

**计算机网络**

**课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 可靠数据传输协议-停等协议和GBN协议的实现 | | | | | |
| 姓名 | 王科龙 | | 院系 | 计算机科学与技术 | | |
| 班级 | 1803104 | | 学号 | 1180801203 | | |
| 任课教师 | 聂兰顺 | | 指导教师 | 聂兰顺 | | |
| 实验地点 | 格物207 | | 实验时间 | 2020/11/7 | | |
| 实验课表现 | 出勤、表现得分(10) |  | 实验报告  得分(40) |  | 实验总分 |  |
| 操作结果得分(50) |  |
| 教师评语 | | | | | | |
|  | | | | | | |

****

|  |
| --- |
| 实验目的： |
| （注：实验报告模板中的各项内容仅供参考，可依照实际实验情况进行修改。）  本次实验的主要目的。  1.理解可靠数据传输的基本原理；掌握停等协议的工作原理；掌握基  于UDP 设计并实现一个停等协议的过程与技术。  2. 理解滑动窗口协议的基本原理；掌握GBN 的工作原理；掌握基于  UDP 设计并实现一个GBN 协议的过程与技术。 |
| 实验内容： |
| 概述本次实验的主要内容，包含的实验项等。  1) 基于UDP 设计一个简单的停等协议，实现单向可靠数据传输（服  务器到客户的数据传输）。  2) 模拟引入数据包的丢失，验证所设计协议的有效性。  3) 改进所设计的停等协议，支持双向数据传输；（选作内容，加分  项目，可以当堂完成或课下完成）  4）基于所设计的停等协议，实现一个C/S 结构的文件传输应用。（选  作内容，加分项目，可以当堂完成或课下完成）  1) 基于UDP 设计一个简单的GBN 协议，实现单向可靠数据传输（服  务器到客户的数据传输）。  2) 模拟引入数据包的丢失，验证所设计协议的有效性。  3) 改进所设计的GBN 协议，支持双向数据传输；（选作内容，加分  项目，可以当堂完成或课下完成）  4）将所设计的GBN 协议改进为SR 协议。（选作内容，加分项目，  可以当堂完成或课下完成） |
| 实验过程： |
| 以文字描述、实验结果截图等形式阐述实验过程，必要时可附相应的代码截图或以附件形式提交。   1. 包格式      1. 停等协议存在发送方和接受方，同时发送方要接受ack来检查。但是因为要实现双向传输，所以可以将发送方和接受放到一个类里，然后实例化是实例同一个类，但是调用不同的函数实现发送和接受。      1. 为了防止接受ack和接受数据冲突，导致ack被接受数据的recvfrom接收到，从而无法进行确认，所以将接受ack和接受数据放到了一起，解析出来数据之后看是否是ack，是ack的话就转到ack的处理程序上去。      1. 模拟引入丢包，丢包有2种，1是丢失pkt，2是丢失ack。丢失pkt可以通过不接受数据模拟，丢失ack可以通过不发送ack模拟        1. 收发文件:由于原本的程序主要是通过在控制台输入文字来交互，并没有考虑收发文件的事情，因此重新写函数来做收发文件。一个要解决的重要问题是，什么时候socket停止接受文件，如果没有标志的话，接受数据的循环无法接受，在python中不关闭文件是无法将数据写入文件中的。   我的主要解决方案是:  Server端发送:   1. Server端发送’file’表示接下来要发送文件 2. Server端读取文件的大小和名字发给Client，让Client利用文件大小来确定是否读完的所有的数据 3. 发送真实的文件内容     由于一般的文件收发都是基于TCP的，需要保证不能有乱序分组出现，但是这个实验要求基于UDP，因此通过使线程睡眠来保证可以准时收到，按序发送  缺点：由于make\_pkt只能打包字符串数据，因此无法收发图片  Client接受：     1. Gbn类的设计思路和停等类相同，只不过遵循的逻辑是gbn的收发逻辑。   发送方通过移动窗口来控制已经发送未确认的分组，保证这些分组收到ack或者是超时重发所有已发送但是未确认的分组。    接受方拥有一个expect\_seq\_num，只有发送的序号和expect\_seq\_num对上了才会接受数据  readable, writeable, errors = select.select([self.socket, ], [], [], 1)  if len(readable) > 0:  is\_rcv = random.random()  byte, addr = self.socket.recvfrom(1024)  message = byte.decode().split()  ret\_seq = int(message[0])  if len(message) <= 1 or 'ack' != message[1]: # 分开接受ack和接受数据  if is\_rcv < 0.3: # 接受之后不处理数据  print("refuse\_pkt", ',content=', message)  return  # print(self.name, 'receive data,seq=', ret\_seq)  if ret\_seq == self.except\_seq\_num:  self.ack = make\_ack(self.except\_seq\_num) # 构造返回的ack  print(self.name, ' receive expected data,seq=', self.except\_seq\_num, ',content=\"', message, '\"')  self.except\_seq\_num = (self.except\_seq\_num + 1) % MAX\_SEQ\_NUM  else:  print('NO expected data,content=', message)  is\_ack = random.random()  if is\_ack < 0.5: # 有0.5的概率不发送ack报文  self.socket.sendto(self.ack, (self.remote\_ip, self.remote\_port))  else:  print('no\_ack=', is\_ack, 'ack\_message=', self.ack)  else:  self.base = ret\_seq + 1  print(self.name, 'receive ACK,seq=', ret\_seq, 'base=', self.base, 'next\_seq\_num=',  self.next\_seq\_num)  if self.base == self.next\_seq\_num:  self.timer = -1 # 收到ack之后后面没有未确认的数据，停止计时器  else:  self.timer = 0 # 收到ack之后，后面还有未确认的数据，重启计时器 else:  if self.next\_seq\_num > self.base: # 读不到资源(ack)并且存在未确认的数据  self.timer = self.timer + 1  if self.timer > MAX\_TIMER:  self.timeout()  7）sr协议不再超时之后发送所有的未确认分组，我的程序里超时之后只发送第一个未确认的分组，同时对于乱序收到的分组，会将其先缓存下来。每收到一个分组之后，接受方的rcv\_base就会移动到第一个目前没有收到分组的序号处，同时将这中间乱序收到没有上交的数据一并上交。 |
| 实验结果： |
| 采用演示截图、文字说明等方式，给出本次实验的实验结果。  1)文件收发        2)gbn双向通信和丢包演示  丢包      同时server也是可以向client发送数据的      3)sr协议效果展示：  发送端快速发送，但是由于丢包使得窗口满：    接受方：    由于server没有返回0ack，使得发送方0分组超时，在这之前，接受方又缓存了1号分组和3号分组，因为0，1号分组都已经接受，可以进行提交，没有收到2号分组，在发送方2分组超时重发，接受方收到2号分组之后，将2号和3号分组一起上交。rcv\_base移动到4 |
| 问题讨论： |
| 对实验过程中的思考问题进行讨论或回答。   1. 线程的阻塞问题，由于recvfrom函数默认是阻塞的，如果没有从系统内核中拿到数据，就会一直阻塞其他进程，导致发送进程也无法进行发送。   解决方法是通过selcet函数去查看socket句柄是否有变化，有变化，存在可读取数据之后在去读取数据。 |
| 心得体会： |
| 结合实验过程和结果给出实验的体会和收获。  要想完整地写完一份代码，要求要又尽量少地错误，一份伪代码或者是流程图之类的东西是必要的，这些可以准确的描述要实现的代码逻辑，将这些东西只放在脑子里是不可靠的，有时候忘记写了一句导致了bug，然后这个bug就有可能花费很长时间去寻找，但是其实上是可以避免的。  通过对于停等协议，gbn协议和sr协议的实现，我对可靠传输数据原理有了更加深刻的认识。 |