ECS 152/ECS 173: Computer Networks

Fall 2025

Project 3: Congestion Control (100 points)

Due Date: 12/3/2025 at 11:59 PM

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

All submissions (including your code, except for the custom protocol) 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 network profile). The training network profile will set buffer size, introduce delays, and set 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 a 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 (20 points)
2. Fixed sliding window protocol with size 100 packets (20 points)
3. TCP Tahoe (20 points)
4. TCP Reno (20 points)
5. Custom protocol (20 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 per-packet delay and jitter 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 acknowledgments for all packets. You have to include sequence numbers in your packets to keep track of acknowledgments.
- To measure the per-packet delay, you will start your timer when you send the packet and stop the timer when you receive an acknowledgment from the receiver for that packet. In case of retransmissions, you should consider the timer to start when you send the packet the first time and stop the timer when you finally receive the acknowledgement.
- To measure the jitter, you will need to get the average value of the difference between the packet delays of two successive packets.

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

$$Metric = \left(\frac{Throughput}{2000}\right) + \frac{15}{Average\ Jitter} + \frac{35}{Average\ delay\ per\ packet}$$

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 up to 500 words. Feel free to draw a diagram to illustrate your key idea. An adequate description is required for your custom protocol.

You will be awarded 10 extra credit points if your custom protocol outperforms TCP Tahoe and TCP Reno by more than 10%.

We will be using packet capture to measure the performance of all the protocols so make sure you use the right sequence numbers while sending the packets. You won't be awarded the extra credit if we cannot extract the sequence number from the first 4 bytes of the message as covered in the discussion. Additionally, your implementation of custom protocol should be able to compete fairly with other sources of traffic on the network. The fairness will be evaluated by running two parallel instances of the protocol when computing the final metrics.

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 network profile provided to you in the assignment and a test network profile. The test network profile will be somewhat similar to the training profile but not exactly like it. The top group in terms of the performance metric on the test profile will receive an automatic A+ for the course. The groups ranked 2 and 3 will receive an automatic A for the course. (Yes, this means that the members of the top-3 ranked groups do not need to show up for the final exam).

You will submit a report that briefly describes your implementation of all congestion control protocols. The report should also include a table of throughput, per-packet delay, jitter, 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 **proj3_[name1]_[student_id1]_[name2]_[student_id2]_[name3]_[student_id3]_sender_custom.py**

Note:

- Due to inherent randomness in the network profile, you are required to measure the throughput, average delay, average jitter, 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 output 4 values at the end: the throughput (in bytes per second), the average packet delay (in seconds), the average jitter (in seconds), and the performance metric (

[1] Give your custom protocol a name.

$(\frac{Throughput}{2000}) + \frac{15}{Average\ Jitter} + \frac{35}{Average\ delay\ per\ packet}$) separated by a comma. All numbers should be reported as floating points, rounded up to 7 decimal places with no units.

You can use ChatGPT or other AI tools for guidance on implementing your custom protocol. But you don't have to submit separate implementations with assistance from ChatGPT or other AI tools.

Report: proj3_[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

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 penalize 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