CLIENT ALGORITHM

ClientDownload Algorithm:

EchoClient(protocol.Protocol)

init(echofactory,id,filename)

generating client handlers with the same id using echofactory;

filename is taken as input;

connectionMade()

This module is used for creating the first connection made by the client after sending the request;

if(self.id == None) then

filename is taken as input;

Blockcreator().createinit() is invoked using self.transport.write;

else

the existing client instance is going to create another handler for the client by callbacks of createBlockForClient(REINIT,filename) in BlockCreater;

def dataRecieved(self,data):

this module is called after the client's request is processed by the server.

if(d[DS.CONTENT\_TYPE]==DS.ACK and[DS.OPERATION]==DS.INIT) then`

id = d[DS.ID]

Call forOperation(("GET " +str(filename)) from the BlockCreator class.

if(self.d[DS.CONTENT\_TYPE]==DS.DATA):

if(self.d[DS.OPERATION]==DS.REINIT) then

creates client handler upon the acknowledgement from the server depending on the size of the data that is transferred.

ef.getMessageFromClient(id,filename)

recieveBlock(self.d)

elif(d[CONTENT\_TYPE]==EOF and d[OPERATION]==GET):

only the last block is received by the client.

expect:

problem with the recieve block;

def recieveblock(data):

flag=0

flag\_dic={} #flag dictionary which stores flag variable

f is used to open a newfile.txt

flag\_dic=self.ef.clientSync(data)

if(flag\_dic['flag']==1):

variable is assigned with block content of flag\_dic;

if(variable[DS.CONTENT\_TYPE]==DS.DATA):

try:

write the [DS.CONTENT] of the variable to f

except:

LOG.debug( "Error in converting from json")

self.transport.loseConnection()

elif(variable[DS.CONTENT\_TYPE]==DS.EOF):

try:

checks whether the end of file is reached and then writes the EOF block to f

expect:

shows an error that connection lost before sending the EOF and loses connection

else

return

class EchoFactory(protocol.ClientFactory):

noc=1

bNum={}

dict\_store\_data={}

def \_\_init\_\_(self):

self.ide=None

self.filename=None

used to objects of the type EchoFactory from input of id and filename.

def buildProtocol(self, addr):

builds the protocol for the EchoClient with id and filename.

def getMessageFromClient(self,Id,filename):

This method intiates the connection with the server but with the id.

So that EchoFactory uses the id for creating the client.noc is incremented and id and filename is specified.

Once the connection is made using the reactor.connectTCP(ip,8000,self)

it is shown in the logs as “connection is made”.

def clientSync(self,data):

cid=data[DS.ID]

flag\_dic={}

flag\_dic['flag']=0 #function variable

if(self.bNum.has\_key(cid)):

if(self.bNum[cid]==data[DS.BLOCK]):

Here the checking is done from second block and returns the flag\_dic from clientSync.

else:

Here the checking is done for first block and the id is added to the bNum.

bNum has got the new id in the client sync.

if(data[DS.BLOCK]==self.bNum[cid]):

This condition is used to check the first block.

if(flag\_dic['flag']==0):

Here if the block is not sent orderly then that block will be added to dictionary(buffer)

blocks are randomnly received.

if(self.dict\_store\_data.has\_key((cid,self.bNum[cid]))):

if there is any required block in the dictionary that should be added to the file.

def clientConnectionFailed(self, connector, reason):

print "Connection failed......"

stop the reactor.

def clientConnectionLost(self, connector, reason):

print "Connection lost......"

if reactor.running:

reactor.stop()

echoFactory=EchoFactory()

1st connection is made

reactor.connectTCP(ip,8000,echoFactory)

reactor.run()