CSC458/2209 - Computer Networking Systems

Programming Assignment 2 - Bufferbloat

Department of Computer Science
University of Toronto

Fall 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.

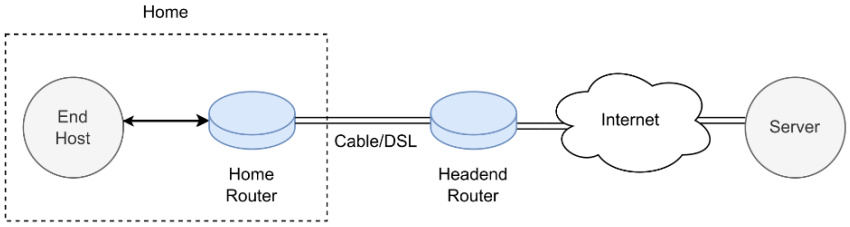

Figure 1: Typical home network topology

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.

1 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.

2 Programming Tasks

Using Mininet, create the following topology. Here, h1 represents your home computer that has a fast connection (1 Gbps) to your home router and a slower uplink connection (10 Mbps). The round-trip propagation delay or the minimum RTT between h1 and h2 is 4ms. (Assume that

the mRTT is uniformly distributed amoung the links). The router buffer can hold up to 100 full-sized Ethernet frames (about 150kB buffer size with a 1500-byte MTU).

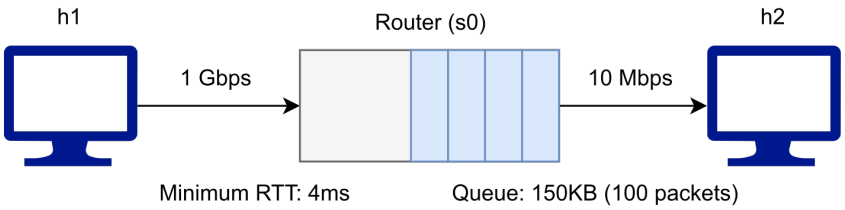

Figure 2: Bufferbloat topology

Having set up the topology,

- Start a long-lived TCP flow, sending data from h1 to h2, using iperf.
- Send pings from h1 to h2 10 times a second and record the RTTs.
- Plot the following time series:
  - CWND for The long-lived TCP flow
  - RTT reported by ping
  - Queue size at the bottleneck
- Spawn a webserver on h1. Periodically download the index.html webpage (three times every five seconds) from h1 and measure how long it takes to be fetched (on average). The starter code has hints on how this can be done. Ensure that both the long-lived TCP flow and webpage downloads send data from h1 to h2, not the reverse.
- The long-lived flow, ping train, and webserver downloads should all be happening simultaneously.
- Reduce the router buffer size to 20 packets, repeat the above experiment, and replot the three graphs.

3 Starting the Assignment

To do the assignment you will need the VM loaded with Mininet and the required dependencies of the assignment. Check the VM Setup documentation for information on how to access this VM.

To help you get started, we have provided a skeleton code. In your prepared VM, clone the starter code for this project:

```
mininet@mininet-vm:~$ git clone https://github.com/yganjali/csc458-pa-fall
    -2025.git
mininet@mininet-vm:~$ cd csc458-pa-fall-2025
mininet@mininet-vm:~/csc458-pa-fall-2025$ git checkout pa2
mininet@mininet-vm:~/csc458-pa-fall-2025$ ./scripts/setup.sh
```

Look for TODOs in the starter code. The files you need to know about are:

- **bufferbloat.py**: Creates the topology; measures CWND, queue sizes, and RTTs; and spawns a webserver.
- **plot_queue.py**: Plots the queue occupancy at the bottleneck router.

- **plot_ping.py**: Parses and plots the RTT reported by ping.
- **plot_tcpprobe.py**: Plots the cwnd time-series for a flow specified by its destination port
- **run.sh**: Runs the experiment and generates all graphs in one go.

4 Questions

Include your answers to the following questions in your report file. Remember to keep answers brief.

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

5 Theoretical Analysis

This section connects the observed TCP behavior from your Mininet experiments to analytical models of congestion control. You will use the AIMD (Additive Increase, Multiplicative Decrease) model to predict TCP throughput and latency, and then compare these predictions to empirical plots. Include all derivations in your report (typed or handwritten) and clearly state any assumptions.

5.1 Assumptions and Reference Formulas

- TCP congestion control follows AIMD.
- Bottleneck buffer is FIFO.
- CWND is measured in packets (MSS-sized payload). MSS explicitly used when converting to bytes/bits.
- Steady-state CWND cycles between $W_{\min} = W_{\max}/2$ and $W_{\max}$.
- Steady-state packet loss probability: p (loss per packet sent). Note that loss occurs only due to buffer overflow.

- Round-trip time:
$$RTT = T_{\text{prop}} + T_{\text{queue}},$$
with T_{prop} base propagation+processing delay, T_{queue} one-way queueing delay.
- Bottleneck capacity: C (packets/sec). Router buffer size: B (packets). Queue occupancy: Q, $0 \leq Q \leq B$.
- Packet serialization time at bottleneck: $\tau = 1/C$ (seconds/packet). Use C in packets/sec; convert to bits/sec via MSS if needed.

5.2 Questions

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 $N_{\text{packets}} \approx 3{W_{\max}}^2/8$
2. [3 pts] **CWND and Throughput Modeling**
Using the term derived above, show that the steady-state relationship between average CWND $\overline{W}$, loss probability p, and throughput is:
$$T \approx \frac{K \cdot \text{MSS}}{RTT\sqrt{p}}$$
and compute constant K. State all assumptions.
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.
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.

6 Submission/Deliverables

Please compress and submit all of the following files in a single file (named "pa2.tar.gz")

- **Final Code**: One goal of this assignment is for you to build a system that is easy to rerun and reproduce the results. Therefore, your final code MUST be runnable as a single shell command (`sudo ./scripts/run.sh`). We will test your code in the same setup.
- **README**: A README file with instructions to reproduce the results. Please list installation instructions for any dependencies required to run your code.
- **Report**: A PDF file containing the answers to all the questions (answers to questions related to Mininet and answers to theoretical questions). Please identify your answers with the question number, and please keep your answers brief. The responses to theoretical questions may be handwritten and subsequently scanned; however, using LaTeX format is preferred.

- **Plots**: There should be 6 plots in total, 3 each for router buffer sizes 100 and 20 packets. They MUST have the following names and be present in your submission folder
  - bb-q20/cwnd-iperf.png
  - bb-q20/q.png
  - bb-q20/rtt.png.
  - bb-q100/cwnd-iperf.png
  - bb-q100/q.png
  - bb-q100/rtt.png.

7 Notes

- When running the curl command to fetch a webpage, please fetch `<webserver_ip_address>/http/index.html`.
- Use `sudo ./scripts/clean.sh` to clean up the program.

8 Grading

Total: 28 pts

- 6 points: Producing working code that generates the graphs
- 10 points: Written questions in Section 4 (Mininet-based)
- 10 points: Theoretical analysis questions in Section 5.2
- 2 points: Producing correct graphs

9 Useful Hints

- If you have been running your code using ssh connection to the VM, you might find it difficult to view your plots with the VM's GUI. You can, instead, start a simple HTTP server inside your assignment directory using

```
python3 -m http.server
```

  which will expose the directory where you started the server on the 8000 port. You just need to use the same port forwarding technique you used when you were setting up the VM, to forward port 8000 of the VM to port 8000 on your host OS. Then you can point a web browser to http://localhost:8000 and browse your results.

- If your Mininet script does not exit cleanly due to an error (or if you pressed Control-C), you may want to issue a clean command `sudo mn -c` before you start Mininet again.

10 Useful Links

- On the bufferbloat problem:
  - https://www.bufferbloat.net/projects/bloat/wiki/Introduction/
- More about Mininet and how it can be used:

  - http://mininet.org/walkthrough/
  - http://geekstuff.org/2018/09/26/mininet-tutorial/

- Monitoring and Tuning the Linux Networking Stack:
  - Monitoring and Tuning the Linux Networking Stack: Receiving Data
  - Illustrated Guide to Monitoring and Tuning the Linux Networking Stack: Receiving Data

- Networking tools
  - tc (Traffic Control) Manual Page
  - tcpdump Manual Page