EECS 589 (Fall 2025) Assignment

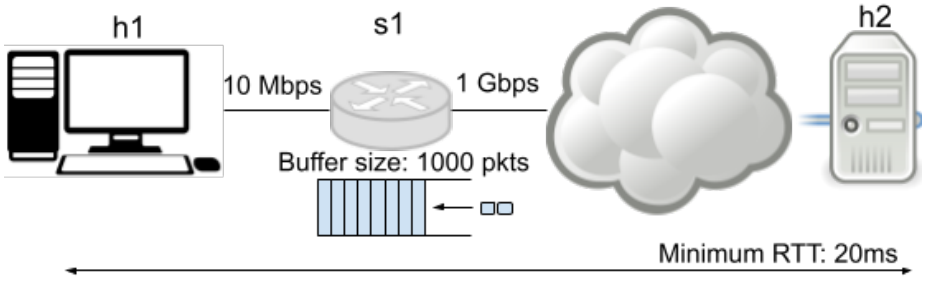
by Z. Morley Mao, Can Carlak

Due: Oct 24th 2025, 11:59 PM EST

1 Part I: Bufferbloat Analysis

1.1 Overview

In this exercise, we will investigate the behavior of TCP in home network environments, with a specific focus on the phenomenon known as "Bufferbloat." Refer to the figure below, a typical home network in which the end host is connected to a home router over a relatively low-capacity link. Beyond the home router, data travels through higher-speed routers and connections across the Internet before reaching the remote server. Our objective is to observe and analyze the TCP dynamics that occur when downloading data from a remote server to the end host similar to this network configuration.



1.2 Tasks

Within Mininet, create the following topology. Here h2 is a remote server (e.g. google.com) on the Internet that has a fast connection (1Gb/s) to your home router with a slow downlink connection (10Mb/s). The round-trip propagation delay or the minimum RTT between h1 and h2 is 20ms. The router buffer size can hold 1000 full-sized ethernet frames (about 1500KB with an MTU of 1500 bytes).

Perform the following tasks:

1. Instantiate the topology with the mentioned link settings.

2. Start a long-lived TCP flow sending data from h2 to h1. You can use iperf/iperf3 to do that. The flow should last at least 30 seconds.

3. Send pings from h1 to h2 10 times a second for at least 30 seconds, and record the RTTs.

4. Spawn a HTTP webserver on h2. Then periodically download an index.html web page (roughly 1 time every 2 seconds) to h1 and measure the average page download time (i.e. how long it takes to fetch it). Make sure that the webpage download data is going in the same direction as the long-lived flow, i.e. from h2 to h1.

5. Reduce the switch s1 buffer size from 1000 to 20, and rerun the above steps.

6. Plot the time series of the following data for each case:

   (a) The RTT reported by the ping test.

   (b) Webpage download time.

   (c) The long-lived TCP flow's congestion window size (CWND).

   (d) Queue size at the switch bottleneck. (The switch buffer/queue monitoring thread is given in the started code.)

Note that the long-lived flow, and the short-flows (ping tests or web server downloads) should be happening **simultaneously**.

1.3 Questions

After finishing the experimental tasks, answer the following questions in your answers.txt file. The answer to each question should be concise within a few sentences.

1. Report your page download time for different queue sizes. Do you see a difference in webpage fetch times with large vs small router buffers? If so, why, if not why not?

2. Does the RTT reported by ping vary with the queue size? Describe the relationship between the two.

3. Can bufferbloat occur in different parts of the end-to-end system other than the routers? Give examples.

4. Identify and describe two ways to mitigate the bufferbloat problem.

2 Part II: SDN Controller Integration

2.1 Overview

In this part, we are tasked with mitigating bufferbloat using POX SDN controller which leverages different QoS queues present on the switch for traffic management. Specifically, we want to configure the network topology, set up different queues on the switch, and use POX SDN controller to program flow rules that assign traffic to these queues.

2.2 Tasks

The topology creation for Part-II, and QoS queue configuration is handled by the starter code and the scripts. Your main objective is to modify the custom POX controller logic to mitigate the bufferbloat problem.

1. Run the template controller code by placing the *qos_template.py* under pox/ext/ directory. Under pox root directory, start the controller by running *python ./pox.py samples.pretty_log qos_template*.
2. Implement custom forwarding rules in your controller script. Consider using different queues for latency sensitive traffic. Check Working with packets and Enqueue of sections in POX API.
3. Start a long-lived TCP flow from h2 to h1 and record the throughput for every 100 ms for at least 30 seconds. Plot the results.
4. Initiate a short-lived RTT measurement from h1 to h2, record the round-trip delay for every 100 ms for at least 30 seconds. Plot the results.

2.3 Questions

After finishing the tasks for Part-II, answer the following question in your answers.txt file.

1. What limitations might you encounter when scaling this bufferbloat mitigation technique to larger or more dynamic real networks?

3 Measurement Tips

While you may use any method for monitoring or measuring, as long as it is documented in your README file, the following approaches may be useful:

- For measuring "CWND" on the sender host: **iperf3** ... -i <interval> ...
- For monitoring the queue size: sudo tc -s qdisc show dev <interface name of the device>. Look for the *backlog* variable.
- For measuring RTT: ping ...

- For starting a simple web server: python -m http.server < *port* >.
- For measuring web page fetch time: curl or wget.

You can execute these commands either externally by attaching Mininet hosts using *mnexec*, or through Mininet's Python API as sub processes or threads. The switch buffer/queue monitoring thread is given in the started code.

4 Mininet + POX Environment

To install Mininet as a virtual machine, follow the official Mininet installation guide, specifically Option 1 (recommended), which includes the POX SDN controller. The Mininet VM Setup Notes provide detailed instructions on how to interact with Mininet after booting the virtual machine. Reviewing the slides from Discussions 1 and 4 may also be helpful for becoming familiar with Mininet usage.

5 Submission Instructions/Deliverables

Assignment Due: Oct, 24th 2025, 11:59 pm (Eastern Time) Please upload all of the following to canvas (using a zip file):

- Format: zip file named < *youruniqname* >-eecs589-a1.zip
- Code: one of the goals of this assignment is for you to build a system that is easy to rerun to reproduce results. Therefore, your final code MUST be runnable via your archived scripts, following your README. Please list installation instructions for any dependencies required to run your code. We will test your code in the mininet-2.3.0 ubuntu-20.04.1 VM setup.
- Answers: A file named answers.txt file with instructions to reproduce the results as well as the answers to questions. Please identify your answers with the question number, and please keep your answers brief.
- Plots: For Part-I, there should be 8 plots in total, 4 each for router buffer sizes of 1000 and 20 packets. For Part-II, 2 plots are needed in total for throughput and latency reports.

6 Grading

- 35 pts: Part-I working code & artifacts.
- 15 pts: Part-II working code & artifacts.
- 50 pts: Answers to the questions (10 pts each).

7 Acknowledgments

The bufferbloat part of this assignment is based on Prof. Changhoon Kim's Assignment 1 from Stanford CS 244: Advanced Topics in Networking.