**Phase 5**

**Network Design**

**Pavan Kumar Revanuru - 01578776**

**Server\_Phase5.py**

import socket #socket library

import sys #includes exit function

import os

import hashlib

HOST = 'localhost'

PORT = 8883 # Assigned port

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

print 'Socket created'

s.bind((HOST, PORT))

print 'Socket bind complete'

#open new file to write into

f = open('Server.jpg','wb')

hel, addr = s.recvfrom(1024)

print (hel)

select, addr = s.recvfrom(1024)

if select == 2:

#corrupted ACK

seq, addr = s.recvfrom(1024)

print(seq)

print "Corrupted"

got = s.recvfrom(1024)

addr = got[1]

rchecksum = got[0]

#recvfrom allows storing of sender address

got2 = s.recvfrom(1024)

#separate data from address

data = got2[0]

hchecksum = hashlib.md5()

hchecksum.update(data)

checksum = hchecksum.hexdigest()

s.sendto("CORRUPTED",addr)

if select == 4:

#ACK Loss

seq = s.recvfrom(1024)

print(seq)

got = s.recvfrom(1024)

addr = got[1]

rchecksum = got[0]

#recvfrom allows storing of sender address

got2 = s.recvfrom(1024)

#separate data from address

data = got2[0]

hchecksum = hashlib.md5()

hchecksum.update(data)

checksum = hchecksum.hexdigest()

if select == "6":

while True:

seq, addr = s.recvfrom(1024)

print(seq)

got = s.recvfrom(1024)

addr = got[1]

rchecksum = got[0]

if rchecksum == "END":

break

#recvfrom allows storing of sender address

got2 = s.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 :

s.sendto("NAK",addr)

seq = s.recvfrom(1024)

print(seq)

got = s.recvfrom(1024)

addr = got[1]

rchecksum = got[0]

if rchecksum == "END":

break

#recvfrom allows storing of sender address

got2 = s.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)

s.sendto("ACK",addr)

f.close()

print "Received!"

#open file to read and send

else:

while True:

seq = s.recvfrom(1024)

print(seq)

got = s.recvfrom(1024)

addr = got[1]

rchecksum = got[0]

if rchecksum == "END":

break

#recvfrom allows storing of sender address

got2 = s.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 :

s.sendto("NAK",addr)

seq = s.recvfrom(1024)

print(seq)

got = s.recvfrom(1024)

addr = got[1]

rchecksum = got[0]

if rchecksum == "END":

break

#recvfrom allows storing of sender address

got2 = s.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)

s.sendto("ACK",addr)

f.close()

print "Received!"

**Client\_Phase5.py**

import socket #socket library

import sys #includes exit function

import os

import hashlib

import signal

from time import sleep

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

host = 'localhost';

port = 8883;

win = input('Set Window: ')

buff = []

#open Test.jpg file

f = open('Test.jpg','rb')

NextSeqNum = 0

#sequence number

PacketNum = 0

#packet number

select = 0

s.sendto ("Hello", (host,port))

print 'Select From Options'

print '1: No Loss/Error'

print '2: ACK Error'

print '3: Data Error'

print '4: ACK Packet Loss'

print '5: Data Packet Loss'

print '6: No recovery'

select = input('Choice: ')

s.sendto(str(select),(host, port))

if select == 3:

#Corrupted Data Packet

fchecksum = "CORRUPTED"

fimg = "FAKE"

s.sendto(str(NextSeqNum),(host,port))

print(NextSeqNum)

s.sendto(fchecksum,(host, port))

s.sendto(fimg,(host, port))

ack = s.recv(1024)

if select == 5:

#Data Packet Loss

#No packet is sent

#loop ensures all packets are collected

NextSeqNum = 0

#create packets and place into buffer

if select == 6:

while True:

img = f.read(1024)

if not img:

break

buff.append(img)

print(PacketNum)

PacketNum = PacketNum + 1

while NextSeqNum < PacketNum:

Base = 0

print 'Batch'

while Base < win:

if not buff[NextSeqNum]:

break

cur = buff[NextSeqNum]

#current packet

hchecksum = hashlib.md5()

hchecksum.update(cur)

checksum = hchecksum.hexdigest()

s.sendto(str(NextSeqNum),(host, port))

print(NextSeqNum)

s.sendto(checksum[:10],(host, port))

s.sendto(cur,(host, port))

ack = s.recv(1024)

NextSeqNum = NextSeqNum + 1

Base = Base + 1

if NextSeqNum == PacketNum:

break

else:

while True:

try:

img = f.read(1024)

if not img:

break

buff.append(img)

print(PacketNum)

PacketNum = PacketNum + 1

while NextSeqNum < PacketNum:

Base = 0

print 'Batch'

while NextSeqNum < Base+win:

if not buff[NextSeqNum]:

break

cur = buff[NextSeqNum]

#current packet

hchecksum = hashlib.md5()

hchecksum.update(cur)

checksum = hchecksum.hexdigest()

s.sendto(str(NextSeqNum),(host, port))

print(NextSeqNum)

s.sendto(checksum[:10],(host, port))

s.sendto(cur,(host, port))

s.settimeout(1)

ack = s.recv(1024)

while ack != "ACK":

hchecksum = hashlib.md5()

hchecksum.update(cur)

checksum = hchecksum.hexdigest()

s.sendto(str(NextSeqNum),(host, port))

print(NextSeqNum)

s.sendto(checksum[:10],(host, port))

s.sendto(cur,(host, port))

ack = s.recv(1024)

NextSeqNum = NextSeqNum + 1

Base = Base + 1

if NextSeqNum == PacketNum:

break

except socket.timeout:

print "Timeout: Retrying"

hchecksum = hashlib.md5()

hchecksum.update(cur)

checksum = hchecksum.hexdigest()

s.sendto(str(NextSeqNum),(host, port))

print(NextSeqNum)

s.sendto(checksum[:10],(host, port))

s.sendto(cur,(host, port))

s.settimeout(1)

ack = s.recv(1024)

#last statement indicates end of packets

s.sendto("LAST",(host, port))

s.sendto("END",(host, port))

print "File Sent"

recv, addr = s.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’.