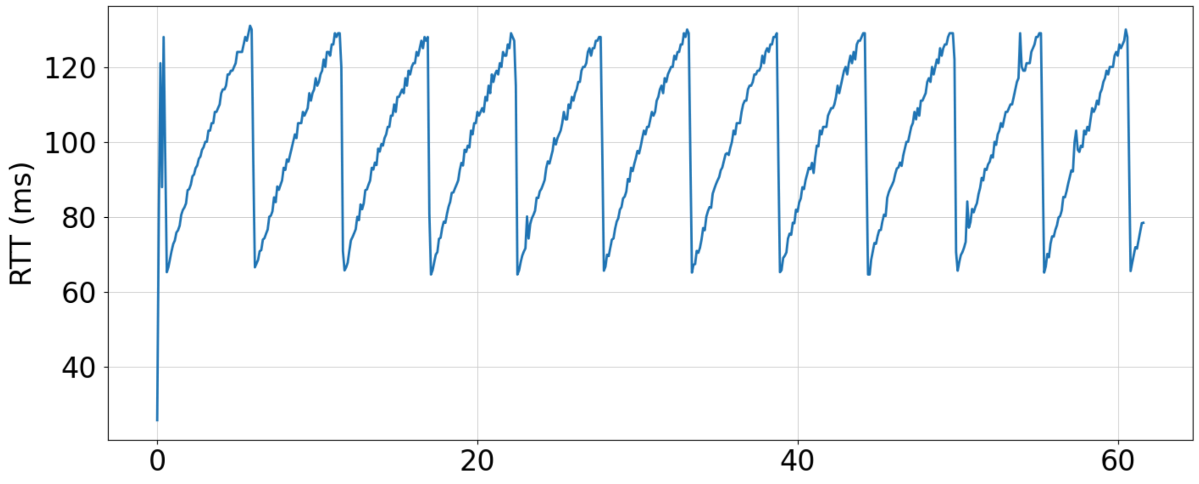
1. Why do you see a difference in webpage fetch times when enabling PIE?
PIE controls the delay by maintaining a drop probability that scales with how large the queue is getting. This keeps the average amount of delay consistent across multiple trials, whereas without PIE, it depends on how close to congestion the flow is.

2. The target queueing delay for PIE is 20 ms. In which case (1 or 10 flows) is PIE better able to keep the queueing delay close to the target? Briefly explain why.
   The case of 1 flow is easier to manage the queueing delay. There is simply less randomness to account for managing the drop probability.

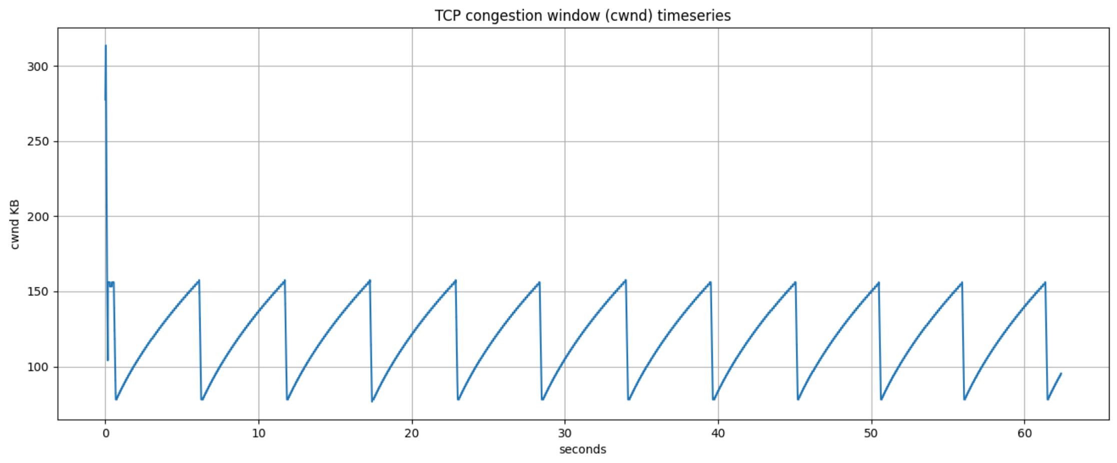
## 1 Flow without PIE

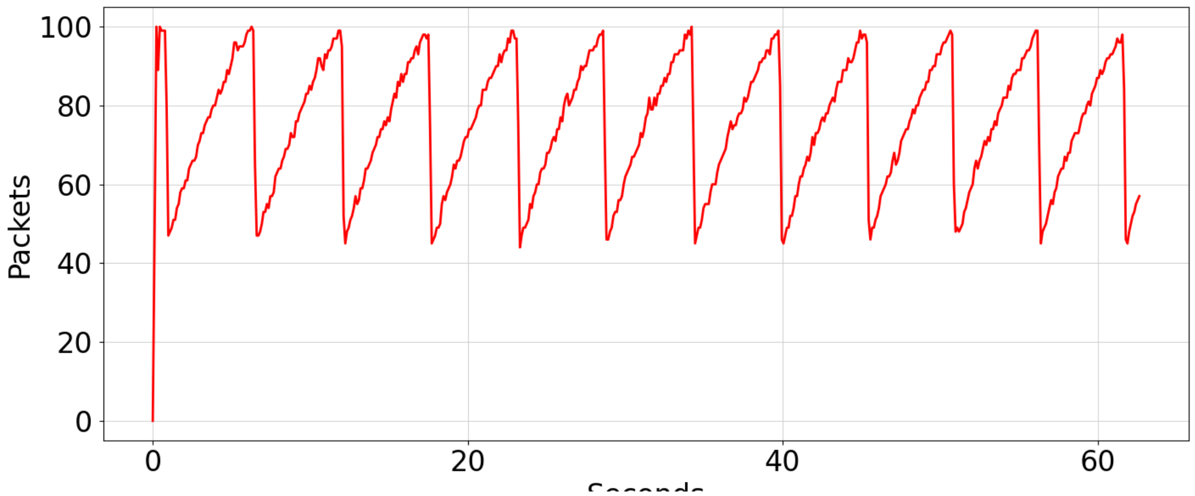


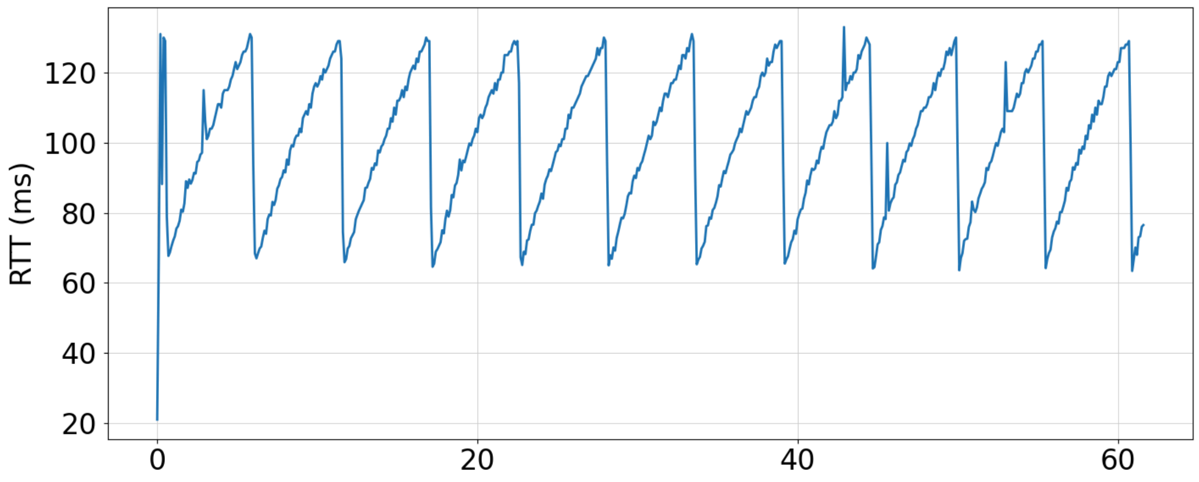




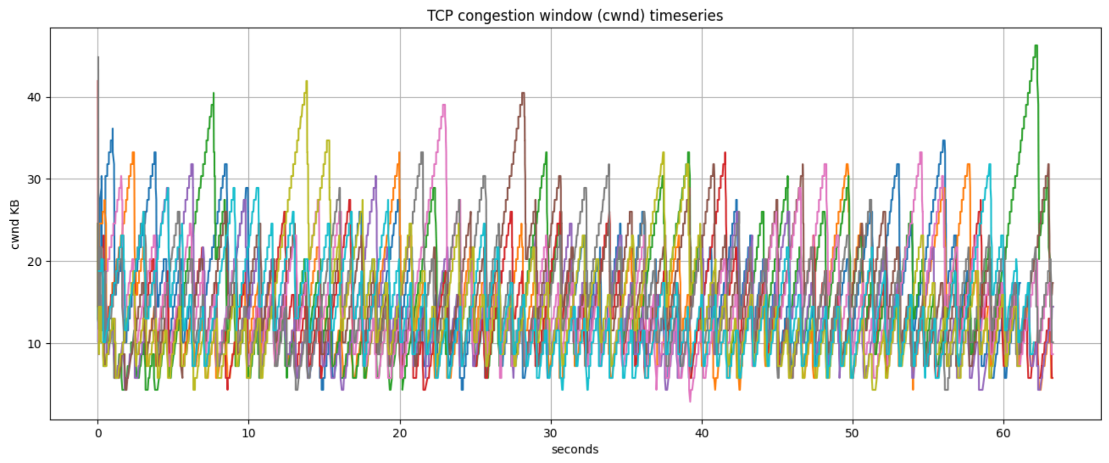
## 1 Flow with PIE

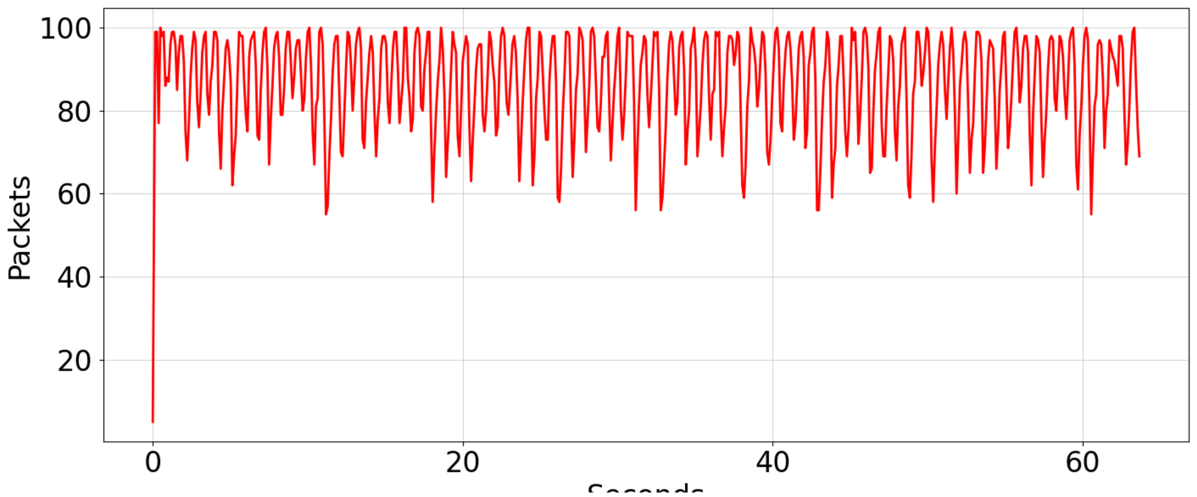


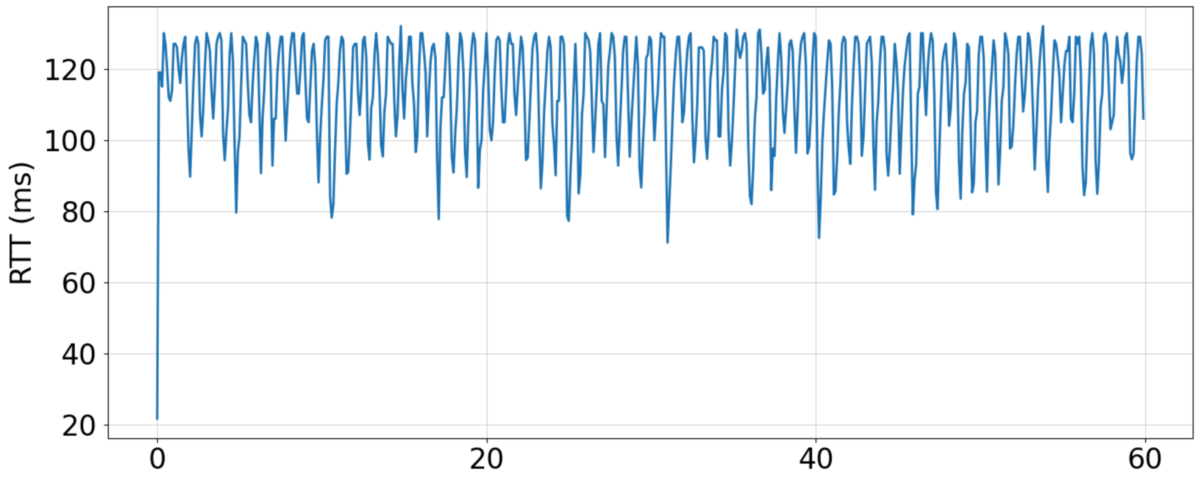




## 10 Flows with PIE


TCP congestion window (cwnd) timeseries
cwnd KB
seconds
10
20
30
40
0
10
20
30
40
50
60


Packets
20
40
60
80
100
0
20
40
60


RTT (ms)
20
40
60
80
100
120
0
20
40
60