#coding=utf-8

import sys

import socket

import getopt

import collections

import Checksum

import BasicSender

import time

from cryptography.fernet import Fernet

'''

This is a skeleton sender class. Create a fantastic transport protocol here.

'''

sys.argv =[sys.argv[0],'-fD:study\sophomore1\ComputerNetwork\TheFirstProgram\CUG.jpg','-p33123','-a127.0.0.1']

class Sender(BasicSender.BasicSender):

    def \_\_init\_\_(self, dest, port, filename, debug=False):

        super(Sender, self).\_\_init\_\_(dest, port, filename, debug)

    def get\_file\_size(file\_path):

        return os.path.getsize(file\_path)

    # Main sending loop.

    def start(self):

        start\_time = time.time()

        end\_time = 0

        size =2990 #读取大小

        seqno = 0 #序列号

        msg\_type = None #记录消息类型的

        msg = self.infile.read(size)

        #新功能

        rtt\_estimate = 0.04  # 初始RTT估计值，单位为秒(这里的环路时间是发送端->接收端->发送端的时间)

        base = 0 #窗口第一个的序号

        window\_size = 5

        window\_max\_size = 10

        loss\_count = 0 #用来动态调整窗口大小（丢包，错误等情况）

        max\_loss = 3#丢包超过这个数，说明网络环境不好，就减少窗口数

        window = []

        send\_time=0

        rtt\_sample =0

        num\_elements = 0

        send\_count=0

        #6Fvkon6hqoiDc0JhOVvxJMBdkLYXY2xKxGrP7BpQ5Z8=

        #irisMX6WX1DoE415qmmVL2igR9EhuH\_986vCsVo-zuE=

        #密钥

        key = b'6Fvkon6hqoiDc0JhOVvxJMBdkLYXY2xKxGrP7BpQ5Z8='

        # 创建一个Fernet对象

        cipher\_suite = Fernet(key)

        while True:

            num\_elements = len(window)

            #填充窗口window

            while num\_elements < window\_size and msg\_type != 'end':

                #协议

                next\_msg = self.infile.read(size)

                msg\_type = 'data'

                if seqno == 0:

                    msg\_type = 'start'

                elif next\_msg == "":

                    msg\_type = 'end'

                # 加密

                msg = cipher\_suite.encrypt(msg)  # 需要加密的数据

                packet = self.make\_packet(msg\_type, seqno, msg)

                self.send(packet)

                print "sent: %s" % seqno

                send\_count += 1

                window.append((packet,seqno))

                num\_elements = len(window)

                msg = next\_msg

                seqno += 1

                if msg\_type == 'end':

                    break

            send\_time = time.time()

            print "window\_size: %s" % window\_size

            print "rtt\_sample: %s" % rtt\_sample

            #获取response

            while window:

                #response = self.receive(rtt\_estimate+0.04)

                response = self.receive(max(0.04,rtt\_estimate+0.001))

                print "recv: %s" % response

                packet, base = window[0]

                if response is not None:

                    if self.handle\_response(response):#校对校验码

                        parts = response.split("|")

                        #动态调整时间

                        rtt\_sample = time.time() - send\_time  # 计算RTT（环路时间）样本值;

                        if rtt\_sample < 0.6 and rtt\_sample > 0.01:#防止样本计算的是程序运行时间导致得到特别离谱的结果(如样本值小于0.01那大概率是算的程序运行的时间)

                            rtt\_estimate = 0.9 \* rtt\_estimate + 0.1 \* rtt\_sample

                            #print "rtt\_estimate: %s" % rtt\_estimate

                        #动态调整滑动窗口

                        if window\_size < window\_max\_size:

                            window\_size += 1  # 如果收到新的确认，增大窗口大小

                        #清理window

                        if parts[0]=='ack':

                            while base < int(parts[1]) and window:#清ack前面的数据

                                if window:

                                    del window[0]

                                    if window:

                                        base = window[0][1]

                        if parts[0]=='buffer':#清buffer所指的数据

                            for index, item in enumerate(window):#enumerate()函数获取元素的索引

                                if item[1] == int(parts[1]):

                                    del window[index]

                                    if window:

                                        base = window[0][1]

                        break #break window

                    else: #校验码不对

                        loss\_count += 1  # 如果校验码不对，增加丢包计数

                        if loss\_count > max\_loss:  # 如果丢包计数超过max\_loss，减小窗口大小

                            window\_size = max(5,window\_size - 1)

                            if rtt\_estimate<0.4:#防止评估rrt过大

                                rtt\_estimate +=0.002

                            loss\_count = 0

                        self.send(packet)  # 重传数据包

                        #print "Resent: %s" % packet

                        print "Resent:%s"% base

                        send\_count += 1

                        window[0] = (packet,int(packet.split('|')[1]))

                        send\_time = time.time()

                elif response is None:  # 超时时间(单位是秒)

                    #time.sleep(rtt\_estimate)#不知道要不要

                    #print "Timeout for packet "

                    loss\_count += 1  # 如果超时，增加丢包计数

                    if loss\_count > max\_loss:  # 如果丢包（超时）计数超过max\_loss，减小窗口大小

                        window\_size = max(5,window\_size - 1)

                        if rtt\_estimate<0.4:#防止评估rrt过大

                            rtt\_estimate +=0.002

                        loss\_count = 0

                    self.send(packet)  # 重传数据包

                    #print "Resent: %s" % packet

                    print "Resent:%s"% base

                    send\_count += 1

                    window[0] = (packet, int(packet.split('|')[1]))

                    send\_time = time.time()

            if msg\_type == 'end' and not window :

                break

        end\_time = time.time()

        #输出传输结果

        finish\_time =end\_time-start\_time

        print "finish\_time:%s"%  finish\_time

        print "send\_count:%s"%  send\_count

        original\_file\_size = seqno\*size/1024

        print "original\_file\_size(KB):%s"%  original\_file\_size

        packet\_loss\_rate=100\*(send\_count-seqno)/send\_count

        print "packet\_loss\_rate:%s%% "% packet\_loss\_rate

        bandwidth\_usage = (send\_count\*size)/1024

        print "sender\_bandwidth\_usage(KB):%s "% bandwidth\_usage

        self.infile.close()

    # Handles a response from the receiver.

    def handle\_response(self,response\_packet):

        if Checksum.validate\_checksum(response\_packet):

            #print "recv: %s" % response\_packet

            return True

        else:

            print "recv: %s <--- CHECKSUM FAILED" % response\_packet

            return False

    def handle\_timeout(self):

        pass

    def handle\_new\_ack(self, ack):

        pass

    def handle\_dup\_ack(self, ack):

        pass

    def log(self, msg):

        if self.debug:

            print msg

'''

This will be run if you run this script from the command line. You should not

change any of this; the grader may rely on the behavior here to test your

submission.

'''

if \_\_name\_\_ == "\_\_main\_\_":

    def usage():

        print "BEARS-TP Sender"

        print "-f FILE | --file=FILE The file to transfer; if empty reads from STDIN"

        print "-p PORT | --port=PORT The destination port, defaults to 33122"

        print "-a ADDRESS | --address=ADDRESS The receiver address or hostname, defaults to localhost"

        print "-d | --debug Print debug messages"

        print "-h | --help Print this usage message"

    try:

        opts, args = getopt.getopt(sys.argv[1:],

                               "f:p:a:d", ["file=", "port=", "address=", "debug="])

    except:

        usage()

        exit()

    port = 33122

    dest = "localhost"

    filename = None

    debug = False

    for o,a in opts:

        if o in ("-f", "--file="):

            filename = a

        elif o in ("-p", "--port="):

            port = int(a)

        elif o in ("-a", "--address="):

            dest = a

        elif o in ("-d", "--debug="):

            debug = True

    s = Sender(dest,port,filename,debug)

    try:

        s.start()

    except (KeyboardInterrupt, SystemExit):

        exit()