Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

Compilation

Python is an interpreted language. Hence, the code is directly executed by the Python interpreter. No compilation is necessary.

Execution/Running

1. Open ***SNW_Sender.py***, **SNW_Receiver.py**, ***simulator.py****, and* ***mainSNW.py*** in VSCode.
2. Change the simulation settings by navigating to the file ***simulator.py***, where ***nsimax*** (Number of messages to simulate)*,* ***lossprob*** (Loss probability), ***corruptprob*** (Corruption probability), and ***Lambda*** (Average time between messages from the Sender's layer 5) can be customized according to your needs.
3. Open a terminal window, by navigating to **Terminal** > **New Terminal**.
4. To simulate the Stop and Wait protocol, run ***python3 mainSNW.py*** on the command line.

Description

High-level description

SNW_Sender.py represents the sender in the Stop-and-Wait protocol, whereas ***SNW_Receiver.py*** represents the receiver in the Stop-and-Wait protocol. ***Simulator.py*** simulates an environment for the Stop-and-Wait protocol, which provides reliable transfer under an unreliable packet delivery system by ensuring that data arrives in order.

Low-level description

In this implementation of the Stop-and-Wait protocol, the function ***S_output(<message>)*** in the sender (***SNW_Sender.py***) would be called when it received a message from layer 5 (application) and would try to create a packet containing the message and send it to layer 3 (network).

The sender would first check whether it is in the ***"WAIT_LAYER5"*** state, where the sender waits for messages from the application layer (layer 5), and sends a packet containing an initial sequence number of 0, and a message of 20 of the same alphabet (e.g: aaaaaaaaaaaaaaaaaaaa) to layer 3, with the method ***to_layer_three('S', <packet>)****,* change state to ***"WAIT_ACK"*** and start the timer for the packet sent. The sender would also assign the packet sent to a variable for packet retransmission in cases where the ACK for the most recently sent packet was not received by the sender in time. If it is in

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

a state to receive ACK from the receiver (***"WAIT_ACK"***), it would not send any messages, and drop any messages that were from the application layer.

The function ***R_input(<received_packet>)*** in the receiver (***SNW_Receiver.py)*** would be called when it receives a packet from layer 3 (network) and would verify the received packet and behave as described below.

The receiver would be expecting a packet from the sender with an initial sequence number of 0 and would verify the integrity of the packet by checking for corruption with checksum, and sequence number. If the packet is not corrupted and with the correct sequence number, the receiver will send the packet to layer 5, with the method ***to_layer_five('R', <received_packet.payload.data>)***, update the next expected sequence number (in this case a 1) and send an ACK corresponding to the sequence number of the packet (in this case, an ACK0). However, if the packet is corrupted or with the wrong sequence number, the receiver sends an implicit NAK to the sender by sending an ACK with the opposite sequence number, that the sender expects (in this case an ACK1).

The function ***S_input(<received_packet>)*** in the sender (***SNW_Sender.py***) would be called when it received a packet from layer 3 and would verify the received packet and behave as described below.

The sender would check whether it is in a state to receive ACK from the receiver *(**"WAIT_ACK"***), then receive the ACK<sequence-number>, and verify the integrity of the packet by checking for corruption with checksum, and ACK number. If the packet is not corrupted and the expected ACK number is received, the server will change its state to ***"WAIT_LAYER5"***, and update its next expected ACK number (in this case, 1). If the packet is corrupted, or the wrong ACK number is received, the sender will drop the packet.

The next message sent by the sender would follow the order of alphabets, and the simulation runs ***nsimax*** times, which represents the number of messages to simulate and can be changed in ***simulator.py***.

In the case where an ACK for the most recently sent packet wasn't received by the sender in time, the function ***S_handle_timer()*** in the sender **(*SNW_Sender.py***) would be called to retransmit the packet.

The sender would check if it is in the ***"WAIT_ACK"*** state, send the packet to layer 3 and start a timer that corresponds to the packet.

In each relevant event listed below, its corresponding statistic counters will be updated.

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

Event:

- Message Sent (in packet)
- Retransmitted Data
- Retransmitted ACK
- Dropped Data
- Dropped ACK
- Corrupted Data
- Corrupted ACK

There would also be counters for total retransmitted, total dropped, and total corrupted packets.

Evaluation

Variables that affect test conditions:

1. ***nsimax*** - Number of messages to simulate
2. ***lossprob*** - Loss probability
3. ***corruptprob*** - Corruption probability
4. ***Lambda*** - Average time between messages from the Sender's layer 5

Test Case 1:

Condition

Nsimmax = 30, lossprob = 0, corruptprob = 0, Lambda = 1000

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

Output

```
data received:  aaaaaaaaaaaaaaaaaaaa
data received:  bbbbbbbbbbbbbbbbbbbb
data received:  cccccccccccccccccccc
data received:  dddddddddddddddddddd
data received:  eeeeeeeeeeeeeeeeeeee
data received:  ffffffffffffffffffff
data received:  gggggggggggggggggggg
data received:  hhhhhhhhhhhhhhhhhhhh
data received:  iiiiiiiiiiiiiiiiiiii
data received:  jjjjjjjjjjjjjjjjjjjj
data received:  kkkkkkkkkkkkkkkkkkkk
data received:  llllllllllllllllllll
data received:  mmmmmmmmmmmmmmmmmmmm
data received:  nnnnnnnnnnnnnnnnnnnn
data received:  oooooooooooooooooooo
data received:  pppppppppppppppppppp
data received:  qqqqqqqqqqqqqqqqqqqq
data received:  rrrrrrrrrrrrrrrrrrrr
data received:  ssssssssssssssssssss
data received:  tttttttttttttttttttt
data received:  uuuuuuuuuuuuuuuuuuuu
data received:  vvvvvvvvvvvvvvvvvvvv
data received:  wwwwwwwwwwwwwwwwwwwwwwwwwwww
data received:  xxxxxxxxxxxxxxxxxxxx
data received:  yyyyyyyyyyyyyyyyyyyy
data received:  zzzzzzzzzzzzzzzzzzzz
data received:  aaaaaaaaaaaaaaaaaaaa
data received:  bbbbbbbbbbbbbbbbbbbb
data received:  cccccccccccccccccccc
data received:  dddddddddddddddddddd
simulation complete
==========STATISTICS==========
 Total Number of Messages Sent               ->  60
 Total Number of Retransmissions             ->  0
   Total Number of Retransmitted Data Packets  ->  0
   Total Number of Retransmitted ACKs        ->  0
 Total Number of Dropped Packets             ->  0
   Total Number of Dropped Data Packets      ->  0
   Total Number of Dropped ACKs              ->  0
 Total Number of Corrupted Packets           ->  0
   Total Number of Corrupted Data Packets    ->  0
   Total Number of Corrupted ACKs            ->  0
 Final Simulation Time                       ->  30011.0147973818
==========STATISTICS==========
```

Analysis

With a loss and corruption probability of zero, packets sent by the sender or receiver would not be lost or corrupted. The sender would send 30 messages, and the receiver would receive them and send an ACK for each message received, thus a total of 60 messages were sent by both sender and receiver.

Test Case 2:

Condition

Nsimmax = 30, lossprob = 0, corruptprob = 0, Lambda = 100

Name: Jun Hao Cheh

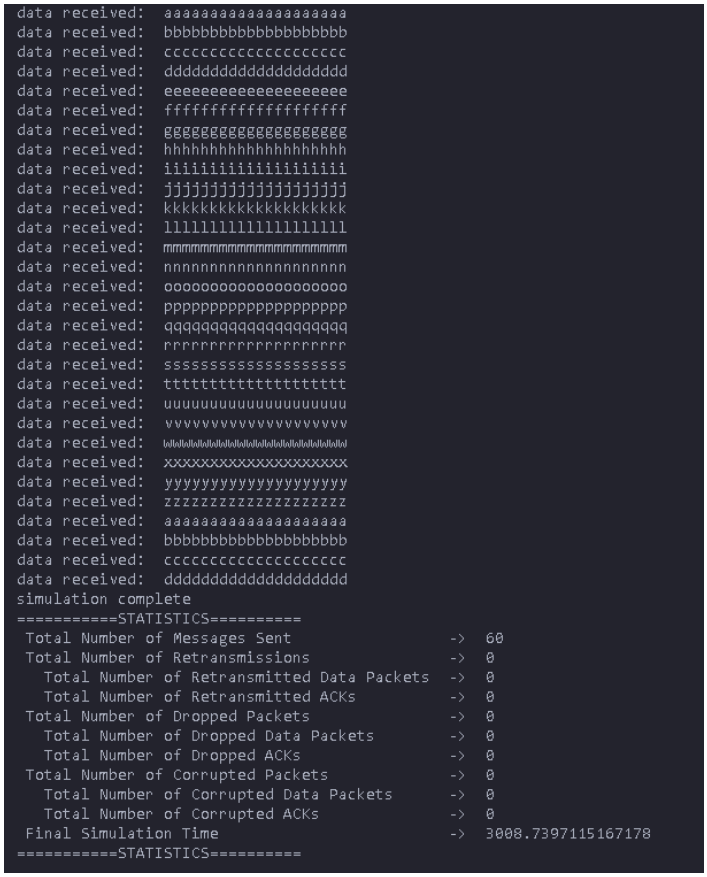
Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

Output



Analysis

With a loss and corruption probability of zero, packets sent by the sender or receiver would not be lost or corrupted. The sender would send 30 messages, and the receiver would receive them and send an ACK for each message received, thus a total of 60 messages were sent by both sender and receiver.

When the Lambda is decreased to 100, the final simulation time decreases, because the average time between messages that the sender received from layer 5 (application) decreased. Hence, the sender was able to send messages to the receiver quicker than previously (when Lambda = 1000).

Test Case 3:

Condition

Nsimmax = 30, lossprob = 0.2, corruptprob = 0, Lambda = 1000

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

Output

```
data received:  aaaaaaaaaaaaaaaaaaaa
data received:  bbbbbbbbbbbbbbbbbbbb
data received:  cccccccccccccccccccc
data received:  dddddddddddddddddddd
data received:  eeeeeeeeeeeeeeeeeeee
data received:  ffffffffffffffffffff
data received:  gggggggggggggggggggg
data received:  hhhhhhhhhhhhhhhhhhhh
data received:  iiiiiiiiiiiiiiiiiiii
data received:  jjjjjjjjjjjjjjjjjjjj
data received:  kkkkkkkkkkkkkkkkkkkk
data received:  llllllllllllllllllll
data received:  mmmmmmmmmmmmmmmmmmmm
data received:  nnnnnnnnnnnnnnnnnnnn
data received:  oooooooooooooooooooo
data received:  pppppppppppppppppppp
data received:  qqqqqqqqqqqqqqqqqqqq
data received:  rrrrrrrrrrrrrrrrrrrr
data received:  ssssssssssssssssssss
data received:  tttttttttttttttttttt
data received:  uuuuuuuuuuuuuuuuuuuu
data received:  vvvvvvvvvvvvvvvvvvvv
data received:  wwwwwwwwwwwwwwwwwwww
data received:  xxxxxxxxxxxxxxxxxxxx
data received:  yyyyyyyyyyyyyyyyyyyy
data received:  zzzzzzzzzzzzzzzzzzzz
data received:  aaaaaaaaaaaaaaaaaaaa
data received:  bbbbbbbbbbbbbbbbbbbb
data received:  cccccccccccccccccccc
data received:  dddddddddddddddddddd
simulation complete
==========STATISTICS==========
 Total Number of Messages Sent                ->  93
 Total Number of Retransmissions              ->  33
   Total Number of Retransmitted Data Packets  ->  21
   Total Number of Retransmitted ACKs          ->  12
 Total Number of Dropped Packets              ->  12
   Total Number of Dropped Data Packets        ->  12
   Total Number of Dropped ACKs                ->  0
 Total Number of Corrupted Packets            ->  0
   Total Number of Corrupted Data Packets      ->  0
   Total Number of Corrupted ACKs              ->  0
 Final Simulation Time                        ->  30072.027498933126
==========STATISTICS==========
```

Analysis

With a corrupt probability of zero, packets sent by the sender and receiver would not be corrupted. Hence, the total number of corrupted packets was zero. When the loss probability is increased to 0.2, there is a 20% chance that a packet will be dropped in the simulation. In the Stop-and-Wait protocol, packets would be dropped when the receiver received a corrupted packet or packet with the wrong sequence number from the sender; when the sender received a corrupted packet or packet with the wrong ACK number from the receiver. With a 0.2 probability of such instances happening, the simulation data shows that the receiver dropped 12 data packets, which might be due to receiving packets with the wrong sequence number, thus, transmitting 12 ACKs (implicit NAKs) to the sender. The 21 retransmitted data packets are due to the sender not receiving their corresponding ACKs in time from the receiver. The data packets were assumed to be lost, which triggered their retransmission. With 30 messages sent to the network layer by the sender, 21 retransmitted data

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

packets from timeout, 30 ACKs received by the sender, and 12 implicit NAKs received by the sender, a total of 93 messages were exchanged between sender and receiver.

Test Case 4:

Condition

Nsimmax = 30, lossprob = 0, corruptprob = 0.3, Lambda = 1000

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

Output

```
data received:  aaaaaaaaaaaaaaaaaaaa
data received:  bbbbbbbbbbbbbbbbbbbb
data received:  cccccccccccccccccccc
data received:  dddddddddddddddddddd
data received:  eeeeeeeeeeeeeeeeeeee
data received:  ffffffffffffffffffff
data received:  gggggggggggggggggggg
data received:  hhhhhhhhhhhhhhhhhhhh
data received:  iiiiiiiiiiiiiiiiiiii
data received:  jjjjjjjjjjjjjjjjjjjj
data received:  kkkkkkkkkkkkkkkkkkkk
data received:  llllllllllllllllllll
data received:  mmmmmmmmmmmmmmmmmmmm
data received:  nnnnnnnnnnnnnnnnnnnn
data received:  oooooooooooooooooooo
data received:  pppppppppppppppppppp
data received:  qqqqqqqqqqqqqqqqqqqq
data received:  rrrrrrrrrrrrrrrrrrrr
data received:  ssssssssssssssssssss
data received:  tttttttttttttttttttt
data received:  uuuuuuuuuuuuuuuuuuuu
data received:  vvvvvvvvvvvvvvvvvvvv
data received:  wwwwwwwwwwwwwwwwwwww
data received:  xxxxxxxxxxxxxxxxxxxx
data received:  yyyyyyyyyyyyyyyyyyyy
data received:  zzzzzzzzzzzzzzzzzzzz
data received:  aaaaaaaaaaaaaaaaaaaa
data received:  bbbbbbbbbbbbbbbbbbbb
data received:  cccccccccccccccccccc
data received:  dddddddddddddddddddd
simulation complete
==========STATISTICS==========
 Total Number of Messages Sent               ->  118
 Total Number of Retransmissions             ->  58
   Total Number of Retransmitted Data Packets  ->  29
   Total Number of Retransmitted ACKs          ->  29
 Total Number of Dropped Packets             ->  58
   Total Number of Dropped Data Packets        ->  29
   Total Number of Dropped ACKs                ->  29
 Total Number of Corrupted Packets           ->  36
   Total Number of Corrupted Data Packets      ->  18
   Total Number of Corrupted ACKs              ->  18
 Final Simulation Time                       ->  30039.87612313465
==========STATISTICS==========
```

Analysis

With a corrupt probability of 0.3, there is a 30% chance for packets to be corrupted. However, even if the loss probability is zero, and a drop chance of zero is expected, corrupted packets would still be dropped in the Stop-and-Wait protocol. Based on the data, both the sender and receiver received 18 corrupted packets. For each of the 18 corrupted packets received by the receiver, the receiver would retransmit an ACK (implicit NAK) and drop the data packet. Since implicit NAKs are sent to the sender, the sender did not receive the correct packet, so the timer for the packet would expire, and the packet would be retransmitted, which explains the total number of retransmitted and dropped data packets and ACKs being more than 19. For each ACK, implicit NAK sent by the receiver, as well as retransmission and initial transmission of packets by the sender, the total message sent counter would be incremented, which explains the increase of total message sent from 60 to 118.

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

Test Case 5:

Condition

Nsimmax = 30, lossprob = 0.2, corruptprob = 0.3, Lambda = 1000

Output

```
data received:  aaaaaaaaaaaaaaaaaaaa
data received:  bbbbbbbbbbbbbbbbbbbb
data received:  cccccccccccccccccccc
data received:  dddddddddddddddddddd
data received:  eeeeeeeeeeeeeeeeeeee
data received:  ffffffffffffffffffff
data received:  gggggggggggggggggggg
data received:  hhhhhhhhhhhhhhhhhhhh
data received:  iiiiiiiiiiiiiiiiiiii
data received:  jjjjjjjjjjjjjjjjjjjj
data received:  kkkkkkkkkkkkkkkkkkkk
data received:  llllllllllllllllllll
data received:  mmmmmmmmmmmmmmmmmmmm
data received:  nnnnnnnnnnnnnnnnnnnn
data received:  oooooooooooooooooooo
data received:  pppppppppppppppppppp
data received:  qqqqqqqqqqqqqqqqqqqq
data received:  rrrrrrrrrrrrrrrrrrrr
data received:  ssssssssssssssssssss
data received:  tttttttttttttttttttt
data received:  uuuuuuuuuuuuuuuuuuuu
data received:  vvvvvvvvvvvvvvvvvvvv
data received:  wwwwwwwwwwwwwwwwwwww
data received:  xxxxxxxxxxxxxxxxxxxx
data received:  yyyyyyyyyyyyyyyyyyyy
data received:  zzzzzzzzzzzzzzzzzzzz
data received:  aaaaaaaaaaaaaaaaaaaa
data received:  bbbbbbbbbbbbbbbbbbbb
data received:  cccccccccccccccccccc
data received:  dddddddddddddddddddd
simulation complete
==========STATISTICS==========
 Total Number of Messages Sent               ->  133
 Total Number of Retransmissions             ->  73
   Total Number of Retransmitted Data Packets  ->  43
   Total Number of Retransmitted ACKs          ->  30
 Total Number of Dropped Packets             ->  52
   Total Number of Dropped Data Packets        ->  30
   Total Number of Dropped ACKs                ->  22
 Total Number of Corrupted Packets           ->  29
   Total Number of Corrupted Data Packets      ->  15
   Total Number of Corrupted ACKs              ->  14
 Final Simulation Time                       ->  30035.2447519157
==========STATISTICS==========
```

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

Analysis

With a loss probability of 0.2 and a corrupt probability of 0.3, packets have a 20% chance of being dropped and a 30% chance of being corrupted. For each ACK, implicit NAK sent by the receiver, as well as retransmission and initial transmission of packets by the sender, the total message sent counter would be incremented. When corrupted packets are received by the receiver, implicit NAKs are sent to the sender. The packet would be retransmitted when the sender did not receive the correct packet or did not receive a packet at all, which causes the timer for the packet to expire. Since corrupted packets would be dropped on top of packets themselves having a 20% chance of being dropped, the total number of retransmissions and dropped packets would be higher than in test case 4, which explains the increase of total messages sent from 118 to 133.

Test Case 6:

Condition

Nsimmax = 30, lossprob = 0.8, corruptprob = 0.8, Lambda = 1000

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

Output

```
data received:  aaaaaaaaaaaaaaaaaaa
waiting for ack, new message is dropped:ccccccccccccccccccc
data received:  bbbbbbbbbbbbbbbbbbb
waiting for ack, new message is dropped:ddddddddddddddddddd
waiting for ack, new message is dropped:eeeeeeeeeeeeeeeeeee
data received:  fffffffffffffffffff
waiting for ack, new message is dropped:ggggggggggggggggggg
waiting for ack, new message is dropped:hhhhhhhhhhhhhhhhhhh
waiting for ack, new message is dropped:iiiiiiiiiiiiiiiiiii
data received:  jjjjjjjjjjjjjjjjjjj
waiting for ack, new message is dropped:kkkkkkkkkkkkkkkkkkk
waiting for ack, new message is dropped:lllllllllllllllllll
waiting for ack, new message is dropped:mmmmmmmmmmmmmmmmmmm
waiting for ack, new message is dropped:nnnnnnnnnnnnnnnnnnn
waiting for ack, new message is dropped:ooooooooooooooooooo
waiting for ack, new message is dropped:qqqqqqqqqqqqqqqqqqq
data received:  ppppppppppppppppppp
data received:  rrrrrrrrrrrrrrrrrrr
waiting for ack, new message is dropped:ttttttttttttttttttt
data received:  sssssssssssssssssss
waiting for ack, new message is dropped:uuuuuuuuuuuuuuuuuuu
waiting for ack, new message is dropped:vvvvvvvvvvvvvvvvvvv
waiting for ack, new message is dropped:wwwwwwwwwwwwwwwwwww
waiting for ack, new message is dropped:xxxxxxxxxxxxxxxxxxx
waiting for ack, new message is dropped:yyyyyyyyyyyyyyyyyyy
waiting for ack, new message is dropped:zzzzzzzzzzzzzzzzzzz
waiting for ack, new message is dropped:aaaaaaaaaaaaaaaaaaa
waiting for ack, new message is dropped:bbbbbbbbbbbbbbbbbbb
waiting for ack, new message is dropped:ccccccccccccccccccc
waiting for ack, new message is dropped:ddddddddddddddddddd
simulation complete
==========STATISTICS==========
 Total Number of Messages Sent                ->  1414
 Total Number of Retransmissions              ->  1400
   Total Number of Retransmitted Data Packets  ->  1156
   Total Number of Retransmitted ACKs          ->  244
 Total Number of Dropped Packets              ->  313
   Total Number of Dropped Data Packets        ->  267
   Total Number of Dropped ACKs                ->  46
 Total Number of Corrupted Packets            ->  247
   Total Number of Corrupted Data Packets      ->  201
   Total Number of Corrupted ACKs              ->  46
 Final Simulation Time                        ->  39108.39008193626
==========STATISTICS==========
```

Analysis

With a higher loss and corrupt probability of 0.8, packets have an 80% chance of being dropped and corrupted. For each ACK, implicit NAK sent by the receiver, as well as retransmission and initial transmission of packets by the sender, the total message sent counter would be incremented. When corrupted packets are received by the receiver, implicit NAKs would be sent to the sender. The packet would be retransmitted when the sender did not receive the correct packet or did not receive a packet at all, which causes the timer for the packet to expire. Since corrupted packets would be dropped on top of packets itself having 80% chance of being dropped, the total number of retransmissions and

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

dropped packets would be higher than test case 5, which explains the increase of total message sent from 133 to 1414. A significant higher number of total messages sent would contribute to a longer simulation time.

Since there are more corrupted packets or incorrect ACKs received by the sender, the state of the sender would not change to **“WAIT_LAYER5”** in time. Hence, new messages from layer 5 (application) would be dropped by the sender since it is still in **“WAIT_ACK”** state, which explains the phenomenon of the print statements **"waiting for ack, new message is dropped: <message>”**

Test Case 7:

Condition

Nsimmax = 30, lossprob = 0.8, corruptprob = 0.8, Lambda = 100000

Name: Jun Hao Cheh

Submission Date: 3/14/2023

Coding language: Python

Sender source code file name: SNW_Sender.py

Receiver source code file name: SNW_Receiver.py

Output

```
data received:  aaaaaaaaaaaaaaaaaaaa
data received:  bbbbbbbbbbbbbbbbbbbb
data received:  cccccccccccccccccccc
data received:  dddddddddddddddddddd
data received:  eeeeeeeeeeeeeeeeeeee
data received:  ffffffffffffffffffff
data received:  gggggggggggggggggggg
data received:  hhhhhhhhhhhhhhhhhhhh
data received:  iiiiiiiiiiiiiiiiiiii
data received:  jjjjjjjjjjjjjjjjjjjj
data received:  kkkkkkkkkkkkkkkkkkkk
data received:  llllllllllllllllllll
data received:  mmmmmmmmmmmmmmmmmmmm
data received:  nnnnnnnnnnnnnnnnnnnn
data received:  oooooooooooooooooooo
data received:  pppppppppppppppppppp
data received:  qqqqqqqqqqqqqqqqqqqq
data received:  rrrrrrrrrrrrrrrrrrrr
data received:  ssssssssssssssssssss
data received:  tttttttttttttttttttt
data received:  uuuuuuuuuuuuuuuuuuuu
data received:  vvvvvvvvvvvvvvvvvvvv
data received:  wwwwwwwwwwwwwwwwwwww
data received:  xxxxxxxxxxxxxxxxxxxx
data received:  yyyyyyyyyyyyyyyyyyyy
data received:  zzzzzzzzzzzzzzzzzzzz
data received:  aaaaaaaaaaaaaaaaaaaa
data received:  bbbbbbbbbbbbbbbbbbbb
data received:  cccccccccccccccccccc
data received:  dddddddddddddddddddd
simulation complete
==========STATISTICS==========
 Total Number of Messages Sent               ->  5503
 Total Number of Retransmissions             ->  5443
   Total Number of Retransmitted Data Packets  ->  4563
   Total Number of Retransmitted ACKs          ->  880
 Total Number of Dropped Packets             ->  1021
   Total Number of Dropped Data Packets        ->  880
   Total Number of Dropped ACKs                ->  141
 Total Number of Corrupted Packets           ->  850
   Total Number of Corrupted Data Packets      ->  710
   Total Number of Corrupted ACKs              ->  140
 Final Simulation Time                       ->  3004993.527533023
==========STATISTICS==========
```

Analysis

When the Lambda is increased from 1000 to 100000, the arrival rate of messages from layer 5 (application) would be slower, so the final simulation time would be higher. Despite the high loss and corruption probabiltiy of 0.8, the sender would be able to change states from **“WAIT_LAYER5”** to **“WAIT_ACK”** before a new message is received from layer 5. With the Lambda increased, the sender would have more time to handle timeouts, when it does not receive an expected packet, which is not corrupted and with a correct sequence. Hence, the total number of retransmitted data packets is higher, which contributes to an overall higher count of total messages sent of 5503.