

Programming Assignment 2 - Bufferbloat Writeup

Allen Xu, xuallen4, 1008265707

November 2025

University of Toronto, Department of Computer Science

CSC458/2209 - Computer Networking Systems, Fall 2025

Assignment 2

Due date: Friday, November 28, 2025.

- In this assignment, we will study the dynamics of TCP in home networks. The figure below shows a "typical" home network with a home router connected to an end-host. The Home Router is connected via cable or DSL to a headend router at the Internet access provider's office. We are going to study what happens when we download data from a remote server to the end-host in this home network. Figure 1: Typical home network topology In a real network, it's hard to measure the sender's CWND and the buffer occupancy (since they are private to the server and the router, respectively). We will therefore use Mininet emulations to measure these metrics in a controlled network environment. Additionally, we will align the Mininet observations with simplified mathematical expressions to enhance our ability to explain our findings.
- Goals:
 - Learn the TCP sawtooth behavior and router buffer occupancy dynamics in a network.
 - Learn why large router buffers may lead to poor performance (which is often called "bufferbloat.")
 - Learn how to use Mininet to run traffic generators, collect statistics, and plot graphs.
 - Learn how to organize your experiments in a reproducible manner.
 - Learn how to link theoretical findings with emulation results.

4. Questions

1. [2 pts] Why do you see a difference in webpage fetch times with small and large router buffers?

With larger router buffers, the queue is able to get filled up more without triggering loss. In these cases, we are left with large queueing delays, which would in turn increase overall RTT and force

webpage packets to wait behind other buffered packets. As a result we see longer webpage fetch times. When we have a smaller buffer, the queue reaches capacity sooner and drops happen earlier. This prevents large queues and long queueing delays. As a result RTT and queueing delay remains low in comparison and we see faster webpage fetch times.

2. [2 pts] Bufferbloat can occur in other places such as your network interface card (NIC). Use `ifconfig` or `ip link show` on your Mininet VM to identify the transmit queue length (`txqueuelen`) of an interface such as `enp0s1` and report the output. For this queue size and a draining rate of 100 Mbps, what is the maximum time a packet might wait in the queue before it leaves the NIC?

Using "`ip link show`" I got a transmit queue length of 1000 for `enp0s8`:

```
enp0s8: BROADCAST,MULTICAST,UP,LOWER_UP mtu 1500 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
```

Given this queue length of 1000 and a draining rate of 100Mbps, we can perform the following calculation to get the maximum time a packet might wait in the queue. We assume queue is full as we want the maximum time a packet might wait

Assume that each packet is equal to MTU which is 1500 bytes. 1500 bytes x 8 bits/byte = 12000 bits. Therefore each packet is 12000 bits in size

Draining rate of 100 megabits/s = 100 000 000 bits/s = 100×10^6 bits/s for simplicity

$$\text{delay} = \frac{1000 \times 12000 \text{ bits}}{100 \times 10^6 \text{ bits/s}} = 0.12 \text{ seconds}$$

Therefore the maximum time a packet might wait in the queue before leaving the NIC is 0.12 seconds.

3. [3 pts] Analyze your plots of CWND, RTT, and queue size.

- Derive or express a symbolic equation showing how RTT varies with queue size (ignore computation overheads in ping that might affect the final result).
- Explain how CWND oscillations correspond to RTT spikes in your plots.
- Discuss how buffer size influences both TCP performance and webpage fetch times.

(a) A symbolic equation showing how RTT varies with queue size is the following:

$\text{RTT} = T_{prop} + T_{queue} = T_{prop} + \frac{Q \times P}{R}$ where Q is the queue size in packets, P is the size of a packet (assumed to be MSS) and R is the capacity or rate of the bottleneck link.

This shows that RTT varies with queue size, as for a fixed bottleneck link rate and propagation delay, a larger queue size results in higher queueing delay and thus higher RTT. A smaller queue size results in smaller queueing delay and thus lower RTT. This can be seen from the graphs as well, as when queue size increases, so does RTT.

(b) CWND oscillations directly correspond to RTT spikes in my graphs. Based on my graphs,

RTT spikes/increases when CWND increases and is near its peak. As CWND increases, there are more packets in flight, and as a result there are more packets occupying the queue. Based on the formula given in part (a) as queue size (occupancy) increases, so does RTT, which aligns with the findings from the graph. When loss occurs, there is a drop in the CWND due to the multiplicative decrease rule. During this, we also see a decrease in RTT. Therefore, CWND oscillations directly affect RTT spikes in my graphs.

(c) Buffer size also directly influences both TCP performance and webpage fetch times. During one trial run of my bufferbloat experiment, there was an average fetch time of roughly 0.74 seconds in the case where buffer size was 20. In comparison, the average fetch time was 1.67 seconds when the buffer size was increased to 100. Increasing buffer size allows large queues to form and causes increased queueing delay. As we can see from the results, larger buffer sizes result in longer webpage fetch times.

However, looking at the graphs it is clear that with larger buffer sizes, we see less loss occurring when compared to smaller buffer sizes. With larger buffer sizes, CWND increases more and the links are well utilized so we have high throughput. With smaller buffer sizes, loss occurs more frequently so CWND is decreased more often and we have smaller throughput. From this perspective of throughput, TCP performs better with larger buffer sizes.

Conclusively, large buffer sizes have better throughput but higher delay, while smaller buffer sizes have lower throughput but improved delay. This shows the classic tradeoff between the two, where the two metrics are at odds.

4. [3 pts] Identify and describe two ways to mitigate bufferbloat. For each, explain the trade-offs and propose an experiment using your Mininet setup to evaluate its quantitative effectiveness. Clearly document your experimental design, including parameter choices, measurement methodology, and justification for your conclusions, so that the TA can reproduce your results.

(1) The first way we can mitigate buffer bloat is by setting the buffer size to some middle ground like 50 packets instead of 20 or 100 packets. Previously, we saw that with large buffer sizes like 100 packets, we have increased latency. By reducing buffer size to something smaller, we can reduce latency and "bufferbloat" while maintaining a higher level of throughput compared to experiments done with buffer size of 20 packets. The tradeoff here would be that although latency is decreased, we would experience more frequent loss. We might not be effectively using the links and would end up with decreased throughput compared to the 100 packet buffer scenario.

The experiment we can do is simply change the QIZES=(20 100) to something like QIZES=(20 50 100) and execute the run script with otherwise the same bufferbloat topology. Measurement methodology will involve looking at average fetch times, average RTT, and frequency of CWND halves, comparing q50 to both q20 and q100. I expect that compared to q100, q50 will have better average fetch times and better average RTT, indicating a mitigation of bufferbloat, although it may also have more frequent CWND halves indicating more frequent loss.

(2) The second way we can mitigate buffer bloat is by matching the rate of the sender to the bottle neck link so the sender cannot overload the buffer. The justification for this is that, the sender is sending packets at around the same speed they are being drained from the queue so the

queue should remain shallow and won't fill up as fast. As a result, we reduce delay and RTT spikes. Effectively, things like webpage fetches would complete faster, mitigating the bufferbloat problem. A tradeoff for this is that by reducing the rate of the sender, we risk under-utilization and reduce burst tolerance. A slower link will pace bursty traffic, and since sender cannot send packets quickly, the queue can no longer absorb big bursts. There may be moments where we can not utilize the bottleneck link and buffer to its max, leading to reduced throughput.

For this experiment, we can keep the same bufferbloat topology with one minor change where we update `-bw-host` default value from 1000 to 10. We then run the experiment as normal. For the measurement methodology, we can also look at average webpage fetch times, average RTT, average queue occupancy, and CWND evolution over time. I expect that with this change, we will have improved average fetch times compared to the default 1000 host bandwidth case. I also expect shallow queue occupancy, where the queue rarely reaches it's max capacity, and overall reduced average RTT.

(3) Note: I was looking into potential methods to implement AQM methods like RED on the bottleneck link. I believe this would have helped mitigate the bufferbloat problem because early detection would prevent the queue from filling completely. Queueing delay would effectively be reduced and therefore we would experience lower RTT and webpage fetch times. However, I was unable to get this working before the deadline for this assignment configuration. Despite this, I believe AQM methods deserve a mention in ways to mitigate the bufferbloat problem.

5. Theoretical Analysis

1. [3 pts] Sent packets per cycle

Consider the CWND-RTT graph. Determine what is the cycle between the minimum congestion window and the maximum congestion window. Assume that no random loss occurs. Now, by summing the packets sent over RTT, show that the total number of packets sent through a cycle is roughly equal to $N_{packets} \approx 3W_{max}^2/8$

Under the AIMD assumptions given, steady state CWND cycles between $W_{min} = W_{max}/2$ and W_{max} . This means one cycle is the growth of CWND from $W_{max}/2$ to W_{max} . Given that CWND increases by 1 packet per RTT, we can calculate the following:

$$\begin{aligned} \text{Number of RTT per cycle} &= W_{max} - W_{min} = W_{max} - W_{max}/2 = W_{max}/2 \\ \text{Number of RTT per cycle} &= W_{max}/2 \end{aligned}$$

In each RTT, TCP will send number of packets \approx CWND. We can calculate the average CWND over the cycle using the min value and the max value and dividing by two.

$$W_{average} \approx \frac{W_{min} + W_{max}}{2} = \frac{W_{max}/2 + W_{max}}{2} = \frac{3}{4}W_{max}$$

The number of packets ($N_{packets}$) sent through a cycle can be estimated by $W_{average} \times RTT_{percycle}$

$$N_{packets} = W_{average} \times RTT_{percycle} = \frac{3}{4}W_{max} \times W_{max}/2 = 3W_{max}^2/8$$

$N_{packets} = 3W_{max}^2/8$ is shown as needed

2. [3 pts] CWND and Throughput Modeling. Using the term derived above, show that the steady-state relationship between average CWND \bar{W} , loss probability p , and throughput is:

$$T \approx \frac{K \cdot MSS}{RTT \cdot \sqrt{p}}$$

and compute the constant K . State all assumptions.

From question one we derived that over one cycle, the total number of packets sent is roughly equal to $N_{packets} \approx 3W_{max}^2/8$

We also assumed that loss happens once per cycle, once CWND reaches W_{max} , meaning that there is one loss per cycle and $N = 3W_{max}^2/8$ packets per cycle.

Taking this, we can model loss probability p as the following

$$p \approx \frac{1}{\text{packets per cycle}} \approx \frac{1}{3W_{max}^2/8} = \frac{8}{3W_{max}^2}$$

$$p \approx \frac{8}{3W_{max}^2}$$

Solving for W_{max} above we get:

$$W_{max}^2 \approx \frac{8}{3p}$$

$$W_{max} \approx \sqrt{\frac{8}{3p}}$$

From question one we computed Average CWND \bar{W} as $\frac{3}{4}W_{max}$

Sub value in for W_{max} we calculated above and we get the following

$$\bar{W} = \frac{3}{4}W_{max} = \frac{3}{4}\sqrt{\frac{8}{3p}} = \sqrt{\frac{72}{48p}} = \sqrt{\frac{3}{2p}}$$

$$\text{Therefore } \bar{W} = \sqrt{\frac{3}{2p}}$$

Since \bar{W} = average number of packets in flight and each packet is MSS bytes, where each RTT delivers roughly \bar{W} packets: Throughput in terms of \bar{W} can be modeled as:

$$T \approx \frac{\bar{W} \cdot MSS}{RTT}$$

$$\text{Subbing in } \bar{W} = \sqrt{\frac{3}{2p}}$$

$$T \approx \frac{MSS}{RTT} \cdot \sqrt{\frac{3}{2p}}$$

Which can be written in the following form as expected

$$T \approx \frac{\sqrt{\frac{3}{2}} \cdot MSS}{RTT \cdot \sqrt{p}} \text{ where constant } k = \sqrt{\frac{3}{2}} \approx 1.225$$

3. [2 pts] Queueing Delay and RTT

Considering the serialization delay, derive maximum one-way queueing delay for buffer B and link capacity C . Express measured RTT as a function of the minimum RTT of the network RTT_0 , queue occupancy Q , MSS, and C . Specify any additional queueing assumptions if you make any.

Assume single FIFO bottleneck link, link capacity C is constant, and no random loss

(a) Assume that each packet has size MSS in bits, and the bottleneck link has rate C in bit-s/s. The time to transmit one singular packet can be then represented as $T = \frac{MSS}{C}$ seconds/packet

For maximum one-way queueing delay for buffer we can consider the worst case scenario where the buffer is maxed out with B packets in the queue, and a new packet arrives and has to wait behind all B packets. In this case, the max one way queueing delay can be represented as the following:

$$T_{queue_{max}} = B \cdot T = B \cdot \frac{MSS}{C}$$

(b) Now, moving on to represent RTT as a function of RTT_0 , Q , MSS , and C

Given that $RTT = T_{prop} + T_{queue}$ with T_{prop} as the base propagation delay + processing delay, minimum RTT of the network RTT_0 is just RTT measured when there is no queueing. Under this assumption, $RTT_0 = T_{prop}$

Replacing T_{prop} with RTT_0 we are now left with the following:

$$RTT = RTT_0 + T_{queue}$$

At some given moment, the queueing delay can be represented in some form of the queueing delay formula calculated in (a) $T_{queue_{max}} = B \cdot \frac{MSS}{C}$. However, the queue occupancy does not always equal to B (as the queue may not be full and $Q \leq B$)

Therefore, replacing B with Q we get the following formula for queueing delay in general:

$$T_{queue} = Q \cdot \frac{MSS}{C}$$

Replacing T_{queue} with $Q \cdot \frac{MSS}{C}$ we are now left with the following:

$$RTT = RTT_0 + Q \cdot \frac{MSS}{C}$$

We have now shown RTT as a function of minimum RTT, Q , MSS and C

4. [2 pts] Effect of Buffer Size on Web Fetch Times Using the previous results, explain algebraically why increasing buffer B can increase short-transfer latency even if long-term throughput remains high.

From $RTT = RTT_0 + Q \cdot \frac{MSS}{C}$ or

$$RTT = T_{prop} + Q \cdot \frac{MSS}{C}$$

it is clear that RTT grows linearly with B, since algebraically, increasing the buffer size B will also subsequently increase maximum possible queue occupancy Q. Basically larger buffer size will allow for larger queue occupancy, resulting in higher RTT values

A simple model for the fetch times of a short web transfer of size F packets is:

$$T_{fetch} \approx RTT + \frac{F}{throughput}$$

However, since F values are small for short-transfers, the RTT term which is inflated due to bufferbloat dominates T_{fetch} values. Short transfers also finish in few RTT compared to large transfers, so their time to complete is dominated by RTT instead of throughput. In a long transfer, the initial inflated RTT is insignificant and the fetch time is basically dominated by F/throughput term. As a result, we see an increase in short transfer latency when we increase buffer size, as we feel the effects of the inflated RTT term more.

Given that earlier we determined that throughput is modeled as:

$$T \approx \frac{K \cdot MSS}{RTT \cdot \sqrt{p}}$$

When buffer size B increases, Q values increase as well since the queue can hold more packets. Loss probability p also decreases since buffer is increased and drops occur less frequently. Same as before, RTT increases due to more queueing delay.

Throughput depends on both RTT and drop probability, so an increase in RTT is compensated by a decrease in drop probability. As a result, it is possible that the throughput still remains high in this case, and thus bottleneck link is well utilized.

Conclusively, we have shown why buffer B increase causes an increase in short-transfer latency even though long-term throughput may remain high.