

**计算机网络课程设计**

**报告**

实习任务名称： Project 2 实习报告

学 号： 20171001561

姓 名： 周麟

**中国地质大学地理信息工程学院**

**软件工程系**

**2020年 5 月**

**1.发送文件**

在发送文件之前，首先读取文件的内容，然后一次性存入一个列表中，这个列表中每个元素保存着每个包的类型，包的sequence number，还有限定范围的文件数据。除了第一个包是start类型其他的都是data类型。然后创建一个线程，运行接收ack的函数，同时运行发送包的函数。

保存文件数据至列表容器中的实现：

packets = []

seq\_num = 0

while True:

data = self.file.read(packet\_size)

if not data:

break

if seq\_num == 0:

msg\_type = "start"

else:

msg\_type = "data"

packets.append(self.make\_packet(msg\_type,seq\_num,data))

seq\_num += 1

当发送完所有包之后，再发送一个空包，类型为end，模拟请求断开连接。

**2.线程**

为了防止发送和接收ack包过程中会导致参数改变而导致的不可靠传输，引入的线程，在发送包时，进程上锁，就不能接收ack，在接收ack时，进程上锁，就不能发送包，这保证了参数的可靠性。

**3.计时器**

为了实现超时重传的协议，引入了一个timer类，这个类主要实现的函数为：

def start(self) 开始计时

def stop(self) 结束计时

def running(self) 检测计时器是否开始计时

def timeout(self) 检测是否超时

当一次性发送5个包之后，开始计时，接收包之后停止计时，然后判断是否超时，如果超时则按照记录的ack重新传ack~ack+4的包。

**4.滑动窗口**

规定窗口的尺寸为5，每次一次性发送五个包。根据更新的包的位置的大小适时的滑动到n~n+4。

**5.接收ack**

为了记录应该发送的包的位置，引入的一个参数base，它记录下次应该发送的ack。当接收到一个包，解析完得到sequence number与base比较，如果这时候的sequence number>base，说明接收方接收到了base自身以及之前的包，则可以滑动窗口，base+1。

即：

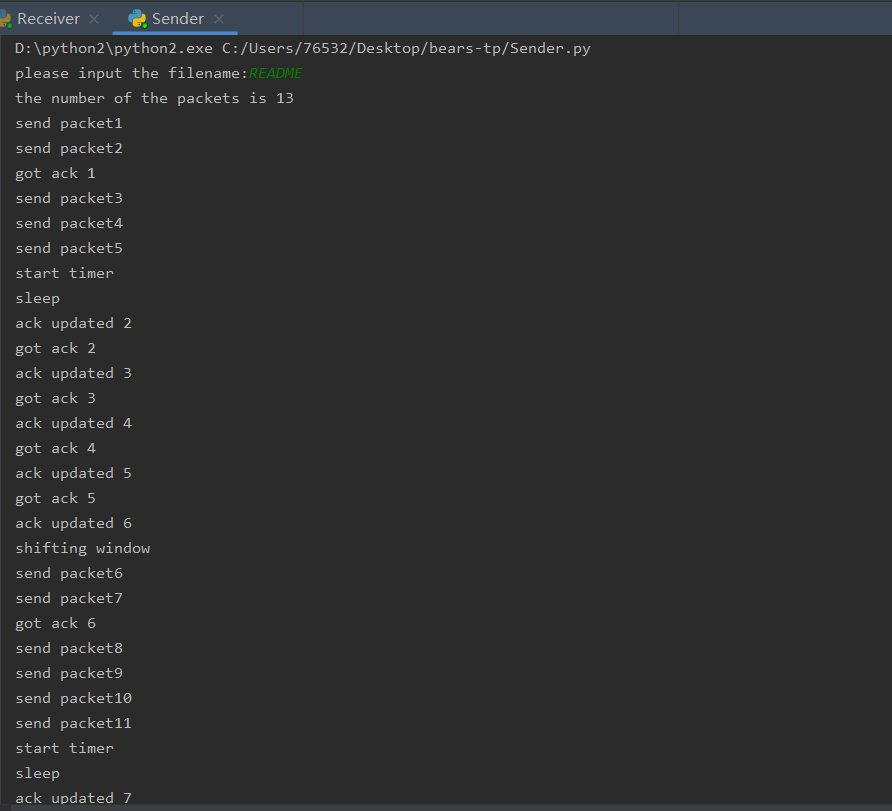
if(seqno > self.base):

mutex.acquire();

self.base = int(seqno) + 1

否则base不改变，继续发送base~base+4的包。

**6.运行截图**





**7.总结**

当使用线程的时候，总是会发生死锁的现象，不断地在修改，网上查阅中来回，才解决了这个问题。因为没引入多线程上锁开锁操作而导致的base值的错误导致发包错误，造成文件和原文件数据不同，在发送包时没有管理好下标而造成溢出的情况，这些问题都是通过多次的修改调试解决。