

CSE 489/589

Programming Assignment 2

Reliable Transport Protocols

Notes: (IMPORTANT)

- One of your group members select <File> - <Make a copy> to make a copy of this report for your group, and share that Google Doc copy with your teammates so that they can also edit it.
- Report your work in each section. Describe the method you used, the obstacles you met, how you solved them, and the results. You can take screenshots at key points. There are NO hard requirements for your description.
- For a certain test, if you successfully implemented it, **take a screenshot of the result from the grader as required in section 5 (required)**. You can just provide the overall result for each test.
- For a certain test, if you tried but failed to implement it, properly describe your work. We will partially grade it based on the work you did.
- **Do NOT claim anything you didn't implement.** If you didn't try on a certain protocol or test, leave that section blank. We will randomly check your code, and if it does not match the work you claimed, you and your group won't get any partial grade score for this WHOLE assignment.
- There will be **15.0** points for this report. These are NOT bonus points and will be given based on the completion of the analysis part (section 6.1).
- If you decide not to attempt the analysis part (section 6.1) of the assignment, you will still NEED to submit this report with the requirements stated in section 6.
- After you finish, export this report as a PDF file and submit it to the UBLearns. For each group, only one member needs to make the submission.

1 - Academic Integrity Policy Statement


I have read and understood the course's academic integrity policy.

2 - Group and Contributions

- Name of member 1:
 - UBITName : Sean Chiu
 - Contributions : The entire thing besides experiment 2 (not attempted)

3 - SANITY Tests

[2.0] ABT



```
timberlake.cse.buffalo.edu - PuTTY
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
Testing with MESSAGES:20, LOSS:0.0, CORRUPTION:1.0, ARRIVAL:1000, WINDOW:0 ...
Running simulator [10 Runs] ...
Run#1 [seed=1234] ... Done!
Run#2 [seed=1111] ... Done!
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
SANITY TESTS: PASS
stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} >
```

For ABT, the timeout scheme is just to wait for the timeout to happen and resend when it times out.

[5.0] GBN



The screenshot shows a PuTTY terminal window titled "timberlake.cse.buffalo.edu - PuTTY". The terminal output displays the results of 10 runs of a GBN (Go-Back-N) test. Each run is labeled "Run#X [seed=XXXX] ... Done!". The seeds used are 2222, 3333, 4444, 5555, 6666, 7777, 8888, and 9999. After the 10 runs, the output shows "PASS!". Below this, it specifies the test parameters: "MESSAGES:20, LOSS:0.0, CORRUPTION:1.0, ARRIVAL:50, WINDOW:50 ...". It then states "Running simulator [10 Runs] ..." and repeats the 10 runs with the same seeds. This second set of runs also results in "PASS!". Finally, it shows "SANITY TESTS: PASS" and a prompt "stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} >".

```
timberlake.cse.buffalo.edu - PuTTY
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
Testing with MESSAGES:20, LOSS:0.0, CORRUPTION:1.0, ARRIVAL:50, WINDOW:50 ...
Running simulator [10 Runs] ...
Run#1 [seed=1234] ... Done!
Run#2 [seed=1111] ... Done!
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
SANITY TESTS: PASS
stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} >
```

For GBN, the timeout scheme is to have the timeout based on the timeout of the packet at `send_base`. When the packet at `send_base` timeouts, we send all packets from `send_base` to `send_base + N`.

[8.0] SR



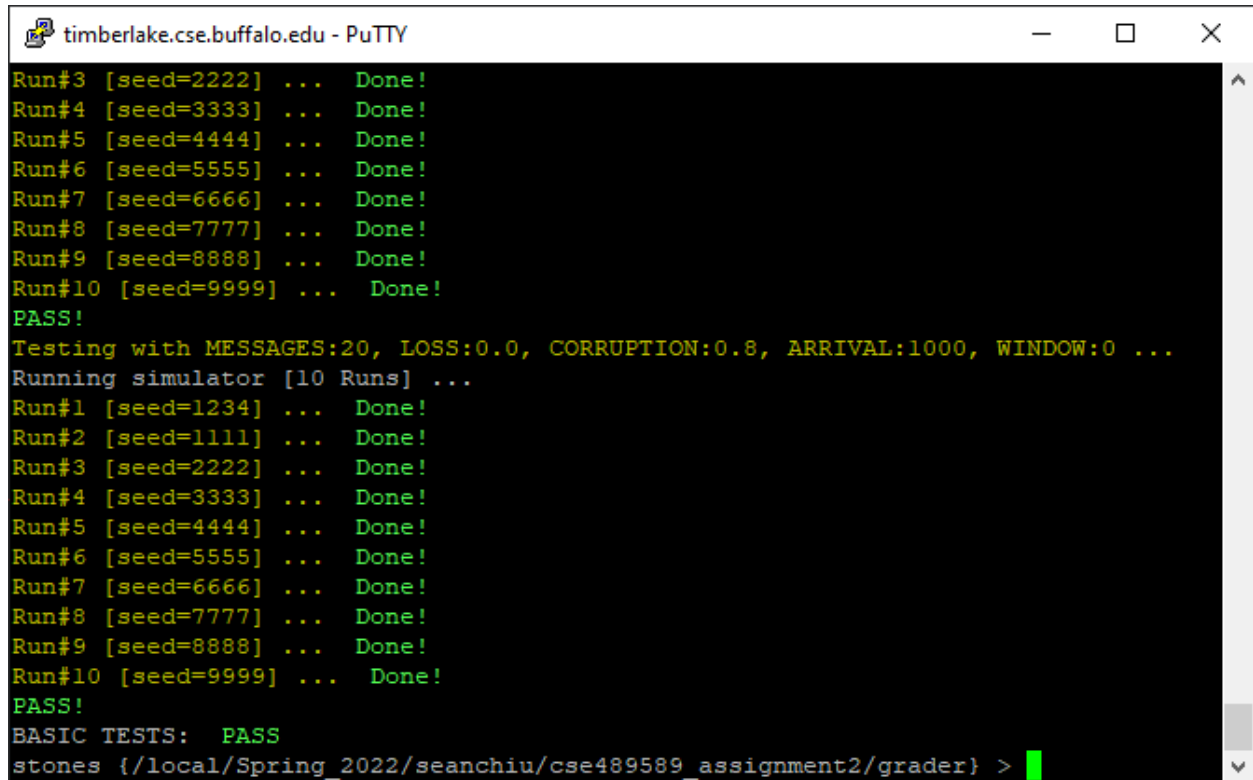
```
timberlake.cse.buffalo.edu - PuTTY
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
Testing with MESSAGES:20, LOSS:0.0, CORRUPTION:1.0, ARRIVAL:50, WINDOW:50 ...
Running simulator [10 Runs] ...
Run#1 [seed=1234] ... Done!
Run#2 [seed=1111] ... Done!
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
SANITY TESTS: PASS
stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} > █
```

For SR, I had a queue to handle multiple timeouts. This is because the order that the packets are added and sent means that we can store the time they are first sent and as they time out, we go to the next number in the queue because that is going to be the next one to timeout. When one packet timeout, it gets added to the back of the queue and the next timeout is going to be (next packet's send_time + TIMEOUT) - current_time.

[No further grading for the protocol that fails a SANITY test.]

4 - BASIC Tests

[5.0] ABT



```
timberlake.cse.buffalo.edu - PuTTY
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
Testing with MESSAGES:20, LOSS:0.0, CORRUPTION:0.8, ARRIVAL:1000, WINDOW:0 ...
Running simulator [10 Runs] ...
Run#1 [seed=1234] ... Done!
Run#2 [seed=1111] ... Done!
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
BASIC TESTS: PASS
stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} >
```

See sanity section for general timeout implementation explanation.

[12.0] GBN



The screenshot shows a PuTTY terminal window titled "timberlake.cse.buffalo.edu - PuTTY". The terminal output displays the results of 10 runs for the GBN protocol. Each run is labeled "Run#X [seed=XXXX] ... Done!". The seeds used are 2222, 3333, 4444, 5555, 6666, 7777, 8888, and 9999. After the 10 runs, the output shows "PASS!". Below this, it says "Testing with MESSAGES:20, LOSS:0.0, CORRUPTION:0.8, ARRIVAL:50, WINDOW:50 ...". Then, it says "Running simulator [10 Runs] ...". This is followed by another set of 10 runs, each labeled "Run#X [seed=XXXX] ... Done!". The seeds used are 1234, 1111, 2222, 3333, 4444, 5555, 6666, 7777, 8888, and 9999. After these runs, it shows "PASS!". Finally, it says "BASIC TESTS: PASS". The prompt "stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} >" is visible at the bottom, followed by a green cursor.

```
timberlake.cse.buffalo.edu - PuTTY
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
Testing with MESSAGES:20, LOSS:0.0, CORRUPTION:0.8, ARRIVAL:50, WINDOW:50 ...
Running simulator [10 Runs] ...
Run#1 [seed=1234] ... Done!
Run#2 [seed=1111] ... Done!
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
BASIC TESTS: PASS
stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} >
```

See sanity section for general timeout implementation explanation.

[18.0] SR



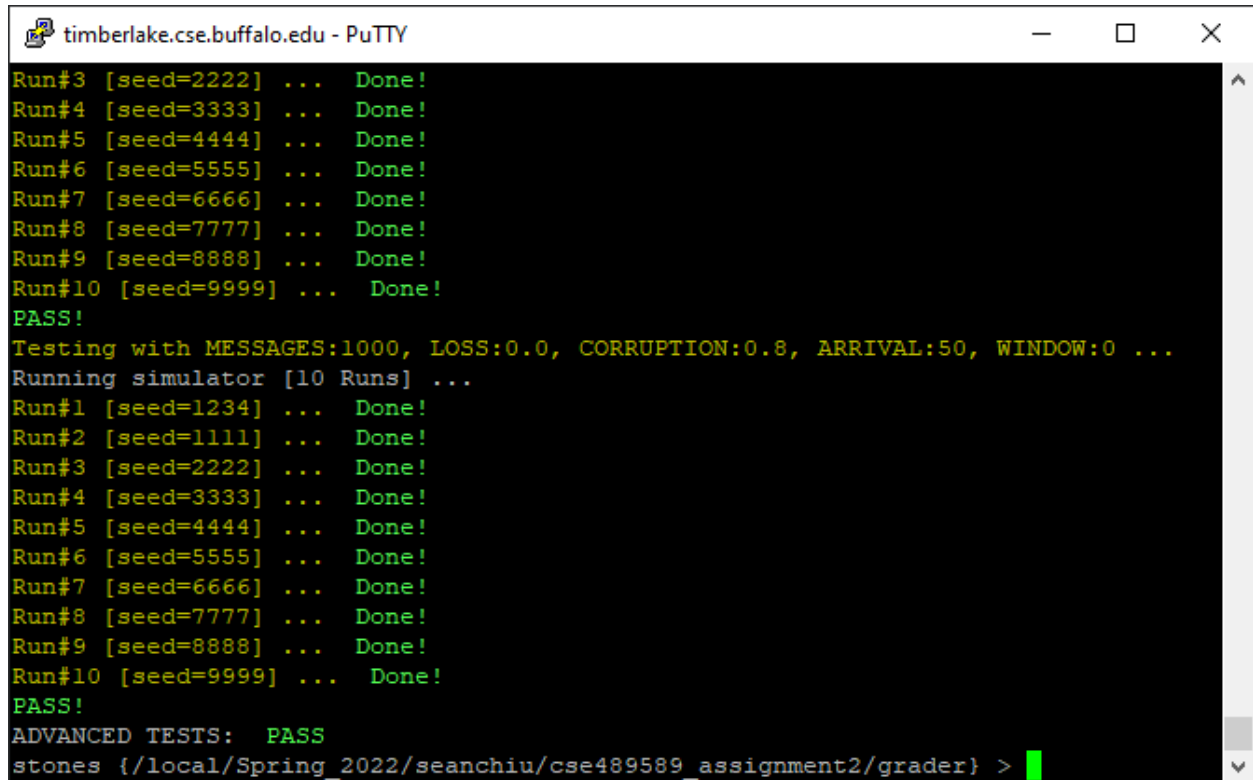
```
timberlake.cse.buffalo.edu - PuTTY
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
Testing with MESSAGES:20, LOSS:0.0, CORRUPTION:0.8, ARRIVAL:50, WINDOW:50 ...
Running simulator [10 Runs] ...
Run#1 [seed=1234] ... Done!
Run#2 [seed=1111] ... Done!
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
BASIC TESTS: PASS
stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} > █
```

See sanity section for general timeout implementation explanation.

[No further grading for the protocol that fails a BASIC test.]

5 - ADVANCED Tests

[5.0] ABT



```
timberlake.cse.buffalo.edu - PuTTY
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
Testing with MESSAGES:1000, LOSS:0.0, CORRUPTION:0.8, ARRIVAL:50, WINDOW:0 ...
Running simulator [10 Runs] ...
Run#1 [seed=1234] ... Done!
Run#2 [seed=1111] ... Done!
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
ADVANCED TESTS: PASS
stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} >
```

See sanity section for general timeout implementation explanation.

[10.0] GBN



The screenshot shows a PuTTY terminal window titled "timberlake.cse.buffalo.edu - PuTTY". The terminal output displays the results of 10 runs for the GBN protocol. Each run (Run#3 to Run#10) shows a seed value and a "Done!" status. After the 10 runs, the output shows "PASS!" and "Testing with MESSAGES:1000, LOSS:0.0, CORRUPTION:0.8, ARRIVAL:50, WINDOW:10 ...". This is followed by another set of 10 runs (Run#1 to Run#10) with different seed values, all showing "Done!". After these runs, the output shows "PASS!" and "ADVANCED TESTS: PASS". The prompt "stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} >" is visible at the bottom, followed by a green cursor.

```
timberlake.cse.buffalo.edu - PuTTY
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
Testing with MESSAGES:1000, LOSS:0.0, CORRUPTION:0.8, ARRIVAL:50, WINDOW:10 ...
Running simulator [10 Runs] ...
Run#1 [seed=1234] ... Done!
Run#2 [seed=1111] ... Done!
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
ADVANCED TESTS: PASS
stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} >
```

See sanity section for general timeout implementation explanation.

[20.0] SR

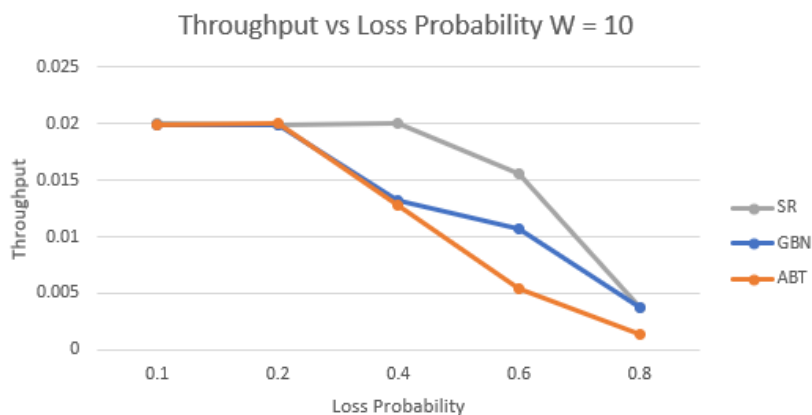
```
timberlake.cse.buffalo.edu - PuTTY
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
Testing with MESSAGES:1000, LOSS:0.0, CORRUPTION:0.8, ARRIVAL:50, WINDOW:10 ...
Running simulator [10 Runs] ...
Run#1 [seed=1234] ... Done!
Run#2 [seed=1111] ... Done!
Run#3 [seed=2222] ... Done!
Run#4 [seed=3333] ... Done!
Run#5 [seed=4444] ... Done!
Run#6 [seed=5555] ... Done!
Run#7 [seed=6666] ... Done!
Run#8 [seed=7777] ... Done!
Run#9 [seed=8888] ... Done!
Run#10 [seed=9999] ... Done!
PASS!
ADVANCED TESTS: PASS
stones {/local/Spring_2022/seanchiu/cse489589_assignment2/grader} >
```

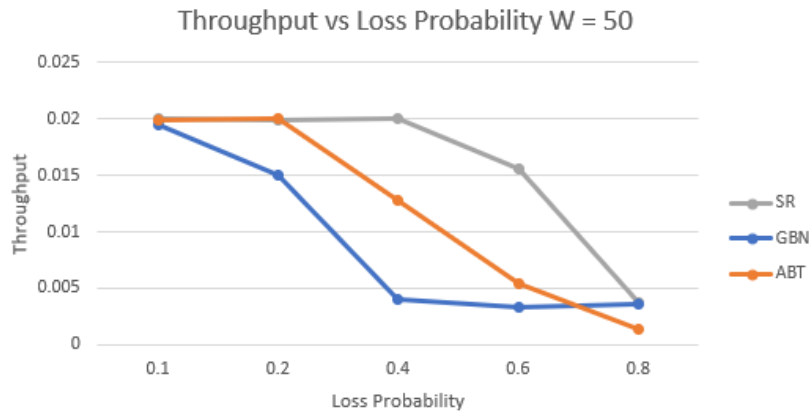
See sanity section for general timeout implementation explanation.

6 - ANALYSIS & REPORT [15.0]

Experiment 1 was attempted, Experiment 2 was not

Experiment 1





I think that increasing loss will decrease throughput and increasing window size will increase throughput. Increasing window size will increase throughput because it allows for more packets to be sent at a time which means greater chance of arriving to the other side and receiving an acknowledgement. Increasing loss will obviously decrease throughput because your packets are getting lost more frequently.

From the two graphs above, you can see that changing window size only affected GBN protocol. By decreasing window size, GBN lost throughput. This could be because GBN resend all packets after send_base. This means that the bigger the window size, the more chance send_base times out and the more packets GBN needs to send to the other side. Since SR and ABT carefully resend only unacked packets, their throughput doesn't change.

I agree with my measurements because of the nature of the protocols. Although they do not align with my initial hypothesis, after looking at the graphs and thinking about why this could be, it makes sense.