Stop and wait:

```
EVENT time: 696.571045,  type: 1, from app layer entity: 0
        MAINLOOP: data given to student: aaaaaaaaaaaaaaaaaa
[A] sending Message to B: aaaaaaaaaaaaaaaaaa, currentACK: 0
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-START TIMER: starting timer at 696.571045
EVENT time: 701.979919,  type: 2, from network layer  entity: 1
        (A=0,B=1)Caller=1-TO_APP_LAYER: data received: aaaaaaaaaaaaaaaaaa
[B] ----- RECEIVING GOOD PACKET ----- packetAckNum=0 BSeq=0
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: scheduling arrival on other side
EVENT time: 706.738098,  type: 2, from network layer  entity: 0
[A] Receiving ACK from B. ACK=0
[A] -----    ACK MATCHES WITH SEQ   -----
        (A=0,B=1)Caller=0-STOP TIMER: stopping timer at 706.738098
```

Case 1: no lost and no corruption:

- A sends a packet 0 and start the timer
- B received a packet 0
- B sends ACK 0
- A received ACK 0
- A stop timer

```
EVENT time: 696.571045,  type: 1, from app layer entity: 0
        MAINLOOP: data given to student: aaaaaaaaaaaaaaaaaa
[A] sending Message to B: aaaaaaaaaaaaaaaaaa, currentACK: 0   [A]A send pkt to B
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-START TIMER: starting timer at 696.571045
EVENT time: 701.979919,  type: 2, from network layer  entity: 1
        (A=0,B=1)Caller=1-TO_APP_LAYER: data received: aaaaaaaaaaaaaaaaaa
[B] ----- RECEIVING GOOD PACKET ----- packetAckNum=0 BSeq=0   [B] send Ack
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet being corrupted
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: scheduling arrival on other side
EVENT time: 706.738098,  type: 2, from network layer  entity: 0
[A] Receiving ACK from B. ACK=0
[A] ----- ACK CORRUPTED or PREV ACK -----  [A] corrupted Ack recived, wait for Time out to handle
EVENT time: 716.571045,  type: 0, timerinterrupt   entity: 0
[A] ---- TIMEOUT ---- currentACK=0   [A]Time out handle
[A] sending Message to B: aaaaaaaaaaaaaaaaaa, currentACK: 0
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost   [A]Resend pkt
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-START TIMER: starting timer at 716.571045
EVENT time: 726.099182,  type: 2, from network layer  entity: 1
[B] ----- Received a DUP Packet ----- packetAckNum=0   [B]detect duplicated pkt
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: scheduling arrival on other side
EVENT time: 735.802429,  type: 2, from network layer  entity: 0
[A] Receiving ACK from B. ACK=0   [B] send same Ack
[A] -----    ACK MATCHES WITH SEQ   -----
        (A=0,B=1)Caller=0-STOP TIMER: stopping timer at 735.802429
```

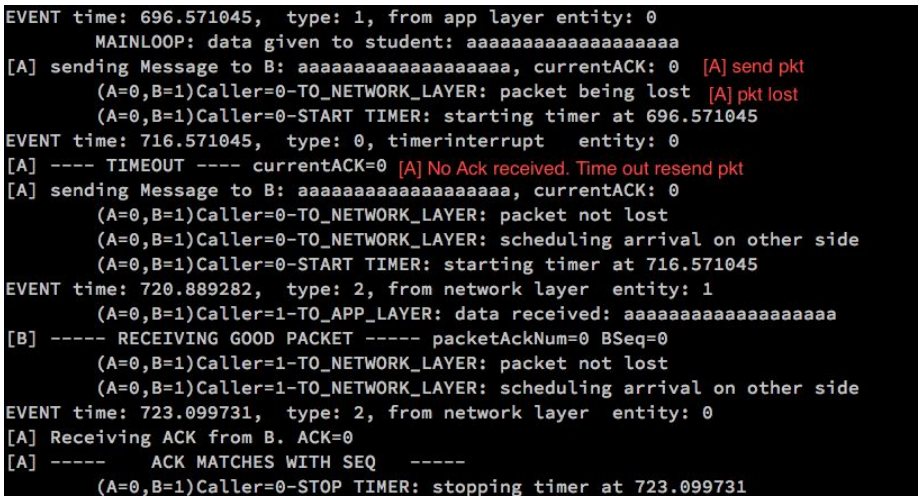
Case 2: ACK corrupted:

- A sends a packet 0
- B received a packet 0
- B sends ACK 0(ACK being corrupted)
- A detected corrupted ACK 0(do nothing)
- A time out then resend the packet and start the timer
- B detect duplicated packet 0
- B sends ACK 0(not being corrupted)
- A received ACK 0
- A stop timer

```
EVENT time: 1966.186157,  type: 1, from app layer entity: 0
        MAINLOOP: data given to student: bbbbbbbbbbbbbbbbbb
[A] sending Message to B: bbbbbbbbbbbbbbbbbb, currentACK: 1   [A] Send pkt
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-START TIMER: starting timer at 1966.186157
EVENT time: 1975.714233,  type: 2, from network layer  entity: 1
        (A=0,B=1)Caller=1-TO_APP_LAYER: data received: bbbbbbbbbbbbbbbbbb
[B] ----- RECEIVING GOOD PACKET ----- packetAckNum=1 BSeq=1   [B] Received pkt
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet being lost   [B] Send ACK(lost)
EVENT time: 1986.186157,  type: 0, timerinterrupt   entity: 0
[A] ---- TIMEOUT ---- currentACK=1
                                                                [A] Time out, resend pkt
[A] sending Message to B: bbbbbbbbbbbbbbbbbb, currentACK: 1
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-START TIMER: starting timer at 1986.186157
EVENT time: 1994.488037,  type: 2, from network layer  entity: 1
[B] ----- Received a DUP Packet ----- packetAckNum=1   [B] Received duplicated pkt,
                                                        send ACK
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: scheduling arrival on other side
EVENT time: 2004.276489,  type: 2, from network layer  entity: 0
[A] Receiving ACK from B. ACK=1
[A] -----    ACK MATCHES WITH SEQ   -----
        (A=0,B=1)Caller=0-STOP TIMER: stopping timer at 2004.276489
```

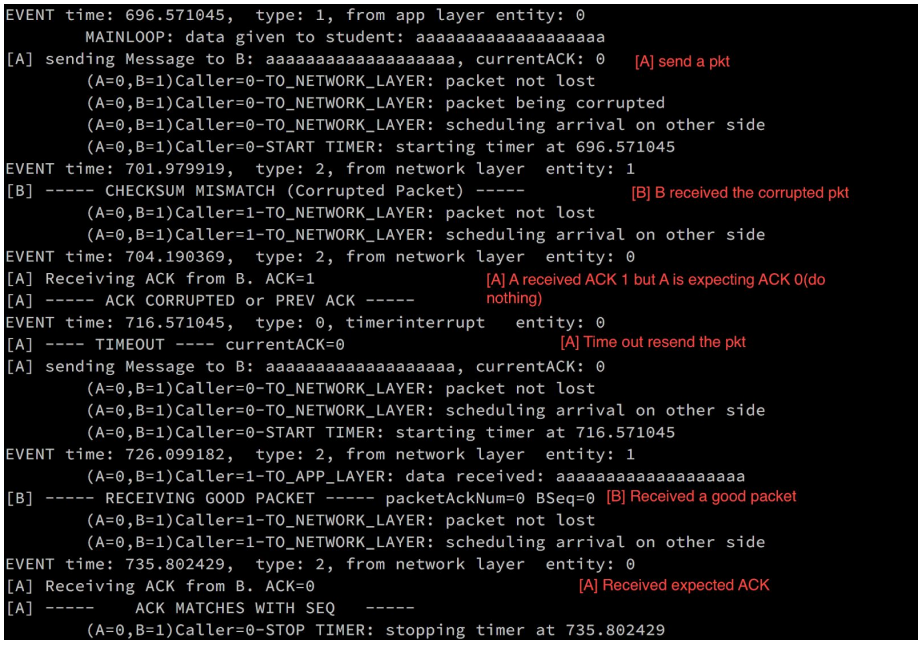
Case 3: ACK loss:

- A sends a packet 1
- B received a packet 1
- B sends ACK 1(lost)
- A time out then resend the packet and start the timer
- B received duplicated packet 1
- B send ACK 1
- A received ACK 1
- A stop timer



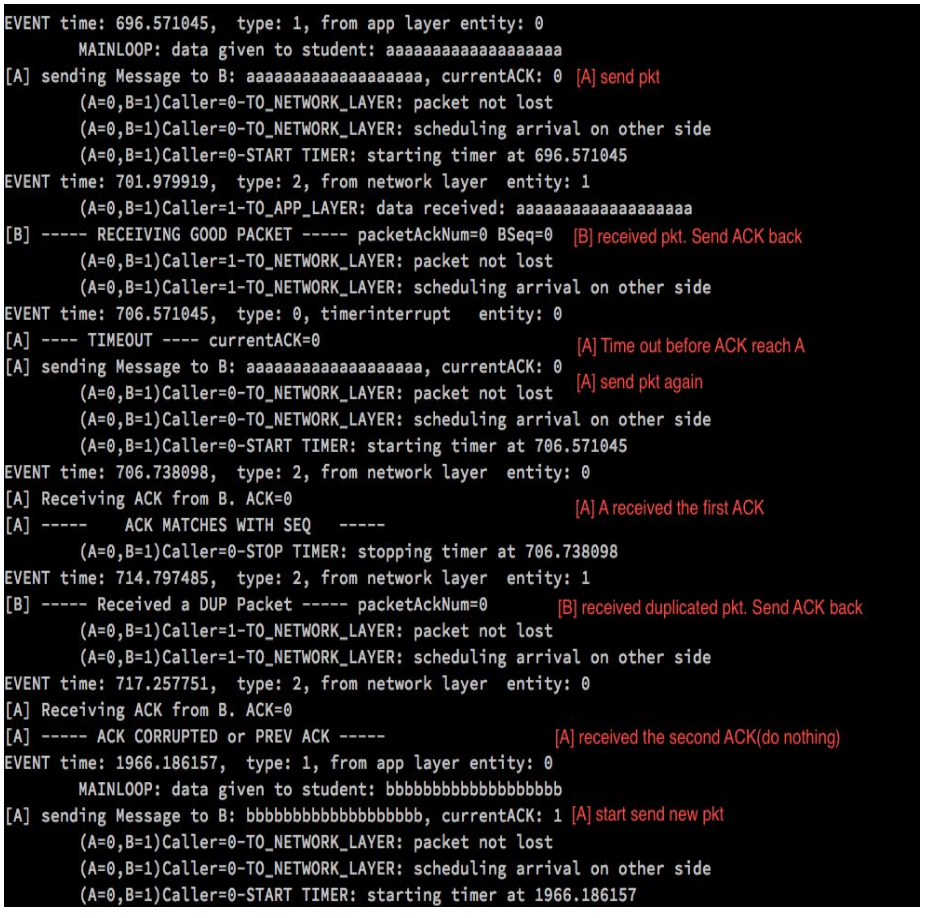
Case 4: packet loss

- A sends a packet with sequence 0
- Network loss the packet
- A didn't receive any ACK, time out
- A resend the packet and start the timer
- B received a packet 0
- B sends the ACK 0
- A received ACK 0
- A stop timer



Case 5: Packet Corrupted

- A sends a packet 0(corrupted)
- B checksum check that received a corrupted packet
- B send a ACK 1
- A received ACK 1 but expecting ACK 0(do nothing)
- A time out then resend the packet and start the timer
- B received the packet(good)
- B send ACK 0
- A received ACK 0
- A stop timer



Case 6: premature timeout/ delayed ACK

- A send a packet 0
- B received packet 0 and send ACK 0 back
- A time out while the ACK 0 on the way and start the timer
- A resend the packet 0
- A received the ACK 0 and stop the timer
- B received the second packet 0 and duplicated packet detected
- B send ACK 0
- A received ACK 0 and do nothing

Go Back And N:

```
[A] ----- NEW PACKET RECEIVED bufferSize=1 freeSlot=7 packetSent=TRUE -----
EVENT time: 12.374607,  type: 2, from network layer  entity: 1

[B] ----- RECEIVING PACKET FROM A expectedSEQ=1 packetSEQ=1
        [B] ----- SEQUENCE MATCH & CHECKSUM PASS -----
        (A=0,B=1)Caller=1-TO_APP_LAYER: data received: aaaaaaaaaaaaaaaaaaa
        [B] ----- SENDING ACK ack=1 -----
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet being corrupted
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: scheduling arrival on other side
EVENT time: 17.132755,  type: 2, from network layer  entity: 0

[A] ----- ACK CORRUPTED -----
EVENT time: 19.661861,  type: 1, from app layer entity: 0
        MAINLOOP: data given to student: bbbbbbbbbbbbbbbbbbb
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet being corrupted
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side

[A] ----- NEW PACKET RECEIVED bufferSize=2 freeSlot=6 packetSent=TRUE -----
EVENT time: 27.956409,  type: 2, from network layer  entity: 1
```

Corrupted packet from B

- A receives a new packet from the application layer
- A sends it to the B
- B receives, sequence Numbers match
- B sends ACK, but network corrupts the packet
- A receives a corrupted Packet, doesn't do anything.
- A receives a new packet, simply sends, while the first packet is still in the queue.
- freeSlots decreases from 7 to 6.

```
EVENT time: 19.661861,  type: 1, from app layer entity: 0
        MAINLOOP: data given to student: bbbbbbbbbbbbbbbbbbb
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet being corrupted
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side

[A] ----- NEW PACKET RECEIVED bufferSize=2 freeSlot=6 packetSent=TRUE -----
EVENT time: 27.956409,  type: 2, from network layer  entity: 1

[B] ----- RECEIVING PACKET FROM A expectedSEQ=2 packetSEQ=2
        [B] ----- PACKET CORRUPTED expected=1866 got=1001863 -----
        [B] ----- SENDING ACK ack=1 -----
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: scheduling arrival on other side
EVENT time: 35.720585,  type: 1, from app layer entity: 0
        MAINLOOP: data given to student: ccccccccccccccccccc
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side

 [A] ----- VALID_ACK & IN_RANGE ack=1 base=1  -----
         [A] ----- UPDATING WINDOW ack=1 -----
         [A] ----- BASE HAS BEEN ACKed base=1 -----
         (A=0,B=1)Caller=0-STOP TIMER: stopping timer at 37.509491
         [A] ----- BASE UPDATED SENDING IN NEW PACKETS base=2 -----
         (A=0,B=1)Caller=0-START TIMER: starting timer at 37.509491
         [A] ----- ALL PACKETS SENT  nextSeqNum=4, freeSlots=6 -----
```

Corrupted packet from A

- A sends a received and send a new packet. However, network corrupts its
- B receives the packet, sees the imbalance in the checksum
- Sends ACK-1, since it already received a packet of seq-1

Non-corrupted ACK

- We see that the ack matches with the base of the window
- A stops the timers. And updates the base
- Sends any new packets (which is none, since there are still 6 freeSlots available.
- Timer gets restarted

```
[A] ----- TIMEOUT RESENDING ALL PACKETS base=2 nextSeq=10 -----
        (A=0,B=1)Caller=0-START TIMER: starting timer at 137.509491
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet being corrupted
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet being corrupted
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet being lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=0-TO_NETWORK_LAYER: scheduling arrival on other side
EVENT time: 141.892502,  type: 2, from network layer  entity: 1
```

Times-out

- Since Packet-2 was corrupted, B hasn't received a packet of sequence 2
- In the meantime, we see that the window has now filled up.
- Upon timeout, A resends all the packets that are in the window (packets-2, … , Packet-9)

```
[B] ----- RECEIVING PACKET FROM A expectedSEQ=19 packetSEQ=19
        [B] ----- SEQUENCE MATCH & CHECKSUM PASS -----
        (A=0,B=1)Caller=1-TO_APP_LAYER: data received: ssssssssssssssssss
        [B] ----- SENDING ACK ack=19 -----
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet being corrupted
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: scheduling arrival on other side
EVENT time: 1186.139404,  type: 2, from network layer  entity: 1

[B] ----- RECEIVING PACKET FROM A expectedSEQ=20 packetSEQ=20
        [B] ----- SEQUENCE MATCH & CHECKSUM PASS -----
        (A=0,B=1)Caller=1-TO_APP_LAYER: data received: ttttttttttttttttttt
        [B] ----- SENDING ACK ack=20 -----
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: packet not lost
        (A=0,B=1)Caller=1-TO_NETWORK_LAYER: scheduling arrival on other side
EVENT time: 1187.619263,  type: 2, from network layer  entity: 0

[A] ----- ACK CORRUPTED -----
EVENT time: 1195.124756,  type: 2, from network layer  entity: 0

[A] ----- VALID_ACK & IN_RANGE ack=20 base=19  -----
        [A] ----- UPDATING WINDOW ack=20 -----
        [A] ----- BASE HAS BEEN ACKed base=19 -----
        [A] ----- BASE HAS BEEN ACKed base=20 -----
        (A=0,B=1)Caller=0-STOP TIMER: stopping timer at 1195.124756
        [A] ----- BASE UPDATED SENDING IN NEW PACKETS base=21 -----
        [A] ----- ALL PACKETS SENT  nextSeqNum=21, freeSlots=8 -----
Terminated at time 1195.124756
```

ACK with a Higher seq than the base

- Here we see that B receives packet-19 and packet-20. Which are both valid since the base is at 19
- B then sends an ack for both, where the ACK for 19 gets corrupted.
- We can see that A ignores the corrupted ACK-19
- A receives an ACK-20 which is between the base and nextSequenceNumber
- A verifies that ACK-20 Packet is not corrupted. Moves the window to next-Sequence-Number 21
  - Since this was the last message in the window, we can see that we now have 8 free slots (max number of slots), therefore, nothing is sent.
  - The program terminates since ‘tt … t’ was the 20th message.

Overview:

Both of the required protocols were implemented. For *Stop-And-Wait Protocol* there was no special data structure that was required as it simply requires the sender to wait for a response from the receiver after sending a packet. Both the sender and the receiver had their own version of the current and the expected sequence number, which allowed each of them to respond back accordingly (ie receiver sending the previous sequence ACK upon receiving a corrupted packet).

For *Go-Back-N Protocol* we used a linked-list in our implementation. Where the entire list represents the buffer (maxSize = 50) and the first 8 elements represent the window. As the sender receives new packets from the application layer they are added to the buffer, and if it is within the window then the packet is sent to the receiver. Upon response, the ACK is evaluated (base, between base and nextSeq) then simply re-adjust window. If the ACK is one before the base, then we leave it to timeout.