Project 2 Report:
Transport Layer Simulator

Ha Vu & Thanh Vu
Professor Amir Sadovnik
CS 305: Networks
April 1, 2017

I. Introduction

1. Goals

In this project, we focus on the logical communication of end host's processes at the transport layer. Our goal is to implement two simple reliable data transfer protocols: Go-Back-N and a packet version of TCP. The protocols should both be able to guarantee the delivery of data from one side to the other, meaning they need to handle corrupted and lost packets.

For simplicity, the model uses packet number as segnum for TCP protocol instead of byte number. The table below shows the functionalities we want to implement for each protocol [1]. The details of implementation will be elaborated in section II of the report.

	Go-Back-N	**TCP**
Sender	● Pipelines packets ● Has timer for oldest unacked packet ● Resends all unacked packet at timeout	● Pipelines packets ● Has timer for oldest unacked packet ● Resends oldest unacked packet at timeout ● Fast retransmits if gets 3 duplicate acks
Receiver	● Sends cumulative ack using highest in-order seqnum ● Discards unordered packets	● Sends cumulative ack using the next expected segnum ● Buffer unordered packets

After implementing the protocols, we aim to verify the correctness of the protocols and compare the effectiveness of the two protocols with regards to different time between messages sent, loss probability, corruption probability, and windows size. The results of both the correctness checking and the experiments are shown in section III of the report. In the conclusion section, we will describe conditions under which one protocol is preferable to the other.

2. Contributions

Ha	**Thanh**
Receiver Transport Layer (Go-back-N + TCP) checksum() ExperimentController	Sender Transport Layer (Go-back-N + TCP)

Overall design Test and Experiment Report

II. Design

1. Reliable Data Transfer Protocols

a. Go-back-N

Sender

Process	**Behaviors**
Send Message	**If the window is open** 1. Save buffer a copy of the message (unacked buffer) 2. Start timer if base == seqnum 3. Create a packet for the message 4. Pass packet to Network Layer **If the window is full** 1. Add message to the queue
Receive Packet	**If packet is not corrupted and acknum is correct** 1. Remove corresponding copies in unacked buffer 2. Update base 3. If there is still unacked messages, restart timer 4. Otherwise, stop timer 5. Call sendMessage for those in the queue
Timeout	1. Restart timer 2. Resend all unacked messages

Receiver

Process	**Behavior**
Receive packet	**If packet is not corrupted and sequence number is cumulative acknum + 1** 1. Extract message from the packet and send message to app layer 2. Increment cumulative acknum 3. Send an ack with cumulative acknum

b. TCP

Sender

Process	Behaviors
Send Message	**If the window is open** 1. Save buffer a copy of the message (unacked buffer) 2. Start timer if base == seqnum 3. Create a packet for the message 4. Pass packet to Network Layer **If the window is full** 1. Add message to the queue
Receive Packet	**If packet is not corrupted and acknum is correct** 1. Remove corresponding copies in unacked buffer 2. Update base and number of duplicate acks 3. If there is still unacked messages, restart timer 4. Otherwise, stop timer 5. Call sendMessage for those in the queue **If packet is not corrupted and acknum is duplicate** 1. Update the number of duplicate acks 2. Resend the message if there are 3 duplicate acks **Else if packet is not corrupted** 1. Send a duplicate ack with cumulative acknum
Timeout	1. Restart timer and count of duplicate acks 2. Resend unacked message with smallest seqnum

Receiver

Process	Behaviors
Receive packet	**If packet is not corrupted and sequence number of packet matches expected cumulative acknum** 1. Extract message from the packet and send message to app layer 2. Increment cumulative acknum 3. Scan the buffer to see if there are packets that could fill in the gap 4. Send the packets that could fill the gap to app layer, and increment cumulative acknum accordingly 5. Send an ack with cumulative acknum **Else if packet is not corrupted and sequence number exceeds cumulative**

	acknum 1. Buffer the packet 2. Send a duplicate ack with cumulative acknum (lower end of the gap) **Else if packet is not corrupted** 1. Send a duplicate ack with cumulative acknum

2. Design & Data Structures

a. TCP and Go-back-N

In this simulation, types of reliable data transfer protocols are set using a boolean flag. Because the model only supports two protocols TCP and Go-back-N, we simply use a *usingTCP* flag, which infers *usingTCP* = *true* as using TCP and *usingTCP* = *false* as using Go-back-N.

For both protocols, the timeout out is set using the following formula. The avgRTT being used is 10 unit time and DevRTT is 5 unit time, based on the implementation of the project's provided framework: `timeout = avgRTT + 4 * DevRTT`

b. Implementation of checksum

To protect itself against corruption, each packet has a checksum field, which is the sum of every character, converted to int type, in every field. Whenever a message arrives, the host will calculate the checksum and compare the checksum field to verify corruption.

c. Sender Transport Data Structures

`SenderTransport` uses a `LinkedList` as a queue to buffer incoming messages when needed. When messages arrive to transport layer from the application layer, if the window is open, they will be sent out immediately. However, if the window is full, messages will have to wait and will be added to the queue, which results in queuing delay. Whenever the window is open again, `SenderTranport` will send out these queuing messages in a first-in-first-out manner.

`SenderTransport` also uses a `LinkedList` to buffer all unacked messages. For Go-back-N, this buffer is used to resend all unacked packets at timeout event. For TCP, this buffer is used to resend the unacked message with smallest seqnum.

By using `LinkedList`, the complexity of buffering and sending one message is constant (O(1)).

d. Receiver Transport Data Structure

In the implementation of TCP protocol, `ReceiverTransport` uses a `TreeSet` (red-black tree) as the data structure to buffer out-of-order packets. This `TreeSet` uses a `Comparator` that sorts the packets using their sequence numbers, thus ensuring that the packets are constantly sorted when they are buffered. The complexity to add a packet to the buffer and remove a packet from the buffer is thus O(log(n)), where n is the number of packets buffered.

III. Results & Analysis

1. Experiment Setup

The goal of the experiment is to compare TCP and Go-back-N protocols in terms of total time taken to send a series of messages under different conditions. In order to verify this goal, we set up the experiment as follows:

Types of variables	**Variable**	**Definition**
Independent variables	Time between sends	The average time taken between the sending of two messages.
	Loss probability	The probability that a packet is lost
	Corruption probability	The probability that a packet, *given that it's not lost*, is corrupted
	Windows size	The windows size of the pipeline in either Go-back-N or TCP
Dependent Variable	Total time	Total time taken to send all messages
Control variables	Number of messages/ packets	Number of lines in the message file
	Timeout	The timeout used for the timer
	RTT	Round-trip delay for a packet between sender and receiver

2. Correctness Results

To verify the correctness of the experiment, we run Go-back-N and TCP twice each (for printout, see Appendix 1). For each of the protocol, we check the correctness of the experiment when there is no loss and corruption, and when there's loss and corruption.

Additionally, due to the existing impel

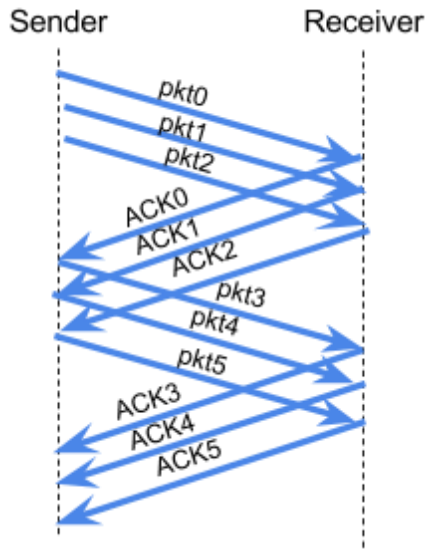
The control variables are below:

- Number of messages: 6
- Timeout: 20
- Time between sends = 3
- Windows size = 3
- RTT = 10

Because of the implementation, there is no transmission delay before sending each packet, but the first packet is sent after 1 time between sends.

a. Without loss/corruption

Without loss and corruption, assuming no out of order packets and fake timeout, TCP and Go-back-N should behave the same way.



Manual calculation:

Total time = ⌈numPackets/winSize⌉ * RTT + remainder(numPackets/winSize) * timeBtwSends

Total time = ⌈6/3⌉ * 10 + 0 * 3 =30

Actual results for Go-back-N: 88

Actual results for TCP: 67

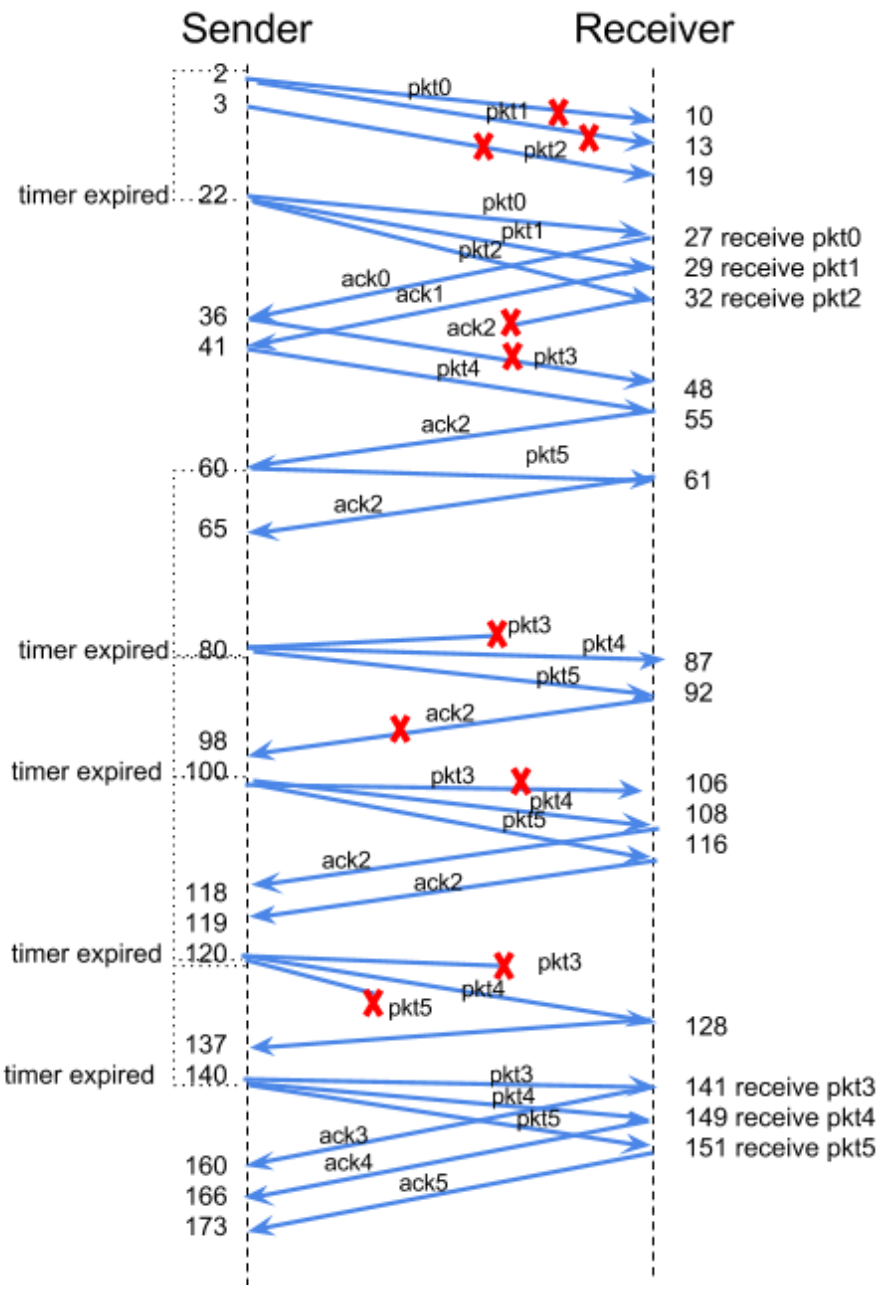
The reasons for the discrepancy is because of fake timeout due to the distribution of the RTT implemented. We took this into consideration, and increases the timeout during the experiment.

b. With loss/corruption

- Loss probability = 0.15
- Corruption probability = 0.15

Go-Back-N

The diagram of the simulation is presented below. As the diagram shows, the simulation correctly behaves according to our Go-back-N protocol.



Manual calculation:

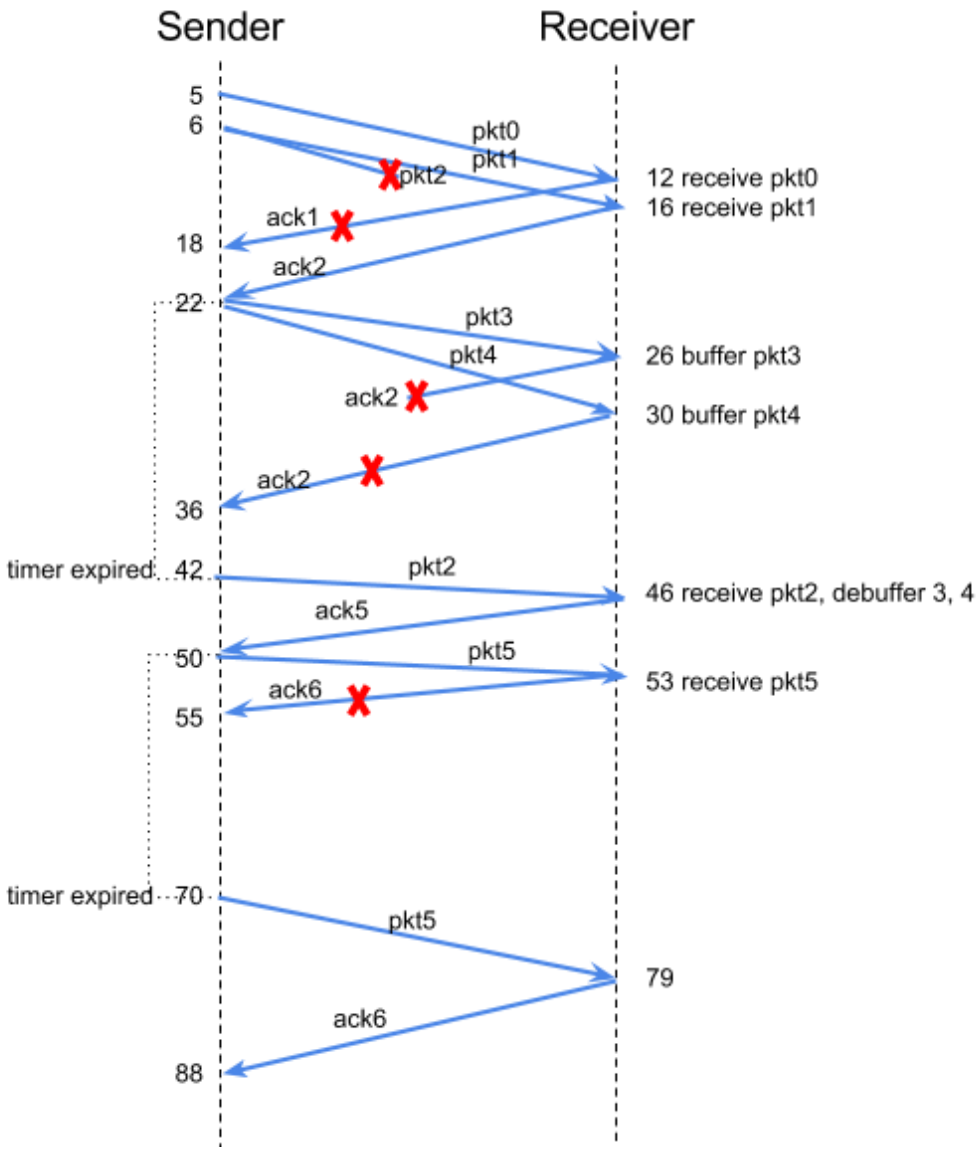
Total time = 5*timeout + 3*RTT + 3*timeBetweenSends = 5*20 + 3*10 + 3*3 = 139

Actual time: 173

The discrepancy between the manual total time and the actual time is due to the distribution of RTT and actual time between sends generated by the program.

TCP

The diagram of the simulation is presented below. As the diagram shows, the simulation correctly behaves according to our TCP protocol.



Manual calculation:

Total time = 2*timeout + 3*RTT + 1*timeBetweenSends = 2*20 + 3*10 + 1*3 = 73

Actual time: 88

The discrepancy between the manual total time and the actual time is due to the distribution of RTT and actual time between sends generated by the program.

3. Results Analysis

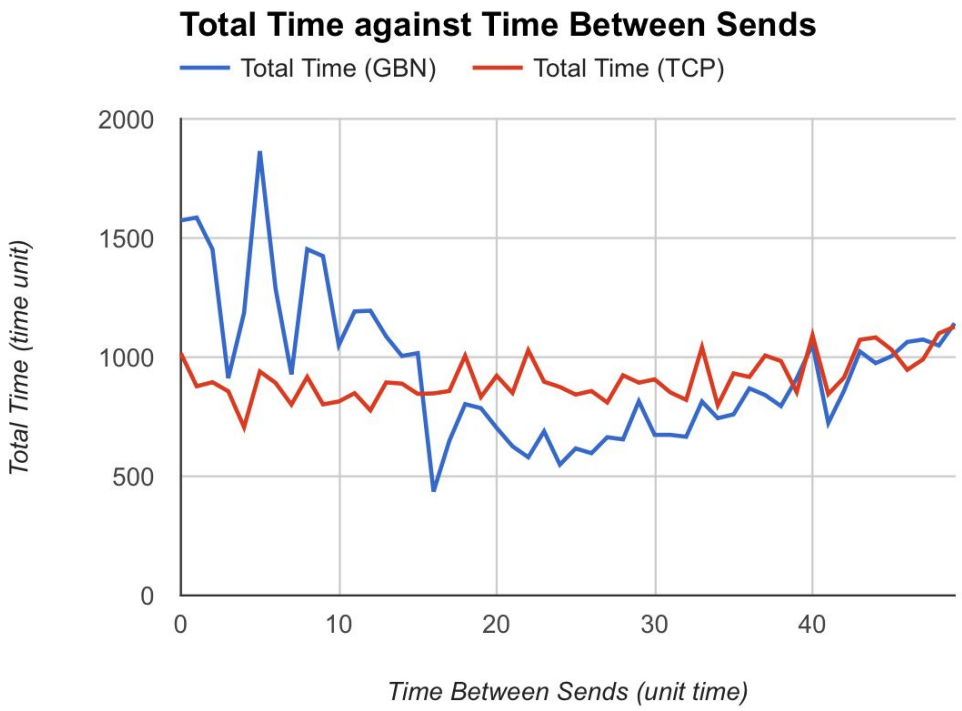
The raw data of the the experiments are shown in Appendix 2.

a. Experiment 1: Time Between Sends (TBS) vs. Total Time

Input parameters

- Number of messages:20
- Number of data points: runs = 50
- Number of trials per run: numTrialsPerRun = 6
- Smallest TBS: initialTimeBtwSends = 0
- Largest TBS: maxTimeBtwSends = 50
- Loss probability: lossProb = 0.1
- Corruption probability: corrProb = 0.1
- Window size: windowsSize = 7

Output results



Discussion

Overall, our result shows that the total time of Go-back-N is higher than that of TCP when timeBtwSends is small, but they seem to converge as timeBtwSend increases. We expect the reason for this behavior is because of the way the two protocols react to loss and corruption. At the beginning, when timeBtwSends is small, many packets are sent in a short period of time, the window tends to be filled more quickly. If packets get lost or arrive late, and timeout happens, Go-back-N will resend all unacked packets (upto 7 packets in this case) instead of 1 packet like TCP. This creates more traffic in the network and results in a higher total time and wider fluctuations.

We can see this this kind of behavior in the simulation because of the way arrival events are generated. In short, packets' arrival times are generated as either the previous arrival or the current time, whichever is greater, plus 0.5 * RTT (time for packet go from one end to the other):

```
lastArrivalTime=1+(int)(ran.nextFloat()*9)+max(lastArrivalTime,timeSoFar)
```

Thus, the more packets are sent at the same time, the larger the arrival time will be for future events, which represents longer packets' travelling time (more traffic in the network).

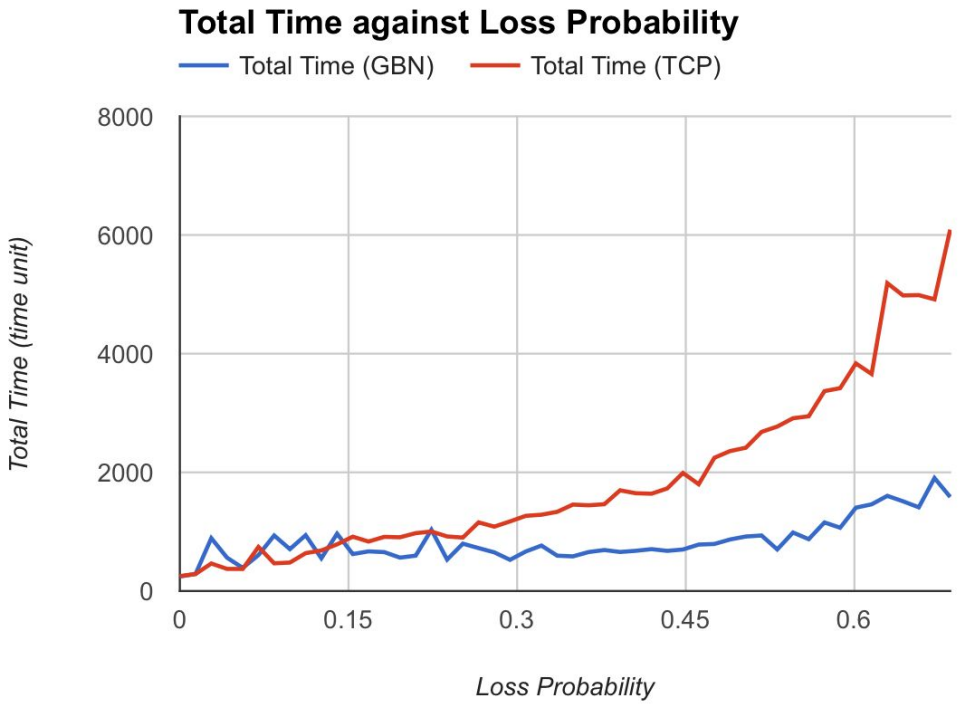
As the time between sends grows, packets are sent less frequently and it is more likely to have only one unacked packet in the window at a time. Thus, at timeout, both Go-back-N and TCP will only resend that one packet, producing the same trends of total time (their total time converge). Especially when timeBtwSends is greater RTT, the effect of pipelining is cancelled, the behaviors of both protocol become more stable.

b. Experiment 2: Loss Probability vs. Total Time

Input parameters:

- Number of messages:20
- Number of data points: runs = 50
- Number of trials per run: numTrialsPerRun = 6
- Smallest loss probability: initialLossProb = 0.0
- Largest loss probability: maxLossProb = 0.7
- Corruption probability: corrProb = 0.0
- Time between send: timeBtwSends = 10
- Window size: windowsSize = 7

Output results:



Discussion

Overall, both Go-back-N and TCP's total times increase as loss probability increases. However, TCP's total time increases at a faster rate than Go-back-N, and except for the few data points when loss probability is below 0.15, it takes a longer time than Go-back-N to transmit all the packets.

The upwards trends of both protocols are expected because both need to retransmit packets when loss occurs, and more loss means more packets retransmitted.

Go-back-N retransmits all packets once timer expires; therefore, most lost packets are transmitted back right away. Additionally, since lost packets do not generate an arrival time, if retransmitted packets get lost, it does not add to the total generated last arrival time. In total, packets get retransmitted faster, and the protocol does not flood the network.

Meanwhile, for TCP's case, it only retransmit one lost packet at a time, and therefore it is slower in filling in gaps. As there are more packets lost, the buffer mechanism becomes less effective. Even the fast retransmit does not work well, because the duplicate acks also have a high probability of being lost as loss probability go up.

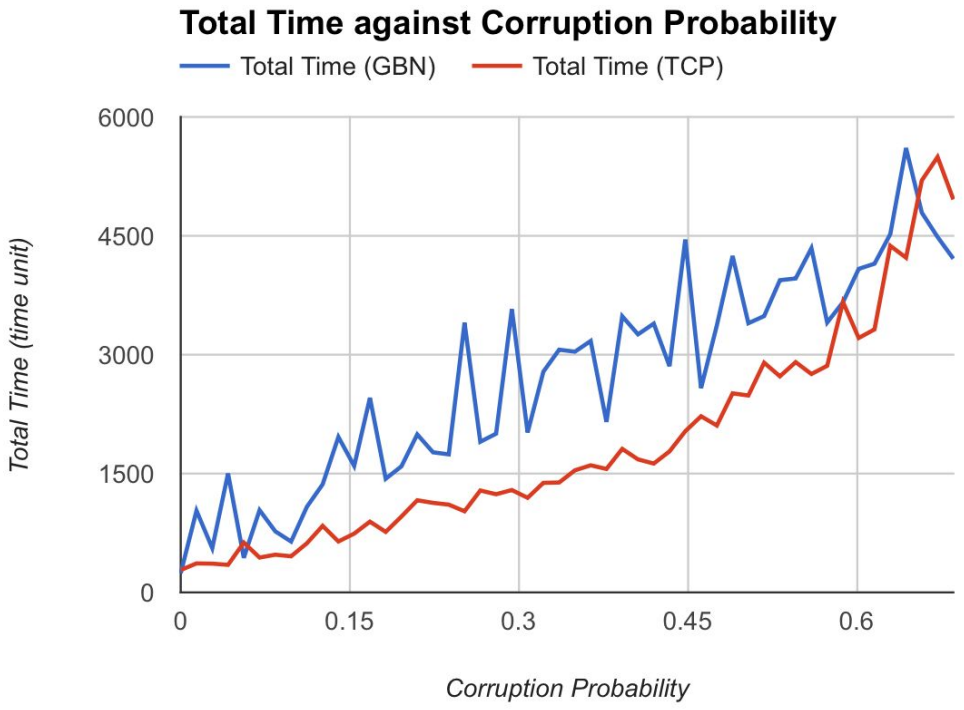
TCP seems somewhat more effective when loss probability is low, because the buffer is more effective when packets only occasionally get lost. In this case, TCP is preferably as Go-back-N would continually retransmit extra packets, leading to higher total time.

c. Experiment 3: Corruption Probability vs. Total Time

Input parameters:

- Number of messages:20
- Number of data points: runs = 50
- Number of trials per run: numTrialsPerRun = 6
- Smallest corruption prob: initialCorrProb = 0.0
- Largest corruption prob: maxCorrProb = 0.7
- Loss probability: lossProb = 0.0
- Time between send: timeBtwSends = 10
- Window size: windowsSize = 7

Output results:



Discussion

Overall, both Go-back-N and TCP's total times increase as corruption probability increases. Both protocols' total times seem to increase at a similar rate. For most of the time, Go-back-N takes more time to transmit all packets than TCP. Go-back-N total times also seem to be a lot more volatile than TCP's total time.

The upwards trends of both protocols are expected because both need to retransmit packets when corruption occurs, and more corruption means more packets retransmitted.

While the two protocols behave similarly when looking into either loss or corruption, corrupted packets do not generate an arrival time whereas lost packets do not. This means that as corruption probability goes up, while the corrupted packets do get retransmitted faster in Go-back-N, the network is more likely to get flooded with packets. This is also the reason why Go-back-N is more volatile: the more packets get sent and corrupted, the more the actual distribution of corruption affects the total time.

Meanwhile, TCP's results for corruption probability are similar to those for loss probability, because it responds similarly in both scenarios.

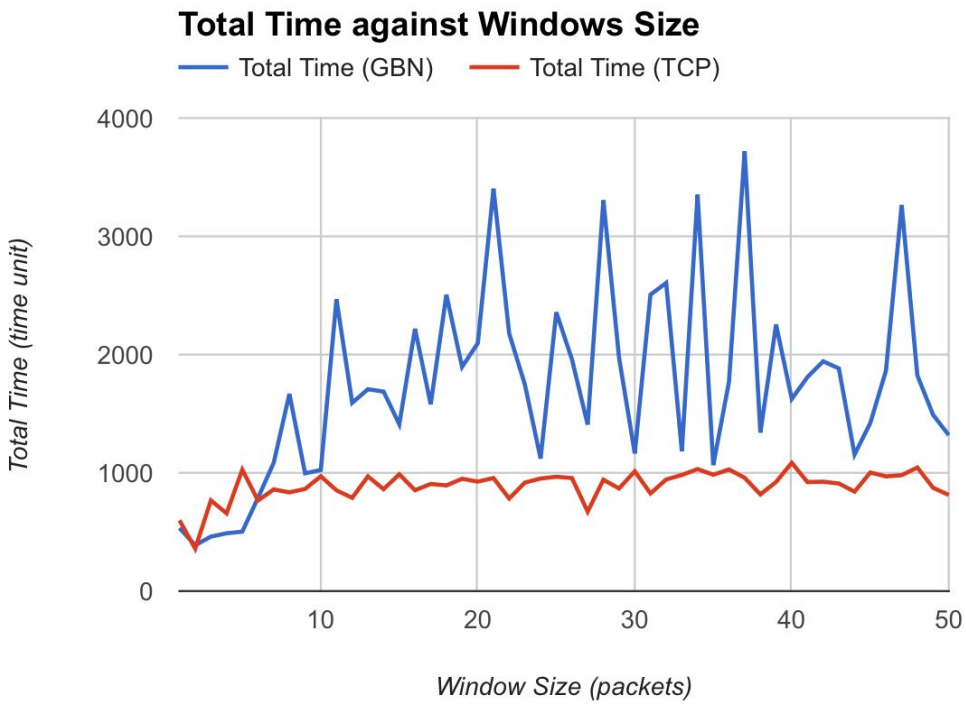
In general, TCP outperforms Go-back-N when there is corruption, in both total time and consistency.

d. Experiment 4: Window Size vs. Total Time

Input parameters:

- Number of messages:20
- Number of data points: runs = 50
- Number of trials per run: numTrialsPerRun = 6
- Initial window size: initialSize = 1
- Increment of window size: sizeIncrement = 1
- Loss probability: lossProb = 0.1
- Corruption probability: corrProb = 0.1
- Time between send: timeBtwSends = 10

Output results:



Discussion

Overall, as window size increases, the total time of both protocols do not trend upwards or downwards. However, Go-back-N's total time fluctuates much more than TCP's. In addition, in general, Go-back-N transmits the packets in higher total time than TCP.

We suspect the volatility of Go-back-N is due to the same reason explained in Experiment 1. Because Go-back-N sends all unacked packets at timeout, its total time is much more sensitive to the positioning of lost and corrupted packets. In addition, the number of retransmitted packets are related to the window size and the timeout. Since our timeout does not change, after the window size increases to a certain point, the number of retransmitted packets stop increasing, and that may explain why volatility stays the same after windows size passes 10 packets.

The reason why TCP has lower total time could be attributed to TCP's fast retransmit mechanism. When the windows size is low, there is a lower possibility that 3 duplicate acks would be retransmitted, as there are not a lot of packets transmitted in a continuous windows, which might explain why Go-back-N has a lower time at the beginning when windows size is very small.

In general, TCP is preferable to Go-back-N as windows size increases.

IV. Conclusion

There seems to be no clear trend as to whether TCP or Go-back-N are preferable across all conditions; depending on the conditions of the network, either protocols would be working better.

Based upon our experiments, Go-back-N outperforms TCP when there is high time between packets sends, high loss probability, and small windows size. TCP is preferable when there is low time between packets sent, any possibility of corruption, low loss probability, and big windows size.

A limitation of our experiment is that in our simulation, there is no consideration for the possibility of out-of-order packets. This possibility exploits TCP's capability of buffering out-of-order at the receivers, potentially making TCP more preferably to Go-back-N, as Go-back-N will flood the network and drop a lot of packets if there are many out-of-order packets.

V. References

[1] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-down Approach*. Addison-Wesley Longman, 2013.

Appendix 1: Raw correctness printout

Check correctness for Go-back-N - No Loss/Corruption

[NS] Message sending from sender to receiver at time 0
[NL] Packet seq: 0 ack: -1 sent Message0

[NS] Message sending from sender to receiver at time 0
[NL] Packet seq: 1 ack: -1 sent Message1

[NS] Message sending from sender to receiver at time 2
[NL] Packet seq: 2 ack: -1 sent Message2

[NS] Message sending from sender to receiver at time 5
[ST] Buffered message
[ST] Current buffered messages: 1

[NS] Message sending from sender to receiver at time 7
[ST] Buffered message
[ST] Current buffered messages: 2

[NS] Message arriving from sender to receiver at time 8
[RT] Receive packet at receiver; seqnum: 0 acknum: -1 msg: Message0
from receiver:Message0
[NL] Packet seq: -1 ack: 0 sent ACK

[NS] Message sending from sender to receiver at time 11
[ST] Buffered message
[ST] Current buffered messages: 3

[NS] Message arriving from sender to receiver at time 13
[RT] Receive packet at receiver; seqnum: 1 acknum: -1 msg: Message1
from receiver:Message1
[NL] Packet seq: -1 ack: 1 sent ACK

[NS] Message arriving from sender to receiver at time 17
[RT] Receive packet at receiver; seqnum: 2 acknum: -1 msg: Message2
from receiver:Message2
[NL] Packet seq: -1 ack: 2 sent ACK

[NS] Message arriving from receiver to sender at time 19
[ST] Receive packet at sender; seqnum: -1 acknum: 0
[ST] Current queuing messages: 3, open windows: 1
[ST] Sending the next message in the queue
[NL] Packet seq: 3 ack: -1 sent Message3

[NS] Message arriving from receiver to sender at time 22
[ST] Receive packet at sender; seqnum: -1 acknum: 1
[ST] Current queuing messages: 2, open windows: 1
[ST] Sending the next message in the queue
[NL] Packet seq: 4 ack: -1 sent Message4

[NS] Message arriving from receiver to sender at time 28
[ST] Receive packet at sender; seqnum: -1 acknum: 2
[ST] Current queuing messages: 1, open windows: 1
[ST] Sending the next message in the queue
[NL] Packet seq: 5 ack: -1 sent Message5

[NS] Message arriving from sender to receiver at time 36
[RT] Receive packet at receiver; seqnum: 3 acknum: -1 msg: Message3
from receiver:Message3
[NL] Packet seq: -1 ack: 3 sent ACK

[NS] Message arriving from sender to receiver at time 41
[RT] Receive packet at receiver; seqnum: 4 acknum: -1 msg: Message4
from receiver:Message4
[NL] Packet seq: -1 ack: 4 sent ACK

[NS] Timer expired at time 48
[NL] Packet seq: 3 ack: -1 sent Message3
[NL] Packet seq: 4 ack: -1 sent Message4
[NL] Packet seq: 5 ack: -1 sent Message5

[NS] Message arriving from sender to receiver at time 49
[RT] Receive packet at receiver; seqnum: 5 acknum: -1 msg: Message5
from receiver:Message5
[NL] Packet seq: -1 ack: 5 sent ACK

[NS] Message arriving from receiver to sender at time 53
[ST] Receive packet at sender; seqnum: -1 acknum: 3

[NS] Message arriving from receiver to sender at time 57
[ST] Receive packet at sender; seqnum: -1 acknum: 4

[NS] Message arriving from sender to receiver at time 58
[RT] Receive packet at receiver; seqnum: 3 acknum: -1 msg: Message3
[NL] Packet seq: -1 ack: 5 sent ACK

[NS] Message arriving from sender to receiver at time 61
[RT] Receive packet at receiver; seqnum: 4 acknum: -1 msg: Message4
[NL] Packet seq: -1 ack: 5 sent ACK

[NS] Message arriving from sender to receiver at time 66
[RT] Receive packet at receiver; seqnum: 5 acknum: -1 msg: Message5
[NL] Packet seq: -1 ack: 5 sent ACK

[NS] Message arriving from receiver to sender at time 70
[ST] Receive packet at sender; seqnum: -1 acknum: 5

[NS] Message arriving from receiver to sender at time 75
[ST] Receive packet at sender; seqnum: -1 acknum: 5

[NS] Message arriving from receiver to sender at time 84
[ST] Receive packet at sender; seqnum: -1 acknum: 5

[NS] Message arriving from receiver to sender at time 88
[ST] Receive packet at sender; seqnum: -1 acknum: 5

Check correctness for TCP - No Loss/Corruption

[NS] Message sending from sender to receiver at time 0
[NL] Packet seq: 0 ack: -1 sent Message0

[NS] Message sending from sender to receiver at time 0
[NL] Packet seq: 1 ack: -1 sent Message1

[NS] Message arriving from sender to receiver at time 1
[RT] Receive packet at receiver; seqnum: 0 acknum: -1 msg: Message0
from receiver:Message0
[NL] Packet seq: -1 ack: 1 sent ACK

[NS] Message arriving from sender to receiver at time 2
[RT] Receive packet at receiver; seqnum: 1 acknum: -1 msg: Message1
from receiver:Message1
[NL] Packet seq: -1 ack: 2 sent ACK

[NS] Message arriving from receiver to sender at time 3
[ST] Receive packet at sender; seqnum: -1 acknum: 1

[NS] Message sending from sender to receiver at time 4
[NL] Packet seq: 2 ack: -1 sent Message2

[NS] Message sending from sender to receiver at time 4
[NL] Packet seq: 3 ack: -1 sent Message3

[NS] Message sending from sender to receiver at time 6
[ST] Buffered message
[ST] Current buffered messages: 1

[NS] Message sending from sender to receiver at time 7
[ST] Buffered message
[ST] Current buffered messages: 2

[NS] Message arriving from receiver to sender at time 12
[ST] Receive packet at sender; seqnum: -1 acknum: 2
[ST] Current queuing messages: 2, open windows: 1
[ST] Sending the next message in the queue
[NL] Packet seq: 4 ack: -1 sent Message4

[NS] Message arriving from sender to receiver at time 15
[RT] Receive packet at receiver; seqnum: 2 acknum: -1 msg: Message2
from receiver:Message2
[NL] Packet seq: -1 ack: 3 sent ACK

[NS] Message arriving from sender to receiver at time 21
[RT] Receive packet at receiver; seqnum: 3 acknum: -1 msg: Message3
from receiver:Message3
[NL] Packet seq: -1 ack: 4 sent ACK

[NS] Message arriving from sender to receiver at time 27
[RT] Receive packet at receiver; seqnum: 4 acknum: -1 msg: Message4
from receiver:Message4
[NL] Packet seq: -1 ack: 5 sent ACK

[NS] Timer expired at time 32
[NL] Packet seq: 2 ack: -1 sent Message2

[NS] Message arriving from receiver to sender at time 35

[ST] Receive packet at sender; seqnum: -1 acknum: 3
[ST] Current queuing messages: 1, open windows: 1
[ST] Sending the next message in the queue
[NL] Packet seq: 5 ack: -1 sent Message5

[NS] Message arriving from receiver to sender at time 43
[ST] Receive packet at sender; seqnum: -1 acknum: 4

[NS] Message arriving from receiver to sender at time 49
[ST] Receive packet at sender; seqnum: -1 acknum: 5

[NS] Message arriving from sender to receiver at time 54
[RT] Receive packet at receiver; seqnum: 2 acknum: -1 msg: Message2
[NL] Packet seq: -1 ack: 5 sent ACK

[NS] Message arriving from sender to receiver at time 57
[RT] Receive packet at receiver; seqnum: 5 acknum: -1 msg: Message5
from receiver:Message5
[NL] Packet seq: -1 ack: 6 sent ACK

[NS] Message arriving from receiver to sender at time 66
[ST] Receive packet at sender; seqnum: -1 acknum: 5

[NS] Message arriving from receiver to sender at time 67
[ST] Receive packet at sender; seqnum: -1 acknum: 6

Check correctness for Go-back-N - With Loss/Corruption

[NS] Message sending from sender to receiver at time 2
[NL] Packet seq: 0 ack: -1 corrupted Message0
[NL] Packet seq: 0 ack: -1 sent 78essage0

[NS] Message sending from sender to receiver at time 2
[NL] Packet seq: 1 ack: -1 corrupted Message1
[NL] Packet seq: 1 ack: -1 sent 78essage1

[NS] Message sending from sender to receiver at time 3
[NL] Packet seq: 2 ack: -1 corrupted Message2
[NL] Packet seq: 2 ack: -1 sent 78essage2

[NS] Message sending from sender to receiver at time 4

[ST] Buffered message
[ST] Current buffered messages: 1

[NS] Message arriving from sender to receiver at time 10
[RT] Receive packet at receiver; seqnum: 0 acknum: -1 msg: 78essage0

[NS] Message arriving from sender to receiver at time 13
[RT] Receive packet at receiver; seqnum: 1 acknum: -1 msg: 78essage1

[NS] Message sending from sender to receiver at time 14
[ST] Buffered message
[ST] Current buffered messages: 2

[NS] Message sending from sender to receiver at time 14
[ST] Buffered message
[ST] Current buffered messages: 3

[NS] Message arriving from sender to receiver at time 19
[RT] Receive packet at receiver; seqnum: 2 acknum: -1 msg: 78essage2

[NS] Timer expired at time 22
[NL] Packet seq: 0 ack: -1 sent Message0
[NL] Packet seq: 1 ack: -1 sent Message1
[NL] Packet seq: 2 ack: -1 sent Message2

[NS] Message arriving from sender to receiver at time 27
[RT] Receive packet at receiver; seqnum: 0 acknum: -1 msg: Message0
from receiver:Message0
[NL] Packet seq: -1 ack: 0 sent ACK

[NS] Message arriving from sender to receiver at time 29
[RT] Receive packet at receiver; seqnum: 1 acknum: -1 msg: Message1
from receiver:Message1
[NL] Packet seq: -1 ack: 1 sent ACK

[NS] Message arriving from sender to receiver at time 32
[RT] Receive packet at receiver; seqnum: 2 acknum: -1 msg: Message2
from receiver:Message2
[NL] Packet seq: -1 ack: 2 lost ACK

[NS] Message arriving from receiver to sender at time 36
[ST] Receive packet at sender; seqnum: -1 acknum: 0
[ST] Current queuing messages: 3, open windows: 1

[ST] Sending the next message in the queue
[NL] Packet seq: 3 ack: -1 corrupted Message3
[NL] Packet seq: 3 ack: -1 sent 78essage3

[NS] Message arriving from receiver to sender at time 41
[ST] Receive packet at sender; seqnum: -1 acknum: 1
[ST] Current queuing messages: 2, open windows: 1
[ST] Sending the next message in the queue
[NL] Packet seq: 4 ack: -1 sent Message4

[NS] Message arriving from sender to receiver at time 48
[RT] Receive packet at receiver; seqnum: 3 acknum: -1 msg: 78essage3

[NS] Message arriving from sender to receiver at time 55
[RT] Receive packet at receiver; seqnum: 4 acknum: -1 msg: Message4
[NL] Packet seq: -1 ack: 2 sent ACK

[NS] Message arriving from receiver to sender at time 60
[ST] Receive packet at sender; seqnum: -1 acknum: 2
[ST] Current queuing messages: 1, open windows: 1
[ST] Sending the next message in the queue
[NL] Packet seq: 5 ack: -1 sent Message5

[NS] Message arriving from sender to receiver at time 61
[RT] Receive packet at receiver; seqnum: 5 acknum: -1 msg: Message5
[NL] Packet seq: -1 ack: 2 sent ACK

[NS] Message arriving from receiver to sender at time 65
[ST] Receive packet at sender; seqnum: -1 acknum: 2

[NS] Timer expired at time 80
[NL] Packet seq: 3 ack: -1 lost Message3
[NL] Packet seq: 4 ack: -1 corrupted Message4
[NL] Packet seq: 4 ack: -1 sent 78essage4
[NL] Packet seq: 5 ack: -1 sent Message5

[NS] Message arriving from sender to receiver at time 87
[RT] Receive packet at receiver; seqnum: 4 acknum: -1 msg: 78essage4

[NS] Message arriving from sender to receiver at time 92
[RT] Receive packet at receiver; seqnum: 5 acknum: -1 msg: Message5
[NL] Packet seq: -1 ack: 2 corrupted ACK
[NL] Packet seq: -1 ack: 2 sent 66CK

[NS] Message arriving from receiver to sender at time 98
[ST] Receive packet at sender; seqnum: -1 acknum: 2

[NS] Timer expired at time 100
[NL] Packet seq: 3 ack: -1 corrupted Message3
[NL] Packet seq: 3 ack: -1 sent 78essage3
[NL] Packet seq: 4 ack: -1 sent Message4
[NL] Packet seq: 5 ack: -1 sent Message5

[NS] Message arriving from sender to receiver at time 106
[RT] Receive packet at receiver; seqnum: 3 acknum: -1 msg: 78essage3

[NS] Message arriving from sender to receiver at time 108
[RT] Receive packet at receiver; seqnum: 4 acknum: -1 msg: Message4
[NL] Packet seq: -1 ack: 2 sent ACK

[NS] Message arriving from sender to receiver at time 116
[RT] Receive packet at receiver; seqnum: 5 acknum: -1 msg: Message5
[NL] Packet seq: -1 ack: 2 sent ACK

[NS] Message arriving from receiver to sender at time 118
[ST] Receive packet at sender; seqnum: -1 acknum: 2

[NS] Message arriving from receiver to sender at time 119
[ST] Receive packet at sender; seqnum: -1 acknum: 2

[NS] Timer expired at time 120
[NL] Packet seq: 3 ack: -1 lost Message3
[NL] Packet seq: 4 ack: -1 sent Message4
[NL] Packet seq: 5 ack: -1 lost Message5

[NS] Message arriving from sender to receiver at time 128
[RT] Receive packet at receiver; seqnum: 4 acknum: -1 msg: Message4
[NL] Packet seq: -1 ack: 2 sent ACK

[NS] Message arriving from receiver to sender at time 137
[ST] Receive packet at sender; seqnum: -1 acknum: 2

[NS] Timer expired at time 140
[NL] Packet seq: 3 ack: -1 sent Message3
[NL] Packet seq: 4 ack: -1 sent Message4
[NL] Packet seq: 5 ack: -1 sent Message5

[NS] Message arriving from sender to receiver at time 141
[RT] Receive packet at receiver; seqnum: 3 acknum: -1 msg: Message3
from receiver:Message3
[NL] Packet seq: -1 ack: 3 sent ACK

[NS] Message arriving from sender to receiver at time 149
[RT] Receive packet at receiver; seqnum: 4 acknum: -1 msg: Message4
from receiver:Message4
[NL] Packet seq: -1 ack: 4 sent ACK

[NS] Message arriving from sender to receiver at time 151
[RT] Receive packet at receiver; seqnum: 5 acknum: -1 msg: Message5
from receiver:Message5
[NL] Packet seq: -1 ack: 5 sent ACK

[NS] Message arriving from receiver to sender at time 160
[ST] Receive packet at sender; seqnum: -1 acknum: 3

[NS] Message arriving from receiver to sender at time 166
[ST] Receive packet at sender; seqnum: -1 acknum: 4

[NS] Message arriving from receiver to sender at time 173
[ST] Receive packet at sender; seqnum: -1 acknum: 5

Check correctness for TCP - With Loss/Corruption

[NS] Message sending from sender to receiver at time 5
[NL] Packet seq: 0 ack: -1 sent Message0

[NS] Message sending from sender to receiver at time 6
[NL] Packet seq: 1 ack: -1 sent Message1

[NS] Message sending from sender to receiver at time 6
[NL] Packet seq: 2 ack: -1 lost Message2

[NS] Message sending from sender to receiver at time 8
[ST] Buffered message
[ST] Current buffered messages: 1

[NS] Message sending from sender to receiver at time 8
[ST] Buffered message

[ST] Current buffered messages: 2

[NS] Message sending from sender to receiver at time 9
[ST] Buffered message
[ST] Current buffered messages: 3

[NS] Message arriving from sender to receiver at time 12
[RT] Receive packet at receiver; seqnum: 0 acknum: -1 msg: Message0
from receiver:Message0
[NL] Packet seq: -1 ack: 1 corrupted ACK
[NL] Packet seq: -1 ack: 1 sent 66CK

[NS] Message arriving from sender to receiver at time 16
[RT] Receive packet at receiver; seqnum: 1 acknum: -1 msg: Message1
from receiver:Message1
[NL] Packet seq: -1 ack: 2 sent ACK

[NS] Message arriving from receiver to sender at time 18
[ST] Receive packet at sender; seqnum: -1 acknum: 1

[NS] Message arriving from receiver to sender at time 22
[ST] Receive packet at sender; seqnum: -1 acknum: 2
[ST] Current queuing messages: 3, open windows: 2
[ST] Sending the next message in the queue
[NL] Packet seq: 3 ack: -1 sent Message3
[ST] Current queuing messages: 2, open windows: 1
[ST] Sending the next message in the queue
[NL] Packet seq: 4 ack: -1 sent Message4

[NS] Message arriving from sender to receiver at time 26
[RT] Receive packet at receiver; seqnum: 3 acknum: -1 msg: Message3
[RT] Buffer packet seqnum 3 msg: Message3
[RT] Number of receiver's buffered pkts: 1
[NL] Packet seq: -1 ack: 2 lost ACK

[NS] Message arriving from sender to receiver at time 30
[RT] Receive packet at receiver; seqnum: 4 acknum: -1 msg: Message4
[RT] Buffer packet seqnum 4 msg: Message4
[RT] Number of receiver's buffered pkts: 2
[NL] Packet seq: -1 ack: 2 corrupted ACK
[NL] Packet seq: 0 ack: 2 sent ACK

[NS] Message arriving from receiver to sender at time 36

[ST] Receive packet at sender; seqnum: 0 acknum: 2

[NS] Timer expired at time 42
[NL] Packet seq: 2 ack: -1 sent Message2

[NS] Message arriving from sender to receiver at time 46
[RT] Receive packet at receiver; seqnum: 2 acknum: -1 msg: Message2
from receiver:Message2
[RT] Remove packet seqnum 3 msg Message3 from buffer
from receiver:Message3
[RT] Remove packet seqnum 4 msg Message4 from buffer
from receiver:Message4
[NL] Packet seq: -1 ack: 5 sent ACK

[NS] Message arriving from receiver to sender at time 50
[ST] Receive packet at sender; seqnum: -1 acknum: 5
[ST] Current queuing messages: 1, open windows: 3
[ST] Sending the next message in the queue
[NL] Packet seq: 5 ack: -1 sent Message5

[NS] Message arriving from sender to receiver at time 53
[RT] Receive packet at receiver; seqnum: 5 acknum: -1 msg: Message5
from receiver:Message5
[NL] Packet seq: -1 ack: 6 corrupted ACK
[NL] Packet seq: -1 ack: 6 sent 66CK

[NS] Message arriving from receiver to sender at time 55
[ST] Receive packet at sender; seqnum: -1 acknum: 6

[NS] Timer expired at time 70
[NL] Packet seq: 5 ack: -1 sent Message5

[NS] Message arriving from sender to receiver at time 79
[RT] Receive packet at receiver; seqnum: 5 acknum: -1 msg: Message5
[NL] Packet seq: -1 ack: 6 sent ACK

[NS] Message arriving from receiver to sender at time 88
[ST] Receive packet at sender; seqnum: -1 acknum: 6

Appendix 2: Raw data for experiments

Experiment 1: Total Time against Time Between Sends

Time Between Sends	Total Time (Go-back-N)	Total Time (TCP)
0	1574	1016
1	1586	878
2	1453	895
3	912	857
4	1186	706
5	1865	940
6	1286	892
7	928	801
8	1453	917
9	1424	802
10	1051	814
11	1192	849
12	1195	777
13	1087	894
14	1005	889
15	1017	846
16	436	848
17	649	858
18	803	1006
19	786	833
20	703	922
21	626	850
22	580	1029
23	689	897
24	549	875
25	617	843
26	597	858

27	664	810
28	655	924
29	815	893
30	674	907
31	674	852
32	666	821
33	814	1042
34	744	798
35	760	932
36	869	917
37	841	1007
38	795	984
39	911	853
40	1054	1092
41	724	845
42	860	915
43	1024	1073
44	975	1083
45	1004	1032
46	1064	947
47	1074	991
48	1049	1100
49	1142	1128

Experiment 2: Total Time against Loss Probability

Loss Probability	Total Time (Go-back-N)	Total Time (TCP)
0	245	251
0.014	288	287
0.027999999	891	464
0.042	562	373
0.055999998	390	372

0.07	606	743
0.084	936	467
0.098	706	480
0.111999996	941	638
0.126	552	684
0.14	966	787
0.154	623	916
0.168	668	834
0.182	656	913
0.196	563	906
0.21	596	975
0.22399999	1032	1003
0.23799999	530	920
0.252	798	901
0.266	724	1157
0.28	652	1085
0.294	526	1173
0.308	666	1267
0.322	766	1286
0.336	596	1335
0.35	584	1457
0.364	657	1445
0.378	691	1463
0.392	658	1697
0.406	678	1648
0.42	706	1640
0.434	677	1729
0.44799998	700	1990
0.46199998	783	1800
0.47599998	793	2246
0.48999998	869	2359
0.504	919	2415
0.518	936	2682
0.532	701	2772

0.546	986	2912
0.56	874	2945
0.574	1156	3369
0.588	1067	3419
0.602	1405	3836
0.616	1461	3658
0.63	1604	5188
0.644	1512	4981
0.658	1411	4987
0.672	1904	4916
0.686	1585	6091

Experiment 3: Total Time against Corruption Probability

Corruption Probability	Total Time (Go-back-N)	Total Time (TCP)
0	244	283
0.014	1031	366
0.027999999	559	363
0.042	1501	348
0.055999998	437	629
0.07	1038	440
0.084	770	476
0.098	642	456
0.111999996	1080	620
0.126	1364	840
0.14	1962	643
0.154	1596	742
0.168	2455	893
0.182	1434	764
0.196	1592	958
0.21	1994	1163
0.22399999	1768	1130

0.23799999	1742	1108
0.252	3403	1024
0.266	1901	1286
0.28	2003	1239
0.294	3574	1292
0.308	2018	1194
0.322	2786	1383
0.336	3063	1387
0.35	3038	1541
0.364	3173	1604
0.378	2152	1558
0.392	3485	1811
0.406	3257	1679
0.42	3392	1626
0.434	2855	1780
0.44799998	4453	2034
0.46199998	2578	2223
0.47599998	3369	2106
0.48999998	4246	2512
0.504	3397	2485
0.518	3487	2899
0.532	3940	2727
0.546	3962	2906
0.56	4344	2756
0.574	3406	2860
0.588	3662	3664
0.602	4083	3211
0.616	4148	3319
0.63	4521	4374
0.644	5608	4224
0.658	4792	5199
0.672	4485	5495
0.686	4211	4959

Experiment 4: Total Time against Windows Size

Windows Size	Total Time (Go-back-N)	Total Time (TCP)
1	533	599
2	388	359
3	461	767
4	489	658
5	503	1027
6	789	766
7	1085	860
8	1667	835
9	996	864
10	1024	971
11	2467	852
12	1592	789
13	1707	970
14	1687	862
15	1411	988
16	2218	853
17	1580	906
18	2506	894
19	1896	950
20	2096	926
21	3403	956
22	2178	783
23	1746	918
24	1121	952
25	2358	967
26	1958	956
27	1408	672
28	3305	942
29	1972	868

30	1164	1013
31	2507	826
32	2606	944
33	1185	982
34	3352	1032
35	1065	984
36	1769	1029
37	3719	958
38	1340	818
39	2254	923
40	1625	1085
41	1810	922
42	1943	925
43	1883	910
44	1155	841
45	1421	1003
46	1864	970
47	3265	980
48	1824	1045
49	1491	874
50	1319	813