

EECE 5830: NETWORK DESIGN

PROJECT PHASE 4: IMPLEMENT RDT 3.0 OVER AN UNRELIABLE UDP CHANNEL WITH BIT-ERRORS AND LOSS

AUTHOR: ASWIN KUMAR MANICKAM, 01624359.

PROGRAM:

- Server.py has the main program for the server to receive a file, Set E_prob OR P_Drop in server side alone if you need only data corruption/ Data packet loss.

```

1  from Globals import *
2  import Functions
3  import Send_Receive
4
5  '''To corrupt acknowledgement packet, Set value from 0 - 99 in E_prob'''
6  E_Prob = 0
7  P_Drop = 0
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
10 sock.bind(addr)
11 print("Server Started")
12
13 p = open('Received Image.jpg', 'wb')
14
15 receiver_sequence = 0
16 loopTimes,address,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence)
17 loops_struct.unpack("!I",loopTimes)[0]
18 print("No. of Loops to send the entire file: ", loop)
19 print("Writing/Receiving process starting soon")
20
21 for i in range(0,loop):
22     print("Loop : ",i+1)
23     if(i>= (loop-2)):
24         E_Prob=0
25         P_Drop=0
26     ImgPkt,address,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence,E_Prob,P_Drop)
27     p.write(ImgPkt)
28     i+=1
29
30 Received_File_Size = Functions.File_size(p)
31
32 p.close()
33 sock.close()
34
35 end = time.time()
36 Elapsed_time = end - start
37 print("Server: File Received\nReceived File size: {} \nTime taken in Seconds: {}".format(Received_File_Size,Elapsed_time))

```

- Client.py has the main program for the client to send a file, Set E_prob or P_Drop in client side alone if you need only Acknowledgement corruption/ Acknowledgement packet loss.

```

1  from Globals import *
2  import Functions
3  import Send_Receive
4
5  '''To corrupt data packet, Set value from 0 - 99 in E_prob'''
6  E_Prob = 0
7  P_Drop = 0
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
10 f = open('Lion.jpg','rb')
11
12 fileSize = Functions.File_size(f)
13
14 loop = Functions.looptimes(fileSize,buffer_size)
15 loop_bytes = struct.pack("!I",loop)
16 print("File has been Extracted \nFile size: {} \nNo. of Loops to send the entire file: {}".format(fileSize,loop))
17 seq_nbr = 0
18 seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,loop_bytes)
19
20 print("Client File Transfer Starts...")
21
22 for i in range(0,loop):
23     print("Unloop : ",i+1)
24     ImgPkt = f.read(buffer_size-3)
25     if(i>= (loop-2)):
26         E_Prob=0
27         P_Drop=0
28     seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,ImgPkt,E_Prob,P_Drop)
29     i+=1
30
31 f.close()
32 sock.close()
33
34 end = time.time()
35 Elapsed_time = end - start
36 print("Client: File Sent\nFile size sent to server: {} \nTime taken in Seconds: {}".format(fileSize,Elapsed_time))

```

- Set E_prob or P_Drop value from 0 to 99.

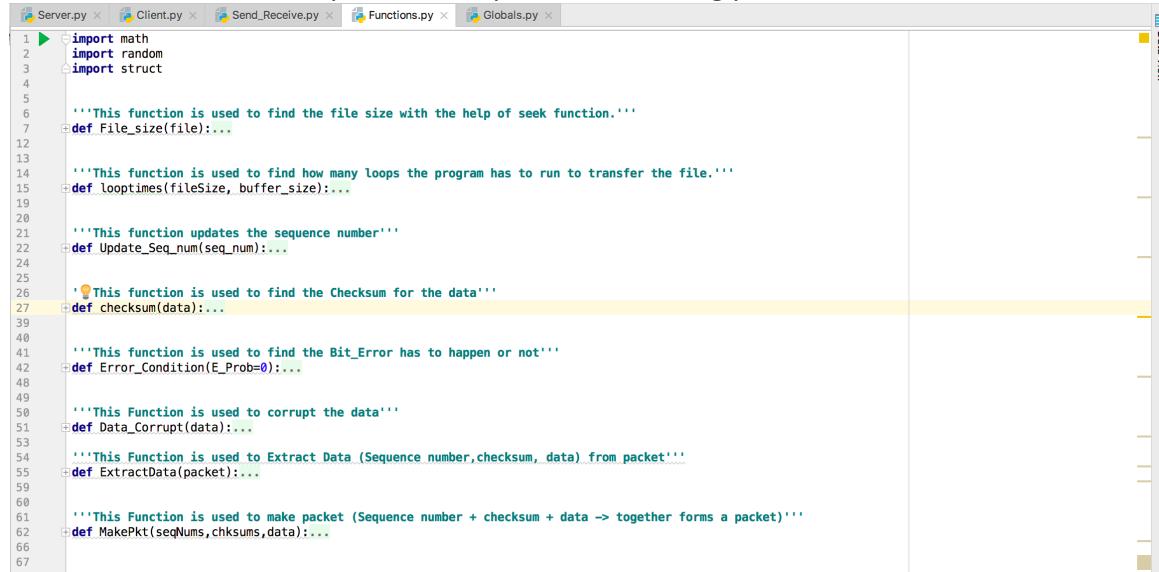
- Globals.py has the common variables for both server and client.

```

1  import socket
2  import struct
3  import time
4
5  start = time.time() #This is used to find the start time of the program, elapsed time can be found by end_time - start_time
6
7  '''Importing IP address, port number, Buffersize'''
8  UDP_IP = "localhost" #localhost is the IP address of this machine
9  UDP_PORT = 5005 #Port Number is assigned to 5005
10 buffer_size = 1024 #Buffer_size is set to 1024. packet size is 1024 with sequence number 1 byte, checksum 2 bytes, data 1021 bytes
11 addr = (UDP_IP,UDP_PORT)

```

- Functions.py has the functions for checksum, file size, number of time loop has to run, Bit-error condition, data corruption, make packet, extracting packet.



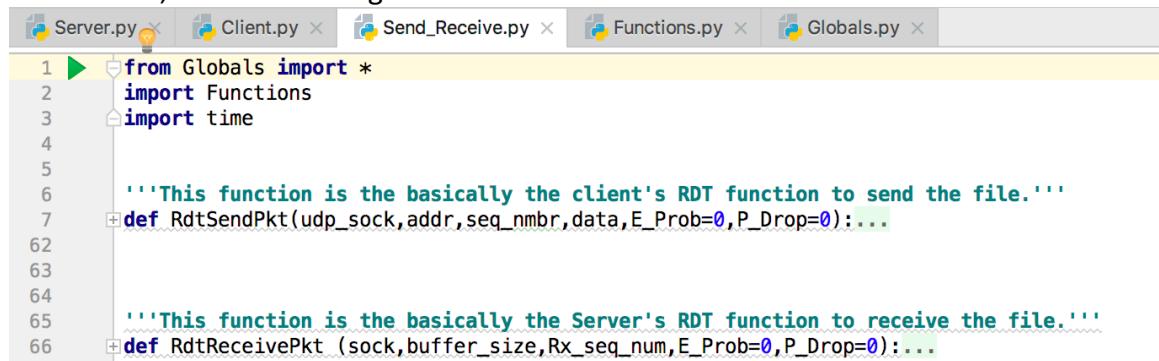
```

1 import math
2 import random
3 import struct
4
5
6 '''This function is used to find the file size with the help of seek function.'''
7 def File_size(file):...
8
9
10 '''This function is used to find how many loops the program has to run to transfer the file.'''
11 def loopTimes(fileSize, buffer_size):...
12
13
14 '''This function updates the sequence number'''
15 def Update_Seq_num(seq_num):...
16
17
18 '''This function is used to find the Checksum for the data'''
19 def checksum(data):...
20
21
22 '''This function is used to find the Bit_Error has to happen or not'''
23 def Error_Condition(E_Prob=0):...
24
25
26 '''This Function is used to corrupt the data'''
27 def Data_Corrupt(data):...
28
29
30 '''This Function is used to Extract Data (Sequence number,checksum, data) from packet'''
31 def ExtractData(packet):...
32
33
34 '''This Function is used to make packet (Sequence number + checksum + data -> together forms a packet)'''
35 def MakePkt(seqNums,chksums,data):...
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

```

*Functions used in this project are simple and code is well commented.

- Send_Receive.py has the RDT protocol for server and client to send/receive file with sequence number, checksum along with the data.



```

1 from Globals import *
2 import Functions
3 import time
4
5
6 '''This function is the basically the client's RDT function to send the file.'''
7 def RdtSendPkt(udp_sock,addr,seq_nmbr,data,E_Prob=0,P_Drop=0):...
8
9
10
11
12
13
14
15
16 '''This function is the basically the Server's RDT function to receive the file.'''
17 def RdtReceivePkt (sock,buffer_size,Rx_seq_num,E_Prob=0,P_Drop=0):...
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
39
40
41
42
43
44
45
46
47
48
49
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66

```

*RDT protocol is designed as per the project/protocol requirements.

PROGRAM EXECUTION:

- Run Server.py first.

The screenshot shows the PyCharm IDE interface with the Server.py file open. The code implements a UDP-based file transfer server. It initializes variables, binds to a port, and enters a loop to receive files from clients. The received files are saved to a local directory. The execution output window shows the server starting and receiving a file named 'Received Image.jpg'.

```

from Globals import*
import Send_Receive
import Functions
#Common variables
#To Send/Receive function
#Functions like checksum, filesize, bit error, etc.

E_Prob = 0
P_Drop = 0

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
sock.bind(addr)
print("Server Started")

p = open('Received Image.jpg', 'wb')
receiver_sequence = 0
loopTimes,address,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence)
print ("No. of Loops to send the entire file: ", loop)
print("Receiving process starting soon")
#opening a new file to copy the transferred image
#Server side Sequence number is initialized to zero
#Socket with IPv4, UDP
#Binding the socket
#Loop Iteration
#Receiving File from Client
#Loop to write entire transferred image in the new file, it runs 'loop' times
#Prints the Current loop, For easier way, initial print value starts from 1 to
#This is used to make sure corruption is not made at last loop, if not client
#Error probability manually set to zero (No corruption) if true,
#Packet Dropping probability manually set to zero (No corruption) if true,
#If packet received successfully, it writes to the file
#Loop Iteration
#file Received from Client at the end of Loop
#Calculating Received Image file size
#closing the file
#closing the socket
for i in range(0,loop):
    print("Loop : ",i+1)
    if(i>= (loop-2)):
        E_Prob=0
        P_Drop=0
    ImgPkt,addr,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence,E_Prob,P_Drop)
    p.write(ImgPkt)
    i=i+1
#file Received from Client at the end of Loop
Received_File_Size = Functions.File_size(p)
p.close()
sock.close()

```

- Run Client.py next and you can see file transfer starts.

The screenshot shows the PyCharm IDE interface with the Client.py file open. The code implements a UDP-based file transfer client. It reads a file from disk and sends it to a server in chunks. The client uses sequence numbers and ACKs to manage the transmission. The execution output window shows the client sending the file 'Lion.jpg' to the server.

```

from Globals import*
import Functions
import Send_Receive
#Common variables
#functions like checksum, filesize, bit error, etc.
#To Send/Receive function

E_Prob = 0
P_Drop = 0

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
f = open('Lion.jpg','rb')
fileSize = Functions.File_size(f)
#finding the loop value
#change loop from integer to byte inorder to send data from client to server
#Sequence Number is set to 0 initially
seq_nmbr = 0
seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,loop_bytes)
#sending the file size to Server
#file size is calculated
loop = Functions.looptimes(fileSize,buffer_size)
loop_bytes = struct.pack("!I", loop)
print("File has been Extracted \nFile size: {} \nNo. of Loops to send the entire file: {} ".format(fileSize,loop))
#reading the file, 1024 bytes at a time,
#this is used to make sure corruption is not made at last loop, if not client
#Error probability manually set to zero (No corruption) if true,
#Packet Dropping probability manually set to zero (No corruption) if true,
#Calls the function rdt_send to send the packet
#Loop Iteration
#it runs 'loop' times
#reading the file, 1024 bytes at a time,
#this is used to make sure corruption is not made at last loop, if not client
#Error probability manually set to zero (No corruption) if true,
#Packet Dropping probability manually set to zero (No corruption) if true,
#Calls the function rdt_send to send the packet
#Loop Iteration
#file closed
#Socket Closed
#Gets the End time
#Gets the Start time
#Creates the sequence number
#Sends the Packet...
#Start_Timer
#Ack: ACK1, sequence_number: 1
#Stop_Timer
Client: File Sent
File size sent to server: 649413
Time taken in Seconds:0.8448469638824463s
Process finished with exit code 0

```

- Check Server.py console again to make sure data has been transferred successfully.

```

from Globals import *
import Functions
import Send_Receive

'''To corrupt data packet, Set value from 0 - 99 in E_prob'''
E_Prob = 0
P_Drop = 0

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
f = open('Lion.jpg','rb')
fileSize = Functions.File_size(f)
loop = Functions.looptimes(fileSize,buffer_size)
loop_bytes = struct.pack("!I", loop)
print("File has been Extracted \nFile size: {} \nNo. of Loops to send the entire file: {}".format(fileSize,loop))
seq_nmbr = 0
seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,loop_bytes)

print('Client File Transfer Starts...')

for i in range(0,loop):
    print("Loop : "+str(i+1))
    ImgPkt = f.read(buffer_size-3)
    if(i>= (loop-2)):
        E_Prob=0
        P_Drop=0
    seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,ImgPkt,E_Prob,P_Drop) #calls the function rdt_send to send the packet
    i+=1

f.close()
sock.close()

end = time.time()
print("Time taken in Seconds: ", end - start)
for i in range(0,loop-2):
    if(i>= (loop-2)):
        break

```

Run: Server Client

Loop : 534
Sequence Number: 0,Receiver_sequence: 0, Checksum from Client: 46884, Checksum for Received File: 46884
Loop : 635
Sequence Number: 1,Receiver_sequence: 1, Checksum from Client: 14565, Checksum for Received File: 14565
Loop : 636
Sequence Number: 0,Receiver_sequence: 0, Checksum from Client: 55709, Checksum for Received File: 55709
Loop : 637
Sequence Number: 1,Receiver_sequence: 1, Checksum from Client: 1301, Checksum for Received File: 1301
Server: File Received
Received File size: 649413
Time taken in Seconds: 5.18013596534729s
Process finished with exit code 0

CORRUPTION (E_Prob is set to 60, that is 60% chance of corrupting the data [Sample Output]):

Data Corruption:

```

from Globals import*
import Send_Receive
import Functions

'''To corrupt acknowledgement packet, Set value from 0 - 99 in E_prob'''
E_Prob = 60
P_Drop = 0

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
sock.bind(addr)
print("Server Started")

p = open('Received Image.jpg', 'wb')

receiver_sequence = 0
loopTimes,address,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence) #Receiving the file size from client
loop= struct.unpack("!I", loopTimes)[0]
print ("No. of Loops to send the entire file: ", loop)
print("write/Receiving process starting soon")

for i in range(0,loop):
    print("Loop : "+str(i+1))
    if(i>= (loop-2)):
        E_Prob=0
        P_Drop=0
    ImgPkt,address,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence,E_Prob,P_Drop) #Calls the function RdtReceive
    p.write(ImgPkt)
    i+=1
#File Received from Client at the end of Loop

Received_File_Size = Functions.File_size(p)

p.close()
sock.close()

```

Run: Server Client

Server Started

```

1  from Globals import *
2  import Functions
3  import Send_Receive
4
5  '''To corrupt data packet, Set value from 0 - 99 in E_Prob'''
6  E_Prob = 0
7  P_Drop = 0
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
10 f = open('Lion.jpg','rb')
11
12 fileSize = Functions.file_size(f)
13 loop = Functions.looptimes(fileSize,buffer_size)
14 loop_bytes = struct.pack("1*",loop)
15 print("File has been Extracted While size: (%d) \nNo. of Loops to send the entire file: (%d) \nfileSize,loop")
16 seq_nbr = 0
17 seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,loop_bytes)
18
19 print("Client File Transfer Starts...")
20
21 for i in range(0,loop):
22     print("Vloop :%d",i+1)
23     ImgPkt = f.read(buffer_size-3)
24     if(i>=loop-2):
25         E_Prob=1
26         P_Drop=0
27     else:
28         P_Drop=1
29     seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,ImgPkt,E_Prob,P_Drop)
30     i+=1
31
32 f.close()
33 sock.close()
34
35 end = time.time()
36
37 print("Gets the End time")

```

Run: Server Client

Loop : 636
Sending the Packet...
Seq: 0, sequence_number: 0
Ack: ACK0, sequence_number: 0
Stop_Timer

Loop : 637
Sending the Packet...
Seq: 1, sequence_number: 1
Ack: ACK1, sequence_number: 1
Stop_Timer

Client Side Sent
File size sent to server: 649413
Time taken in Seconds:29.5139362812842245

Process finished with exit code 0

You can see Data Corrupted intentionally in the Server

```

1  from Globals import *
2  import Functions
3  import Send_Receive
4
5  '''To corrupt data packet, Set value from 0 - 99 in E_Prob'''
6  E_Prob = 0
7  P_Drop = 0
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
10 f = open('Lion.jpg','rb')
11
12 fileSize = Functions.file_size(f)
13 loop = Functions.looptimes(fileSize,buffer_size)
14 loop_bytes = struct.pack("1*",loop)
15 print("File has been Extracted While size: (%d) \nNo. of Loops to send the entire file: (%d) \nfileSize,loop")
16 seq_nbr = 0
17 seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,loop_bytes)
18
19 print("Client File Transfer Starts...")
20
21 for i in range(0,loop):
22     print("Vloop :%d",i+1)
23     ImgPkt = f.read(buffer_size-3)
24     if(i>=loop-2):
25         E_Prob=1
26         P_Drop=0
27     else:
28         P_Drop=1
29     seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,ImgPkt,E_Prob,P_Drop)
30     i+=1
31
32 f.close()
33 sock.close()
34
35 end = time.time()
36
37 print("Gets the End time")

```

Run: Server Client

Server Requested to Resend the Packet
Sequence Number: 1,Receiver_sequence: 1, Checksum From Client: 37022, Checksum for Received File: 55378
Sequence Number: 1,Receiver_sequence: 1, Checksum From Client: 37022, Checksum for Received File: 37022

Loop : 632
Sequence Number: 0,Receiver_sequence: 0, Checksum From Client: 20376, Checksum for Received File: 20376

Loop : 633
Sequence Number: 1,Receiver_sequence: 1, Checksum From Client: 34283, Checksum for Received File: 34283

Loop : 634
?

Data Corrupted
Server Requested to Resend the Packet
Sequence Number: 0,Receiver_sequence: 0, Checksum From Client: 46884, Checksum for Received File: 59886
Sequence Number: 0,Receiver_sequence: 0, Checksum From Client: 46884, Checksum for Received File: 46884

Process finished with exit code 0

Client resends the packet

```

1  from Globals import *
2  import Functions
3  import Send_Receive
4
5  '''To corrupt data packet, Set value from 0 - 99 in E_Prob'''
6  E_Prob = 0
7  P_Drop = 0
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
10 f = open('Lion.jpg','rb')
11
12 fileSize = Functions.file_size(f)
13 loop = Functions.looptimes(fileSize,buffer_size)
14 loop_bytes = struct.pack("1*",loop)
15 print("File has been Extracted While size: (%d) \nNo. of Loops to send the entire file: (%d) \nfileSize,loop")
16 seq_nbr = 0
17 seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,loop_bytes)
18
19 print("Client File Transfer Starts...")
20
21 for i in range(0,loop):
22     print("Vloop :%d",i+1)
23     ImgPkt = f.read(buffer_size-3)
24     if(i>=loop-2):
25         E_Prob=1
26         P_Drop=0
27     else:
28         P_Drop=1
29     seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,ImgPkt,E_Prob,P_Drop)
30     i+=1
31
32 f.close()
33 sock.close()
34
35 end = time.time()
36
37 print("Gets the End time")

```

Run: Server Client

Loop : 634
sequence_number: 1
Stop_Timer

Loop : 634
Sending the Packet...
Start_Timer
Seq: 0, sequence_number: 0
Resending the Packet
TMD OUT !!!

?

Resending the Packet...
Start_Timer
Seq: 0, sequence_number: 0
Stop_Timer

Loop : 635
Sending the Packet...
Stop_Timer

ACK CORRUPTION:

```

1  from Globals import *           #Common variables
2  import Functions              #Functions like checksum, filesize, bit_error, etc.
3  import Send_Receive            #To Send/Receive function
4
5  """To corrupt data packet, Set value from 0 - 99 in E_prob"""
6  E_Prob = 60                   #E_prob is the error probability and can be set from 0-99
7  P_Drop = 0                     #P_Drop is the packet dropping probability and can be set from 0-99
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)      #Socket with IPV4, UDP
10 f = open('Lion.jpg','rb')          #opening a new file, this file will be transferred to the server
11
12 fileSize = Functions.File_size(f)        #File size is calculated
13
14 loop = Functions.looptimes(fileSize,buffer_size)    #finding the loop value
15 loop_bytes = struct.pack("!I", loop)                #change loop from integer to byte in order to send data
16 print("File has been Extracted \nfile size: {} \nNo. of Loops to send the entire file: {}".format(fileSize,loop)) #Sequence Number is set to 0 initially
17 seq_nbr = 0
18 seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,loop_bytes) #sending the file size to Server
19
20 print('Client File Transfer Starts...')
21
22 for i in range(0,loop):          #it runs 'loop' times
23     print("\nLoop : ",i+1)        #Prints the Current loop, For easier way, initial print value starts from 1 to
24     ImgPkt = f.read(buffer_size-3) #reading the file, 1021 bytes at a time.
25     if(i>= (loop-2)):           #This is used to make sure corruption is not made at last loop, if not client
26         E_Prob=0                 #Error probability manually set to zero (No corruption) if true,
27         P_Drop=0                 #Packet Dropping probability manually set to zero (No corruption) if true.
28     seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,E_Prob,P_Drop) #calls the function rdt_send to send the packet
29     i=i+1
30
31 f.close()                      #File closed
32 sock.close()                   #Socket Closed
33
34 end = time.time()              #Gets the End time
35 Elapsed_time = end - start    #Gets the elapsed time
36 print("Client: File Sent\nfile size sent to server: {} \nTime taken in Seconds:{}\n".format(fileSize,Elapsed_time))
37

```

You can see ACK Corrupted intentionally in the client

```

1  from Globals import *           #Common variables
2  import Functions              #Functions like checksum, filesize, bit_error, etc.
3  import Send_Receive            #To Send/Receive function
4
5  """To corrupt data packet, Set value from 0 - 99 in E_prob"""
6  E_Prob = 60                   #E_prob is the error probability and can be set from 0-99
7  P_Drop = 0                     #P_Drop is the packet dropping probability and can be set from 0-99
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)      #Socket with IPV4, UDP
10 f = open('Lion.jpg','rb')          #opening a new file, this file will be transferred to the server
11
12 fileSize = Functions.File_size(f)        #File size is calculated
13
14 loop = Functions.looptimes(fileSize,buffer_size)    #finding the loop value
15 loop_bytes = struct.pack("!I", loop)                #change loop from integer to byte in order to send data
16 print("File has been Extracted \nfile size: {} \nNo. of Loops to send the entire file: {}".format(fileSize,loop)) #Sequence Number is set to 0 initially
17 seq_nbr = 0
18 seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,loop_bytes) #sending the file size to Server
19
20 print('Client File Transfer Starts...')
21
22 for i in range(0,loop):          #it runs 'loop' times
23     print("\nLoop : ",i+1)        #Prints the Current loop, For easier way, initial print value starts from 1 to
24     ImgPkt = f.read(buffer_size-3) #reading the file, 1021 bytes at a time.
25     if(i>= (loop-2)):           #This is used to make sure corruption is not made at last loop, if not client
26         E_Prob=0                 #Error probability manually set to zero (No corruption) if true,
27         P_Drop=0                 #Packet Dropping probability manually set to zero (No corruption) if true.
28     seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,E_Prob,P_Drop) #calls the function rdt_send to send the packet
29     i=i+1
30
31 f.close()                      #File closed
32 sock.close()                   #Socket Closed
33
34 end = time.time()              #Gets the End time
35 Elapsed_time = end - start    #Gets the elapsed time
36 print("Client: File Sent\nfile size sent to server: {} \nTime taken in Seconds:{}\n".format(fileSize,Elapsed_time))
37

```

Server request to resend the packet

```

1  from Globals import *           #Common variables
2  import Functions              #Functions like checksum, filesize, bit_error, etc.
3  import Send_Receive            #To Send/Receive function
4
5  """To corrupt data packet, Set value from 0 - 99 in E_prob"""
6  E_Prob = 60                   #E_prob is the error probability and can be set from 0-99
7  P_Drop = 0                     #P_Drop is the packet dropping probability and can be set from 0-99
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)      #Socket with IPV4, UDP
10 f = open('Lion.jpg','rb')          #opening a new file, this file will be transferred to the server
11
12 fileSize = Functions.File_size(f)        #File size is calculated
13
14 loop = Functions.looptimes(fileSize,buffer_size)    #finding the loop value
15 loop_bytes = struct.pack("!I", loop)                #change loop from integer to byte in order to send data
16 print("File has been Extracted \nfile size: {} \nNo. of Loops to send the entire file: {}".format(fileSize,loop)) #Sequence Number is set to 0 initially
17 seq_nbr = 0
18 seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,loop_bytes) #sending the file size to Server
19
20 print('Client File Transfer Starts...')
21
22 for i in range(0,loop):          #it runs 'loop' times
23     print("\nLoop : ",i+1)        #Prints the Current loop, For easier way, initial print value starts from 1 to
24     ImgPkt = f.read(buffer_size-3) #reading the file, 1021 bytes at a time.
25     if(i>= (loop-2)):           #This is used to make sure corruption is not made at last loop, if not client
26         E_Prob=0                 #Error probability manually set to zero (No corruption) if true,
27         P_Drop=0                 #Packet Dropping probability manually set to zero (No corruption) if true.
28     seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,E_Prob,P_Drop) #calls the function rdt_send to send the packet
29     i=i+1
30
31 f.close()                      #File closed
32 sock.close()                   #Socket Closed
33
34 end = time.time()              #Gets the End time
35 Elapsed_time = end - start    #Gets the elapsed time
36 print("Client: File Sent\nfile size sent to server: {} \nTime taken in Seconds:{}\n".format(fileSize,Elapsed_time))
37

```

PACKET LOSS (P_Drop is set to 60, that is 60% chance of dropping the packet [Sample Output]):

Data Packet Loss:

```

1: Project 2: SourceCode_P4_Manickam [Phase 3] ~/Doc 3: External Libraries 4: Server.py 5: Client.py 6: Send_Receive.py 7: Functions.py 8: Globals.py
1
from Globals import*
import Functions
import Send_Receive
import Functions
#Common variables
#To Send/Receive function
#Functions like checksum, filesize, bit error, etc.
E_Prob = 0
#E_Prob is the error probability and can be set from 0-99
P_Drop = 60
#P_Drop is the packet dropping probability and can be set from 0-99
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
#Socket with IPV4, UDP
sock.bind(addr)
#Binding the socket
print("Server Started")
#Server Started
p = open('Received Image.jpg', 'wb')
#opening a new file to copy the transferred image
receiver_sequence = 0
#Server side Sequence number is initialised to zero
loopTimes,address,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence) #Receiving the file size from client
loop= struct.unpack("!I", loopTimes)[0]
#This is used to make sure corruption is not made at last loop, if not
print("No. of Loops to send the entire file: ", loop)
#changing loop from byte to integer
print("write/Receiving process starting soon")
#Receiving File from Client
for i in range(0,loop):
    print("Loop : ",i+1)
    if(i<=(loop-2)):
        E_Prob=0
        P_Drop=0
        ImgPkt,address,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence,E_Prob,P_Drop) #Calls the function RdtReceivePkt
        p.write(ImgPkt)
        i+=1
    #File Received from Client at the end of Loop
    Received_File_Size = Functions.File_size(p)
    #Calculating Received Image file size
p.close()
#closing the file
sock.close()
#closing the socket

```

Run: Server
Server Started

4: Run Python Console

7:1 LF: UTF-8: 8

```

1: Project 2: SourceCode_P4_Manickam [Phase 3] ~/Doc 3: External Libraries 4: Server.py 5: Client.py 6: Send_Receive.py 7: Functions.py 8: Globals.py
1
from Globals import*
import Functions
import Send_Receive
#Common variables
#Functions like checksum, filesize, bit error, etc.
#To Send/Receive function
E_Prob = 0
#E_Prob is the error probability and can be set from 0-99
P_Drop = 0
#P_Drop is the packet dropping probability and can be set from 0-99
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
f = open('Lion.jpg','rb')
#Socket with IPV4, UDP
#opening a new file, this file will be transferred to the server
fileSize = Functions.File_size(f)
#File size is calculated
loop = Functions.looptimes(fileSize,buffer_size)
#finding the loop value
loop_bytes = struct.pack("!I", loop)
#change loop from integer to byte inorder to send data from client to server
print("File has been Extracted \nFile size: {} \nNo. of Loops to send the entire file: {}".format(fileSize,loop))
#Sequence Number is set to 0 initially
seq_nmbr = 0
seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,loop_bytes)
#Sequence Number is set to 0 initially
#sending the file size to Server
print('Client File Transfer Starts...')
#it runs 'loop' times
for i in range(0,loop):
    print("\nLoop : ",i+1)
    ImgPkt = f.read(buffer_size-3)
    if(i<=(loop-2)):
        E_Prob=0
        P_Drop=0
        seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,ImgPkt,E_Prob,P_Drop) #Calls the function rdt_send to send the packet
        i+=1
    #File closed
    #Socket closed
f.close()
sock.close()
#Gets the End time
end = time.time()

Run: Server Client
Loop : 636
Sending the Packet...
Start_Timer
Ack: ACK0, sequence_number: 0
Stop_Timer
Loop : 637
Sending the Packet...
Start_Timer
Ack: ACK1, sequence_number: 1
Stop_Timer
Client: File Sent
File size sent to server: 649413
Time taken in Seconds:32.664512157440186s
Process finished with exit code 0

```

4: Run Python Console

7:1 LF: UTF-8: 8

You can see Data packet lost intentionally in the Server

```

from Globals import *
import Functions
import Send_Receive
#Common variables
#Functions like checksum, filesize, bit error, etc.
#To Send/Receive function

'''To corrupt data packet, Set value from 0 - 99 in E_prob'''
E_Prob = 0
P_Drop = 0

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
f = open('Lion.jpg','rb')
fileSize = Functions.File_size(f)

loop = Functions.loops(fileSize,buffer_size)
loop_bytes = struct.pack("!I", loop)
print("File has been Extracted \nFile size: {} \nNo. of Loops to send the entire file: [{}]\n".format(fileSize,loop))
seq_nbr = 0
seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,loop_bytes)

print("Client File Transfer Starts...")
for i in range(loop):
    print("InLoop :{},i+1".format(i))
    ImgPkt = f.read(buffer_size-3)
    if(i>= (loop-2)):
        E_Prob=100
        P_Drop=1
    seq_nbr = seq_nbr + 1
    seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,ImgPkt,E_Prob,P_Drop) #calls the function rdt_send to send the packet
    i+=1
f.close()
sock.close()
end = time.time()

#File closed
#Socket Closed
#Gets the End time

```

Run: Server Client

- Loop : 632 DATA PACKET DROPPED
- Sequence Number: 0,Receiver_sequence: 0, Checksum from Client: 20376, Checksum for Received File: 20376
- Loop : 633 Sequence Number: 1,Receiver_sequence: 1, Checksum from Client: 34283, Checksum for Received File: 34283
- Loop : 634 Sequence Number: 0,Receiver_sequence: 0, Checksum from Client: 46884, Checksum for Received File: 46884
- Loop : 635 Sequence Number: 1,Receiver_sequence: 1, Checksum from Client: 14565, Checksum for Received File: 14565
- Loop : 636 Sequence Number: 0,Receiver_sequence: 0, Checksum from Client: 55709, Checksum for Received File: 55709

Run: Run Python Console

Client resends the packet

```

from Globals import *
import Functions
import Send_Receive
#Common variables
#Functions like checksum, filesize, bit error, etc.
#To Send/Receive function

'''To corrupt data packet, Set value from 0 - 99 in E_prob'''
E_Prob = 0
P_Drop = 0

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
f = open('Lion.jpg','rb')
fileSize = Functions.File_size(f)

loop = Functions.loops(fileSize,buffer_size)
loop_bytes = struct.pack("!I", loop)
print("File has been Extracted \nFile size: {} \nNo. of Loops to send the entire file: [{}]\n".format(fileSize,loop))
seq_nbr = 0
seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,loop_bytes)

print("Client File Transfer Starts...")
for i in range(loop):
    print("InLoop :{},i+1".format(i))
    ImgPkt = f.read(buffer_size-3)
    if(i>= (loop-2)):
        E_Prob=100
        P_Drop=1
    seq_nbr = seq_nbr + 1
    seq_nbr = Send_Receive.RdtSendPkt(sock,addr,seq_nbr,ImgPkt,E_Prob,P_Drop) #calls the function rdt_send to send the packet
    i+=1
f.close()
sock.close()
end = time.time()

#File closed
#Socket Closed
#Gets the End time

```

Run: Server Client

- Loop : 632 Sending the Packet... Start_Timer STOP_TIMER !!!
- Sending the Packet...
 - Ack: ACK0, sequence_number: 0 Stop_Timer
- Loop : 633 Sending the Packet...
 - Ack: ACK1, sequence_number: 1 Stop_Timer
- Loop : 634 - - -

Run: Run Python Console

ACK Packet Loss:

```

1  from Globals import *           #Common variables
2  import Functions               #Functions like checksum, filesize, bit error, etc.
3  import Send_Receive             #To Send/Receive function
4
5  '''To corrupt data packet, Set value from 0 - 99 in E_Prob'''   #E_Prob is the error probability and can be set from 0-99
6  E_Prob = 0                   #P_Drop is the packet dropping probability and can be set from 0-99
7  P_Drop = 60
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)          #Socket with IPv4, UDP
10 f = open('Lion.jpg','rb')                                         #opening a new file, this file will be transferred to server
11
12 fileSize = Functions.File_size(f)                                #file size is calculated
13
14 loop = Functions.looptimes(fileSize,buffer_size)                 #finding the loop value
15 loop_bytes = struct.pack("!I", loop)                               #change loop from integer to byte inorder to send data
16 print("File has been Extracted \nFile size: {0} \nNo. of Loops to send the entire file: {1}\n".format(fileSize,loop)) #Sequence Number is set to 0 initially
17 seq_nmbr = 0
18 seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,loop_bytes) #sending the file size to Server
19
20 print('Client File Transfer Starts...')
21
22 for i in range(0,loop):                                           #it runs 'loop' times
23     print("\nLoop :{0}, Sequence Number: {1} ".format(i+1))        #Prints the Current loop, For easier way, initial print value starts from 1
24     ImgPkt = f.read(buffer_size-3)                                 #reading the file, 1021 bytes at a time.
25     if(i>=(loop-2)):                                              #This is used to make sure corruption is not made at last loop, if not client
26         E_Prob=0                                                    #Error probability manually set to zero (No corruption) if true,
27         P_Drop=0                                                   #Packet Dropping probability manually set to zero (No dropping) if true.
28     seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,E_Prob,P_Drop) #calls the function rdt_send to send the packet
29     i+=1
30
31 f.close()                                                       #File closed
32 sock.close()                                                     #Socket Closed
33
34 end = time.time()                                              #Gets the End time
35 Elapsed_time = end - start                                     #Gets the elapsed time
36 print("Client: File Sent\nFile size sent to server: {0}\nTime taken in Seconds:{1}\n".format(fileSize,Elapsed_time))

```

You can see ACK packet dropped intentionally in the client

```

1  from Globals import *           #Common variables
2  import Functions               #Functions like checksum, filesize, bit error, etc.
3  import Send_Receive             #To Send/Receive function
4
5  '''To corrupt data packet, Set value from 0 - 99 in E_Prob'''   #E_Prob is the error probability and can be set from 0-99
6  E_Prob = 0                   #P_Drop is the packet dropping probability and can be set from 0-99
7  P_Drop = 60
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)          #Socket with IPv4, UDP
10 f = open('Lion.jpg','rb')                                         #opening a new file, this file will be transferred to the server
11
12 fileSize = Functions.File_size(f)                                #file size is calculated
13
14 loop = Functions.looptimes(fileSize,buffer_size)                 #finding the loop value
15 loop_bytes = struct.pack("!I", loop)                               #change loop from integer to byte inorder to send data from client to server
16 print("File has been Extracted \nFile size: {0} \nNo. of Loops to send the entire file: {1}\n".format(fileSize,loop)) #Sequence Number is set to 0 initially
17 seq_nmbr = 0
18 seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,loop_bytes) #sending the file size to Server
19
20 print('Client File Transfer Starts...')
21
22 for i in range(0,loop):                                           #it runs 'loop' times
23     print("\nLoop :{0}, Sequence Number: {1} ".format(i+1))        #Prints the Current loop, For easier way, initial print value starts from 1
24     ImgPkt = f.read(buffer_size-3)                                 #reading the file, 1021 bytes at a time.
25     if(i>=(loop-2)):                                              #This is used to make sure corruption is not made at last loop, if not client
26         E_Prob=0                                                    #Error probability manually set to zero (No corruption) if true,
27         P_Drop=0                                                   #Packet Dropping probability manually set to zero (No dropping) if true.
28     seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,E_Prob,P_Drop) #calls the function rdt_send to send the packet
29     i+=1
30
31 f.close()                                                       #File closed
32 sock.close()                                                     #Socket Closed
33
34 end = time.time()                                              #Gets the End time
35 Elapsed_time = end - start                                     #Gets the elapsed time

```

Run: Server Client
Loop : 682
Sending the Packet...
Start_Timer
Sequence Number: 0
Stop_Timer
Loop : 683
Sending the Packet...
Start_Timer
Sequence Number: 1
Stop_Timer
Loop : 684
ACKNOWLEDGEMENT PACKET DROPPED !!!
TIMED OUT !!!
? Sending the Packet...
Start_Timer
Ack_Timer
Sequence Number: 1
Stop_Timer
Loop : 684

Server request to resend the packet

```

1  from Globals import *           #Common variables
2  import Functions               #Functions like checksum, filesize, bit error, etc.
3  import Send_Receive             #To Send/Receive function
4
5  '''To corrupt data packet, Set value from 0 - 99 in E_Prob'''   #E_Prob is the error probability and can be set from 0-99
6  E_Prob = 0                   #P_Drop is the packet dropping probability and can be set from 0-99
7  P_Drop = 60
8
9  sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)          #Socket with IPv4, UDP
10 f = open('Lion.jpg','rb')                                         #opening a new file, this file will be transferred to the server
11
12 fileSize = Functions.File_size(f)                                #file size is calculated
13
14 loop = Functions.looptimes(fileSize,buffer_size)                 #finding the loop value
15 loop_bytes = struct.pack("!I", loop)                               #change loop from integer to byte inorder to send data from client to server
16 print("File has been Extracted \nFile size: {0} \nNo. of Loops to send the entire file: {1}\n".format(fileSize,loop)) #Sequence Number is set to 0 initially
17 seq_nmbr = 0
18 seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,loop_bytes) #sending the file size to Server
19
20 print('Client File Transfer Starts...')
21
22 for i in range(0,loop):                                           #it runs 'loop' times
23     print("\nLoop :{0}, Sequence Number: {1} ".format(i+1))        #Prints the Current loop, For easier way, initial print value starts from 1
24     ImgPkt = f.read(buffer_size-3)                                 #reading the file, 1021 bytes at a time.
25     if(i>=(loop-2)):                                              #This is used to make sure corruption is not made at last loop, if not client
26         E_Prob=0                                                    #Error probability manually set to zero (No corruption) if true,
27         P_Drop=0                                                   #Packet Dropping probability manually set to zero (No dropping) if true.
28     seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,E_Prob,P_Drop) #calls the function rdt_send to send the packet
29     i+=1
30
31 f.close()                                                       #File closed
32 sock.close()                                                     #Socket Closed
33
34 end = time.time()                                              #Gets the End time
35 Elapsed_time = end - start                                     #Gets the elapsed time

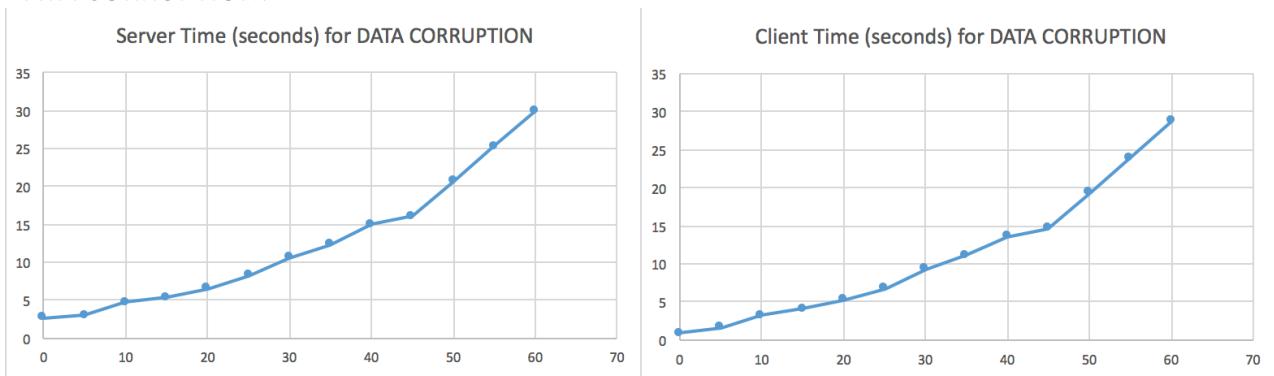
```

Run: Server Client
Sequence Number: 0, Receiver_sequence: 0, Checksum from Client: 2557, Checksum for Received File: 2557
Loop : 681
Sequence Number: 1, Receiver_sequence: 1, Checksum from Client: 28943, Checksum for Received File: 28943
Loop : 682
Sequence Number: 0, Receiver_sequence: 0, Checksum from Client: 59813, Checksum for Received File: 59813
Loop : 683
Sequence Number: 1, Receiver_sequence: 1, Checksum from Client: 19133, Checksum for Received File: 19133
Loop : 684
? Server Requested to Resend the Packet
Sequence Number: 1, Receiver_sequence: 0, Checksum from Client: 19133, Checksum for Received File: 19133
Sequence Number: 0, Receiver_sequence: 0, Checksum from Client: 22899, Checksum for Received File: 22899

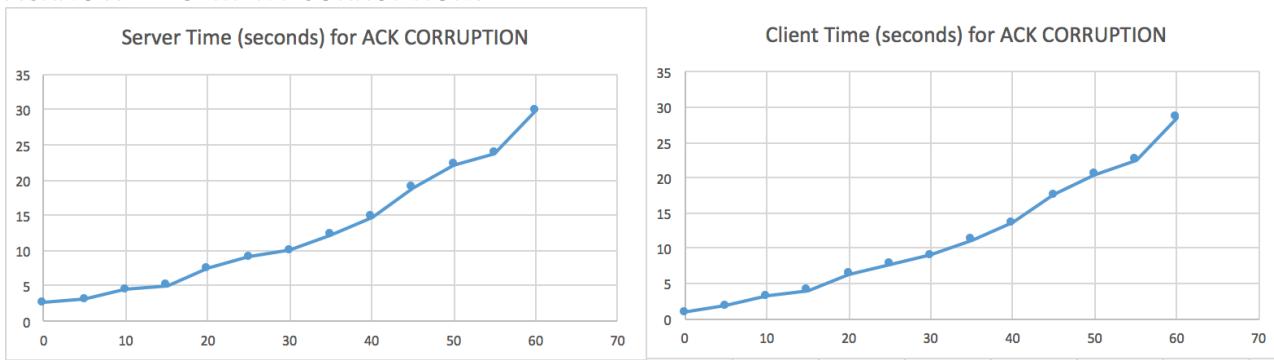
CHARTS FOR COMPLETION TIME FOR TRANSFERRING THE SAME FILE 0% LOSS/ERROR TO 60% LOSS/ERROR IN INCREMENTS OF 5%:

GRAPH -> X-AXIS: PROBABILITY 0 TO 60 %, Y-AXIS: TIME IN SECONDS

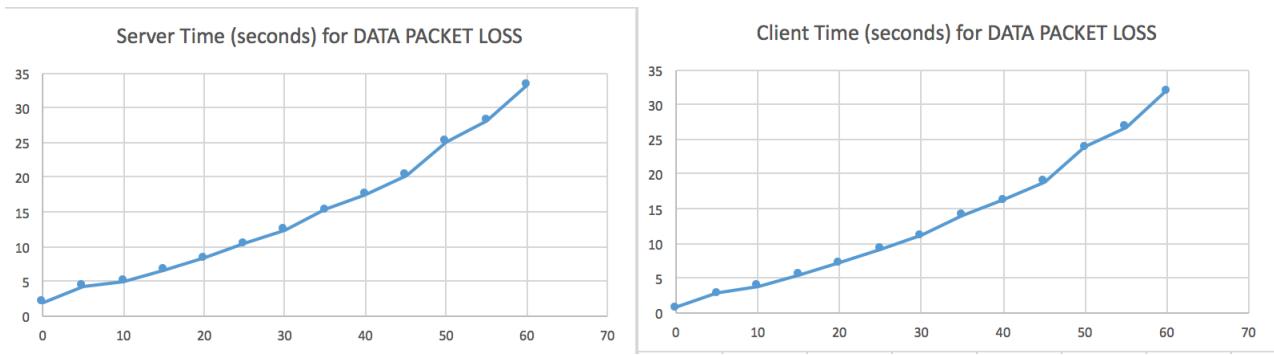
DATA CORRUPTION:



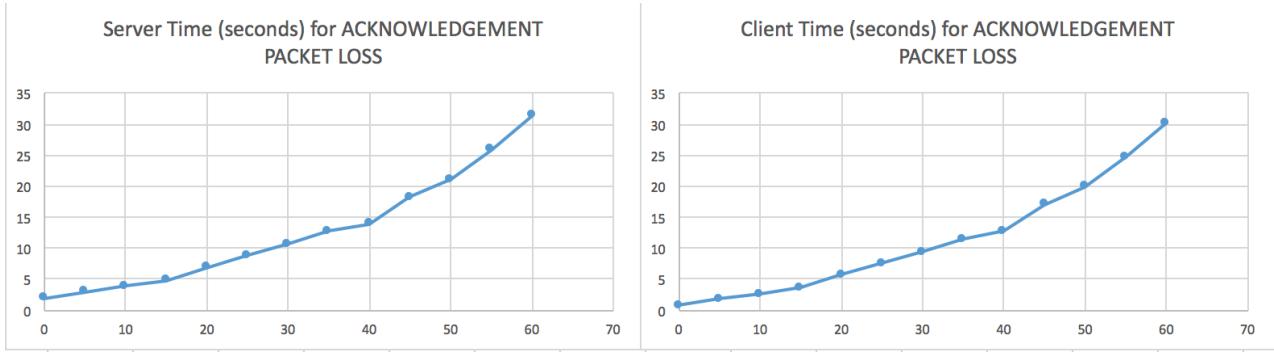
ACKNOWLEDGEMENT CORRUPTION:



DATA PACKET LOSS:

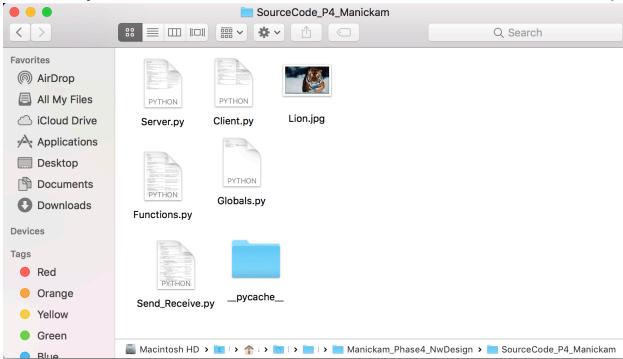


ACKNOWLEDGEMENT PACKET LOSS:

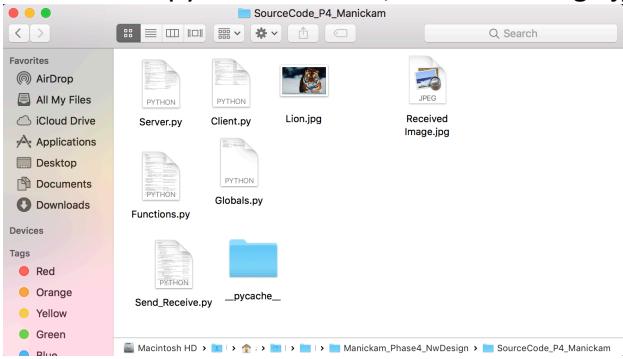


FILE TRANSFER OUTPUT:

Initially the folder looks like below screenshot (Without Received Image.jpg)



After Server.py executed first, Received Image.jpg is created



After Running the Program, New file called Received Image.jpg will be in the folder which shows successful File transfer

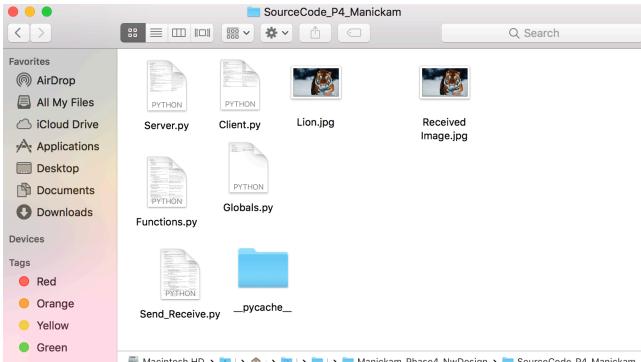
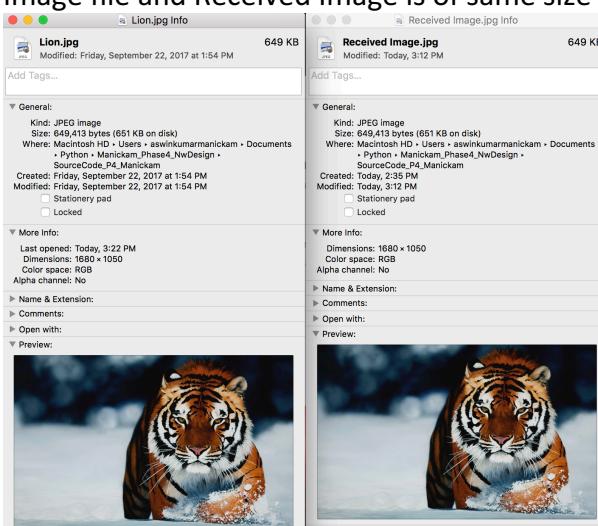


Image file and Received Image is of same size (Reliable Data Transfer)



Result:

Thus RDT 3.0 over an unreliable UDP channel with bit-errors and loss has been implemented.