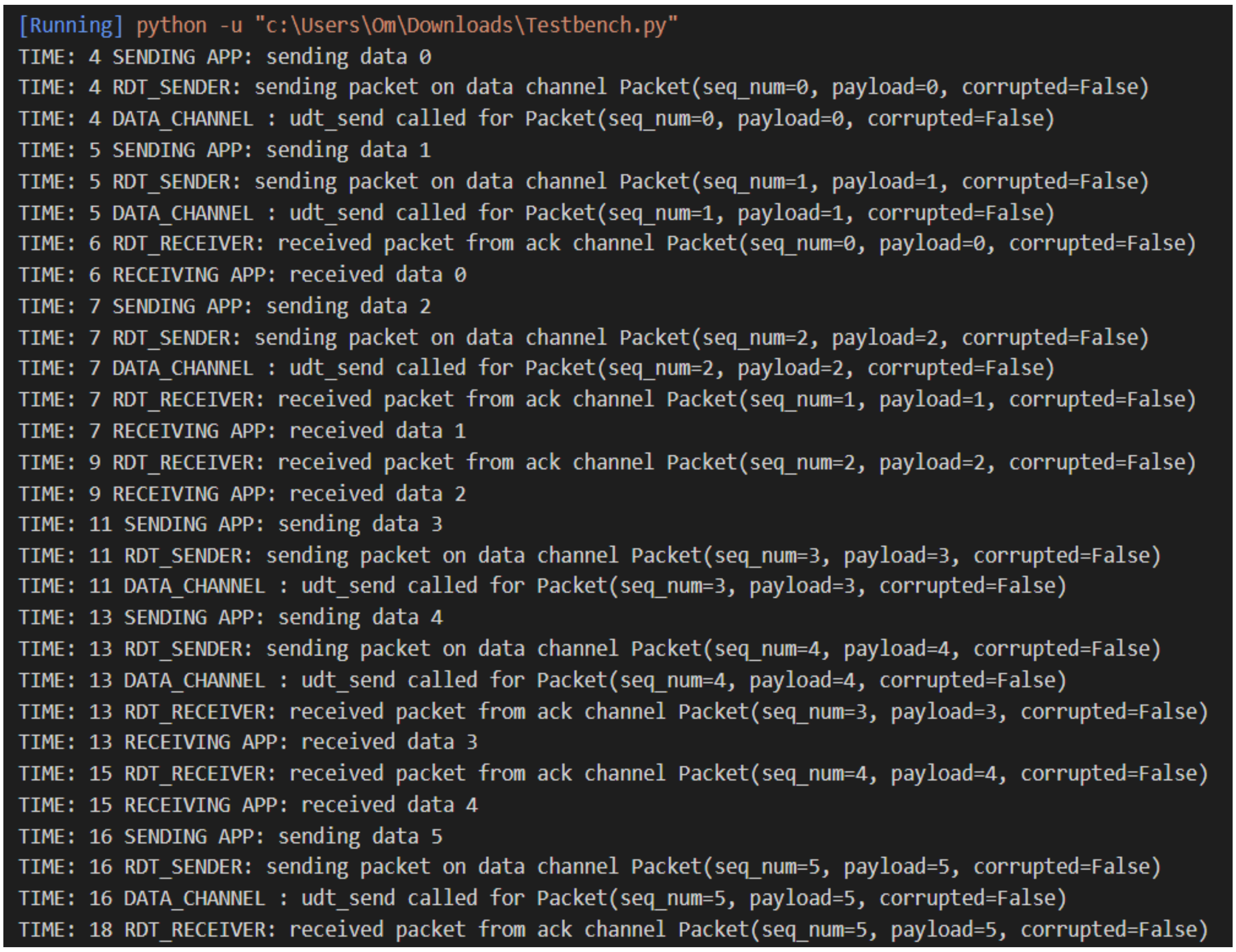
LAB 3: Principles of Reliable Data Transfer

- *Ujjwal (1904139)*
  Om Patil (1904125)

Q1) We have implemented Protocol_rdt1 which works only when the channel is assumed to be ideal i.e. when Pc = 0 and Pl = 0.

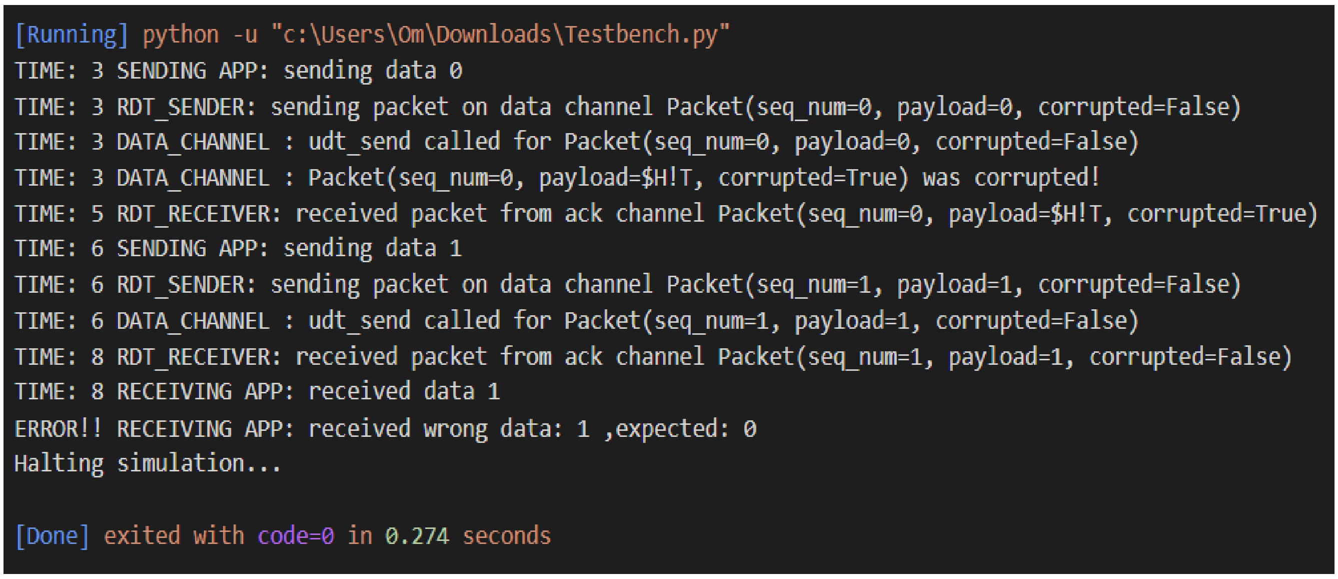
Lets first run the protocol with **Pc = 0** and **Pl = 0.** The screenshot of the output is shown below:



```
[Running] python -u "c:\Users\Om\Downloads\Testbench.py"
TIME: 4 SENDING APP: sending data 0
TIME: 4 RDT_SENDER: sending packet on data channel Packet(seq_num=0, payload=0, corrupted=False)
TIME: 4 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=0, corrupted=False)
TIME: 5 SENDING APP: sending data 1
TIME: 5 RDT_SENDER: sending packet on data channel Packet(seq_num=1, payload=1, corrupted=False)
TIME: 5 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 6 RDT_RECEIVER: received packet from ack channel Packet(seq_num=0, payload=0, corrupted=False)
TIME: 6 RECEIVING APP: received data 0
TIME: 7 SENDING APP: sending data 2
TIME: 7 RDT_SENDER: sending packet on data channel Packet(seq_num=2, payload=2, corrupted=False)
TIME: 7 DATA_CHANNEL : udt_send called for Packet(seq_num=2, payload=2, corrupted=False)
TIME: 7 RDT_RECEIVER: received packet from ack channel Packet(seq_num=1, payload=1, corrupted=False)
TIME: 7 RECEIVING APP: received data 1
TIME: 9 RDT_RECEIVER: received packet from ack channel Packet(seq_num=2, payload=2, corrupted=False)
TIME: 9 RECEIVING APP: received data 2
TIME: 11 SENDING APP: sending data 3
TIME: 11 RDT_SENDER: sending packet on data channel Packet(seq_num=3, payload=3, corrupted=False)
TIME: 11 DATA_CHANNEL : udt_send called for Packet(seq_num=3, payload=3, corrupted=False)
TIME: 13 SENDING APP: sending data 4
TIME: 13 RDT_SENDER: sending packet on data channel Packet(seq_num=4, payload=4, corrupted=False)
TIME: 13 DATA_CHANNEL : udt_send called for Packet(seq_num=4, payload=4, corrupted=False)
TIME: 13 RDT_RECEIVER: received packet from ack channel Packet(seq_num=3, payload=3, corrupted=False)
TIME: 13 RECEIVING APP: received data 3
TIME: 15 RDT_RECEIVER: received packet from ack channel Packet(seq_num=4, payload=4, corrupted=False)
TIME: 15 RECEIVING APP: received data 4
TIME: 16 SENDING APP: sending data 5
TIME: 16 RDT_SENDER: sending packet on data channel Packet(seq_num=5, payload=5, corrupted=False)
TIME: 16 DATA_CHANNEL : udt_send called for Packet(seq_num=5, payload=5, corrupted=False)
TIME: 18 RDT_RECEIVER: received packet from ack channel Packet(seq_num=5, payload=5, corrupted=False)
```

Here, we can clearly verify that the Protocol runs perfectly under the above mentioned conditions.

In the next case let us consider **Pc = 0.5** (instead of 0). Following is the screenshot of the output:



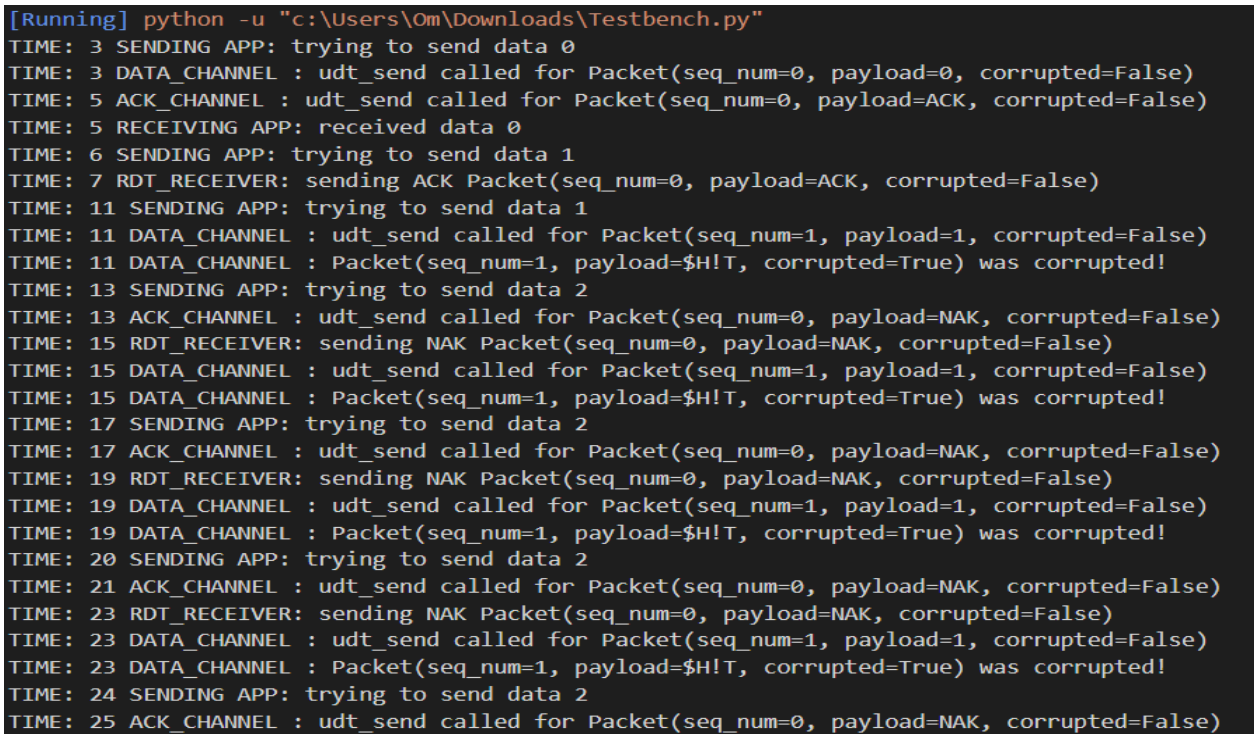
```
[Running] python -u "c:\Users\Om\Downloads\Testbench.py"
TIME: 3 SENDING APP: sending data 0
TIME: 3 RDT_SENDER: sending packet on data channel Packet(seq_num=0, payload=0, corrupted=False)
TIME: 3 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=0, corrupted=False)
TIME: 3 DATA_CHANNEL : Packet(seq_num=0, payload=$H!T, corrupted=True) was corrupted!
TIME: 5 RDT_RECEIVER: received packet from ack channel Packet(seq_num=0, payload=$H!T, corrupted=True)
TIME: 6 SENDING APP: sending data 1
TIME: 6 RDT_SENDER: sending packet on data channel Packet(seq_num=1, payload=1, corrupted=False)
TIME: 6 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 8 RDT_RECEIVER: received packet from ack channel Packet(seq_num=1, payload=1, corrupted=False)
TIME: 8 RECEIVING APP: received data 1
ERROR!! RECEIVING APP: received wrong data: 1 ,expected: 0
Halting simulation...

[Done] exited with code=0 in 0.274 seconds
```

In the output we can very well see that the Protocol fails as in the end it outputs that there is some error due to receiving the wrong data packet and thus it is halting the simulation.

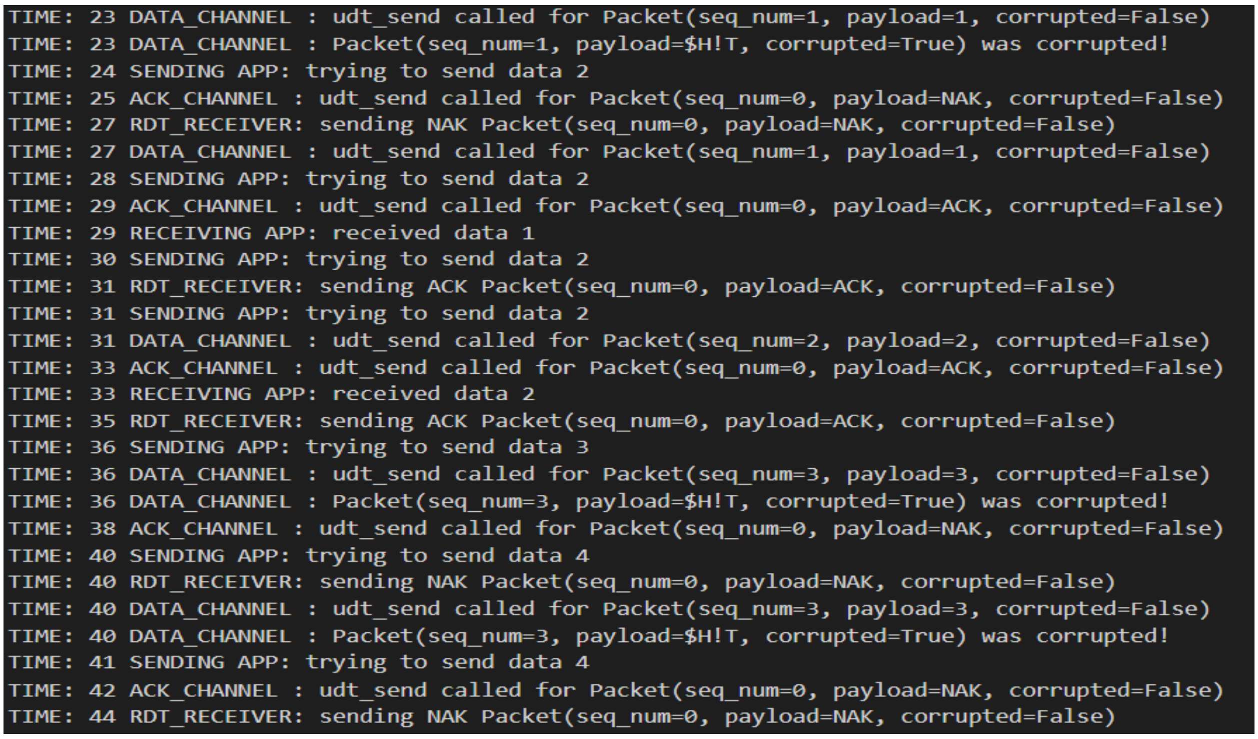
This error occured because the first packet which was sent (packet ‘0’) got corrupted (corrupted = true, payload = $h!T) while flowing through the unreliable channel. Thus, the receiver received the corrupted packet but was waiting for the uncorrupted packet ‘0’. In the next iteration when the sender sends packet ‘1’ it does not get corrupted, now the receiver receives packet ‘1’ but it was expecting packet ‘0’ so it halts the simulation as it realizes something went wrong and it can’t rectify it.

Q2) Implementing Protocol_rdt2 which is simple ACK/NAK based protocol which can work even if the data packets get corrupted.

We can verify its functionality by setting Pc > 0 (for eg. Pc = 0.5) for data packets. Following is the screenshot of the output :



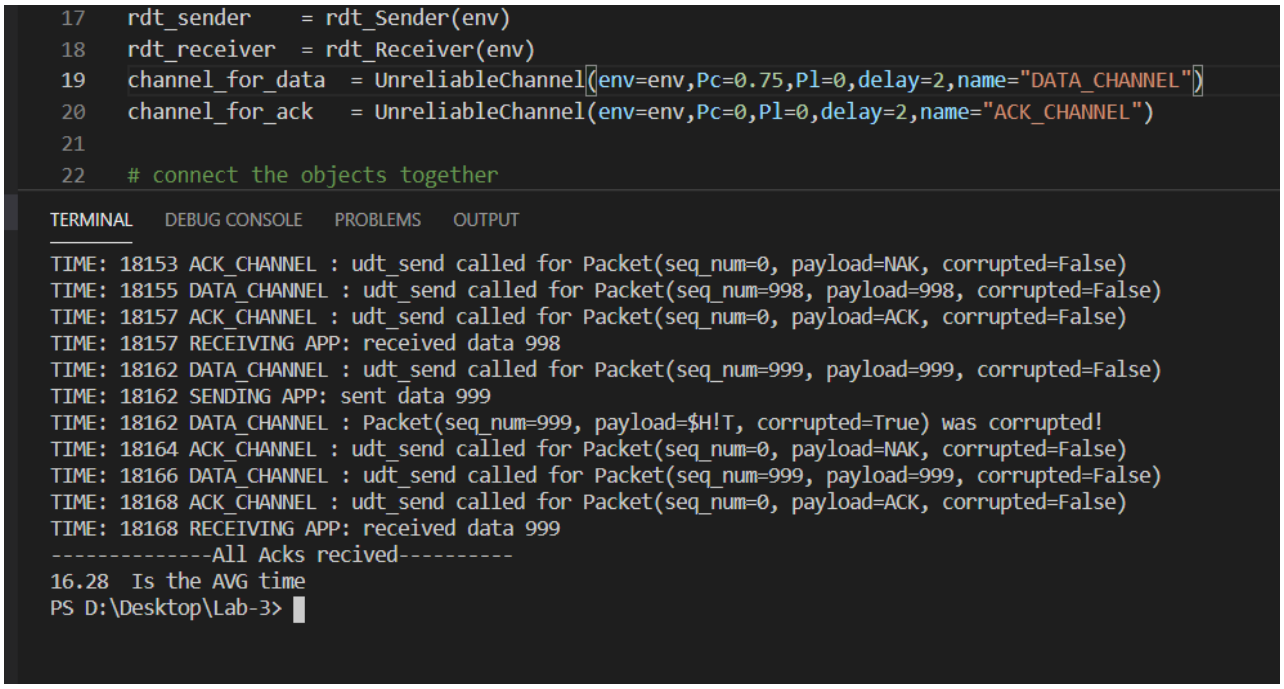
```
[Running] python -u "c:\Users\Om\Downloads\Testbench.py"
TIME: 3 SENDING APP: trying to send data 0
TIME: 3 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=0, corrupted=False)
TIME: 5 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 5 RECEIVING APP: received data 0
TIME: 6 SENDING APP: trying to send data 1
TIME: 7 RDT_RECEIVER: sending ACK Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 11 SENDING APP: trying to send data 1
TIME: 11 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 11 DATA_CHANNEL : Packet(seq_num=1, payload=$H!T, corrupted=True) was corrupted!
TIME: 13 SENDING APP: trying to send data 2
TIME: 13 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 15 RDT_RECEIVER: sending NAK Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 15 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 15 DATA_CHANNEL : Packet(seq_num=1, payload=$H!T, corrupted=True) was corrupted!
TIME: 17 SENDING APP: trying to send data 2
TIME: 17 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 19 RDT_RECEIVER: sending NAK Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 19 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 19 DATA_CHANNEL : Packet(seq_num=1, payload=$H!T, corrupted=True) was corrupted!
TIME: 20 SENDING APP: trying to send data 2
TIME: 21 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 23 RDT_RECEIVER: sending NAK Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 23 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 23 DATA_CHANNEL : Packet(seq_num=1, payload=$H!T, corrupted=True) was corrupted!
TIME: 24 SENDING APP: trying to send data 2
TIME: 25 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=NAK, corrupted=False)
```



```
TIME: 23 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 23 DATA_CHANNEL : Packet(seq_num=1, payload=$H!T, corrupted=True) was corrupted!
TIME: 24 SENDING APP: trying to send data 2
TIME: 25 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 27 RDT_RECEIVER: sending NAK Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 27 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 28 SENDING APP: trying to send data 2
TIME: 29 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 29 RECEIVING APP: received data 1
TIME: 30 SENDING APP: trying to send data 2
TIME: 31 RDT_RECEIVER: sending ACK Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 31 SENDING APP: trying to send data 2
TIME: 31 DATA_CHANNEL : udt_send called for Packet(seq_num=2, payload=2, corrupted=False)
TIME: 33 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 33 RECEIVING APP: received data 2
TIME: 35 RDT_RECEIVER: sending ACK Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 36 SENDING APP: trying to send data 3
TIME: 36 DATA_CHANNEL : udt_send called for Packet(seq_num=3, payload=3, corrupted=False)
TIME: 36 DATA_CHANNEL : Packet(seq_num=3, payload=$H!T, corrupted=True) was corrupted!
TIME: 38 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 40 SENDING APP: trying to send data 4
TIME: 40 RDT_RECEIVER: sending NAK Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 40 DATA_CHANNEL : udt_send called for Packet(seq_num=3, payload=3, corrupted=False)
TIME: 40 DATA_CHANNEL : Packet(seq_num=3, payload=$H!T, corrupted=True) was corrupted!
TIME: 41 SENDING APP: trying to send data 4
TIME: 42 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 44 RDT_RECEIVER: sending NAK Packet(seq_num=0, payload=NAK, corrupted=False)
```

From the above screenshot we can verify that the Protocol_rdt2 works correctly even when there is some probability of data packets getting corrupted.

Q3) Here is a sample screenshot for **Pc = 0.75 .**

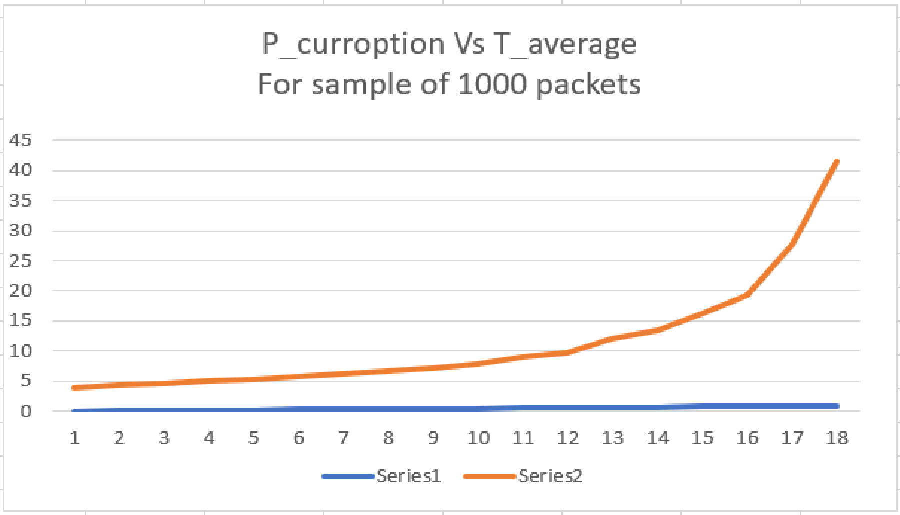


```
rdt_sender     = rdt_Sender(env)
rdt_receiver   = rdt_Receiver(env)
channel_for_data   = UnreliableChannel(env=env,Pc=0.75,Pl=0,delay=2,name="DATA_CHANNEL")
channel_for_ack    = UnreliableChannel(env=env,Pc=0,Pl=0,delay=2,name="ACK_CHANNEL")

# connect the objects together
```

```
TIME: 18153 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 18155 DATA_CHANNEL : udt_send called for Packet(seq_num=998, payload=998, corrupted=False)
TIME: 18157 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 18157 RECEIVING APP: received data 998
TIME: 18162 DATA_CHANNEL : udt_send called for Packet(seq_num=999, payload=999, corrupted=False)
TIME: 18162 SENDING APP: sent data 999
TIME: 18162 DATA_CHANNEL : Packet(seq_num=999, payload=$H!T, corrupted=True) was corrupted!
TIME: 18164 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=NAK, corrupted=False)
TIME: 18166 DATA_CHANNEL : udt_send called for Packet(seq_num=999, payload=999, corrupted=False)
TIME: 18168 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 18168 RECEIVING APP: received data 999
--------------All Acks recived----------
16.28  Is the AVG time
PS D:\Desktop\Lab-3>
```

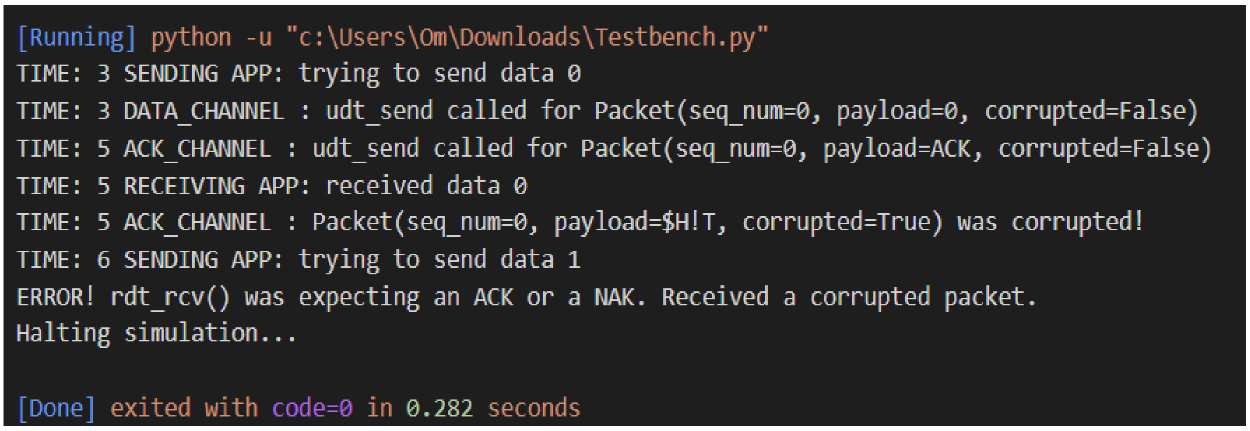
Similarly, doing for all we get the below graph which is exponential in nature.

pc(data)	T_avg
0	4
0.1	4.46
0.15	4.696
0.2	5.04
0.25	5.228
0.3	5.832
0.35	6.276
0.4	6.696
0.45	7.212
0.5	7.96
0.55	9.044
0.6	9.692
0.65	12.128
0.7	13.368
0.75	16.28
0.8	19.316
0.85	27.732
0.9	41.424



The graph is exponential because as the Probability of Corruption increases more packets will get corrupted which implies we need to send even more number of packets for that one corrupted packet that means these packets may also get corrupted so more and more packets need to be sent which in turn increases the time.

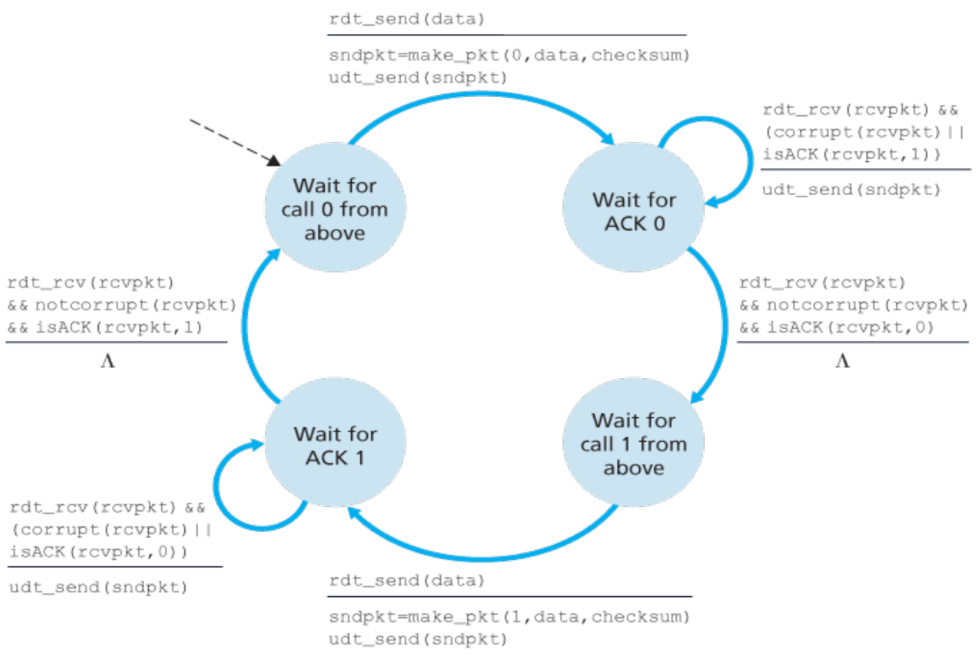
Q4) The Protocol_rdt2 which we implemented in the 2nd question will not work when the acknowledgement packets start to get corrupted.
Following is the screenshot of the output when the Pc for the ACK/NAK channel is greater than zero (Let, Pc = 0.5) :



```
[Running] python -u "c:\Users\Om\Downloads\Testbench.py"
TIME: 3 SENDING APP: trying to send data 0
TIME: 3 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=0, corrupted=False)
TIME: 5 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 5 RECEIVING APP: received data 0
TIME: 5 ACK_CHANNEL : Packet(seq_num=0, payload=$H!T, corrupted=True) was corrupted!
TIME: 6 SENDING APP: trying to send data 1
ERROR! rdt_rcv() was expecting an ACK or a NAK. Received a corrupted packet.
Halting simulation...

[Done] exited with code=0 in 0.282 seconds
```

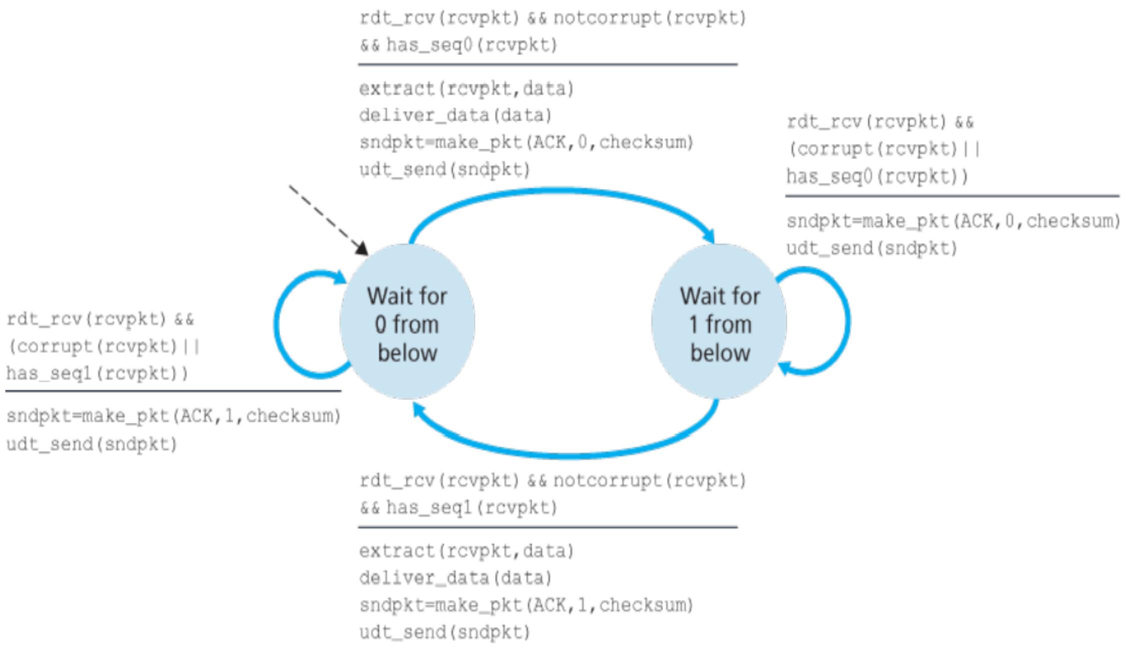
Here, we can observe that the output gives us an error specifying that the ACK/NAK which the sender received from the receiver was corrupted and therefore can’t continue further with the simulation.

Q5) Following is the FSM model , we need to develop in our code :

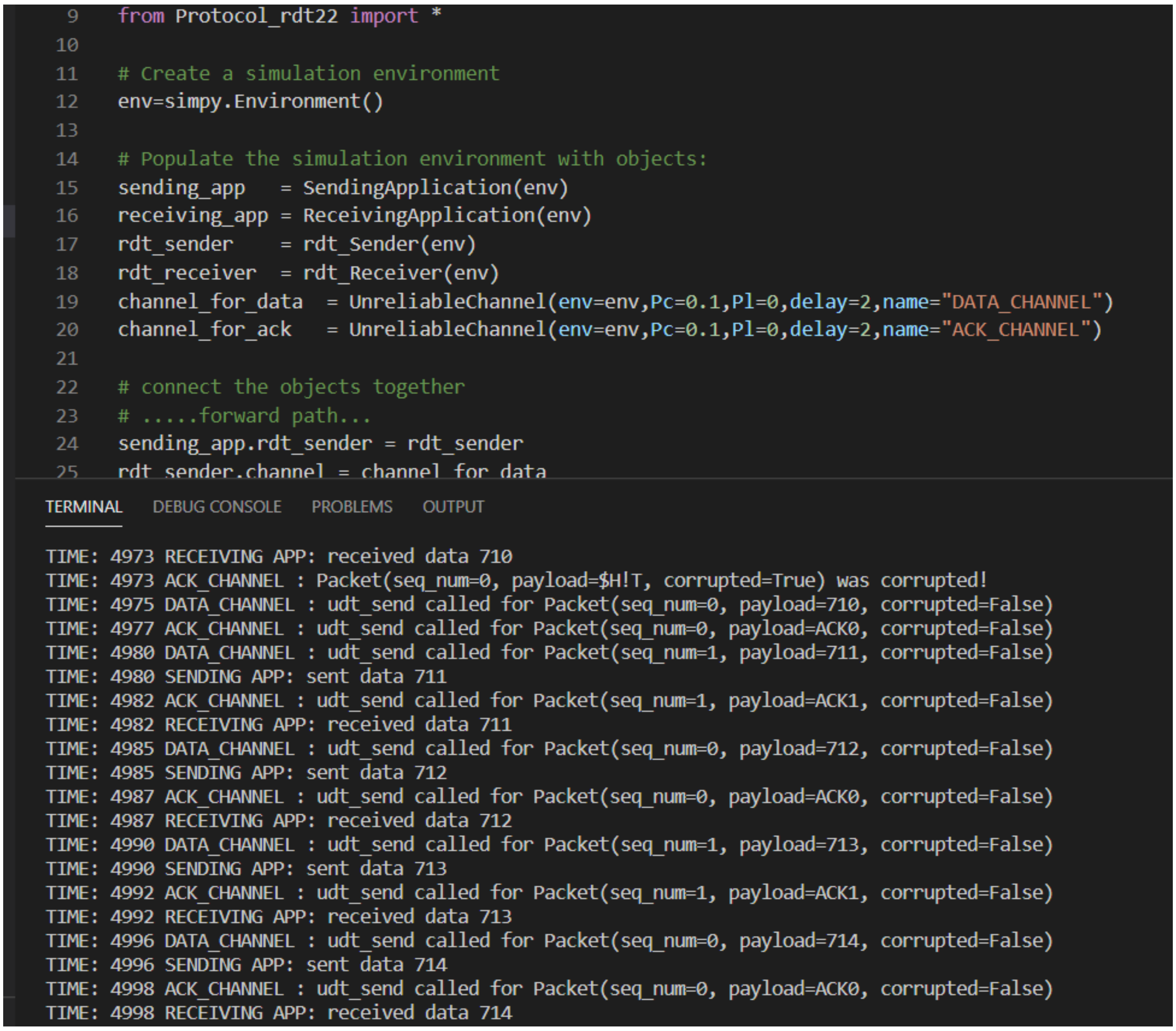

rdt_send(data)
sndpkt=make_pkt(0,data,checksum)
udt_send(sndpkt)
Wait for call 0 from above
Wait for ACK 0
rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || isACK(rcvpkt,1))
udt_send(sndpkt)
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt,0)
Λ
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt,1)
Λ
Wait for ACK 1
Wait for call 1 from above
rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || isACK(rcvpkt,0))
udt_send(sndpkt)
rdt_send(data)
sndpkt=make_pkt(1,data,checksum)
udt_send(sndpkt)

Now there is no need for NAK, instead, we can give ACK of that packet which was not expected to serve as NAK.

Similarly, we develop at receiver :


rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq0(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(ACK,0,checksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || has_seq0(rcvpkt))
sndpkt=make_pkt(ACK,0,checksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || has_seq1(rcvpkt))
sndpkt=make_pkt(ACK,1,checksum)
udt_send(sndpkt)
Wait for 0 from below
Wait for 1 from below
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq1(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(ACK,1,checksum)
udt_send(sndpkt)

The protocol Works well for both channels having probability of corruption 0.1 as shown below



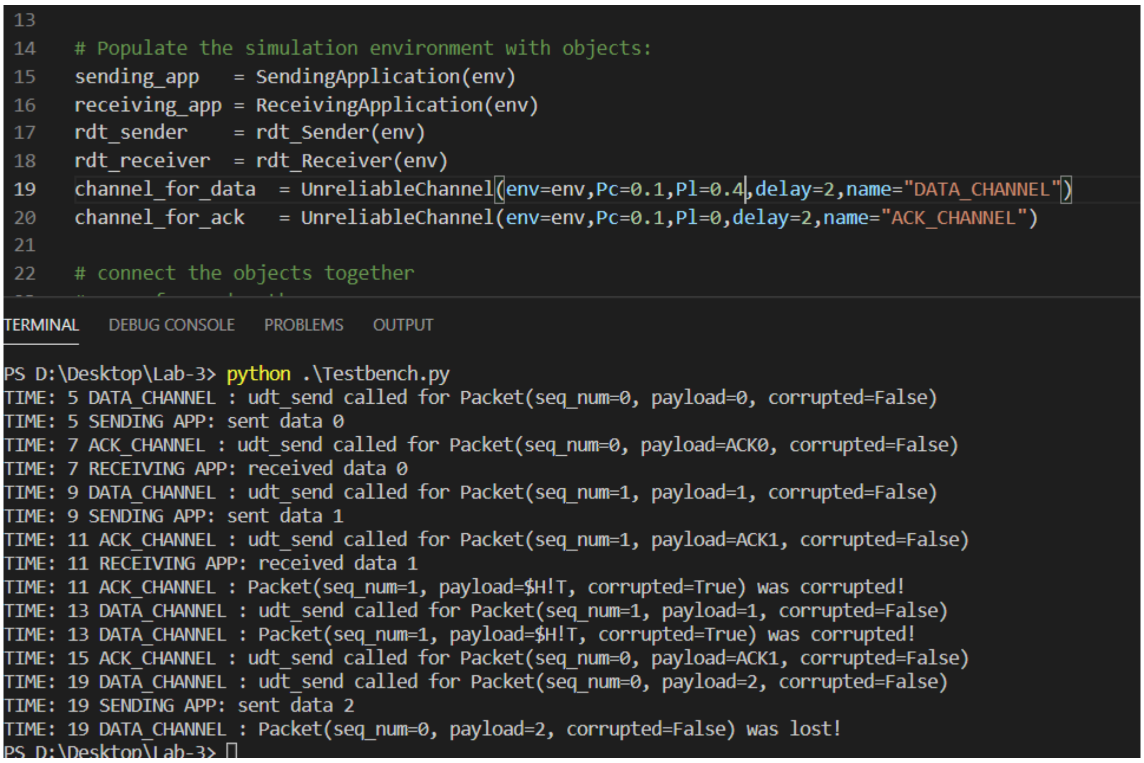
```
from Protocol_rdt22 import *

# Create a simulation environment
env=simpy.Environment()

# Populate the simulation environment with objects:
sending_app    = SendingApplication(env)
receiving_app = ReceivingApplication(env)
rdt_sender     = rdt_Sender(env)
rdt_receiver   = rdt_Receiver(env)
channel_for_data   = UnreliableChannel(env=env,Pc=0.1,Pl=0,delay=2,name="DATA_CHANNEL")
channel_for_ack    = UnreliableChannel(env=env,Pc=0.1,Pl=0,delay=2,name="ACK_CHANNEL")

# connect the objects together
# .....forward path...
sending_app.rdt_sender = rdt_sender
rdt_sender.channel = channel_for_data
```

TERMINAL   DEBUG CONSOLE   PROBLEMS   OUTPUT

```
TIME: 4973 RECEIVING APP: received data 710
TIME: 4973 ACK_CHANNEL : Packet(seq_num=0, payload=$H!T, corrupted=True) was corrupted!
TIME: 4975 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=710, corrupted=False)
TIME: 4977 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 4980 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=711, corrupted=False)
TIME: 4980 SENDING APP: sent data 711
TIME: 4982 ACK_CHANNEL : udt_send called for Packet(seq_num=1, payload=ACK1, corrupted=False)
TIME: 4982 RECEIVING APP: received data 711
TIME: 4985 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=712, corrupted=False)
TIME: 4985 SENDING APP: sent data 712
TIME: 4987 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 4987 RECEIVING APP: received data 712
TIME: 4990 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=713, corrupted=False)
TIME: 4990 SENDING APP: sent data 713
TIME: 4992 ACK_CHANNEL : udt_send called for Packet(seq_num=1, payload=ACK1, corrupted=False)
TIME: 4992 RECEIVING APP: received data 713
TIME: 4996 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=714, corrupted=False)
TIME: 4996 SENDING APP: sent data 714
TIME: 4998 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 4998 RECEIVING APP: received data 714
```

Q6)

A) If the packet is lost, the sender is struck in a listing state as it will not hear back any acknowledgment and thus, would be always truck in that state until simulation times out as we can observe below

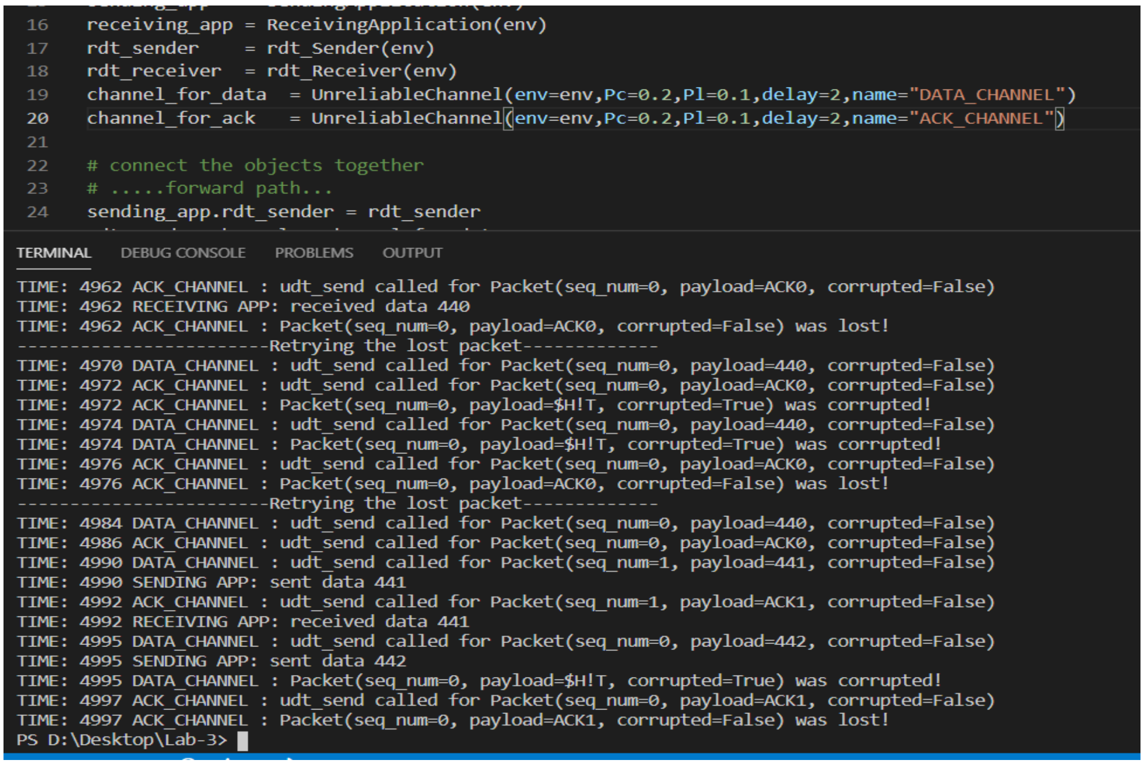


```
# Populate the simulation environment with objects:
sending_app   = SendingApplication(env)
receiving_app = ReceivingApplication(env)
rdt_sender    = rdt_Sender(env)
rdt_receiver  = rdt_Receiver(env)
channel_for_data  = UnreliableChannel(env=env,Pc=0.1,Pl=0.4,delay=2,name="DATA_CHANNEL")
channel_for_ack   = UnreliableChannel(env=env,Pc=0.1,Pl=0,delay=2,name="ACK_CHANNEL")

# connect the objects together
```

TERMINAL   DEBUG CONSOLE   PROBLEMS   OUTPUT

```
PS D:\Desktop\Lab-3> python .\Testbench.py
TIME: 5 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=0, corrupted=False)
TIME: 5 SENDING APP: sent data 0
TIME: 7 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 7 RECEIVING APP: received data 0
TIME: 9 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 9 SENDING APP: sent data 1
TIME: 11 ACK_CHANNEL : udt_send called for Packet(seq_num=1, payload=ACK1, corrupted=False)
TIME: 11 RECEIVING APP: received data 1
TIME: 11 ACK_CHANNEL : Packet(seq_num=1, payload=$H!T, corrupted=True) was corrupted!
TIME: 13 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 13 DATA_CHANNEL : Packet(seq_num=1, payload=$H!T, corrupted=True) was corrupted!
TIME: 15 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK1, corrupted=False)
TIME: 19 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=2, corrupted=False)
TIME: 19 SENDING APP: sent data 2
TIME: 19 DATA_CHANNEL : Packet(seq_num=0, payload=2, corrupted=False) was lost!
PS D:\Desktop\Lab-3> 
```

B)



```
receiving_app = ReceivingApplication(env)
rdt_sender    = rdt_Sender(env)
rdt_receiver  = rdt_Receiver(env)
channel_for_data  = UnreliableChannel(env=env,Pc=0.2,Pl=0.1,delay=2,name="DATA_CHANNEL")
channel_for_ack   = UnreliableChannel(env=env,Pc=0.2,Pl=0.1,delay=2,name="ACK_CHANNEL")

# connect the objects together
# .....forward path...
sending_app.rdt_sender = rdt_sender
```

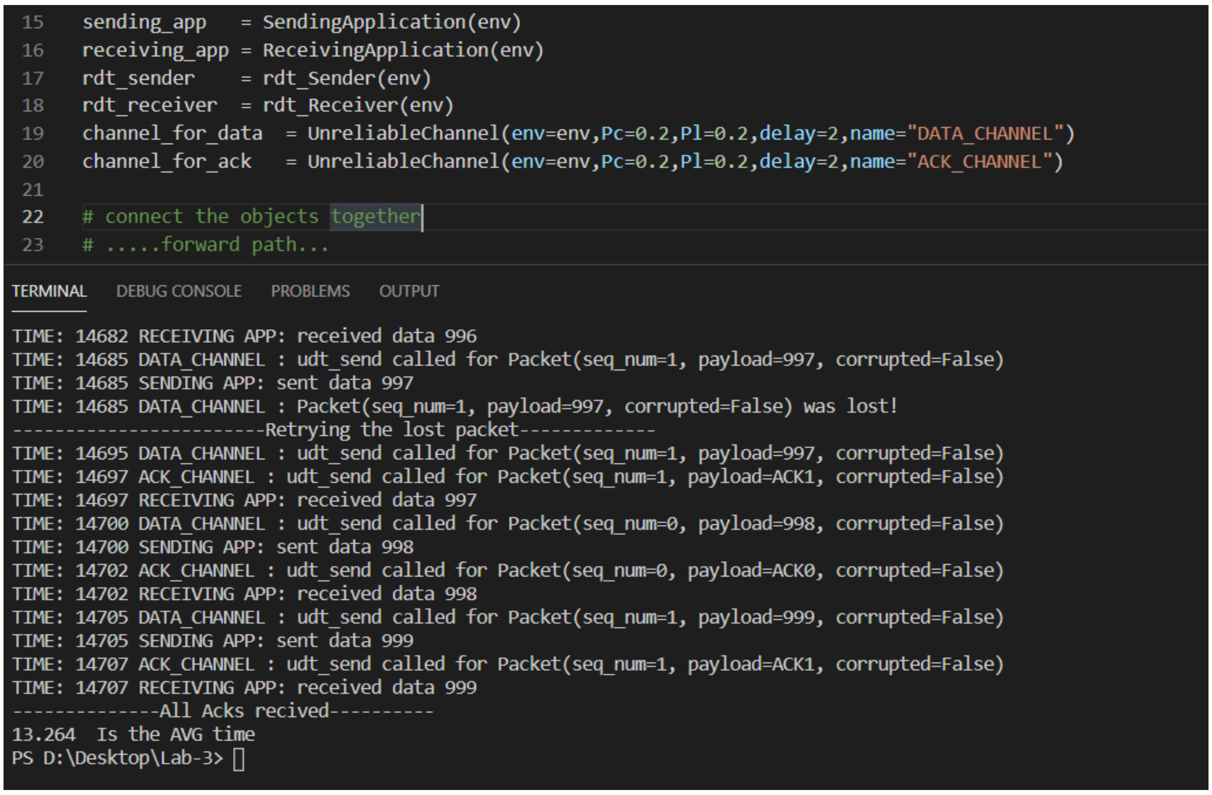
TERMINAL   DEBUG CONSOLE   PROBLEMS   OUTPUT

```
TIME: 4962 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 4962 RECEIVING APP: received data 440
TIME: 4962 ACK_CHANNEL : Packet(seq_num=0, payload=ACK0, corrupted=False) was lost!
------------------------Retrying the lost packet-------------
TIME: 4970 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=440, corrupted=False)
TIME: 4972 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 4972 ACK_CHANNEL : Packet(seq_num=0, payload=$H!T, corrupted=True) was corrupted!
TIME: 4974 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=440, corrupted=False)
TIME: 4974 DATA_CHANNEL : Packet(seq_num=0, payload=$H!T, corrupted=True) was corrupted!
TIME: 4976 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 4976 ACK_CHANNEL : Packet(seq_num=0, payload=ACK0, corrupted=False) was lost!
------------------------Retrying the lost packet-------------
TIME: 4984 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=440, corrupted=False)
TIME: 4986 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 4990 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=441, corrupted=False)
TIME: 4990 SENDING APP: sent data 441
TIME: 4992 ACK_CHANNEL : udt_send called for Packet(seq_num=1, payload=ACK1, corrupted=False)
TIME: 4992 RECEIVING APP: received data 441
TIME: 4995 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=442, corrupted=False)
TIME: 4995 SENDING APP: sent data 442
TIME: 4995 DATA_CHANNEL : Packet(seq_num=0, payload=$H!T, corrupted=True) was corrupted!
TIME: 4997 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK1, corrupted=False)
TIME: 4997 ACK_CHANNEL : Packet(seq_num=0, payload=ACK1, corrupted=False) was lost!
PS D:\Desktop\Lab-3> 
```

The developed protocol works well with the both loss and corruption probability in both Ack and Data channels

C)

Following is one example of the average time .

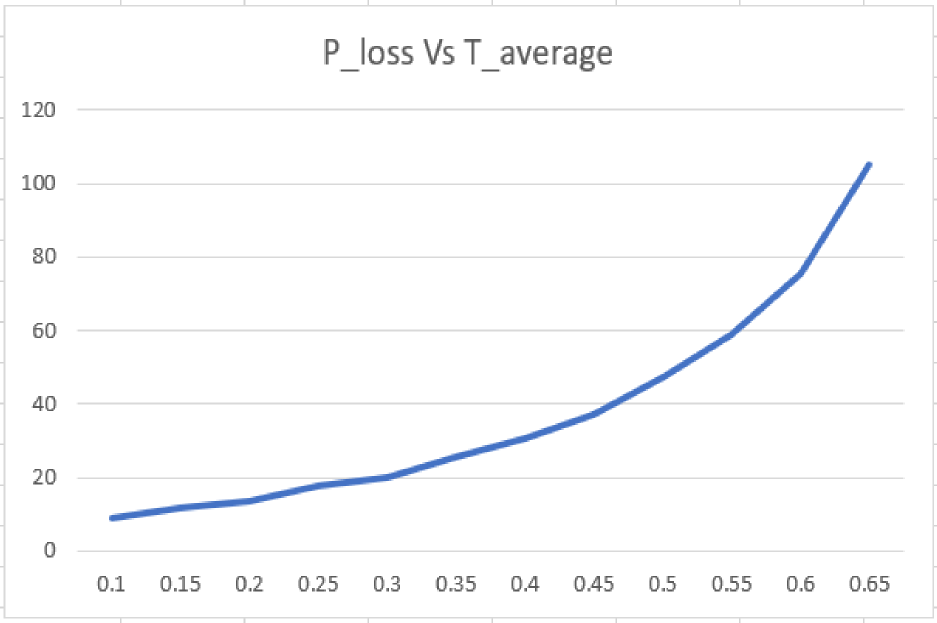


```
sending_app    = SendingApplication(env)
receiving_app = ReceivingApplication(env)
rdt_sender     = rdt_Sender(env)
rdt_receiver   = rdt_Receiver(env)
channel_for_data  = UnreliableChannel(env=env,Pc=0.2,Pl=0.2,delay=2,name="DATA_CHANNEL")
channel_for_ack   = UnreliableChannel(env=env,Pc=0.2,Pl=0.2,delay=2,name="ACK_CHANNEL")

# connect the objects together
# .....forward path...
```

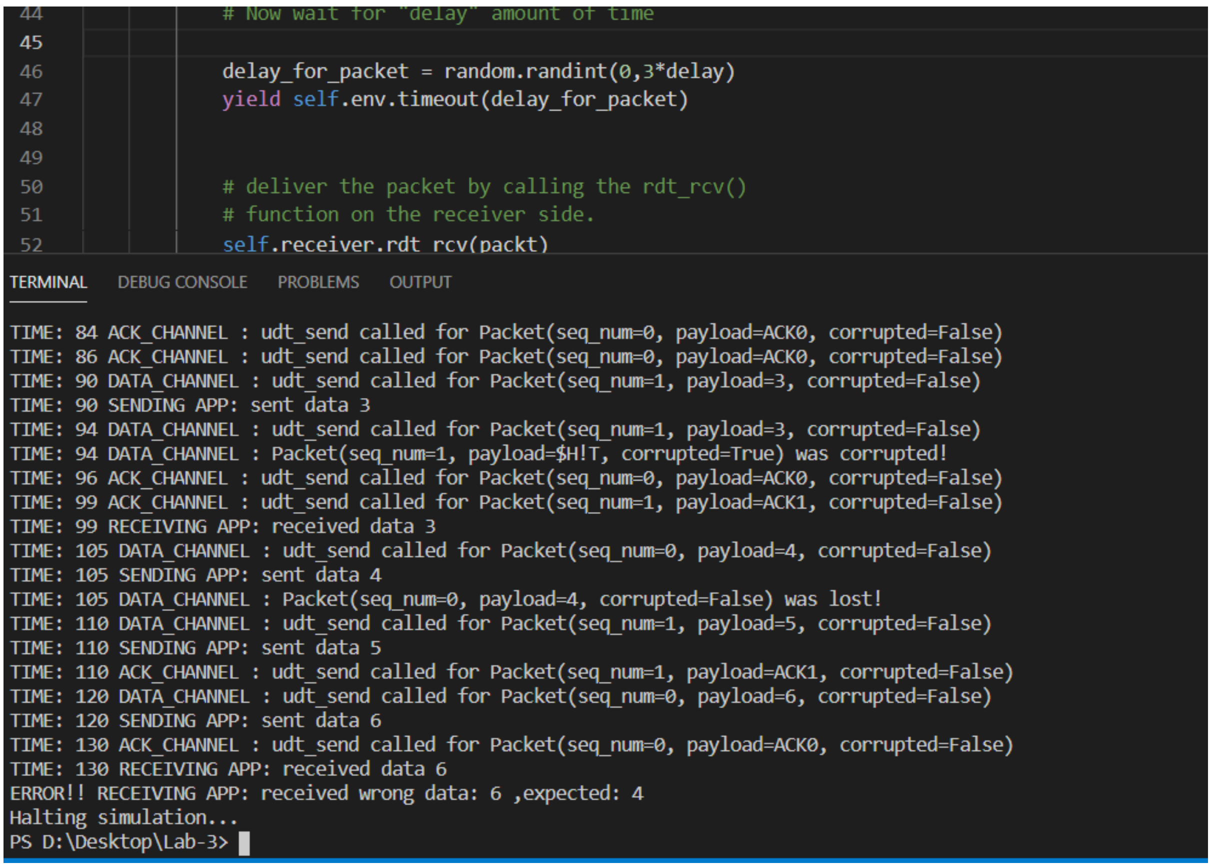
```
TIME: 14682 RECEIVING APP: received data 996
TIME: 14685 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=997, corrupted=False)
TIME: 14685 SENDING APP: sent data 997
TIME: 14685 DATA_CHANNEL : Packet(seq_num=1, payload=997, corrupted=False) was lost!
-----------------------Retrying the lost packet------------
TIME: 14695 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=997, corrupted=False)
TIME: 14697 ACK_CHANNEL : udt_send called for Packet(seq_num=1, payload=ACK1, corrupted=False)
TIME: 14697 RECEIVING APP: received data 997
TIME: 14700 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=998, corrupted=False)
TIME: 14700 SENDING APP: sent data 998
TIME: 14702 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 14702 RECEIVING APP: received data 998
TIME: 14705 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=999, corrupted=False)
TIME: 14705 SENDING APP: sent data 999
TIME: 14707 ACK_CHANNEL : udt_send called for Packet(seq_num=1, payload=ACK1, corrupted=False)
TIME: 14707 RECEIVING APP: received data 999
-------------All Acks recived---------
13.264  Is the AVG time
PS D:\Desktop\Lab-3> 
```

We observe the following graph in this simulation

Pl	Time
0.1	9.088
0.15	11.576
0.2	13.77
0.25	17.676
0.3	20.022
0.35	25.492
0.4	30.878
0.45	36.966
0.5	47.262
0.55	59.042
0.6	75.536
0.65	105.202



E)



```
            # Now wait for "delay" amount of time

            delay_for_packet = random.randint(0,3*delay)
            yield self.env.timeout(delay_for_packet)


            # deliver the packet by calling the rdt_rcv()
            # function on the receiver side.
            self.receiver.rdt_rcv(packt)
```

TERMINAL   DEBUG CONSOLE   PROBLEMS   OUTPUT

```
TIME: 84 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 86 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 90 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=3, corrupted=False)
TIME: 90 SENDING APP: sent data 3
TIME: 94 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=3, corrupted=False)
TIME: 94 DATA_CHANNEL : Packet(seq_num=1, payload=$H!T, corrupted=True) was corrupted!
TIME: 96 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 99 ACK_CHANNEL : udt_send called for Packet(seq_num=1, payload=ACK1, corrupted=False)
TIME: 99 RECEIVING APP: received data 3
TIME: 105 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=4, corrupted=False)
TIME: 105 SENDING APP: sent data 4
TIME: 105 DATA_CHANNEL : Packet(seq_num=0, payload=4, corrupted=False) was lost!
TIME: 110 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=5, corrupted=False)
TIME: 110 SENDING APP: sent data 5
TIME: 110 ACK_CHANNEL : udt_send called for Packet(seq_num=1, payload=ACK1, corrupted=False)
TIME: 120 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=6, corrupted=False)
TIME: 120 SENDING APP: sent data 6
TIME: 130 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK0, corrupted=False)
TIME: 130 RECEIVING APP: received data 6
ERROR!! RECEIVING APP: received wrong data: 6 ,expected: 4
Halting simulation...
PS D:\Desktop\Lab-3>
```

The only modification we did in the code was, change the delay to , be any random number between 0 and 3*delay. So 3*delay servers as maximum delay faced by a packet.
And as we can see clearly, the code fails by reordering the packet.