ECS 152: Computer Networks

Fall 2023

Project 3: Congestion Control (100 points)

Due Date: 12/6/2023 at 5:00 PM

Team: The project is to be done in a team of at most 3 students. You cannot discuss your code/data with other classmates (*except* your project partners).

All submissions (including your code) will be checked for **plagiarism** against other submissions as well as the Internet including ChatGPT and other such tools. Plagiarized submissions will be entitled to **zero** points.

In this project, you are given a docker container that contains a large file, a UDP receiver and an emulated network profile (called the training profile). The network profile will set buffer size, introduce delays, determine the throughput of the link between the sender and the receiver. You can find the docker container with the file, receiver and network profile at this link. The link contains a README with instructions on scripts to build and deploy the container.

Your goal is to implement the UDP sender in Python that will send data from the large file to the receiver in a series of packets, where the size of each packet is 1024 bytes. You will implement 5 variants of the sender that implement the following congestion control protocols:

1. Stop-and-wait protocol (10 points)
2. Fixed sliding window protocol with size 100 packets (10 points)
3. TCP Tahoe (25 points)
4. TCP Reno (25 points)
5. Custom protocol (30 points)

For each UDP sender, you will measure and report the throughput (size of transmitted data/time taken to send data) in the units of bytes per second and the average of per-packet delay in the units of seconds.

- To measure throughput, start your timer as soon as you create your socket and stop your timer once you have received acknowledgements for all packets. You have to include sequence numbers in your packets to keep track of acknowledgements.
- To measure the per-packet delay, you will start your timer when you send the packet and stop the timer when you receive an acknowledgement from the receiver for that packet. In case of retransmissions, you should consider the timer to start when you sent the packet the first time and stop the timer when you finally receive the acknowledgement.

Your goal is to maximize the throughput while also minimizing the average delay. To evaluate the performance of your UDP sender, you are required to compute the following metric: **throughput/average delay per packet**.

For TCP Tahoe and TCP Reno, set the initial window size = 1 packet and the initial slow start threshold to 64 packets.

You are asked to implement a custom sender for your custom protocol. You have to describe your custom protocol in at least 200 words. Without an adequate description, your custom protocol submission will be considered invalid. You will be awarded 20 points for your custom protocol, if it outperforms TCP Tahoe and TCP Reno. You will be awarded 30 points if your custom protocol outperforms TCP Tahoe and TCP Reno by more than 10%.

We also have a competition for the best custom congestion control protocol.[1] We will test your custom protocol on two different network profiles: the training profile provided to you in the assignment and a secret test profile. The test profile will be somewhat similar to the training profile but not exactly like it. The top-3 groups in terms of the performance metric on the test profile will receive an automatic A+ for the course. The groups ranked 4 and 5 will receive an automatic A for the course.

You will submit a report that briefly describes your implementation of all congestion control protocols. The custom protocol should be adequately described in at least 200 words (use more words if the protocol is more complex). There is no word limit for the description of non-custom protocols. Your report should also include a table of throughput, delay, and the performance metric for each protocol on the training profile.

Submit separate Python files for each of your senders. You should name them as follows: **sender_stop_and_wait.py**, **sender_fixed_sliding_window.py**, **sender_tahoe.py**, **sender_reno.py** and **sender_custom.py**

Note:

- Due to inherent randomness in the network profile, you are required to measure the throughput, average delay, and performance metric 10 times. You should report the average and standard deviation of 10 measurements.
- In your final submission, make sure you remove all print/debugging statements since it could impact the performance of your implementation as measured by our wrapper testing program.
- Each program should only output 3 lines: the throughput (in bytes per second), the average packet delay (in seconds), and the performance metric (throughput/average per packet delay) separated by a comma. All numbers should be reported as floating points, rounded up to 2 decimal places with no units.

You can use ChatGPT for guidance on implementing the different protocols. But you don't have to submit separate implementations with assistance from ChatGPT.

Report: proj1_[name1]_[student_id1]_[name2]_[student_id2]_[name3]_[student_id3].pdf

At the beginning of the page, specify the following:

1. Full name of student 1 (Student ID)
2. Full name of student 2 (Student ID)
3. Full name of student 3 (Student ID)
4. Team name to mention in the contest leaderboard
5. Name of the python source code

[1] Give your custom protocol a name.

Testing Environment:

All submissions will be tested on Python 3+.

Late Submission Policy:

No late submissions are allowed. However, if you barely miss the deadline, you can get partial points upto 24 hours. The percentage of points you will lose is given by the equation below. This will give you partial points up to 24 hours after the due date and penalizes you less if you narrowly miss the deadline.

Total marks = (Actual Marks you would get if NOT late) x [1 - hours late/24]

Late Submissions (later than 24 hours from the due date) will result in zero points, *unless you have our prior permission or documented accommodation*.

——————————————— ***Best of luck*** ———————————————

Include this signed page along with your submission

Submission Page

I certify that all submitted work is my own work. I have completed all of the assignments on my own without assistance from others except as indicated by appropriate citation. I have read and understand the university policy on plagiarism and academic dishonesty. I further understand that official sanctions will be imposed if there is any evidence of academic dishonesty in this work. I certify that the above statements are true.

Team Member 1:

_____	_____	_____
Full Name (Printed)	Signature	Date

Team Member 2:

_____	_____	_____
Full Name (Printed)	Signature	Date

Team Member 3:

_____	_____	_____
Full Name (Printed)	Signature	Date