IIT Goa CS 378 (Computer Networks LAB)
Instructor: Neha Karanjkar

Name: **ParasYadav**
Roll No.: **170010007**

LAB 4: Reliable Transport Protocols

1. (a)
When we set Pc=0.5 , the packet gets corrupted in the channel with 50% probability.
udt_send() is called by the data channel for packet 1. The packet gets corrupted in the data channel. Then udt_send() is called for packet 2. The receiver correctly receives packet 2, but the receiver was expecting packet 1. **So, the simulation is halted.**

Attached below is the corresponding output in terminal:

```
paras@paras-X556UQK:~$ cd protocol
paras@paras-X556UQK:~/protocol$ python3 Testbench.py
TIME: 3 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=0, corrupted=False)
TIME: 3 SENDING APP: sent data 0
TIME: 5 RECEIVING APP: received data 0
TIME: 6 DATA_CHANNEL : udt_send called for Packet(seq_num=1, payload=1, corrupted=False)
TIME: 6 SENDING APP: sent data 1
TIME: 6 DATA_CHANNEL : Packet(seq_num=1, payload=$H!T, corrupted=True) was corrupted!
TIME: 9 DATA_CHANNEL : udt_send called for Packet(seq_num=2, payload=2, corrupted=False)
TIME: 9 SENDING APP: sent data 2
TIME: 11 RECEIVING APP: received data 2
ERROR!! RECEIVING APP: received wrong data: 2 ,expected: 1
Halting simulation...
paras@paras-X556UQK:~/protocol$
```

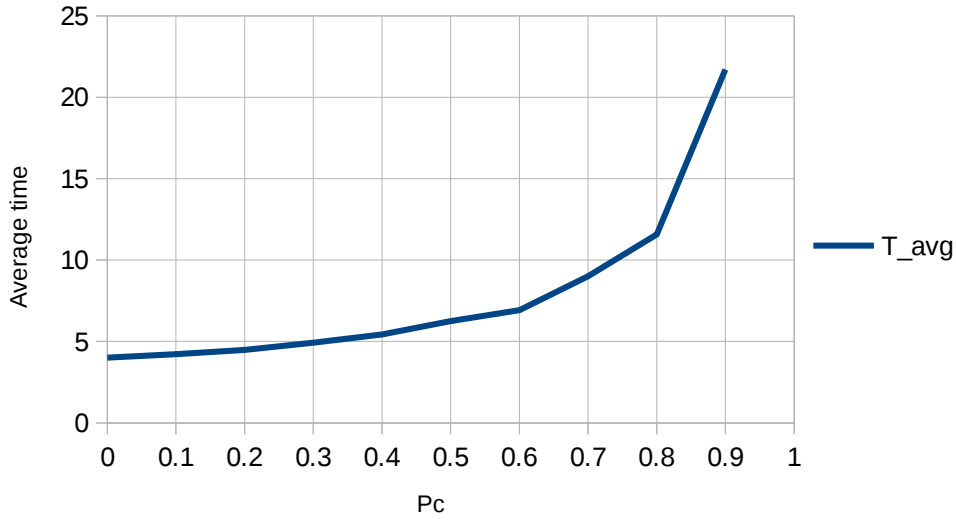
1. (b)
Yes, the protocol works for Pc>0 for the data channel , by sending ACKs and NAKs corresponding to the packet received by the receiver. Whenever the sender receives a NAK, it resends the packet. Whenever the sender receives an ACK, it sends the next packet.

Attached below is the corresponding output in terminal:

```
TIME: 6861 DATA_CHANNEL : udt_send called for Packet(seq_num=995, payload=995, corrupted=False)
TIME: 6861 SENDING APP: sent data 995
TIME: 6863 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 6863 RECEIVING APP: received data 995
TIME: 6867 DATA_CHANNEL : udt_send called for Packet(seq_num=996, payload=996, corrupted=False)
TIME: 6867 SENDING APP: sent data 996
TIME: 6869 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 6869 RECEIVING APP: received data 996
TIME: 6873 DATA_CHANNEL : udt_send called for Packet(seq_num=997, payload=997, corrupted=False)
TIME: 6873 SENDING APP: sent data 997
TIME: 6875 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 6875 RECEIVING APP: received data 997
TIME: 6879 DATA_CHANNEL : udt_send called for Packet(seq_num=998, payload=998, corrupted=False)
TIME: 6879 SENDING APP: sent data 998
TIME: 6881 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 6881 RECEIVING APP: received data 998
TIME: 6885 DATA_CHANNEL : udt_send called for Packet(seq_num=999, payload=999, corrupted=False)
TIME: 6885 SENDING APP: sent data 999
TIME: 6887 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 6887 RECEIVING APP: received data 999
TIME: 6891 DATA_CHANNEL : udt_send called for Packet(seq_num=1000, payload=1000, corrupted=False)
TIME: 6891 SENDING APP: sent data 1000
TIME: 6893 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 6893 RECEIVING APP: received data 1000
paras@paras-X556UQK:~/protocol$
```

1. (c)

Pc	T_avg
0	4.000
0.1	4.222
0.2	4.472
0.3	4.925
0.4	5.430
0.5	6.247
0.6	6.919
0.7	9.004
0.8	11.575
0.9	21.699



If you observe, you'll notice that the graph is **hyperbolic.** This is easily justifiable because the simulation takes a constant time, approximately equal to the sum channel delay for the data and ack channels. However, this same time increases as the packets get corrupted and tends to infinity when Pc=1 (i.e. the channel corrupts every packet passing through it), because then the packets gets corrupted every time with probability 1 and the sender never gets an ACK.

1. (d)

Let us assume we're trying to send packet 1.
udt_send() is called by the data channel for the packet 1. The application sends the data packet.
udt_send() is called by the ack channel for sending ACK, but the ACK gets corrupted on the way.
The sender checks that the payload is neither ACK nor NAK, and the simulation is halted.

Attached below is the corresponding output in terminal:

```
paras@paras-X556UQK:~/protocol$ python3 Testbench.py
TIME: 3 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=0, corrupted=False)
TIME: 3 SENDING APP: sent data 0
TIME: 5 ACK_CHANNEL : udt_send called for Packet(seq_num=0, payload=ACK, corrupted=False)
TIME: 5 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=0, payload=ACK, corrupted=False)
TIME: 11 RECEIVING APP: received data 1
TIME: 11 ACK_CHANNEL : Packet(seq_num=0, payload=$H!T, corrupted=True) was corrupted!
ERROR! rdt_rcv() was expecting an ACK or a NAK. Received a corrupted packet.
Halting simulation...
paras@paras-X556UQK:~/protocol$
```

2. Submitted as **Protocol_rdt22.py**

3. (a)
The sender sends data packet 0, but it gets lost in the data channel. The receiver doesn't receive anything so it doen't send any ACK or NAK. The sender does't receive any ACK or NAK, so it doen't send the next packet. As there is timeout concept, the whole process comes to a standstill. Finally the process terminates because of the time constraint in the **Testbench.py** file, *env.run(until=100).*

Attached below is the corresponding output in terminal:

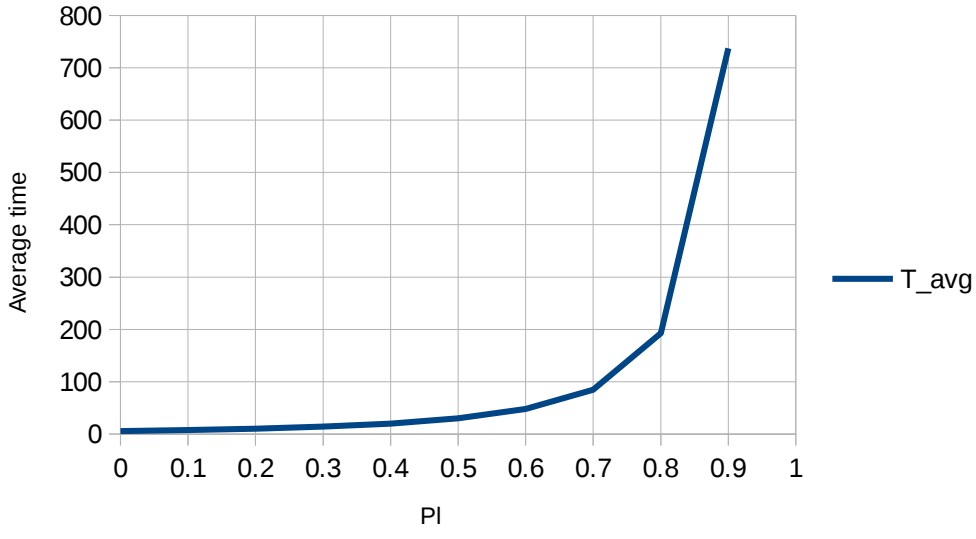
```
paras@paras-X556UQK:~/protocol$ python3 Testbench.py
TIME: 3 DATA_CHANNEL : udt_send called for Packet(seq_num=0, payload=0, corrupted=False)
TIME: 3 SENDING APP: sent data 0
TIME: 3 DATA_CHANNEL : Packet(seq_num=0, payload=0, corrupted=False) was lost!
paras@paras-X556UQK:~/protocol$
```

3. (b) Submitted as **Protocol_rdt3.py**

3.(c)

(Pc=0.2 is fixed for all cases)

Pl	T_avg
0	6.008
0.1	8.032
0.2	10.235
0.3	14.668
0.4	20.064
0.5	30.088
0.6	47.817
0.7	84.566
0.8	192.689
0.9	737.306



3. (d)

Channel.py code :

```
# SimPy model for an unreliable communication channel.
#
#	A packet sent over this channel:
#		- can get corrupted, with probability Pc
#		- can get lost, with probability Pl
#		- reaches the other end after "delay" amount of time, if it is not lost.
#
# Author: Neha Karanjkar


import simpy
import random
from Packet import Packet
import copy
class UnreliableChannel(object):

	def __init__(self,env,Pc,Pl,delay,name):
		# Initialize variables
		self.env=env
		self.Pc=Pc
		self.Pl=Pl
		self.delay=delay
		self.receiver=None
		self.name=name

	def udt_send(self,packt_to_be_sent):
		# this function is called by the sending-side
		# to send a new packet over the channel.
		packt=copy.copy(packt_to_be_sent) #!!! BEWARE: Python uses pass-by-reference by default.Thus a copy() is
necessary
		print("TIME:",self.env.now,self.name,": udt_send called for",packt)
		# start a process to deliver this packet across the channel.
		self.env.process(self.deliver_packet_over_channel(self.delay, packt))



	def deliver_packet_over_channel(self, delay, packt_to_be_delivered):
		packt=copy.copy(packt_to_be_delivered)
		# Is this packet lost?
		if random.random()<self.Pl:
			print("TIME:",self.env.now,self.name,":",packt,"was lost!")
		else:
			# Is this packet corrupted?
			if random.random()<self.Pc:
				packt.corrupt()
				print("TIME:",self.env.now,self.name,":",packt,"was corrupted!")
			# Now wait for "delay" amount of time
			delay=random.randint(1,9)
			yield self.env.timeout(delay)
			# deliver the packet by calling the rdt_rcv()
			# function on the receiver side.
			self.receiver.rdt_rcv(packt)
```

Output in Terminal :