NAME - GAUTAM KUMAR MAHAR(2103114) AND AUMKAR LOREKAR (2003108)

LAB 4 (Sliding Window Protocols)

Question 1.

You are provided with a Template that consists of Python code for simulating the Go-Back-N Protocol. Go through the code carefully and try to understand the behavior of each block in the Template.

a) Check if the implementation of the rdt_Sender and rdt_Receiver classes matches the Finite State Machine (FSM) description given in the textbook1 .

Answer - The FSM description provided in the textbook fits the implementation, yes.

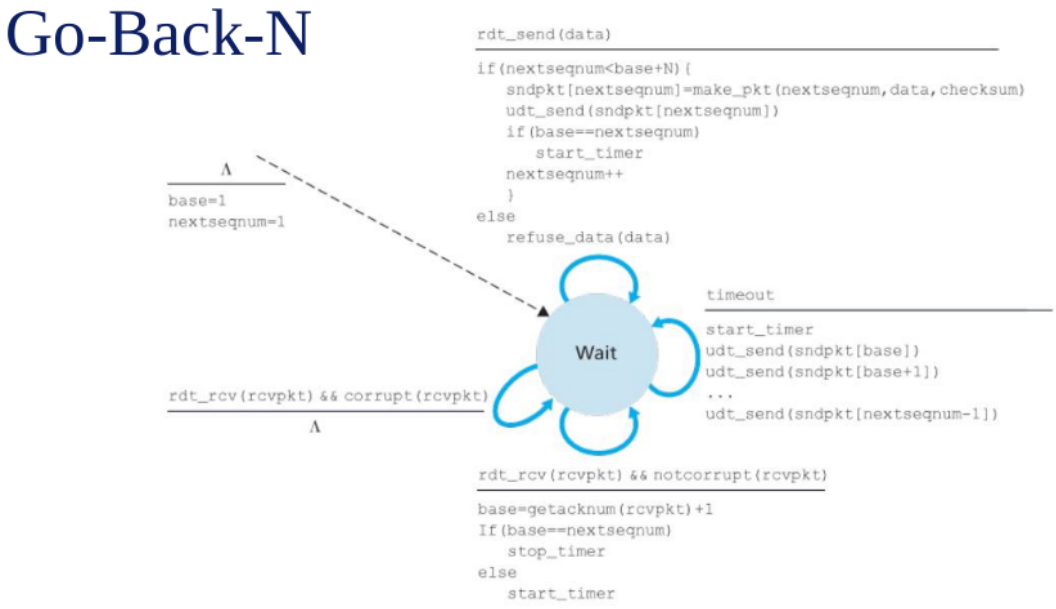


Figure 3.20 Extended FSM description of the GBN sender

Go-Back-N

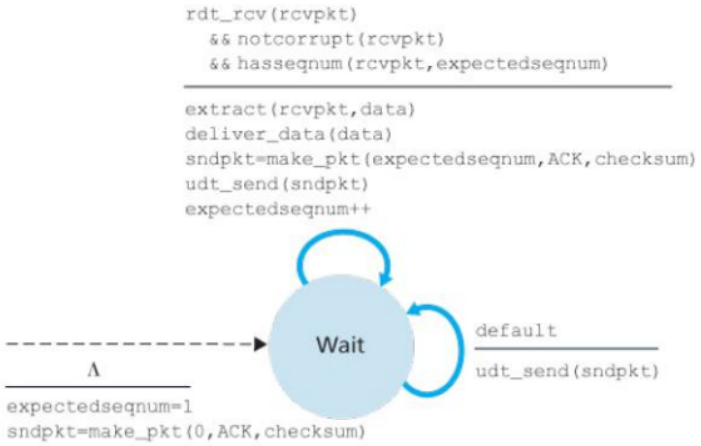


Figure 3.21 Extended FSM description of the GBN receiver

b) List at-least one major difference you observe between the Python implementation for the rdt sender/receiver classes and the FSM description in the textbook.

Answer -

The following are some of the differences between the python implementation provided and the fsm from the textbook -

→When sending packets in the window in FSM, the timeout is restarted first. But, in this case, the timeout is restarted after the packets have been sent again in code.

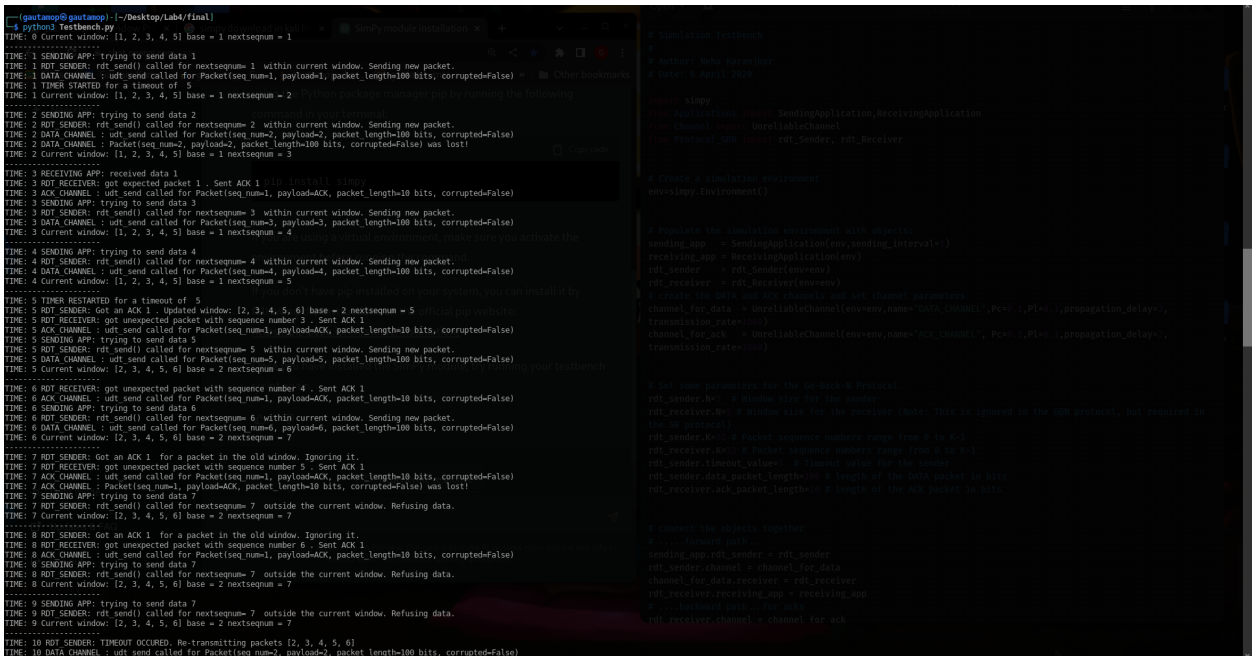
In this case, the maximum value of the sequence number is not set to k.

→ Every time the seq num is increased, the buffer size is taken into consideration in order to loop back to 0 following an overflow.

c) Simulate the system by running the command $ python3 Testbench.py Check if the simulation runs without any errors for the case where Pc (probability of packet corruption) and Pl (probability of packet loss) are non zero for both the DATA and ACK channels.

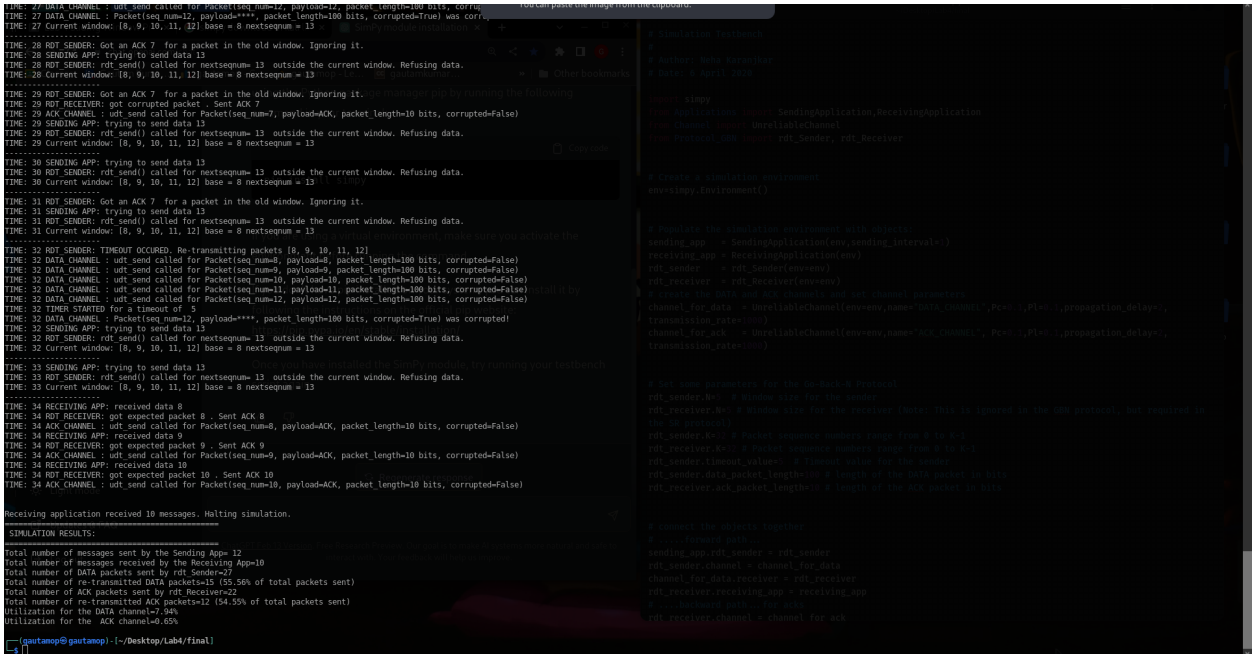
Answer -

It appears from the output -



```
TIME: 10 RDT_SENDER: rdt_send() called for nextseqnum= 7  outside the current window. Refusing data.
TIME: 10 Current window: [2, 3, 4, 5, 6] base = 2 nextseqnum = 7
--------------------
TIME: 11 SENDING APP: trying to send data 7
TIME: 11 RDT_SENDER: rdt_send() called for nextseqnum= 7  outside the current window. Refusing data.
TIME: 11 Current window: [2, 3, 4, 5, 6] base = 2 nextseqnum = 7
--------------------
TIME: 12 RECEIVING APP: received data 2
TIME: 12 RDT_RECEIVER: got expected packet 2 . Sent ACK 2
TIME: 12 ACK_CHANNEL : udt_send called for Packet(seq_num=2, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 12 RECEIVING APP: received data 3
TIME: 12 RDT_RECEIVER: got expected packet 3 . Sent ACK 3
TIME: 12 ACK_CHANNEL : udt_send called for Packet(seq_num=3, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 12 RDT_RECEIVER: got unexpected packet with sequence number 5 . Sent ACK 3
TIME: 12 ACK_CHANNEL : udt_send called for Packet(seq_num=3, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 12 RDT_RECEIVER: got corrupted packet . Sent ACK 3
TIME: 12 ACK_CHANNEL : udt_send called for Packet(seq_num=3, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 12 SENDING APP: trying to send data 7
TIME: 12 RDT_SENDER: rdt_send() called for nextseqnum= 7  outside the current window. Refusing data.
TIME: 12 Current window: [2, 3, 4, 5, 6] base = 2 nextseqnum = 7
--------------------
TIME: 13 SENDING APP: trying to send data 7
TIME: 13 RDT_SENDER: rdt_send() called for nextseqnum= 7  outside the current window. Refusing data.
TIME: 13 Current window: [2, 3, 4, 5, 6] base = 2 nextseqnum = 7
--------------------
TIME: 14 TIMER RESTARTED for a timeout of  5
TIME: 14 RDT_SENDER: Got an ACK 2 . Updated window: [3, 4, 5, 6, 7] base = 3 nextseqnum = 7
TIME: 14 TIMER RESTARTED for a timeout of  5
TIME: 14 RDT_SENDER: Got an ACK 3 . Updated window: [4, 5, 6, 7, 8] base = 4 nextseqnum = 7
TIME: 14 RDT_SENDER: Got an ACK 3  for a packet in the old window. Ignoring it.
TIME: 14 RDT_SENDER: Got an ACK 3  for a packet in the old window. Ignoring it.
TIME: 14 SENDING APP: trying to send data 7
TIME: 14 RDT_SENDER: rdt_send() called for nextseqnum= 7  within current window. Sending new packet.
TIME: 14 DATA_CHANNEL : udt_send called for Packet(seq_num=7, payload=7, packet_length=100 bits, corrupted=False)
TIME: 14 Current window: [4, 5, 6, 7, 8] base = 4 nextseqnum = 8
--------------------
TIME: 15 SENDING APP: trying to send data 8
TIME: 15 RDT_SENDER: rdt_send() called for nextseqnum= 8  within current window. Sending new packet.
TIME: 15 DATA_CHANNEL : udt_send called for Packet(seq_num=8, payload=8, packet_length=100 bits, corrupted=False)
TIME: 15 Current window: [4, 5, 6, 7, 8] base = 4 nextseqnum = 9
--------------------
TIME: 16 RDT_RECEIVER: got unexpected packet with sequence number 7 . Sent ACK 3
TIME: 16 ACK_CHANNEL : udt_send called for Packet(seq_num=3, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 16 SENDING APP: trying to send data 9
TIME: 16 RDT_SENDER: rdt_send() called for nextseqnum= 9  outside the current window. Refusing data.
TIME: 16 Current window: [4, 5, 6, 7, 8] base = 4 nextseqnum = 9
--------------------
TIME: 17 RDT_RECEIVER: got unexpected packet with sequence number 8 . Sent ACK 3
TIME: 17 ACK_CHANNEL : udt_send called for Packet(seq_num=3, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 17 SENDING APP: trying to send data 9
TIME: 17 RDT_SENDER: rdt_send() called for nextseqnum= 9  outside the current window. Refusing data.
TIME: 17 Current window: [4, 5, 6, 7, 8] base = 4 nextseqnum = 9
--------------------
TIME: 18 RDT_SENDER: Got an ACK 3  for a packet in the old window. Ignoring it.
TIME: 18 SENDING APP: trying to send data 9
TIME: 18 RDT_SENDER: rdt_send() called for nextseqnum= 9  outside the current window. Refusing data.
TIME: 18 Current window: [4, 5, 6, 7, 8] base = 4 nextseqnum = 9
--------------------
TIME: 19 RDT_SENDER: TIMEOUT OCCURED. Re-transmitting packets [4, 5, 6, 7, 8]
TIME: 19 DATA_CHANNEL : udt_send called for Packet(seq_num=4, payload=4, packet_length=100 bits, corrupted=False)
TIME: 19 DATA_CHANNEL : udt_send called for Packet(seq_num=5, payload=5, packet_length=100 bits, corrupted=False)
TIME: 19 DATA_CHANNEL : udt_send called for Packet(seq_num=6, payload=6, packet_length=100 bits, corrupted=False)
TIME: 19 DATA_CHANNEL : udt_send called for Packet(seq_num=7, payload=7, packet_length=100 bits, corrupted=False)
TIME: 19 DATA_CHANNEL : udt_send called for Packet(seq_num=8, payload=8, packet_length=100 bits, corrupted=False)
TIME: 19 TIMER STARTED for a timeout of  5
TIME: 19 DATA_CHANNEL : Packet(seq_num=8, payload=8, packet_length=100 bits, corrupted=False) was lost!
TIME: 19 RDT_SENDER: Got an ACK 3  for a packet in the old window. Ignoring it.
```

```
--------------------
TIME: 20 SENDING APP: trying to send data 9
TIME: 20 RDT_SENDER: rdt_send() called for nextseqnum= 9  outside the current window. Refusing data.
TIME: 20 Current window: [4, 5, 6, 7, 8] base = 4 nextseqnum = 9
--------------------
TIME: 21 RECEIVING APP: received data 4
TIME: 21 RDT_RECEIVER: got expected packet 4 . Sent ACK 4
TIME: 21 ACK_CHANNEL : udt_send called for Packet(seq_num=4, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 21 RECEIVING APP: received data 5
TIME: 21 RDT_RECEIVER: got expected packet 5 . Sent ACK 5
TIME: 21 ACK_CHANNEL : udt_send called for Packet(seq_num=5, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 21 RECEIVING APP: received data 6
TIME: 21 RDT_RECEIVER: got expected packet 6 . Sent ACK 6
TIME: 21 ACK_CHANNEL : udt_send called for Packet(seq_num=6, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 21 RECEIVING APP: received data 7
TIME: 21 RDT_RECEIVER: got expected packet 7 . Sent ACK 7
TIME: 21 ACK_CHANNEL : udt_send called for Packet(seq_num=7, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 21 ACK_CHANNEL : Packet(seq_num=7, payload=ACK, packet_length=10 bits, corrupted=False) was lost!
TIME: 21 SENDING APP: trying to send data 9
TIME: 21 RDT_SENDER: rdt_send() called for nextseqnum= 9  outside the current window. Refusing data.
TIME: 21 Current window: [4, 5, 6, 7, 8] base = 4 nextseqnum = 9
--------------------
TIME: 22 SENDING APP: trying to send data 9
TIME: 22 RDT_SENDER: rdt_send() called for nextseqnum= 9  outside the current window. Refusing data.
TIME: 22 Current window: [4, 5, 6, 7, 8] base = 4 nextseqnum = 9
--------------------
TIME: 23 TIMER RESTARTED for a timeout of  5
TIME: 23 RDT_SENDER: Got an ACK 4 . Updated window: [5, 6, 7, 8, 9] base = 5 nextseqnum = 9
TIME: 23 TIMER RESTARTED for a timeout of  5
TIME: 23 RDT_SENDER: Got an ACK 5 . Updated window: [6, 7, 8, 9, 10] base = 6 nextseqnum = 9
TIME: 23 TIMER RESTARTED for a timeout of  5
TIME: 23 RDT_SENDER: Got an ACK 6 . Updated window: [7, 8, 9, 10, 11] base = 7 nextseqnum = 9
TIME: 23 SENDING APP: trying to send data 9
TIME: 23 RDT_SENDER: rdt_send() called for nextseqnum= 9  within current window. Sending new packet.
TIME: 23 DATA_CHANNEL : udt_send called for Packet(seq_num=9, payload=9, packet_length=100 bits, corrupted=False)
TIME: 23 Current window: [7, 8, 9, 10, 11] base = 7 nextseqnum = 10
--------------------
TIME: 24 SENDING APP: trying to send data 10
TIME: 24 RDT_SENDER: rdt_send() called for nextseqnum= 10  within current window. Sending new packet.
TIME: 24 DATA_CHANNEL : udt_send called for Packet(seq_num=10, payload=10, packet_length=100 bits, corrupted=False)
TIME: 24 Current window: [7, 8, 9, 10, 11] base = 7 nextseqnum = 11
--------------------
TIME: 25 RDT_RECEIVER: got unexpected packet with sequence number 9 . Sent ACK 7
TIME: 25 ACK_CHANNEL : udt_send called for Packet(seq_num=7, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 25 SENDING APP: trying to send data 11
TIME: 25 RDT_SENDER: rdt_send() called for nextseqnum= 11  within current window. Sending new packet.
TIME: 25 DATA_CHANNEL : udt_send called for Packet(seq_num=11, payload=11, packet_length=100 bits, corrupted=False)
TIME: 25 Current window: [7, 8, 9, 10, 11] base = 7 nextseqnum = 12
--------------------
TIME: 26 RDT_RECEIVER: got unexpected packet with sequence number 10 . Sent ACK 7
TIME: 26 ACK_CHANNEL : udt_send called for Packet(seq_num=7, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 26 SENDING APP: trying to send data 12
TIME: 26 RDT_SENDER: rdt_send() called for nextseqnum= 12  outside the current window. Refusing data.
TIME: 26 Current window: [7, 8, 9, 10, 11] base = 7 nextseqnum = 12
--------------------
TIME: 27 TIMER RESTARTED for a timeout of  5
TIME: 27 RDT_SENDER: Got an ACK 7 . Updated window: [8, 9, 10, 11, 12] base = 8 nextseqnum = 12
TIME: 27 RDT_RECEIVER: got unexpected packet with sequence number 11 . Sent ACK 7
TIME: 27 ACK_CHANNEL : udt_send called for Packet(seq_num=7, payload=ACK, packet_length=10 bits, corrupted=False)
TIME: 27 SENDING APP: trying to send data 12
TIME: 27 RDT_SENDER: rdt_send() called for nextseqnum= 12  within current window. Sending new packet.
TIME: 27 DATA_CHANNEL : udt_send called for Packet(seq_num=12, payload=12, packet_length=100 bits, corrupted=False)
TIME: 27 DATA_CHANNEL : Packet(seq_num=12, payload='***', packet_length=100 bits, corrupted=True) was corrupted!
TIME: 27 Current window: [8, 9, 10, 11, 12] base = 8 nextseqnum = 13
--------------------
TIME: 28 RDT_SENDER: Got an ACK 7  for a packet in the old window. Ignoring it.
TIME: 28 SENDING APP: trying to send data 13
```



```
Receiving application received 10 messages. Halting simulation.
================================================
 SIMULATION RESULTS:
================================================
Total number of messages sent by the Sending App= 12
Total number of messages received by the Receiving App=10
Total number of DATA packets sent by rdt_Sender=27
Total number of re-transmitted DATA packets=15 (55.56% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=22
Total number of re-transmitted ACK packets=12 (54.55% of total packets sent)
Utilization for the DATA channel=7.94%
Utilization for the  ACK channel=0.65%
```

In the scenario described above, the code effectively implements the go back n protocol with Pl = Pc = 0.1.

<u>Question 2.</u>

Assuming the parameter values for the DATA channel and the ACK channel are identical, set the parameter values (in file Testbench.py) as follows: ◦ Pc=0.2, Pl=0.2 , channel propagation delay = 2 seconds ◦ Packet_length for DATA packets = 1000 bits, packet length for ACK packets = 10 bits ◦ Transmission rate = 1000 bits per second ◦ Window size(N)=10, Range of sequence numbers(K)=16

Modify the file Testbench.py so that a single simulation is run exactly until the receiving application receives the first 1000 messages.

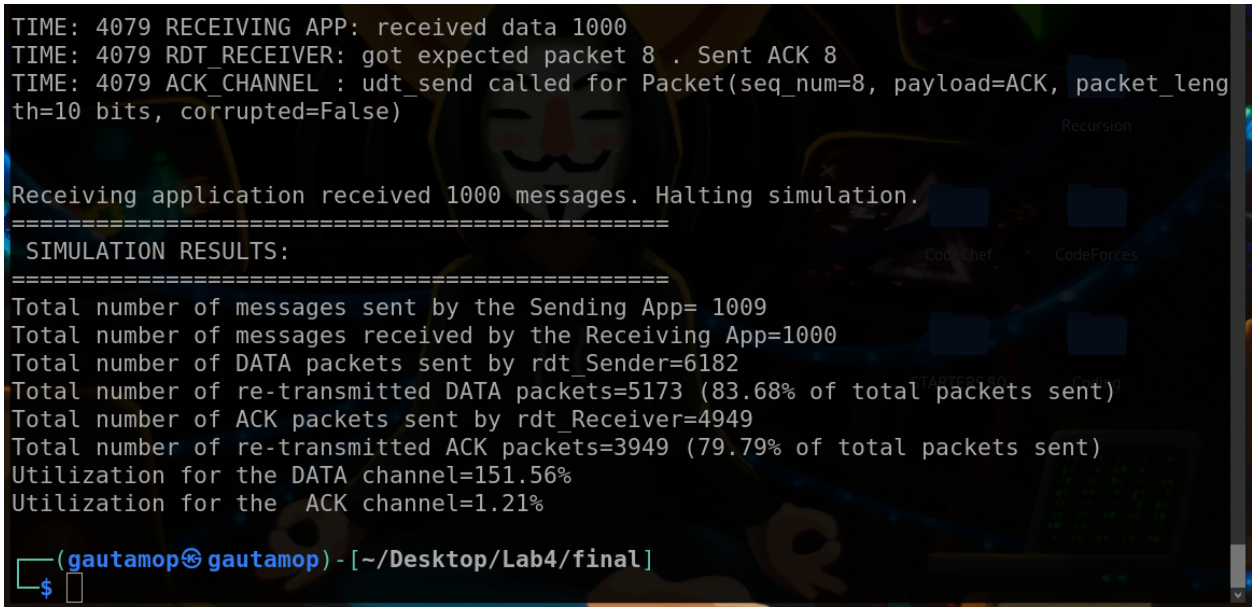
a) Report the time at which the receiving application receives the first 1000 messages, averaged across five such simulation runs.

Answer - Applying the test bench's modifications listed below

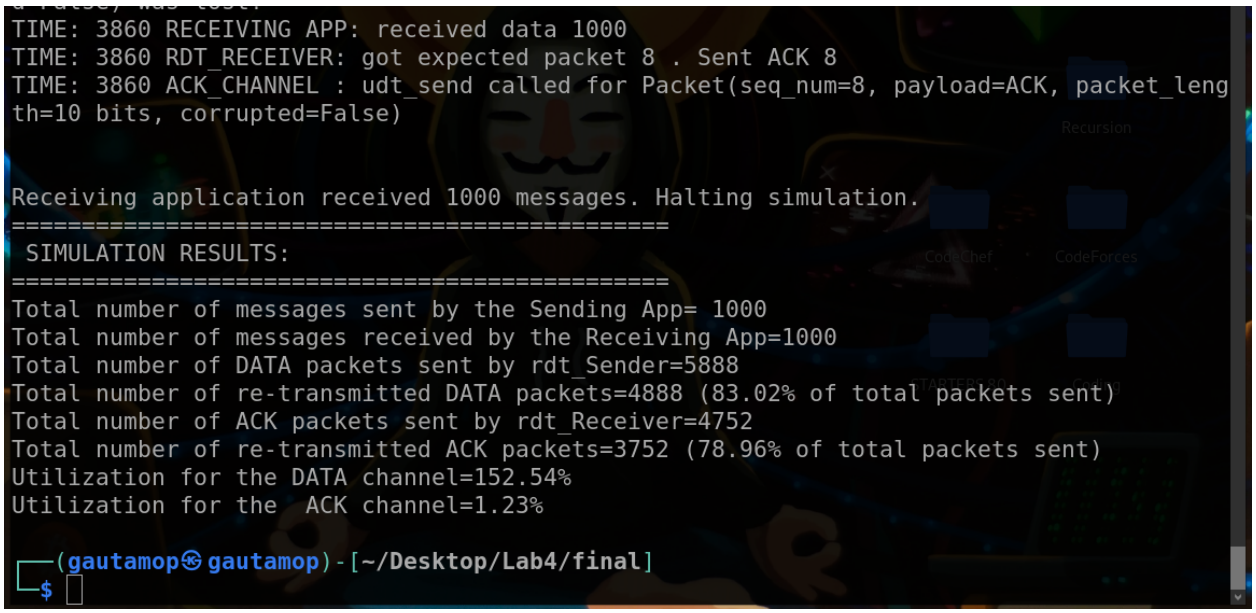
```
# Populate the simulation environment with objects:
sending_app   = SendingApplication(env,sending_interval=1)
receiving_app = ReceivingApplication(env)
rdt_sender    = rdt_Sender(env=env)
rdt_receiver  = rdt_Receiver(env=env)
# create the DATA and ACK channels and set channel parameters
channel_for_data  = UnreliableChannel(env=env,name="DATA_CHANNEL",Pc=0.2,Pl=0.2,propagation_delay=2,
transmission_rate=1000)
channel_for_ack   = UnreliableChannel(env=env,name="ACK_CHANNEL", Pc=0.2,Pl=0.2,propagation_delay=2,
transmission_rate=1000)


# Set some parameters for the Go-Back-N Protocol
rdt_sender.N=10 # Window size for the sender
rdt_receiver.N=10 # Window size for the receiver (Note: This is ignored in the GBN protocol, but required in
the SR protocol)
rdt_sender.K=16 # Packet sequence numbers range from 0 to K-1
rdt_receiver.K=16 # Packet sequence numbers range from 0 to K-1
rdt_sender.timeout_value=5  # Timeout value for the sender
rdt_sender.data_packet_length=1000 # length of the DATA packet in bits
rdt_receiver.ack_packet_length=10 # length of the ACK packet in bits
```

```
# Run simulation, and print status information every now and then.
# Run the simulation until TOTAL_SIMULATION_TIME elapses OR the receiver receives a certain
# number of messages in total, whichever occurs earlier.

TOTAL_SIMULATION_TIME=100000000 # ⟸ Total simulation time. Increase it as you like.
t=0
while env.peek() ⩽ TOTAL_SIMULATION_TIME:
    if(env.peek()>t):
        rdt_sender.print_status()
    env.step()
    t=int(env.now)
    # We may wish to halt the simulation if some condition occurs.
    # For example, if the receiving application recives 100 messages.
    num_msg = receiving_app.total_messages_received
    if num_msg ⩾ 1000: # ⟸ Halt simulation when receiving application receives these many messages.
        print("\nTime Taken", t)
        print("\n\nReceiving application received",num_msg,"messages. Halting simulation.")
        break
if t==TOTAL_SIMULATION_TIME:
    print("\n\nTotal simulation time has elapsed. Halting simulation.")
```

Simulation_1



Simulation_2



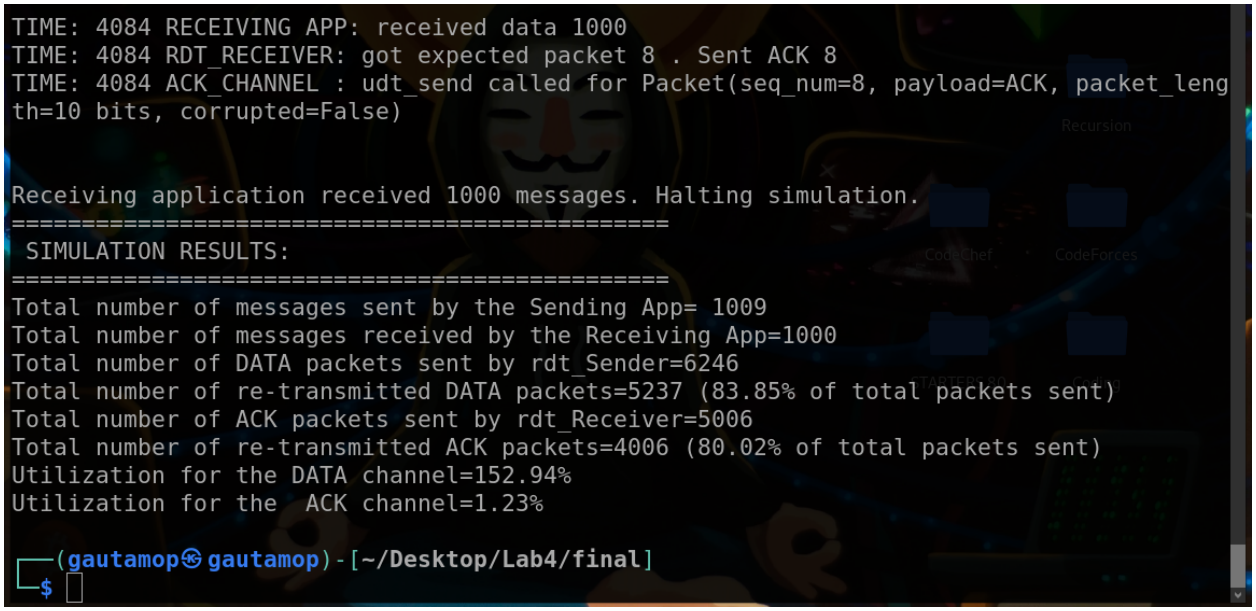
Simulation_3

```
TIME: 4212 RECEIVING APP: received data 1000
TIME: 4212 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 4212 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, packet_leng
th=10 bits, corrupted=False)


Receiving application received 1000 messages. Halting simulation.
================================================
 SIMULATION RESULTS:
================================================
Total number of messages sent by the Sending App= 1008
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=6479
Total number of re-transmitted DATA packets=5471 (84.44% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=5128
Total number of re-transmitted ACK packets=4128 (80.50% of total packets sent)
Utilization for the DATA channel=153.82%
Utilization for the  ACK channel=1.22%

┌──(gautamop㉿gautamop)-[~/Desktop/Lab4/final]
└─$
```

Simulation_4

```
TIME: 4106 RECEIVING APP: received data 1000
TIME: 4106 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 4106 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, packet_leng
th=10 bits, corrupted=False)


Receiving application received 1000 messages. Halting simulation.
================================================
 SIMULATION RESULTS:
================================================
Total number of messages sent by the Sending App= 1003
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=6159
Total number of re-transmitted DATA packets=5156 (83.71% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=4921
Total number of re-transmitted ACK packets=3921 (79.68% of total packets sent)
Utilization for the DATA channel=150.00%
Utilization for the  ACK channel=1.20%

┌──(gautamop㉿gautamop)-[~/Desktop/Lab4/final]
└─$
```

Simulation_5



The average amount of time it takes to get the messages throughout these five simulations is (4179+3860+4212+4106+4084)/5 = 4088.2

b) Measure and report channel utilization on the sender side (for the DATA channel) averaged across five simulation runs.

Answer -

The average channel utilization across these 5 simulations to receive the messages is (151.56+152.54+153.82+150.00+152.94)/5 = 152.17%

When transmission delays are taken into account when determining the channel's overall utilisation, this number is higher than 100%. As a result, the sender continues to transmit packets after it should have stopped to wait for previously sent packets to be received on the other side.

c) Measure and report (across five simulation runs) the fraction of the packets sent by the rdt_Sender that are simply retransmissions of previously sent packets.

Answer -

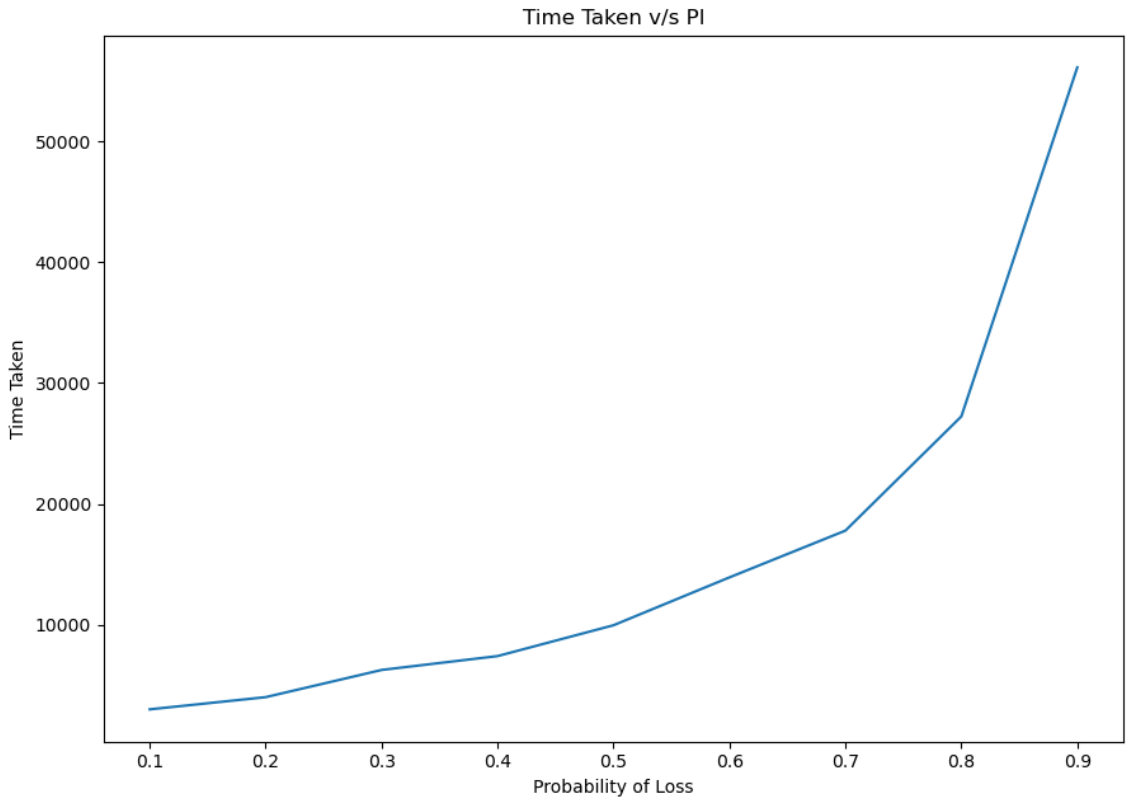
The average number of data packets that must be retransmitted in order to get the messages in these 5 scenarios is (83.68+83.02+84.44+83.71+83.85)/5 = 83.74

Question 3.

We wish to study the effect of channel parameters (such as Pc and Pl) and the protocol parameters (such as the Window size N) on the performance of this protocol. It is reasonable to believe that as Pc and Pl increase, more and more retransmissions will occur that will increase the channel utilization.

Answer the following questions keeping all parameter values the same as those listed in question 2 (changing only those parameter values as asked) and assuming that the parameter values for the DATA and ACK channels are identical.

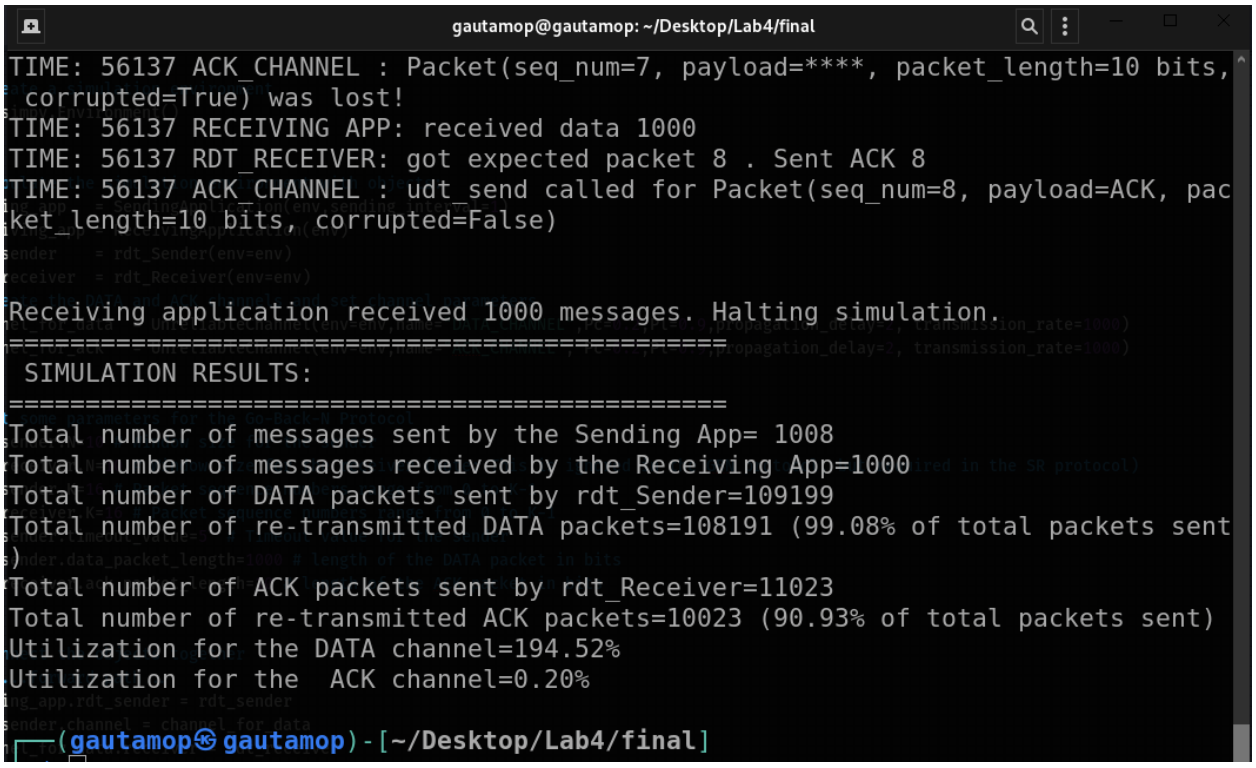
a) Let T be the simulation time at which the receiving application receives the first 1000 messages. Draw a plot to show how T varies with Pl (as Pl ranges from 0.1 to 0.9).



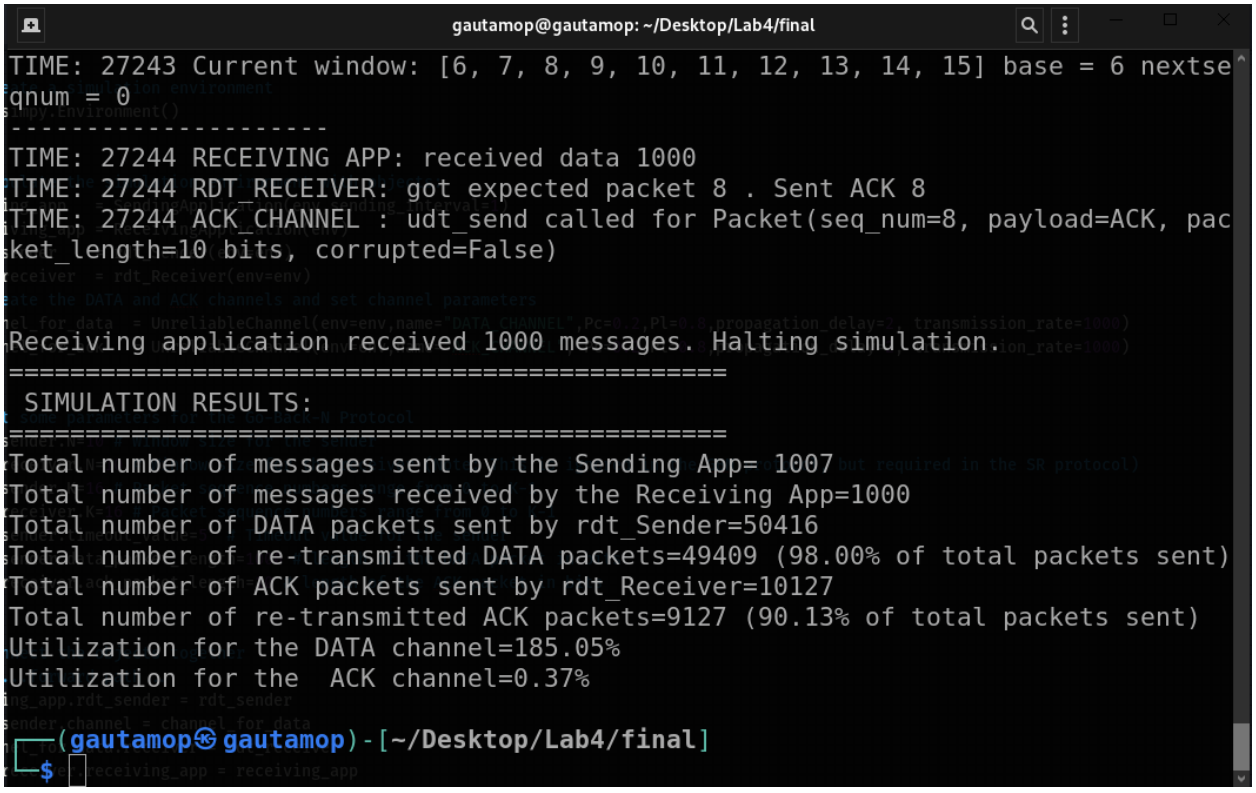

```
# x axis values - pl
x = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
# corresponding y axis values - time
y = [2993,  4002, 6254, 7402, 9955, 13918, 17788, 27244, 56137]
```

Pl Ranges From 0.1 to 09

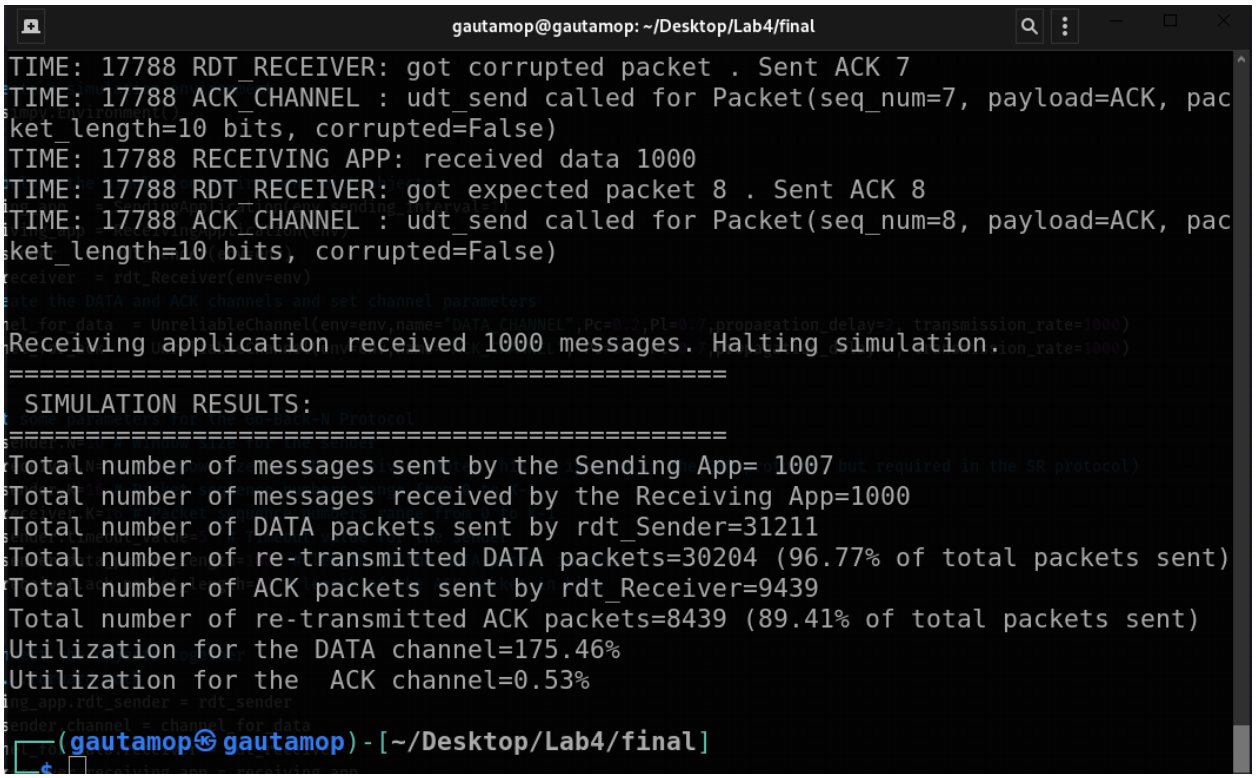
Pl = 0.9



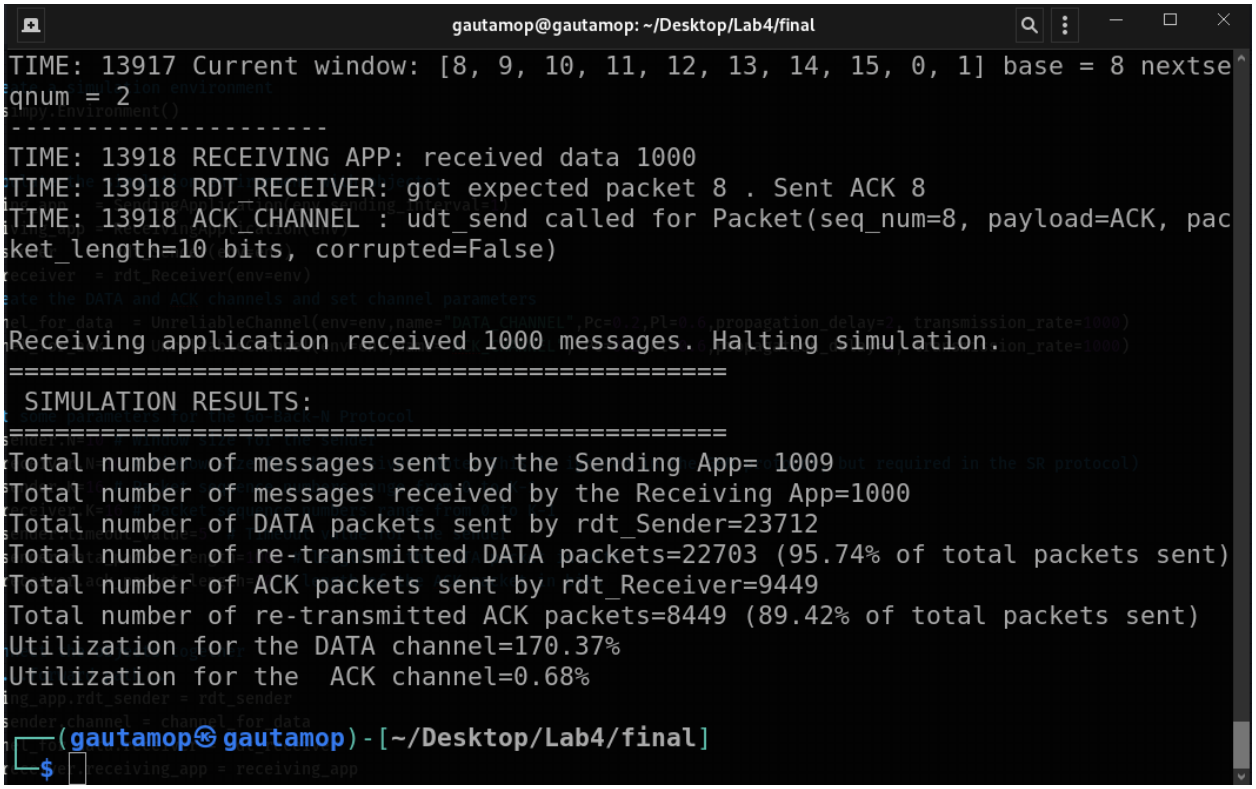
Pl = 0.8


gautamop@gautamop: ~/Desktop/Lab4/final
TIME: 27243 Current window: [6, 7, 8, 9, 10, 11, 12, 13, 14, 15] base = 6 nextse
qnum = 0
TIME: 27244 RECEIVING APP: received data 1000
TIME: 27244 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 27244 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, pac
ket_length=10 bits, corrupted=False)
Receiving application received 1000 messages. Halting simulation.
SIMULATION RESULTS:
Total number of messages sent by the Sending App= 1007
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=50416
Total number of re-transmitted DATA packets=49409 (98.00% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=10127
Total number of re-transmitted ACK packets=9127 (90.13% of total packets sent)
Utilization for the DATA channel=185.05%
Utilization for the ACK channel=0.37%
(gautamop㉿gautamop)-[~/Desktop/Lab4/final]
$

Pl=0.7

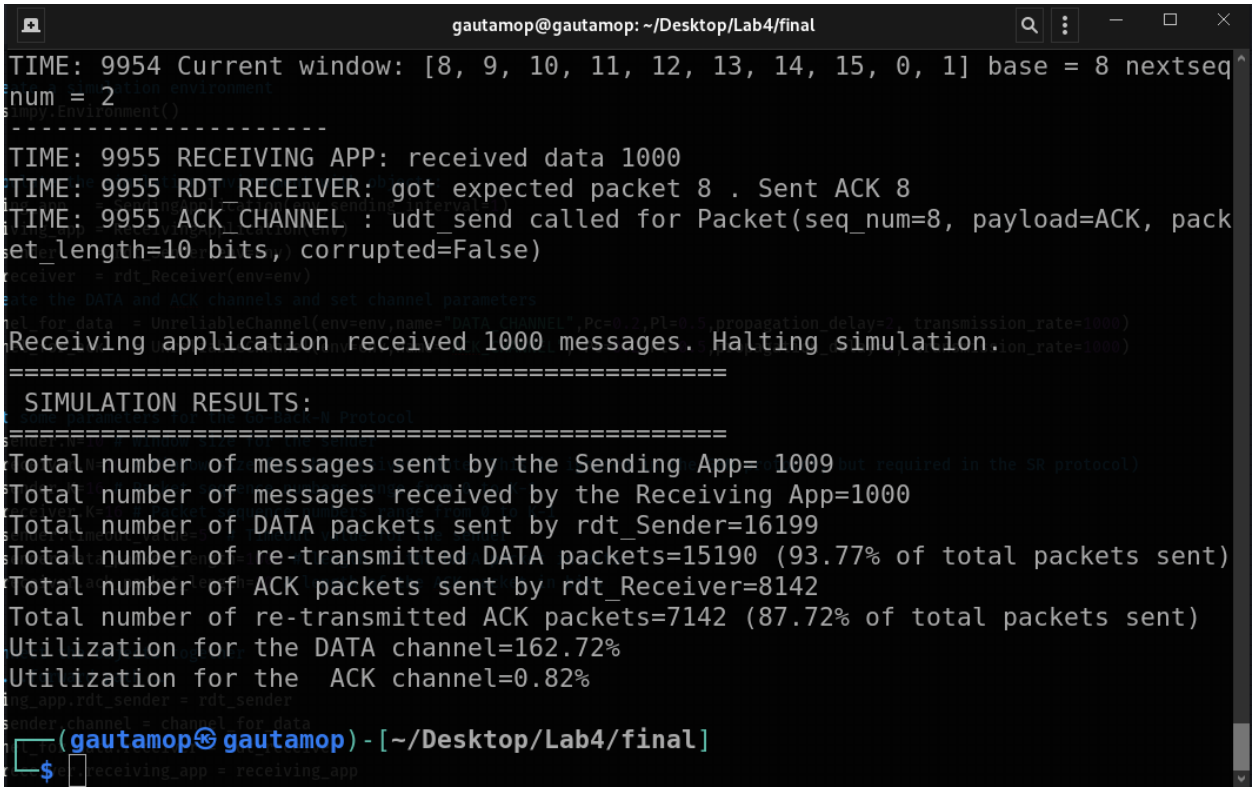

gautamop@gautamop: ~/Desktop/Lab4/final
TIME: 17788 RDT_RECEIVER: got corrupted packet . Sent ACK 7
TIME: 17788 ACK_CHANNEL : udt_send called for Packet(seq_num=7, payload=ACK, pac
ket_length=10 bits, corrupted=False)
TIME: 17788 RECEIVING APP: received data 1000
TIME: 17788 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 17788 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, pac
ket_length=10 bits, corrupted=False)
Receiving application received 1000 messages. Halting simulation.
SIMULATION RESULTS:
Total number of messages sent by the Sending App= 1007
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=31211
Total number of re-transmitted DATA packets=30204 (96.77% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=9439
Total number of re-transmitted ACK packets=8439 (89.41% of total packets sent)
Utilization for the DATA channel=175.46%
Utilization for the ACK channel=0.53%
(gautamop㉿gautamop)-[~/Desktop/Lab4/final]

Pl = 0.6


gautamop@gautamop: ~/Desktop/Lab4/final
TIME: 13917 Current window: [8, 9, 10, 11, 12, 13, 14, 15, 0, 1] base = 8 nextse
qnum = 2

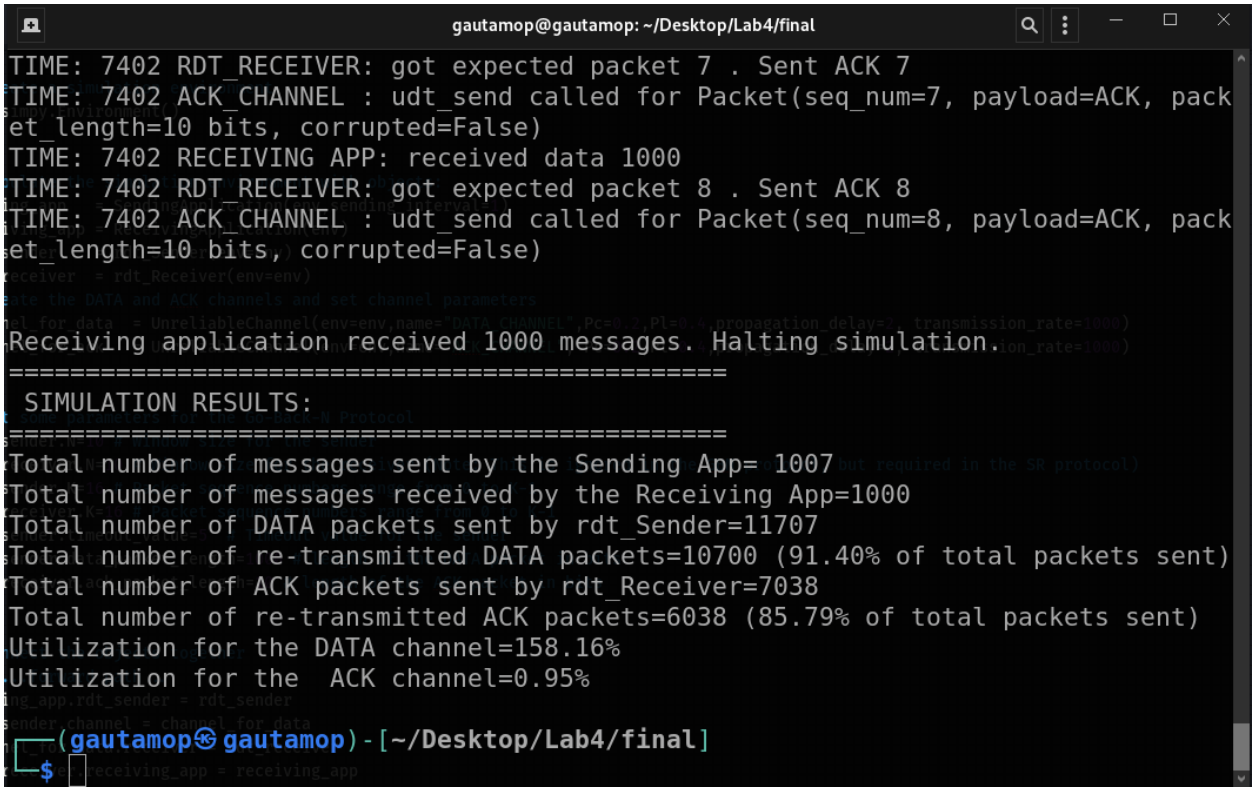
TIME: 13918 RECEIVING APP: received data 1000
TIME: 13918 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 13918 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, pac
ket_length=10 bits, corrupted=False)
Receiving application received 1000 messages. Halting simulation.
SIMULATION RESULTS:
Total number of messages sent by the Sending App= 1009
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=23712
Total number of re-transmitted DATA packets=22703 (95.74% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=9449
Total number of re-transmitted ACK packets=8449 (89.42% of total packets sent)
Utilization for the DATA channel=170.37%
Utilization for the ACK channel=0.68%
(gautamop㉿gautamop)-[~/Desktop/Lab4/final]
$

Pl = 0.5

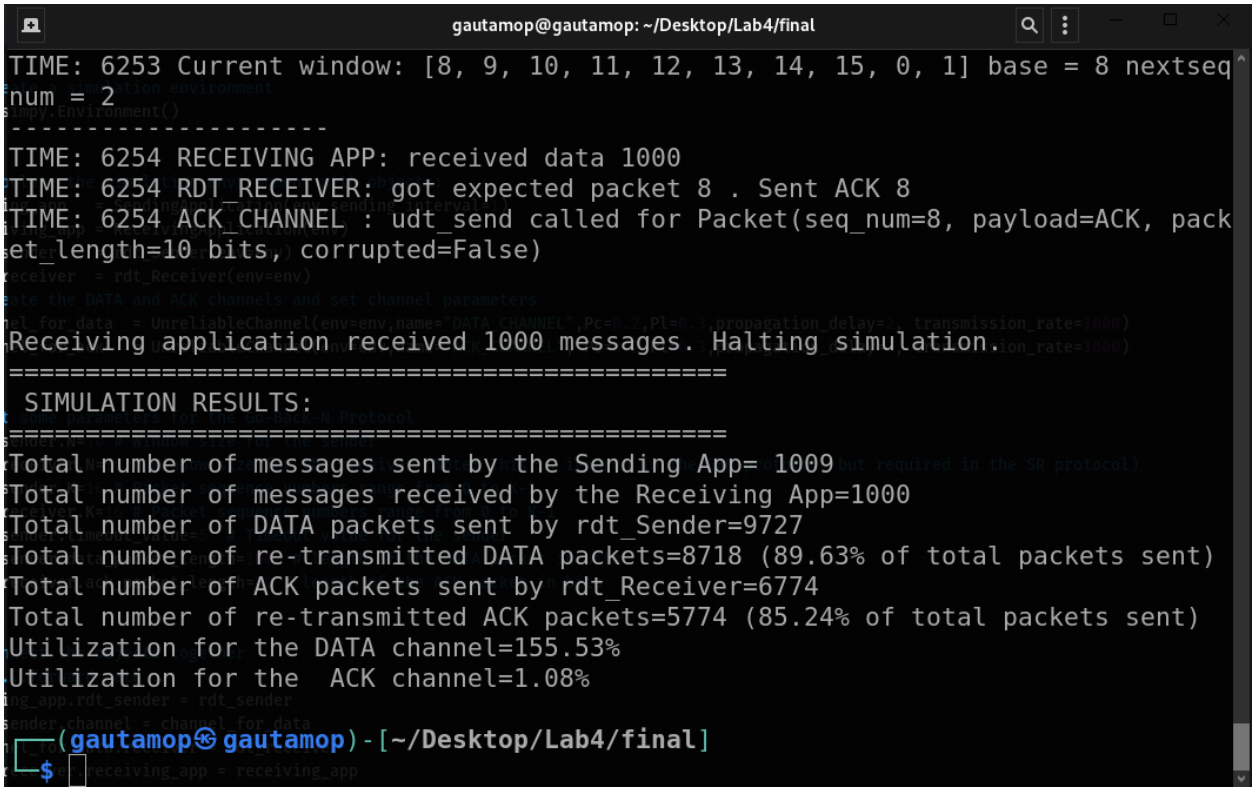

gautamop@gautamop: ~/Desktop/Lab4/final
TIME: 9954 Current window: [8, 9, 10, 11, 12, 13, 14, 15, 0, 1] base = 8 nextseq
num = 2

TIME: 9955 RECEIVING APP: received data 1000
TIME: 9955 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 9955 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, pack
et_length=10 bits, corrupted=False)
Receiving application received 1000 messages. Halting simulation.
SIMULATION RESULTS:
Total number of messages sent by the Sending App= 1009
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=16199
Total number of re-transmitted DATA packets=15190 (93.77% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=8142
Total number of re-transmitted ACK packets=7142 (87.72% of total packets sent)
Utilization for the DATA channel=162.72%
Utilization for the ACK channel=0.82%
(gautamop㉿gautamop)-[~/Desktop/Lab4/final]
$

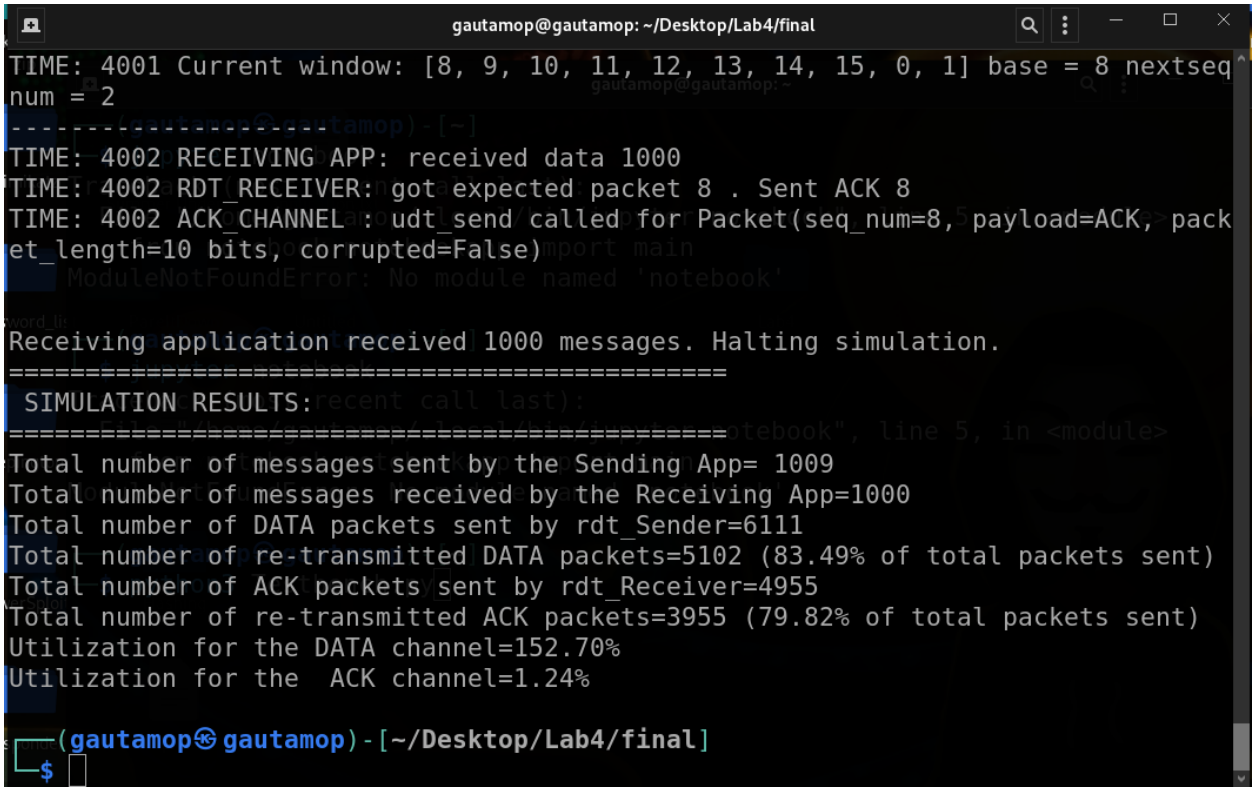
Pl = 0.4


gautamop@gautamop: ~/Desktop/Lab4/final
TIME: 7402 RDT_RECEIVER: got expected packet 7 . Sent ACK 7
TIME: 7402 ACK_CHANNEL : udt_send called for Packet(seq_num=7, payload=ACK, pack
et_length=10 bits, corrupted=False)
TIME: 7402 RECEIVING APP: received data 1000
TIME: 7402 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 7402 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, pack
et_length=10 bits, corrupted=False)
Receiving application received 1000 messages. Halting simulation.
SIMULATION RESULTS:
Total number of messages sent by the Sending App= 1007
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=11707
Total number of re-transmitted DATA packets=10700 (91.40% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=7038
Total number of re-transmitted ACK packets=6038 (85.79% of total packets sent)
Utilization for the DATA channel=158.16%
Utilization for the ACK channel=0.95%
(gautamop㉿gautamop)-[~/Desktop/Lab4/final]
$

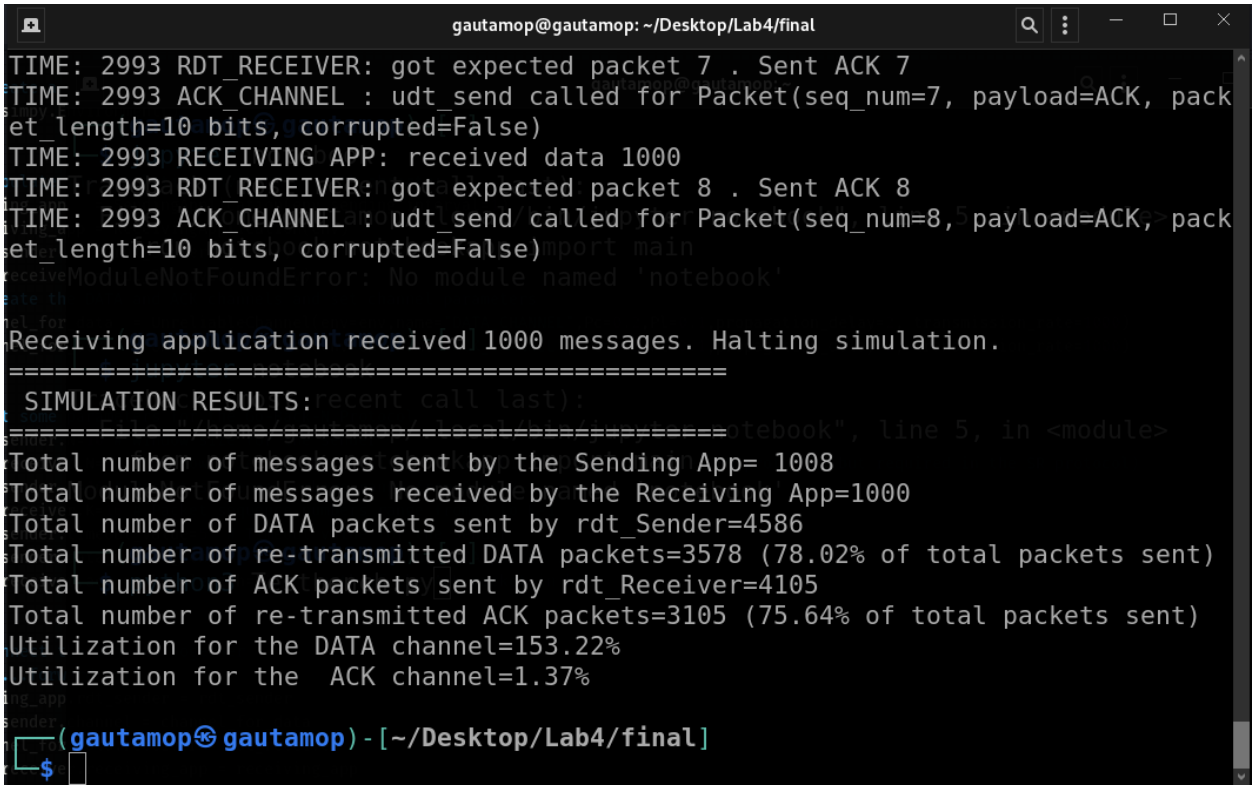
Pl =0.3


gautamop@gautamop: ~/Desktop/Lab4/final
TIME: 6253 Current window: [8, 9, 10, 11, 12, 13, 14, 15, 0, 1] base = 8 nextseq
num = 2
TIME: 6254 RECEIVING APP: received data 1000
TIME: 6254 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 6254 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, pack
et_length=10 bits, corrupted=False)
Receiving application received 1000 messages. Halting simulation.
SIMULATION RESULTS:
Total number of messages sent by the Sending App= 1009
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=9727
Total number of re-transmitted DATA packets=8718 (89.63% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=6774
Total number of re-transmitted ACK packets=5774 (85.24% of total packets sent)
Utilization for the DATA channel=155.53%
Utilization for the ACK channel=1.08%
(gautamop㉿gautamop)-[~/Desktop/Lab4/final]
$

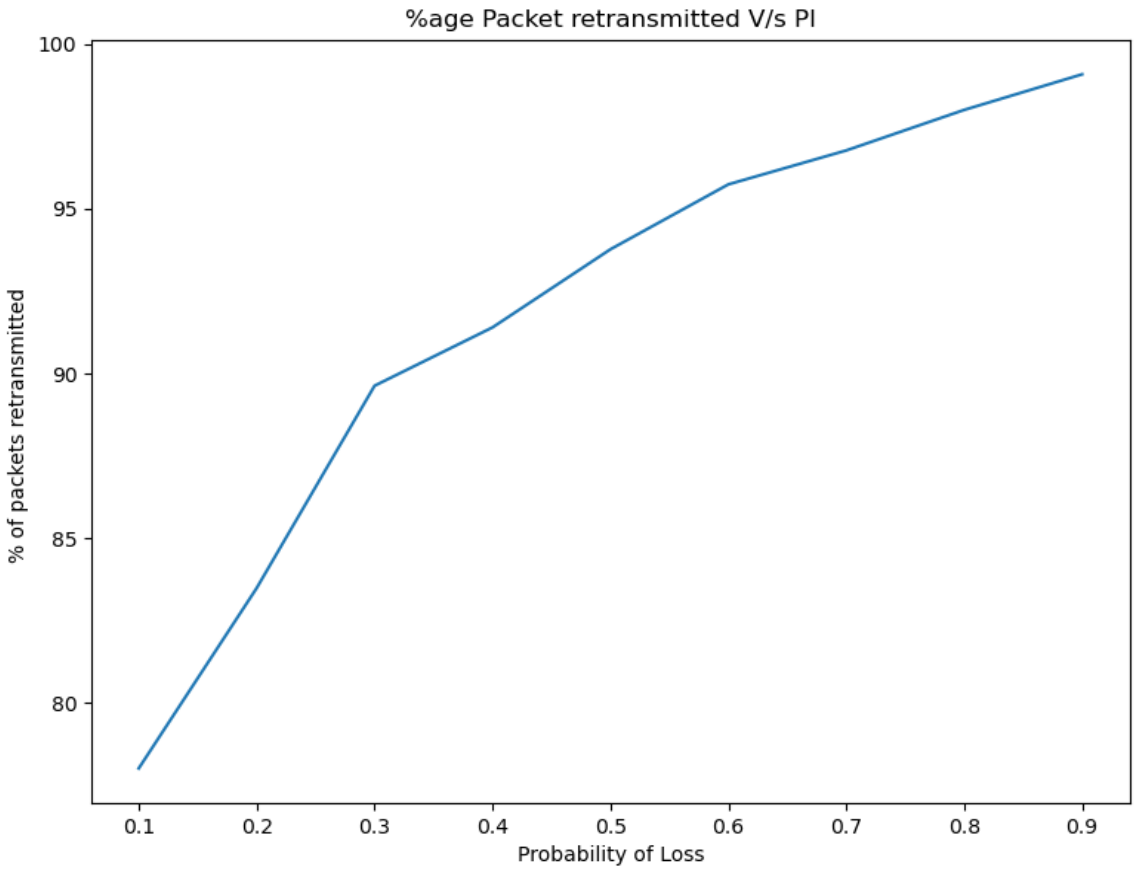
Pl = 0.2


gautamop@gautamop: ~/Desktop/Lab4/final
TIME: 4001 Current window: [8, 9, 10, 11, 12, 13, 14, 15, 0, 1] base = 8 nextseq
num = 2
TIME: 4002 RECEIVING APP: received data 1000
TIME: 4002 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 4002 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, pack
et_length=10 bits, corrupted=False)
Receiving application received 1000 messages. Halting simulation.
SIMULATION RESULTS:
Total number of messages sent by the Sending App= 1009
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=6111
Total number of re-transmitted DATA packets=5102 (83.49% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=4955
Total number of re-transmitted ACK packets=3955 (79.82% of total packets sent)
Utilization for the DATA channel=152.70%
Utilization for the ACK channel=1.24%
(gautamop㉿gautamop)-[~/Desktop/Lab4/final]
$

Pl = 0.1

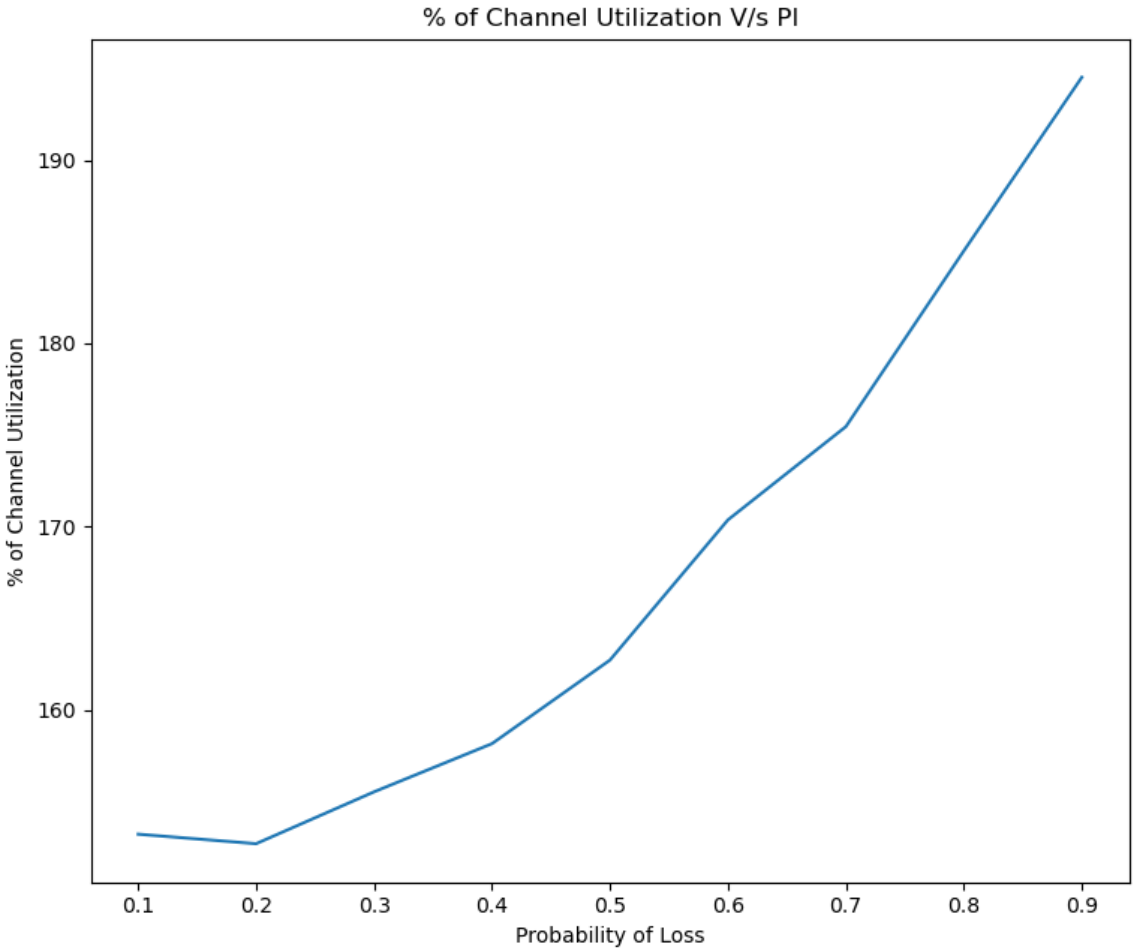

gautamop@gautamop: ~/Desktop/Lab4/final
TIME: 2993 RDT_RECEIVER: got expected packet 7 . Sent ACK 7
TIME: 2993 ACK_CHANNEL : udt_send called for Packet(seq_num=7, payload=ACK, pack
et_length=10 bits, corrupted=False)
TIME: 2993 RECEIVING APP: received data 1000
TIME: 2993 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 2993 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, pack
et_length=10 bits, corrupted=False)
Receiving application received 1000 messages. Halting simulation.
SIMULATION RESULTS:
Total number of messages sent by the Sending App= 1008
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=4586
Total number of re-transmitted DATA packets=3578 (78.02% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=4105
Total number of re-transmitted ACK packets=3105 (75.64% of total packets sent)
Utilization for the DATA channel=153.22%
Utilization for the ACK channel=1.37%
(gautamop㉿gautamop)-[~/Desktop/Lab4/final]
$

b) Let x be the percentage of packets sent by the rdt_sender that are just retransmissions of previously sent packets. Draw a plot to show how x varies with Pl (as Pl ranges from 0.1 to 0.9, for a simulation performed until the receiving application receives the first 1000 messages).




```
# x axis values - pl
x = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
# corresponding y axis values - time
y = [78.02, 83.49, 89.63, 91.40, 93.77, 95.74, 96.77, 98.00, 99.08]
```

c) Let U be the channel utilization on the sender-side. Draw a plot to show how U varies with Pl (as Pl ranges from 0.1 to 0.9, for a simulation performed until the receiving application receives the first 1000 messages).




```
# x axis values - pl
x = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
# corresponding y axis values - time
y = [153.22, 152.70, 155.53, 158.16, 162.72, 170.37, 175.46, 185.05, 194.52]
```

Again, the transmission delay that is employed in the above calculations but not taken into account in the total simulation duration is the reason why these channel utilization%ages are greater than 100.

Question 4.

As an engineer, suppose you are tasked with deploying such a protocol in a real world scenario where Pc/Pl values for the channel have been estimated beforehand. The following information is known: Channel transmission rate is 1000 bits per second, Pc=0.5, Pl=0.5, propagation delay =2, Packet_length for DATA packets = 1000 bits, packet length for ACK packets = 10 bits. 3 You need to tune your protocol parameters (Window size N, K and timeout_value) for achieveing the best performance.

a) How would you define/quantify "best performance" in this case?

Answer -

The time it takes to transmit a series of packets depends on variables like N and the timeout setting, even though K should just be a valid number—in this example, a number bigger than N. By reducing the timeout value as close to RTT as possible, time will be cut short, but channel utilisation will also increase. Thus we shouldn't treat the channel improperly. Moreover, as N and K are increased, time is reduced while channel consumption increases and a constant timeout value (something bigger than RTT) is maintained. As a result, we must pick a sound middle ground between the two.

In terms of time, the best performance will be considered. When all packets are delivered in less time, performance is enhanced. We must be cautious, though, and continue to use the channel effectively.

b) What would happen if the window size N is too small or too large? What limits its value?

Answer -

Because fewer packets can be sent at a time if the window size is too small, it will take longer overall to send the packets. On the other hand, keeping the window size too large won't help either because the channel utilisation will continue to rise with window size and may eventually reach 100%. To deal with this, we would either need to increase the timeout or stop sending packets.

c) List the values for N, K and timeout that you would choose for achieving the best performance, and explain your reasoning behind this choice.

Answer -

Channel transmission rate is 1000 bits per second
Pc = Pl = 0.5
Delay = 2s
Data Packet size = 1000 bits
Ack Packet size = 10 bits

→ A bigger timeout value must be selected for a window with a larger size, while a smaller window size requires a smaller timeout value.
→Also, the delay should be at least as long as the RTT to prevent pointless packet retransmissions. For instance, if the window size is 3, the timeout can be 5 to 6 seconds, while for a window with a size of 4, it can be 10 seconds.

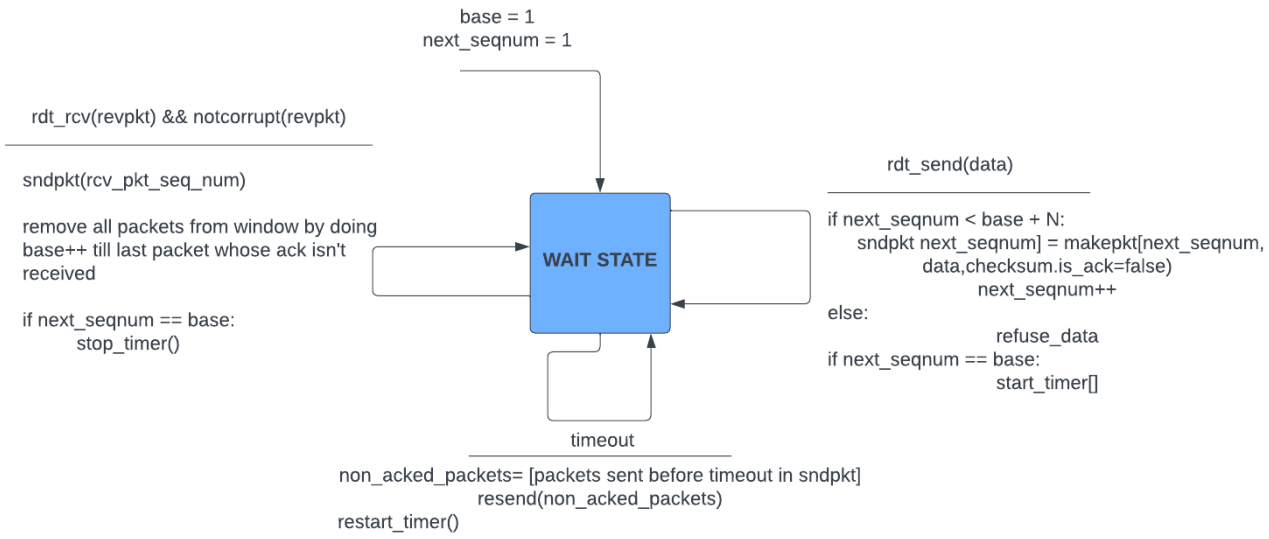
Selective Repeat Protocol

Question 5.

Describe the Sender-side and Receiver-side logic for the Selective Repeat protocol as a Finite State Machine (in a format similar to the FSM description of the Go-Back-N protocol).

Ans:

Sender FSM (made on Lucid app**)**

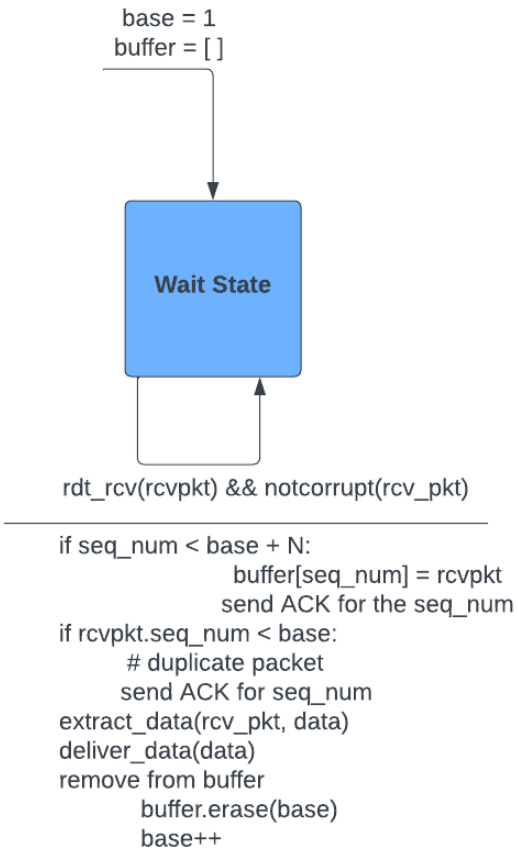


1. **Initially** :- base is initialized to 1 and next_seqnum is set to 1,
2. **rdt_send(data)** :- upon calling this function we first check if it is possible to send the packet by checking if the next_seqnum is less than base + Window Size,
If yes, we make a packet corresponding to its data and increment next_seqnum,
if no then we refuse sending the data, also if next_seqnum is equal to the base
here this means that the window is empty and we must start the timer.
3. **rdt_rcv(rcvpkt)** :- here if the sender gets an ACK this means that he must remove all those packets whose ack it has received until the last unACK'ed packet by doing base++ and if the window becomes empty this means that all the

packets were sent successfully so we stop the timer.

4. **Timeout** :- Upon a timeout the sender looks for all of the packets whose ACK it hasn't received in the sending window, it resends them again and restarts the timer.

Receiver FSM (made on Lucid app**)**



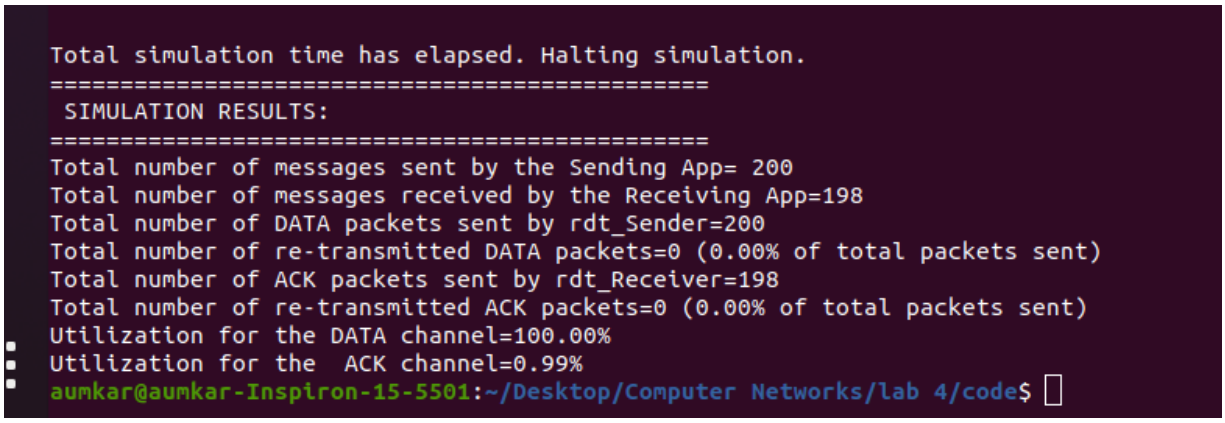
1.) **Initially** :- base is set to 1 and a buffer is being initialized

2.) **rdt_rcv(rcvpkt)** :- if a packet is received we first check if it's seq_num is in the current window of the receiver if so we add it to the buffer. If the packet we received has the seq_num less than base this means we received a duplicate packet so we resend an ACK for it, while doing all this we keep checking if the base has crossed the next expected seq_num and reached the buffered data packet we erase that data packet from the buffer.

Question 6.

Keeping the rest of the template as it is, modify only the rdt_Sender and rdt_Receiver classes to implement the Selective Repeat Protocol, and save this as file Protocol_SR.py. Debug and validate your implementation by running a long enough simulation (say, until the receiving application receives 1000 messages). Your simulation should run with no errors for each of the following cases:

a) Pl=0, Pc=0 for both DATA and ACK channels



```
TIME: 1002 RDT_RECEIVER: packets now delivering  8  and removed from buffer
TIME: 1002 RECEIVING APP: received data 1000


Receiving application received 1000 messages. Halting simulation.
================================================
 SIMULATION RESULTS:
================================================
Total number of messages sent by the Sending App= 1001
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=1001
Total number of re-transmitted DATA packets=0 (0.00% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=1000
Total number of re-transmitted ACK packets=0 (0.00% of total packets sent)
Utilization for the DATA channel=99.90%
Utilization for the  ACK channel=1.00%
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 4/code$
```

b) Pl=0, Pc=0.5 for both DATA and ACK channels

```
Total simulation time has elapsed. Halting simulation.
================================================
 SIMULATION RESULTS:
================================================
Total number of messages sent by the Sending App= 41
Total number of messages received by the Receiving App=36
Total number of DATA packets sent by rdt_Sender=121
Total number of re-transmitted DATA packets=80 (66.12% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=59
Total number of re-transmitted ACK packets=22 (37.29% of total packets sent)
Utilization for the DATA channel=60.50%
Utilization for the  ACK channel=0.30%
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 4/code$
```

```
TIME: 7741 RDT_RECEIVER: packets now delivering  8  and removed from buffer
TIME: 7741 RECEIVING APP: received data 1000


Receiving application received 1000 messages. Halting simulation.
================================================
 SIMULATION RESULTS:
================================================
Total number of messages sent by the Sending App= 1000
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=3781
Total number of re-transmitted DATA packets=2781 (73.55% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=1917
Total number of re-transmitted ACK packets=812 (42.36% of total packets sent)
Utilization for the DATA channel=48.84%
Utilization for the  ACK channel=0.25%
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 4/code$
```

c) Pl=0.5, Pc=0.5 for both DATA and ACK channels

```
Total simulation time has elapsed. Halting simulation.
================================================
 SIMULATION RESULTS:
================================================
Total number of messages sent by the Sending App= 11
Total number of messages received by the Receiving App=11
Total number of DATA packets sent by rdt_Sender=104
Total number of re-transmitted DATA packets=93 (89.42% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=31
Total number of re-transmitted ACK packets=18 (58.06% of total packets sent)
Utilization for the DATA channel=52.00%
Utilization for the  ACK channel=0.16%
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 4/code$
```

```
TIME: 36380 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, packet_
TIME: 36380 RDT_RECEIVER: packets now delivering  8  and removed from buffer
TIME: 36380 RECEIVING APP: received data 1000


Receiving application received 1000 messages. Halting simulation.
================================================
 SIMULATION RESULTS:
================================================
Total number of messages sent by the Sending App= 1000
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=16847
Total number of re-transmitted DATA packets=15847 (94.06% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=4193
Total number of re-transmitted ACK packets=2919 (69.62% of total packets sent)
Utilization for the DATA channel=46.31%
Utilization for the  ACK channel=0.12%
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 4/code$
```

d) Various values of the Window size N in the sender and the receiver. Submit the file Protocol_SR.py and paste a screenshot or partial simulation log for one of the simulations.

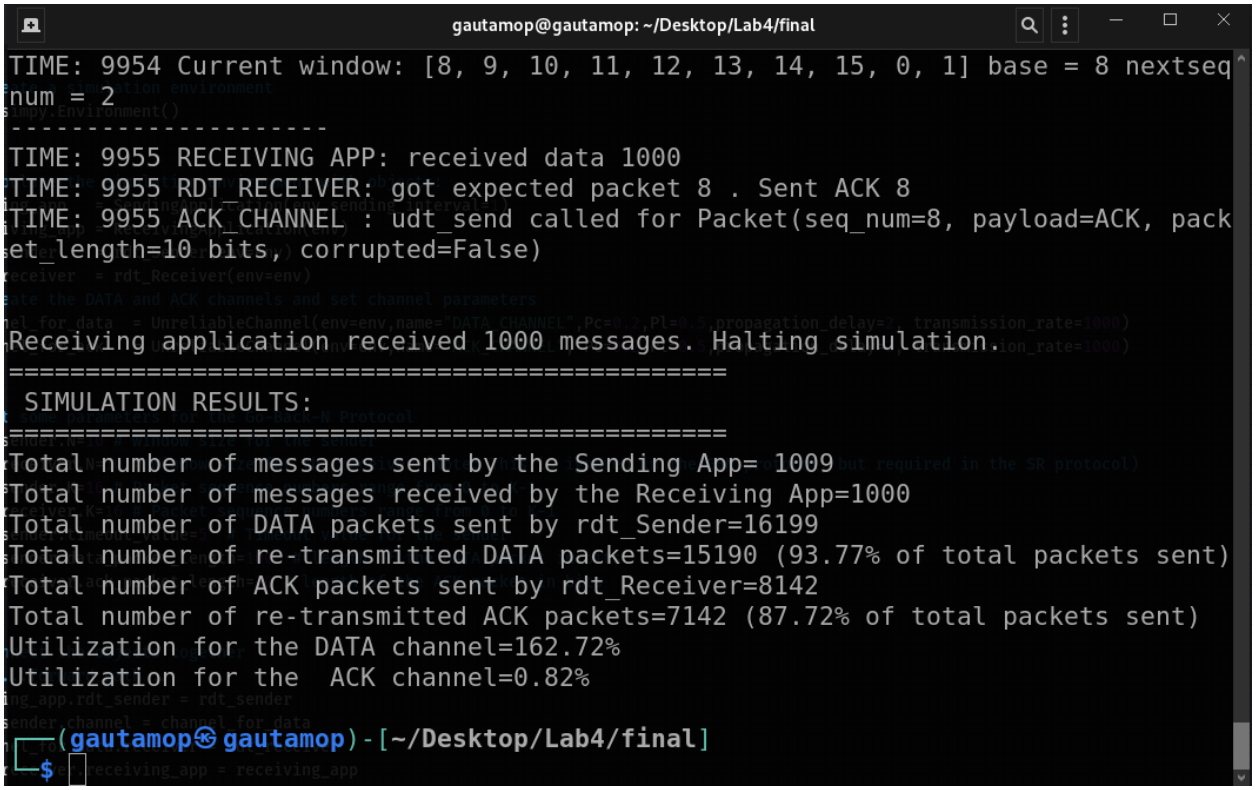
N=10 K=25

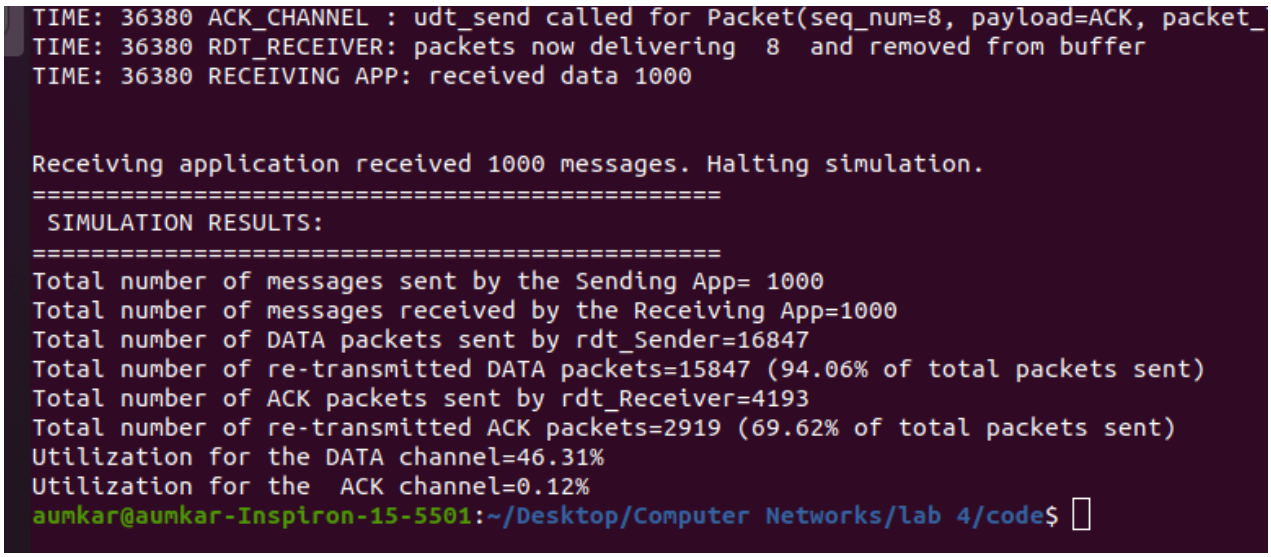
```
TIME: 1002 RECEIVING APP: received data 1000


Receiving application received 1000 messages. Halting simulation.
================================================
 SIMULATION RESULTS:
================================================
Total number of messages sent by the Sending App= 1001
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=1001
Total number of re-transmitted DATA packets=0 (0.00% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=1000
Total number of re-transmitted ACK packets=0 (0.00% of total packets sent)
Utilization for the DATA channel=99.90%
Utilization for the  ACK channel=1.00%
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 4/code$
```

Question 7.

For the same values of channel parameters (pc, pl, transmission rates etc), does the SR protocol show a better performance as compared to the GBN protocol? Describe your observations.

Ans: Yes, Selective Repeat protocol performs better than Go Back-N protocol for the same set of parameters of Pc and Pl since SR protocol has fewer retransmissions than GBN protocol as only a particular frame is retransmitted instead of the entire window.


gautamop@gautamop: ~/Desktop/Lab4/final
TIME: 9954 Current window: [8, 9, 10, 11, 12, 13, 14, 15, 0, 1] base = 8 nextseq
num = 2
TIME: 9955 RECEIVING APP: received data 1000
TIME: 9955 RDT_RECEIVER: got expected packet 8 . Sent ACK 8
TIME: 9955 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, pack
et_length=10 bits, corrupted=False)
Receiving application received 1000 messages. Halting simulation.
SIMULATION RESULTS:
Total number of messages sent by the Sending App= 1009
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=16199
Total number of re-transmitted DATA packets=15190 (93.77% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=8142
Total number of re-transmitted ACK packets=7142 (87.72% of total packets sent)
Utilization for the DATA channel=162.72%
Utilization for the ACK channel=0.82%
(gautamop㉿gautamop)-[~/Desktop/Lab4/final]
$


TIME: 36380 ACK_CHANNEL : udt_send called for Packet(seq_num=8, payload=ACK, packet_
TIME: 36380 RDT_RECEIVER: packets now delivering 8 and removed from buffer
TIME: 36380 RECEIVING APP: received data 1000
Receiving application received 1000 messages. Halting simulation.
SIMULATION RESULTS:
Total number of messages sent by the Sending App= 1000
Total number of messages received by the Receiving App=1000
Total number of DATA packets sent by rdt_Sender=16847
Total number of re-transmitted DATA packets=15847 (94.06% of total packets sent)
Total number of ACK packets sent by rdt_Receiver=4193
Total number of re-transmitted ACK packets=2919 (69.62% of total packets sent)
Utilization for the DATA channel=46.31%
Utilization for the ACK channel=0.12%
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 4/code$

Question 8.

For the SR protocol to work correctly, the value of K (representing the range of sequence numbers) must be at-least twice as large as N (the Window size). That is, N<=K/2. Try setting N=K and check if you get any errors during simulation. Explain why setting N<=K/2 would avoid these errors.

Ans.

When K is less than N, many packets with the same number are delivered, making it difficult for the recipient to distinguish which one it is receiving and increasing the likelihood that wrong data will be transmitted.

If the N<=K/2 then the packets might get recognised incorrectly.

If an ACK is lost, the receiver may confuse the new packets sent with the retransmitted packet. The protocol will run smoothly otherwise for K > 2N.

Thank You !