Xingkun Yin

U44255956

Jingyu Su

U58115442

Compilation instruction:

To run GBN or SR, go to GBN or SR folder and run:

- javac *.java
- java Project

Description of trace file header:

To get our trace, we piped the output of “java Project” to a txt files.

The header of the trace files are as below (from trace/sw/sw_loss_corruption.txt)

Inputs are :

Enter number of messages to simulate (> 0): [10]

Enter packet loss probability (0.0 for no loss): [0.0]

Enter packet corruption probability (0.0 for no corruption): [0.0]

Enter average time between messages from sender's layer 5 (> 0.0): [1000]

Enter window size (> 0): [8]

Enter retransmission timeout (>0.0) [15.0]

Enter trace level (>= 0): [0]

Enter random seed: [0]

100

0.2

0.2

1000

1

30

3

1234

Each input is listed below the requirement. This should be straight forward.

==========

Stop & Wait (window = 1) Working correctly

Score is out of 10 pts

==========

For stop and wait, we tested several cases by setting the window size to 1 in SR.

works for no loss and no corruption

- Our code work for this one and the trace is in trace/sw/sw_noloss_nocorruption.txt.

```
==============STATISTICS=======================
Number of original packets transmitted by A:30
Number of retransmissions by A:0
Number of data packets delivered to layer 5 at B:30
Number of ACK packets sent by B:30
Number of corrupted packets:0
Ratio of lost packets:0.0
Ratio of corrupted packets:0.0
Average RTT:9.825333414716075
Average communication time:9.825333414716075
================================================
```

works for loss and no corruption

- Our code show recovery from loss
- Trace can be found in trace/sw/sw_loss_nocorruption.txt
- recovery from DATA loss, error detection by timeout

File Edit Format View Help

```
generateNextArrival(): called
generateNextArrival(): time is 28585.39384136478
generateNextArrival(): future time for event 1 at entity 0 will be 29141.776787621686
------- A, aOutput -------
Before:
nextSeqnumOverall: 84
aBaseOverall: 84
aIncommingSeqnum: 2
A, send: 1
toLayer3: seqnum: 1  acknum: 0  checksum: 321  payload: gggggggggggggggggggg
toLayer3: packet being lost
stopTimer: stopping timer at 28585.39384136478
stopTimer: Warning: Unable to cancel your timer
startTimer: starting timer at 28585.39384136478
After:
nextSeqnumOverall: 85
aBaseOverall: 84
aIncommingSeqnum: 1
---------------------------

EVENT time: 28630.39384136478  type: 0  entity: 0
A, timeout
A, send: 1
toLayer3: seqnum: 1  acknum: 0  checksum: 321  payload: gggggggggggggggggggg
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 28630.39384136478
stopTimer: Warning: Unable to cancel your timer
startTimer: starting timer at 28630.39384136478

EVENT time: 28631.520462660596  type: 2  entity: 1
B, in order: 1
B, send ACK: 1
toLayer3: seqnum: 0  acknum: 1  checksum: 1  payload:
toLayer3: scheduling arrival on other side
```

    - The screen shot above shows a data packet from A is lost , it is detected by timeout and resent. The packet is then successfully collected by B.
- recovery from ACK loss, error detection by timeout

```
EVENT time: 26637.35038300729  type: 0  entity: 0
A, timeout
A, send: 1
toLayer3: seqnum: 1  acknum: 0  checksum: 201  payload: aaaaaaaaaaaaaaaaaaaa
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 26637.35038300729
stopTimer: Warning: Unable to cancel your timer
startTimer: starting timer at 26637.35038300729

EVENT time: 26643.65337076937  type: 2  entity: 1
B, not in order: 1
B, send ACK: 1
toLayer3: seqnum: 0  acknum: 1  checksum: 1  payload:
toLayer3: packet being lost

EVENT time: 26682.35038300729  type: 0  entity: 0
A, timeout
A, send: 1
toLayer3: seqnum: 1  acknum: 0  checksum: 201  payload: aaaaaaaaaaaaaaaaaaaa
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 26682.35038300729
stopTimer: Warning: Unable to cancel your timer
startTimer: starting timer at 26682.35038300729
```

    -
    - The above is a screen of a loss in ACK packet and was detected by timeout.

```
==============STATISTICS=======================
Number of original packets transmitted by A:89
Number of retransmissions by A:102
Number of data packets delivered to layer 5 at B:89
Number of ACK packets sent by B:131
Number of corrupted packets:0
Ratio of lost packets:0.3167701863354037
Ratio of corrupted packets:0.0
Average RTT:9.682287057100394
Average communication time:61.56031949107483
================================================
```

-
- The above is the result for loss but no corruption, stop and wait protocol. It can be found in trace/sw/sw_loss_nocorruption.txt.

works for corruption and no loss

- Our code show recovery from corruption
- In our design, when receiving a corrupt packet the sender and receiver both send the previous packet, so it does not wait for time out
- Trace can be found in trace/sw/sw_noloss_corruption.txt
- recovery from DATA corruption

```
A, retransmit, not in window: 1
A, send: 2
toLayer3: seqnum: 2  acknum: 0  checksum: 222  payload: bbbbbbbbbbbbbbbbbbbb
toLayer3: packet being corrupted
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 104.43626209295938
startTimer: starting timer at 104.43626209295938

EVENT time: 109.11761833552049  type: 2  entity: 1
B, corrupt packet
B, send ACK: 1
toLayer3: seqnum: 0  acknum: 1  checksum: 1  payload:
toLayer3: scheduling arrival on other side

EVENT time: 112.78710287789544  type: 2  entity: 0
A, retransmit, not in window: 1
A, send: 2
toLayer3: seqnum: 2  acknum: 0  checksum: 222  payload: bbbbbbbbbbbbbbbbbbbb
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 112.78710287789544
startTimer: starting timer at 112.78710287789544

EVENT time: 115.76525162511062  type: 2  entity: 1
B, in order: 2
B, send ACK: 2
toLayer3: seqnum: 0  acknum: 2  checksum: 2  payload:
toLayer3: scheduling arrival on other side
```

-
- The screen shot above shows a data packet from A is corrupted and detected by B. The packet is resent and then successfully collected by B.

- recovery from ACK corruption

```
EVENT time: 2809.884792626728  type: 2  entity: 1
B, in order: 1
B, send ACK: 1
toLayer3: seqnum: 0  acknum: 1  checksum: 1  payload:
toLayer3: packet being corrupted
toLayer3: scheduling arrival on other side

EVENT time: 2819.7993713187047  type: 2  entity: 0
A, corrupt
A, send: 1
toLayer3: seqnum: 1  acknum: 0  checksum: 401  payload: kkkkkkkkkkkkkkkkkkk
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 2819.7993713187047
startTimer: starting timer at 2819.7993713187047

EVENT time: 2828.1299478133487  type: 2  entity: 1
B, not in order: 1
B, send ACK: 1
toLayer3: seqnum: 0  acknum: 1  checksum: 1  payload:
toLayer3: scheduling arrival on other side

EVENT time: 2835.5491805780207  type: 2  entity: 0
A, in window: 1
aBaseLocal: 1
aBaseOverall: 10nextSeqnumOverall: 11
aBaseOverall: 11nextSeqnumOverall: 11
```

-
- The above is a screen of a corruption in ACK packet and was detected by A.

```
===============STATISTICS=======================
Number of original packets transmitted by A:99
Number of retransmissions by A:97
Number of data packets delivered to layer 5 at B:99
Number of ACK packets sent by B:196
Number of corrupted packets:125
Ratio of lost packets:0
Ratio of corrupted packets:0.2976190476190476
Average RTT:10.527384640033944
Average communication time:20.660686580148017
================================================
```

-
- The above is the result for corruption but no lost, stop and wait protocol. It can be found in trace/sw/sw_noloss_corruption.txt.

works for both loss and corruption

```
===============STATISTICS=======================
Number of original packets transmitted by A:28
Number of retransmissions by A:36
Number of data packets delivered to layer 5 at B:28
Number of ACK packets sent by B:51
Number of corrupted packets:15
Ratio of lost packets:0.1826086956521739
Ratio of corrupted packets:0.1595744680851064
Average RTT:9.411897422929528
Average communication time:91.43664575422103
================================================
```

-
- The above is the result for corruption andlost, stop and wait protocol. It can be found in trace/sw/sw_loss_corruption.txt.

```
================================================
```

SR (window > 1)

Score is out of 90 pts

```
================================================
```

C1: works for no loss + no corruption

```
===============STATISTICS=======================
Number of original packets transmitted by A:30
Number of retransmissions by A:0
Number of data packets delivered to layer 5 at B:30
Number of ACK packets sent by B:30
Number of corrupted packets:0
Ratio of lost packets:0.0
Ratio of corrupted packets:0.0
Average RTT:9.825333414716075
Average communication time:9.825333414716075
================================================
```

-
- Our code work for this one and the trace is in trace/sr/ sr_noloss_nocorruption.txt

For the following scenario, C2 to C5, the trace are in trace/sr/ sr_loss_corruption.txt

C2: identify (on output trace) case where ack is lost/corrupted and a later cumulative ack moves sender window by more than 1

```
EVENT time: 22019.099826044498  type: 2  entity: 1
B, in order: 11
B, send ACK: 11
toLayer3: seqnum: 0  acknum: 11  checksum: 11  payload:
toLayer3: packet being lost

EVENT time: 22100.559404278698  type: 0  entity: 0
A, timeout
A, send: 10
toLayer3: seqnum: 10  acknum: 0  checksum: 710  payload: zzzzzzzzzzzzzzzzzzzz
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 22100.559404278698
stopTimer: Warning: Unable to cancel your timer
startTimer: starting timer at 22100.559404278698

EVENT time: 22105.398968474383  type: 2  entity: 1
B, not in order: 10
B, send ACK: 11
toLayer3: seqnum: 0  acknum: 11  checksum: 11  payload:
toLayer3: scheduling arrival on other side

EVENT time: 22112.24826807459  type: 2  entity: 0
A, in window: 11
aBaseLocal: 10
aBaseOverall: 25 nextSeqnumOverall: 27
*** More than 1 ***
aBaseOverall: 27nextSeqnumOverall: 27
aBaseLocal: 12
A, reach the end of buffer, stop timer
stopTimer: stopping timer at 22112.24826807459
```

- In the image above, the ACK packet is lost can cause the window in A to move from base at 10 to 12.

C3: identify (on output trace) case where when data packet is lost/corrupted, and data is retransmitted after RTO

```
EVENT time: 24040.406506546224  type: 1  entity: 0
generateNextArrival(): called
generateNextArrival(): time is 24040.406506546224
generateNextArrival(): future time for event 1 at entity 0 will be 24341.929380169076
------- A, aOutput -------
Before:
nextSeqnumOverall: 28
aBaseOverall: 28
aIncommingSeqnum: 13
A, send: 13
toLayer3: seqnum: 13  acknum: 0  checksum: 253  payload: ccccccccccccccccccc
toLayer3: packet being lost
stopTimer: stopping timer at 24040.406506546224
stopTimer: Warning: Unable to cancel your timer
startTimer: starting timer at 24040.406506546224
After:
nextSeqnumOverall: 29
aBaseOverall: 28
aIncommingSeqnum: 14
--------------------------

EVENT time: 24130.406506546224  type: 0  entity: 0
A, timeout
A, send: 13
toLayer3: seqnum: 13  acknum: 0  checksum: 253  payload: ccccccccccccccccccc
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 24130.406506546224
stopTimer: Warning: Unable to cancel your timer
startTimer: starting timer at 24130.406506546224
```

- In the screen shot above, the Data packet is dropped

C4: identify (on output trace) case where when data packet is lost/corrupted, and data is retransmitted after receiving duplicate ack

```
EVENT time: 24343.167516098518  type: 2  entity: 1
B, corrupt packet
B, send ACK: 13
toLayer3: seqnum: 0  acknum: 13  checksum: 13  payload:
toLayer3: scheduling arrival on other side

EVENT time: 24346.49174474319  type: 2  entity: 0
A, retransmit, not in window: 13
A, send: 14
toLayer3: seqnum: 14  acknum: 0  checksum: 274  payload: ddddddddddddddddddd
toLayer3: packet being corrupted
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 24346.49174474319
startTimer: starting timer at 24346.49174474319

EVENT time: 24352.75627918333  type: 2  entity: 1
B, corrupt packet
B, send ACK: 13
toLayer3: seqnum: 0  acknum: 13  checksum: 13  payload:
toLayer3: scheduling arrival on other side

EVENT time: 24356.753715628533  type: 2  entity: 0
A, retransmit, not in window: 13
A, send: 14
toLayer3: seqnum: 14  acknum: 0  checksum: 274  payload: ddddddddddddddddddd
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 24356.753715628533
startTimer: starting timer at 24356.753715628533
```

- In the screen shot above, the data packet is lost and it is retransmitted after a duplicate ack is received.

C5: identify (on output trace) case where when data packet is lost/corrupted, and the retransmitted data is delivered and a cumulative ack moves the sender window by more than 1

```
EVENT time: 142833.61400189216  type: 0  entity: 0
A, timeout
A, send: 12
toLayer3: seqnum: 12  acknum: 0  checksum: 392  payload: jjjjjjjjjjjjjjjjjjj
toLayer3: packet being lost
stopTimer: stopping timer at 142833.61400189216
stopTimer: Warning: Unable to cancel your timer
startTimer: starting timer at 142833.61400189216

EVENT time: 142914.8228400525  type: 1  entity: 0
generateNextArrival(): called
generateNextArrival(): time is 142914.8228400525
generateNextArrival(): future time for event 1 at entity 0 will be 144440.26001770076
------- A, aOutput -------
Before:
nextSeqnumOverall: 140
aBaseOverall: 139
aIncommingSeqnum: 13
A, send: 13
toLayer3: seqnum: 13  acknum: 0  checksum: 413  payload: kkkkkkkkkkkkkkkkkkk
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 142914.8228400525
startTimer: starting timer at 142914.8228400525
After:
nextSeqnumOverall: 141
aBaseOverall: 139
aIncommingSeqnum: 14
--------------------------

EVENT time: 142924.58827478866  type: 2  entity: 1
B, in order: 13
B, send ACK: 13
toLayer3: seqnum: 0  acknum: 13  checksum: 13  payload:
toLayer3: scheduling arrival on other side

EVENT time: 142927.25138096255  type: 2  entity: 0
A, in window: 13
aBaseLocal: 12
aBaseOverall: 139 nextSeqnumOverall: 141
*** More than 1 ***
aBaseOverall: 141nextSeqnumOverall: 141
aBaseLocal: 14
A, reach the end of buffer, stop timer
stopTimer: stopping timer at 142927.25138096255
```

- In the screen shot above, a data packet is lost and cause the sender window to go from 12 to 14.

Logistics

- For our GBN Sack Protcol
    - Sender
        - We have an array list to hold all packets from layer 5.
        - On receiving the message from layer 5, we convert the message into a packet with appropriate sequence number and store it in to the buffer. We also check if the window is at its capacity, it not we send window until we reach the capacity.
        - On receiving ACK
            - If corrupt, we send the base packet

- If not corrupt
    - If the seqnum is duplicated, we send the base
    - If it is in window, we move the base to the sequence number after ACK number and send more packets to fit the window capacity

- Receiver
    - Receiver has a buffer of the same size of the window
    - On receiving the corrupted or duplicated packet, reply an ACK with the base
    - On receiving a packet with seqnum == base, move the base up until there are no packet with the sequence number, upload the newly received packet and the packets with seqnum base just looped through and delete them from base.

-
- Our checksum related functions include addChecksum(), calculateChecksum() and evaluateChecksum()
    - The way the checksum is calculated is:
        - Checksum = seqnum + seqnum + Character.getNumericVale() for evey character in the payload.
- Our code is readable as each state in the FSM is in a single clause of if else clauses. We have a lot of comments describe how the code works. The helper functions and attributes we added are in java regions, i.e.
    - //region helper function
    - //endregion
- Possible tradeoff:
    - We are storing all packet from layer 5 to A, and related time in a array list which is the same length as the amount of packets A receive from layer 5.
- Extension
    - We may reduce the usage of memory by using a constant array to store the information and delete the packets in the past that we no longer need.
- Compilation instruction is at the top of this report.

Statistics

Justification for retransmission timer:

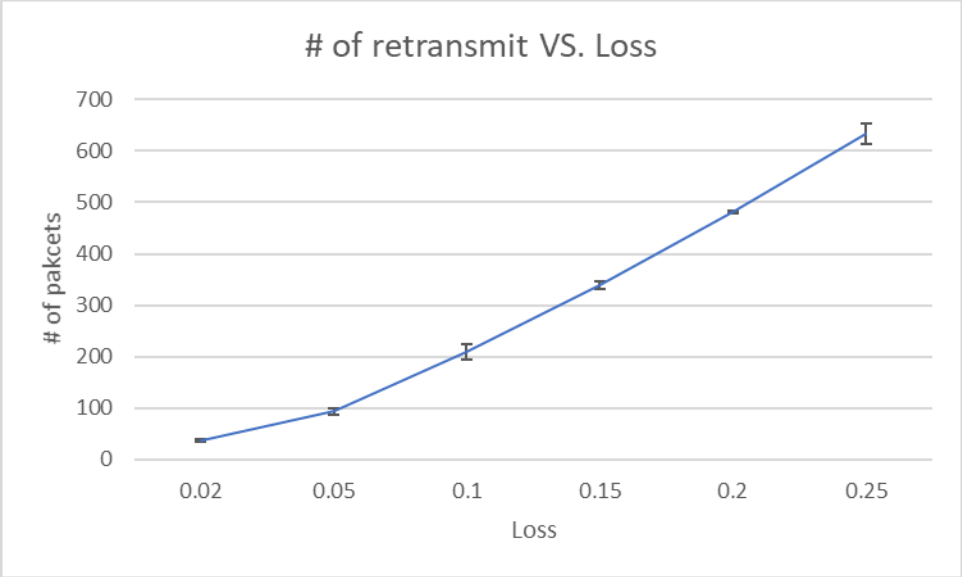
Our retransmission timer is always 90. This is because under RTO of 90, our program is able to work correctly, receiving everything sent in order. (Doesn't mean it can't work correctly with other RTO value!)

retransmits under no loss & no corruption

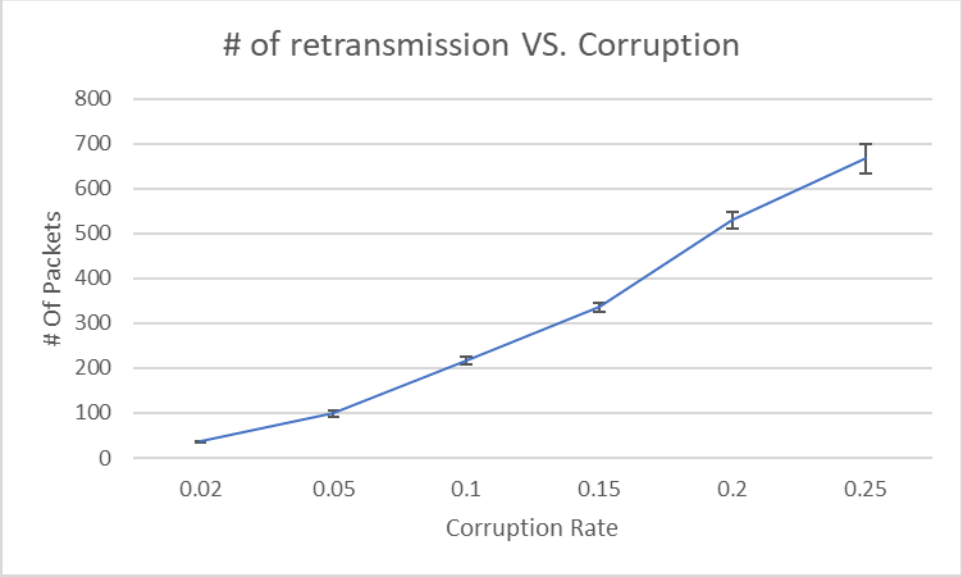
```
===============STATISTICS=======================
Number of original packets transmitted by A:30
Number of retransmissions by A:0
Number of data packets delivered to layer 5 at B:30
Number of ACK packets sent by B:30
Number of corrupted packets:0
Ratio of lost packets:0.0
Ratio of corrupted packets:0.0
Average RTT:9.825333414716075
Average communication time:9.825333414716075
================================================
```

From previous sections, we can see that when there is no loss and corruption, the number of retransmissions is 0.

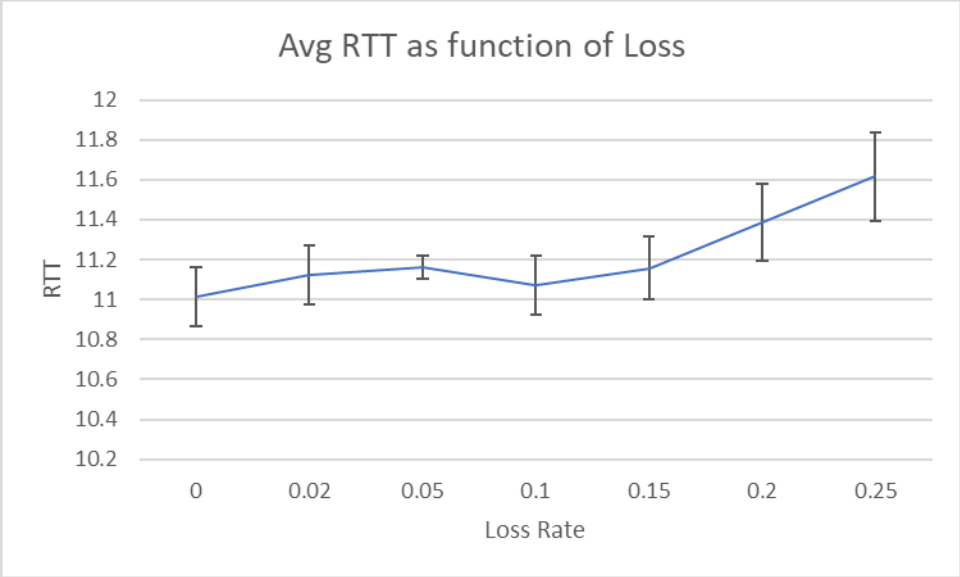
retransmits as function of loss



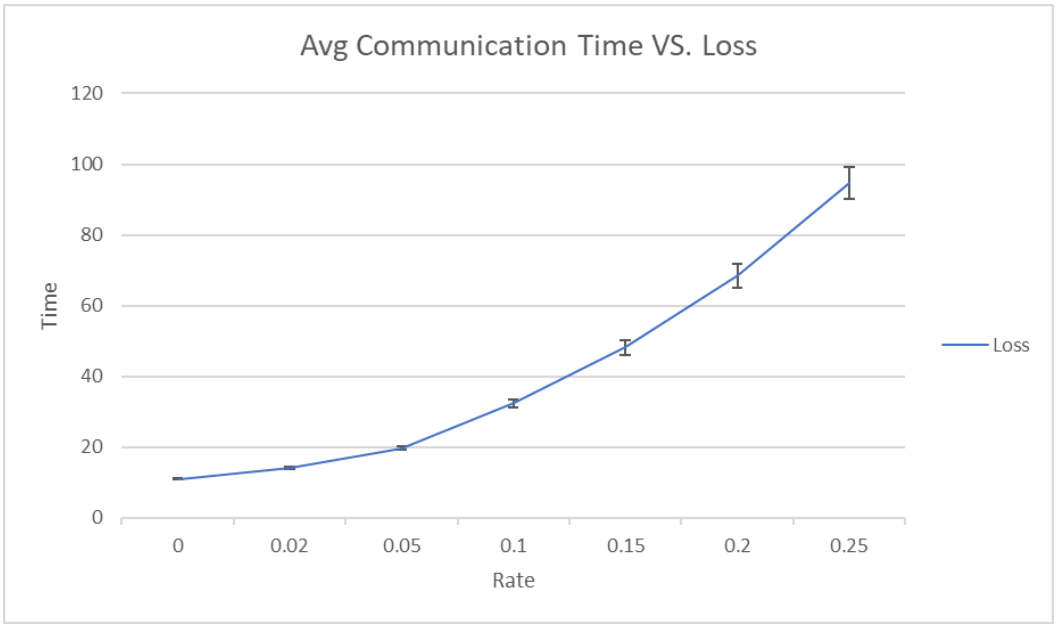
retransmits as function of corruption



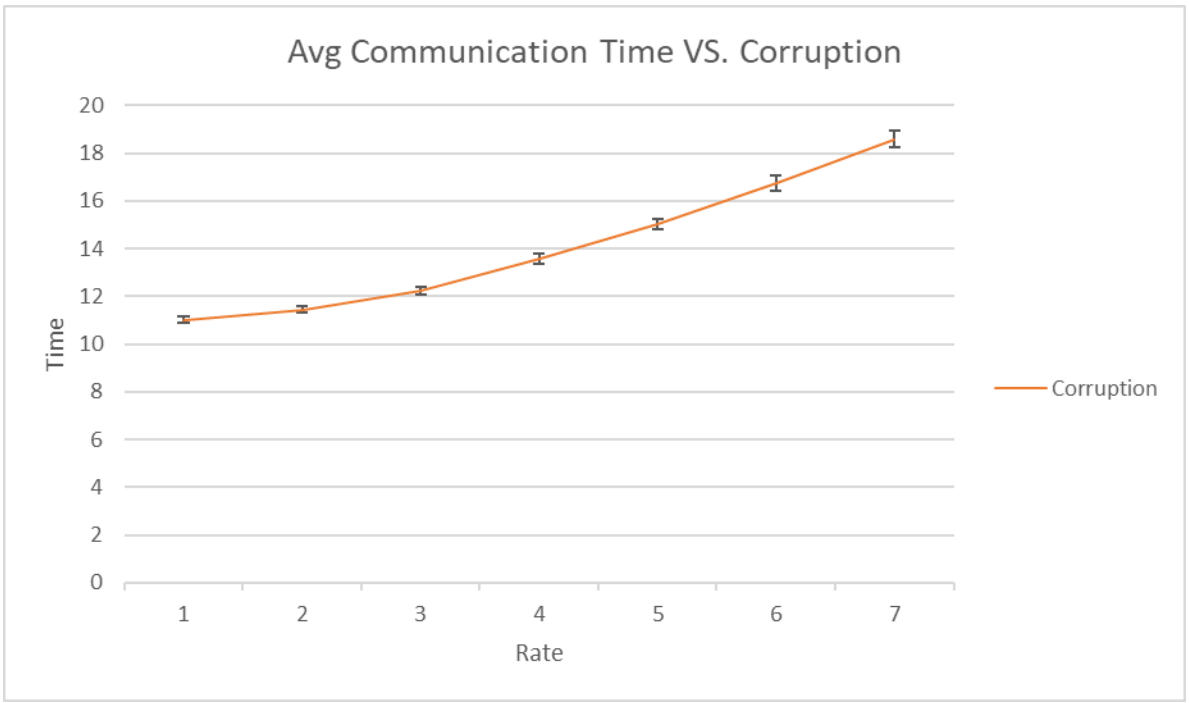
Average RTT as function of loss/corruption



Average time to communicate packet vs. Loss



Average time to communicate packet vs. Corruption



Evidence of several runs

This can be found in the .csv and.excel files submitted on csa machines.

Our program is well tested, we go over the traces thoroughly.

==========

GBN with SACK option (window >1)

Score is out 100 pts

==========

same behavior as SR for no loss & no corruption

```
==============STATISTICS=======================
Number of original packets transmitted by A:30
Number of retransmissions by A:0
Number of data packets delivered to layer 5 at B:30
Number of ACK packets sent by B:30
Number of corrupted packets:0
Ratio of lost packets:0.0
Ratio of corrupted packets:0.0
Average RTT:9.825333414716075
Average communication time:9.825333414716075
================================================
```

- From trace trace/gbn/gbn_noloss_nocorruption.txt

works for loss and no corruption

```
==============STATISTICS=======================
Number of original packets transmitted by A:30
Number of retransmissions by A:8
Number of data packets delivered to layer 5 at B:30
Number of ACK packets sent by B:34
Number of corrupted packets:0
Ratio of lost packets:0.1111111111111111
Ratio of corrupted packets:0.0
Average RTT:10.167828913235896
Average communication time:35.22833338419746
================================================
```

- From trace trace/gbn/gbn_loss_nocorruption.txt

works for no loss and corruption

```
==============STATISTICS=======================
Number of original packets transmitted by A:30
Number of retransmissions by A:5
Number of data packets delivered to layer 5 at B:30
Number of ACK packets sent by B:35
Number of corrupted packets:6
Ratio of lost packets:0.0
Ratio of corrupted packets:0.08450704225352113
Average RTT:9.797202063051067
Average communication time:11.858578244371563
================================================
```

- From trace trace/gbn/gbn_noloss_corruption.txt

works for both loss and corruption

```
==============STATISTICS=======================
Number of original packets transmitted by A:30
Number of retransmissions by A:17
Number of data packets delivered to layer 5 at B:30
Number of ACK packets sent by B:41
Number of corrupted packets:8
Ratio of lost packets:0.10227272727272728
Ratio of corrupted packets:0.10126582278481013
Average RTT:9.294828638569346
Average communication time:45.19521571499744
================================================
```

- From trace trace/gbn/gbn_loss_corruption.txt

Annotations on traces show difference between SN and GBN+SACK

```
EVENT time: 418.0058595538194  type: 2  entity: 1
B, corrupt packet
B, send ACK: 1
toLayer3: seqnum: 0  acknum: 1  checksum: 1  payload:
toLayer3: scheduling arrival on other side

EVENT time: 419.9732352671895  type: 2  entity: 0
A, retransmit, not in window: 1
A, send: 2
toLayer3: seqnum: 2  acknum: 0  checksum: 222  payload: bbbbbbbbbbbbbbbbbbbb
toLayer3: scheduling arrival on other side
stopTimer: stopping timer at 419.9732352671895
startTimer: starting timer at 419.9732352671895
^^^ retransmit window ^^^
A, send: 3
toLayer3: seqnum: 3  acknum: 0  checksum: 243  payload: ccccccccccccccccccc
toLayer3: packet being lost
stopTimer: stopping timer at 419.9732352671895
startTimer: starting timer at 419.9732352671895
```

- Notice GBN send a window, 2 packets here which does not happen in SR
- From trace trace/gbn/gbn_loss_corruption.txt

Logistics

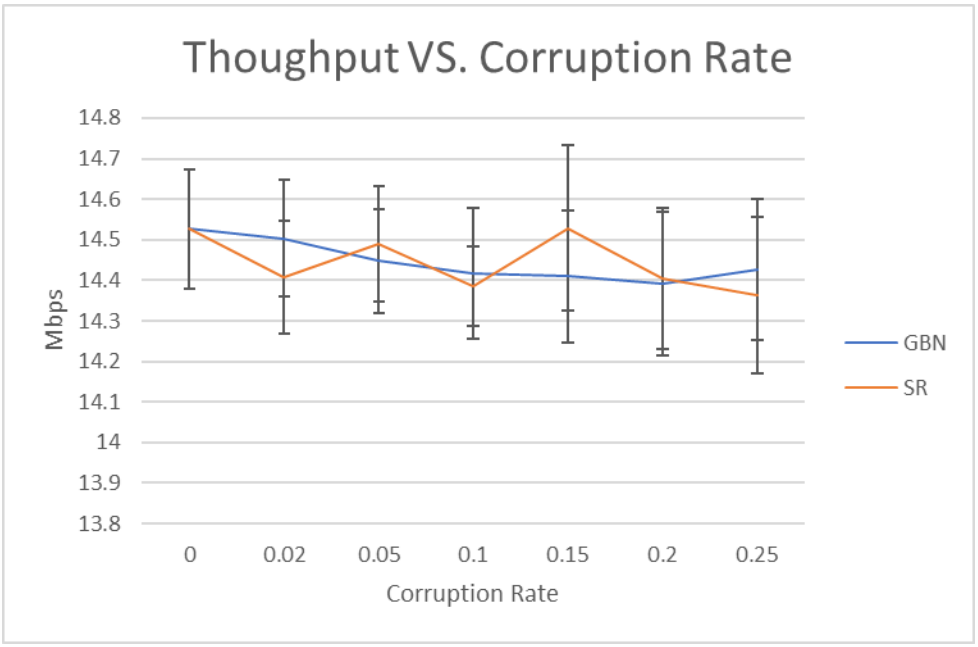
- Our code is readable as each state in the FSM is in a single clause of if else clauses. We have a lot of comments describe how the code works. The helper functions and attributes we added are in java regions, i.e.

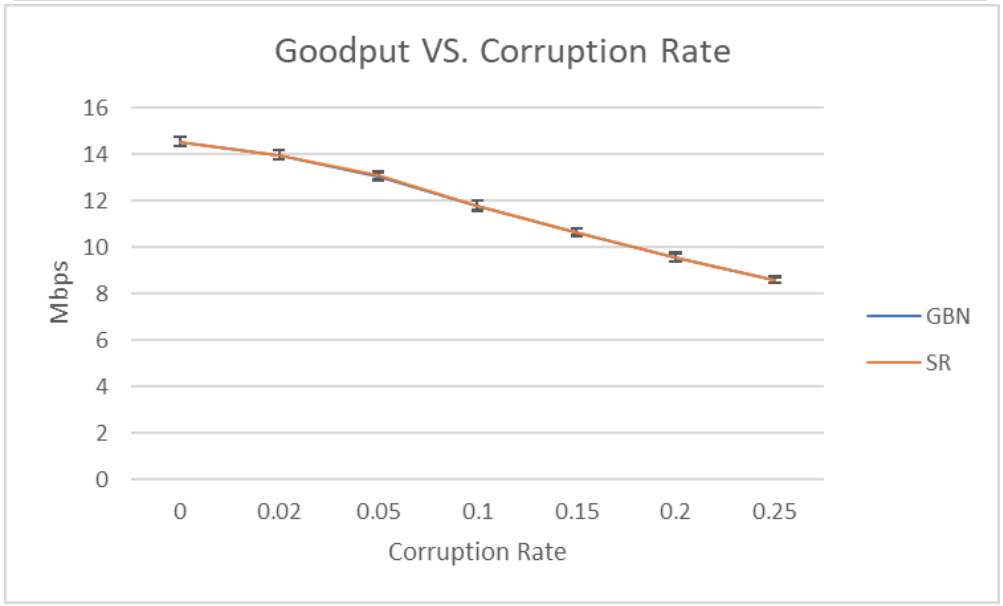
    - //region helper function
    - //endregion
- For our GBN Sack Protcol
    - Sender
        - As SR, we have an array list to hold all packets from layer 5.
        - On receiving the message from layer 5, we convert the message into a packet with appropriate sequence number and store it in to the buffer. We also check if the window is at its capacity, it not we send window until we reach the capacity.
        - On receiving ACK
            - If corrupt, we send the whole window
            - If not corrupt
                - If the seqnum is duplicated, we send the packets in window that is not in SACK
                - If it is in window, we move the base to the sequence number after ACK number and send more packets to fit the window capacity
    - Receiver
        - Receiver has a array list for SACK to record the 5 latest received packets that are in window
        - Receiver has a buffer of the same size of the window
        - On receiving the corrupted or duplicated packet, reply an ACK with the base and a SACK
        - On receiving a packet with seqnum == base, move the base up until there are no packet with the sequence number, upload the newly received packet and the packets with seqnum base just looped through and delete them from base.

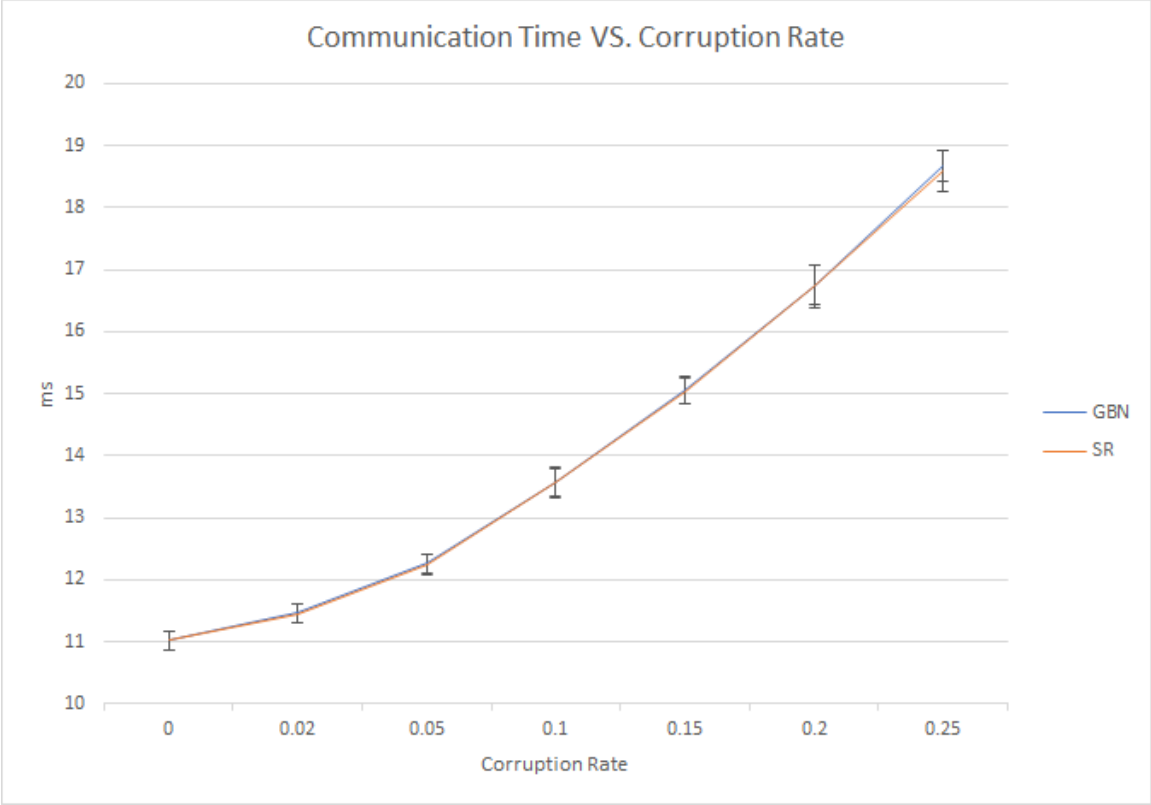
Statistics

We plotted Throughput, Goodput, average packet delay of SR vs. GBN with ACK under corruption as loss, as follows:

Throughput, Goodput, average packet delay, SR vs. GBN+SACK under corruption:

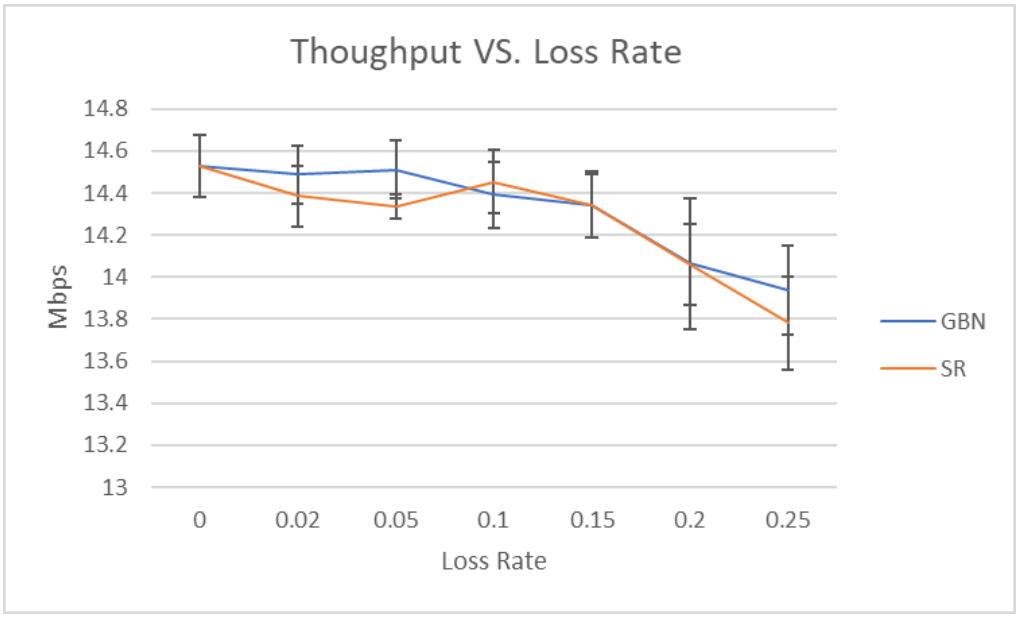

Thoughput VS. Corruption Rate
14.8
14.7
14.6
14.5
14.4
14.3
14.2
14.1
14
13.9
13.8
Mbps
0
0.02
0.05
0.1
0.15
0.2
0.25
Corruption Rate
GBN
SR


Goodput VS. Corruption Rate
16
14
12
10
8
6
4
2
0
Mbps
0
0.02
0.05
0.1
0.15
0.2
0.25
Corruption Rate
GBN
SR

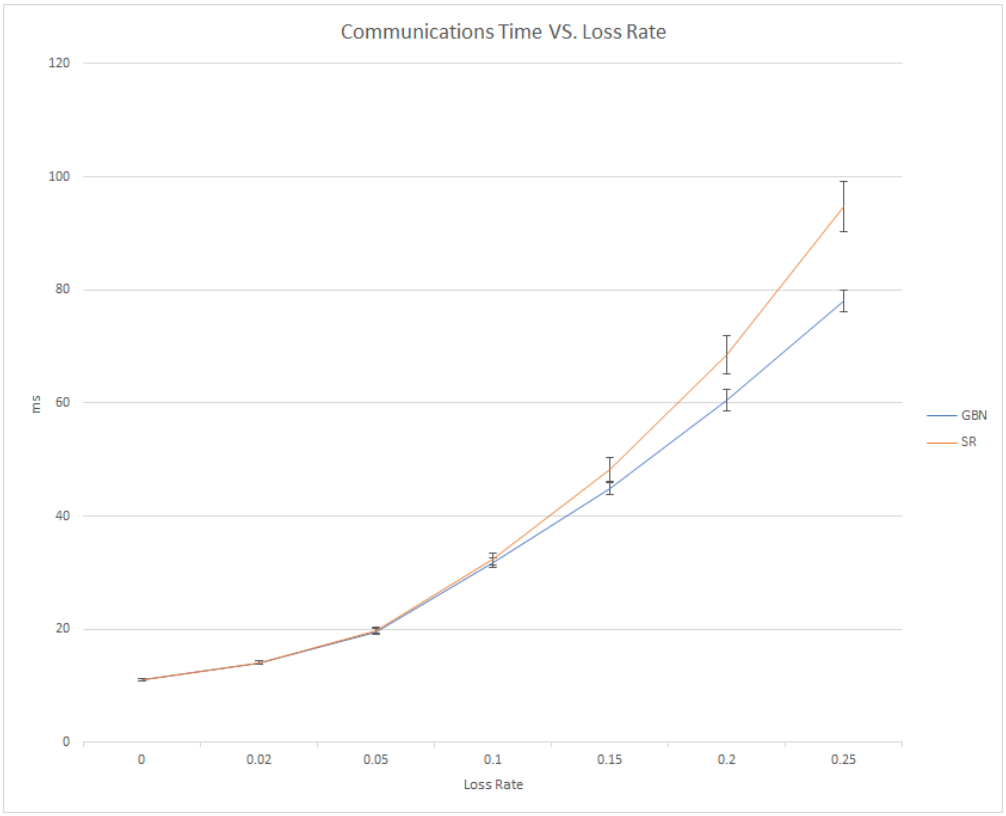


We can see that the two difference mechanisms show very similar performance under corruption, this is because in both mechanisms, both sender and receiver retransmit when corrupted packet is received.

Throughput, Goodput, average packet delay, SR vs. GBN+SACK under losses：


Thoughput VS. Loss Rate
14.8
14.6
14.4
14.2
14
13.8
13.6
13.4
13.2
13
Mbps
0
0.02
0.05
0.1
0.15
0.2
0.25
Loss Rate
GBN
SR


Goodput VS. Loss Rate
16
14
12
10
8
6
4
2
0
Mbps
0
0.02
0.05
0.1
0.15
0.2
0.25
Loss Rate
GBN
SR



The difference of performance is much larger compared to corruption. This is because when packets are lost, it is likely to trigger timeout. Since higher loss rate will result in higher possibility of other lost packets in the window, applying GBN will speed up the retransmission of successive packets, therefore result in better performance.

Evidence of several runs

This can be found in the .csv and.excel files submitted on csa machines.

Our program is well tested, we go over the traces thoroughly.