**Phase 3**

**Network Design**

**Pavan Kumar Revanuru**

**Server\_Phase4.py**

import socket #socket library

import sys #includes exit function

import os

import hashlib

Host = 'localhost'

Port = 15001

ss = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

print 'Socket created'

# Bind socket to local host and port

ss.bind((Host, Port))

print 'Socket bind complete'

f = open('Server.jpg','wb') #open new file to write into

sel, addr = ss.recvfrom(1024)

select = int(sel)

if select == 2:

#corrupted ACK

seq, addr = ss.recvfrom(1024)

print(seq)

print 'corrupt'

got = ss.recvfrom(1024)

addr = got[1]

rchecksum = got[0]

#recvfrom allows storing of sender address

got2 = ss.recvfrom(1024)

#separate data from address

data = got2[0]

hchecksum = hashlib.md5()

hchecksum.update(data)

checksum = hchecksum.hexdigest()

ss.sendto("CORRUPTED",addr)

if select == 4:

#ACK Loss

seq = ss.recvfrom(1024)

print(seq)

got = ss.recvfrom(1024)

addr = got[1]

rchecksum = got[0]

#recvfrom allows storing of sender address

got2 = ss.recvfrom(1024)

#separate data from address

data = got2[0]

hchecksum = hashlib.md5()

hchecksum.update(data)

checksum = hchecksum.hexdigest()

while True:

seq, addr = ss.recvfrom(1024)

print(seq)

got = ss.recvfrom(1024)

addr = got[1]

rchecksum = got[0]

if rchecksum == "END":

break

#recvfrom allows storing of sender address

got2 = ss.recvfrom(1024)

#separate data from address

data = got2[0]

if data == "END":

break

hchecksum = hashlib.md5()

hchecksum.update(data)

checksum = hchecksum.hexdigest()

while checksum[:10] != rchecksum :

ss.sendto("NAC",addr)

seq = ss.recvfrom(1024)

print(seq)

got = ss.recvfrom(1024)

addr = got[1]

rchecksum = got[0]

if rchecksum == "END":

break

#recvfrom allows storing of sender address

got2 = ss.recvfrom(1024)

#separate data from address

data = got2[0]

if data == "END":

break

hchecksum = hashlib.md5()

hchecksum.update(data)

checksum = hchecksum.hexdigest()

f.write(data)

ss.sendto("ACK",addr)

print "Received!"

ss.sendto("Transmission complete", addr)

f.close()

**Client\_Phase4.py**

import socket #socket library

import sys #Library file

import os #Library file

import hashlib #Library file

from time import sleep

cs = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM) #udp socket

Host = 'localhost';

Port = 15001;

f = open('image.jpg','rb') #Open image file

i = 1

select = 0

print 'Select From Options' #Selection of one choice

print '1: No Loss/Error'

print '2: ACK Error'

print '3: Data Error'

print '4: ACK Packet Loss'

print '5: Data Packet Loss'

select = input('Option number: ')

cs.sendto(str(select),(Host, Port)) #Sending the input option

select = int(select) #converting from str to int

if select == 3: #Data error packet

#Corrupted Data Packet

fchecksum = "CORRUPTED"

print 'data error'

fimg = "FAKE"

cs.sendto(str(i),(Host,Port))

print(i)

cs.sendto(fchecksum,(Host, Port))

cs.sendto(fimg,(Host, Port))

ack = cs.recv(1024)

if select == 5: #data packet loss

print "Packet loss"

#Data Packet Loss

#No packet is sent

#loop ensures all packets are collected

i = 1

while True: #No packet/data loss

try:

img = f.read(1024) #Read 1024 bytes of data

if not img: #If the read file is empty

break

hchecksum = hashlib.md5() #Calculating the checksum

hchecksum.update(img)

checksum = hchecksum.hexdigest()

cs.sendto(str(i),(Host, Port)) #Sending the sequence number

print(i)

cs.sendto(checksum[:10],(Host, Port)) #Sending the checksum

cs.sendto(img,(Host, Port)) #Sending the data

cs.settimeout(1) #Timer starts

ack = cs.recv(1024) #Ack receives from the receiver

while ack != "ACK": #If the ACK is not received

hchecksum = hashlib.md5() #Send the packet again

hchecksum.update(img)

checksum = hchecksum.hexdigest()

cs.sendto(str(i),(Host, Port))

print(i)

cs.sendto(checksum[:10],(Host, Port))

cs.sendto(img,(Host, Port))

ack = cs.recv(1024)

i = i + 1 #Increment i

except socket.timeout: #If the timer expires, control comes here

print "timeout" #Make the packet again and send

hchecksum = hashlib.md5()

hchecksum.update(img)

checksum = hchecksum.hexdigest()

cs.sendto(str(i),(Host, Port))

print(i)

cs.sendto(checksum[:10],(Host, Port))

cs.sendto(img,(Host, Port))

cs.settimeout(1)

ack = cs.recv(1024)

while ack != "ACK":

hchecksum = hashlib.md5()

hchecksum.update(img)

checksum = hchecksum.hexdigest()

cs.sendto(str(i),(Host, Port))

print(i)

cs.sendto(checksum[:10],(Host, Port))

cs.sendto(img,(Host, Port))

ack = cs.recv(1024)

i = i + 1

#last statements

cs.sendto("LAST",(Host, Port))

cs.sendto("END",(Host, Port))

print "File Sent"

recv, addr = cs.recvfrom(4096) #Confirmation stating that the file is received.

print (recv)

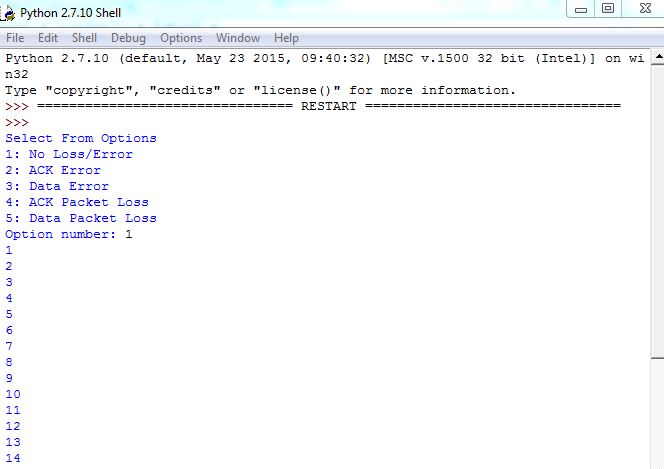
f.close()

**Explanation:**

1. When both client and server starts, the client side asks for the option to execute which type of function.
2. After the initial connection setup, if the user asks for option 2 which is ACK error, then the statements which are related to it gets executed first. At the server, the ack error is detected and the ‘corrupt’ reply is given in the ack signal. Then the packet 1 is sent again which is executed by the no data/error loss function.
3. In the final block where all the packets are sent, the sequence number, checksum and the data are sent. After sending the data packet, the timer is turned on and the countdown begins. It has to receive the ack within the period. Else, the packet is sent again. If the ack received is correct, the sequence number is increased and the loop is executed. If the ack received is not correct, then the packet is made again and sent. The timer again is made to start.
4. If the user selects option 3, that is Data error, there will be a change in the checksum and the final value doesn’t tally with the original one. There will be no ack and the packet will be made to send again.
5. If the user selects option 4, that is ack loss, the sender will send the packet again as the timer will run out.
6. After the image is successfully transferred, the server sends a command saying that the transmission is complete by ‘Transmission Complete’.

**Screenshots:**

**Client side output:**





Server side output:

