

**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-GBN协议的设计与实现 | | | | | |
| 姓名 | 徐柯炎 | | 院系 | 计算机科学与技术学院 | | |
| 班级 | 2103602 | | 学号 | 2021110683 | | |
| 任课教师 | 刘亚维 | | 指导教师 | 刘亚维 | | |
| 实验地点 | 格物207 | | 实验时间 | 2023.10.28 | | |
| 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： |
| 1. 理解可靠数据传输的基本原理 2. 理解滑动窗口协议的基本原理 3. 掌握停等协议的工作原理 4. 掌握GBN 的工作原理 5. 掌握基于UDP 设计并实现一个停等协议的过程与技术 6. 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： |
| 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. 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）。 3. 模拟引入数据包的丢失，验证所设计协议的有效性。 4. 改进所设计的停等和GBN协议，支持双向数据传输； 5. 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. 6. 将所设计的GBN 协议改进为SR 协议 |
| 实验过程： |
| 1. GBN客户端（停等协议只需将GBN协议窗口中大小改为1）   进入客户端功能之后，当匹配到输入的是“-time”或者“-quit”则作为数据包发送给服务端，向服务端获取当前时间或者结束测试。    而如果匹配到的是“testgbn [X][Y]”的话（X为包丢失率，Y为ACK丢失率），则开始初始化数据包。将“testgbn [X][Y]”这段报文发到服务端（服务端也会有相应函数操作）。在等待服务端回复设置为UDP为非阻塞模式的时候，客户端发起握手连接给服务端，两者进入三次握手阶段，接着握手完了以后进入接收数据阶段。分别记为stage 0和stage 1，接下来将具体阐述功能。    第一个stage 0是等待握手阶段。如果服务端发回来的报文的seq匹配的话，就说明准备文件传输，并初始化recvSeq（上一个收到的序列号）和waitSeq（我希望收到的序列号），并转到阶段1。    第二个stage 1是等待接受数据阶段。这里面我们引入随机变量b，模拟数据包丢失的情况。如果没有丢失，那么判断是不是期望的数据包，即waitSeq – Seq是否为0，如果是期望收到的序列号，我们将缓存内容拷贝一份到recvpaper（存放最终报文）中，并更新recvNum（总共收到的报文数）、waitseq和recvSeq，并发送确认报文（包括我确认收到的序列号、我一共收到了多少packet）。    如果不是我期望收到的数据包，当上一个收到的序列号为0时不发送ACK，除此之外需要返回ACK。不过这次返回的ACK的数据包里面要返回上一个收到的序列号，也就还是原来的序列号，还要返回收到的报文的数量。  接下来是模拟丢失ACK的过程，最后发送ACK给服务端。     1. GBN服务端（停等协议只需将GBN协议窗口中大小改为1）   首先是服务器端的初始化套接字和设置非阻塞模式。在服务端运行之后，接着会监听端口地址，监听到客户端发送的命令执行函数，如果是“-time”和“-quit”则执行返回时间并打包发给客户端和退出程序。    如果收到的是“-testgbn [X][Y]”的话，就会进入三次握手阶段，握手阶段过后就是数据发送阶段。Stage 0和stage 1是握手阶段，stage 2是数据传输阶段。    在stage 0也就是连接允许阶段，服务器向客户端发送一个 205 大小的状态码表示服务器准备好了，可以发送数据。    在stage 1也就是接收ACK阶段，服务端首先调用recvfrom函数，接收到来自客户端的握手报文，如果recvSize小于0则说明该数据包丢失或者有错误，此时waitCount++（即等待时间加一）当waitCount超过3次，则会提示超时。如果收到的是我们想要的握手的数据包，那就会初始化curSeq（现在的序列号），curAck（现在确认的ack），并重置waitCount，并转到阶段2。    在stage 2中也就是数据传输阶段。服务器首先调用seqIsAvailable函数来确认是否有剩余的窗口，如果有剩余，则说明还能继续发送新的分组，然后将序列号加一存入buffer中，将分组的长度送给buffer，接着将报文传送给客户端。然后对当前序列号加一并对SEQ\_SIZE取模并得到新的curSeq。    接下来是等待ACK，如果没有收到ACK，也就是没有收到报文，则会返回-1，并且计时器+1，直到计时器超过10（即10次没有收到），则会重传刚刚的数据包。如果等到了想要的ACK，则会重置计时器，并且如果buffer[1]和报文总组数相等的话，数据传输就完成了。     1. 模拟引入丢包   在这里我们在客户端实现模拟丢包。首先用lossInLossRatio函数来产生随机数模拟丢包。这个函数产生一个一百以内的随机整数，如果这个数小于我们指定的数，则返回true，否则返回false。    将返回的bool值存入b中。接着如果b为真的话，就打印提示信息并模拟丢包，也就是继续执行下一个循环，如果b为假，则继续执行本次循环的后面部分。  ACK丢失也是如此，这里不过多阐述。     1. 支持双向数据传输   这里采用线程函数来实现客户端和服务器的双向传输。本质上就是在一个程序中既运行客户端又运行服务器。如下图所示。可以看到这个线程执行客户端的功能。    主函数如下，可以看出，在进行玩初始化后，主函数执行服务器函数而线程函数来执行客户端函数，利用多线程的方法来实现了双向传输的功能。     1. 实现C/S结构的文件传输应用   如下图所示，服务器端通过open函数来打开测试文件，并读取其中的内容发给客户端。    在传输完成之后，客户端通过printf函数将服务器发送的报文打印在屏幕上，并且将报文保存到本地，如下图所示。     1. SR客户端   SR客户端和GBN客户端的区别在于GBN没有接收窗口的缓存，接收到不是期望的seq就直接丢弃，而SR会将后面的seq保存在接收窗口中，这样服务器重传只需重传丢失的包。    当然如果收到的序列号和我期望收到的序号相同那么就让窗口基数++。如果在窗口基数后面有已经接收并保存下来的包，则将窗口向前一格，如此循环，直到当前期望收到的包还没有到。     1. SR服务端   SR服务端和GBN服务端的区别在于GBN期望收到的ACK只有一个，而SR可以期望收到多个ACK，这个ACK范围在send\_base到send\_base + SEND\_SIZE。在接受ACK阶段，程序会遍历期望收到的ACK，查看哪些发送的数据包还没有接收到，然后对应分组i的计时器加一，计时器的值超过三则重新发送该分组。    当然如果收到了ACK，则会对第i个分组的ACK进行处理，也就是将当前的ACK设置为true。 |
| 实验结果： |
| **必做功能：**   1. 获取时间：      1. 退出客户端：      1. 单向传递：   开始进行数据传输；    结束数据传输。    **选做功能：**   1. 模拟数据包丢失：   数据包丢失：    ACK丢失：     1. 双向数据传递：   客户端向服务器传输文件：       1. 文件传输：   左边re\_test.txt为接收到的文件，右边test.txt为发送的文件。     1. SR协议：   图中显示了当前的接收窗口和发送窗口。 |
| 问题讨论： |
| 1. GBN流程的一些讨论   一开始对GBN客户端和服务端的流程不是很了解，于是查阅相关资料得到了如下图所示的流程：    有了这张图之后就好了很多，在写程序时思路也更加清晰；   1. 关于如何模拟丢包的讨论   实验中采用对某数取余，当余数为0时则视为该数据分组丢失，不返回确认ACK；  客户端对收到的数据帧个数计数，当对某数取余后余数为0时则不返回确认ACK从而模拟丢包现象；   1. 关于双向传输的一些讨论：   一开始想用让用户自己选择的方法，也就是在两个程序上都实现客户端和服务器的功能然后自行选择是用哪一种功能来进行双向传输，但这种双向是半双工的；后来发现可以利用多线程的方法来实现双向传输，这样就实现了全双工的双向传输。   1. GBN（停等）协议数据分组格式、确认分组格式、各个域的作用   在以太网中，数据帧的MTU为1500字节，所以UDP数据报的数据部分应小于1472字节（除去IP头部20字节与UDP头的8字节），为此，定义UDP数据报的数据部分格式为：    Seq为1个字节，取值为0~255，（故序列号最多为256个）；  Data≤1024个字节，为传输的数据；  最后一个字节放入EOF0，表示结尾。  ACK数据帧定义：    由于是从服务器端到客户端的单向数据传输，因此ACK数据帧不包含任何数据，只需要将ACK发送给服务器端即可。ACK字段为一个字节，表示序列号数值。末尾放入0，表示数据结束。   1. GBN（停等）协议典型交互过程   **服务端：**   1. 开启端口访问，等待客户端发起请求。 2. 收到客户请求消息后，读取请求数据的内容，构造数据包并准备发送，开启新的发送数据报端口，在新端口上向客户端进行发送数据报同时开启计时器。 3. 接受客户端返回的ACK确认数据报，根据序列号判断选择滑动窗口发送帧还是重传操作。   **客户端：**   1. 向服务端发送请求，然后等待服务端反馈； 2. 接收到服务端数据报后进行判断是否为当前欲接收的数据报，是则接收，不是则丢弃，然后返回ACK确认。   典型交互中，出现的情况有发送的数据报丢失、超时或收到重复ACK导致重传；   1. 程序实现的主要类（或函数）及其主要作用   void timeoutHandler();//超时重传处理函数，滑动窗口内的数据帧都要重传  void ackHandler(char c);//收到 ack，累积确认，取数据帧的第一个字节  bool seqIsAvailable();//当前序列号curSeq是否可用  void printTips();//打印提示信息  BOOL lossInLossRatio(float lossRatio);//根据丢失率随机生成一个数字，判断是否丢失,丢失则返回 TRUE，否则返回 FALSE  void getCurTime(char\* ptime);//获取当前系统时间，结果存入 ptime 中   1. UDP编程主要特点    1. udp发送和接收没有缓冲区，发送和接收都是整包，自动保持包的边界    2. udp包的发送和接收不保证一定成功，不保证按正确顺序抵达    3. 在接收udp包时，如果接收包时给定的buffer太小的话，会发生异常，要捕获异常，相应调整buffer的大小，和给出反馈信息。    4. 如果不允许丢包的情况出现的话，要有重发机制来保证，如：每发一条信息，只有收到正确的ack的时候，才证明成功 |
| 心得体会： |
| 通过本次实验，我理解了滑动窗口协议的基本原理，掌握了GBN的工作原理，并且掌握了基于UDP设计并实现了一个GBN协议的过程和技术，了解了UDP编程的主要特点，增强了多线程编程的能力，加深了对传输层协议的理解。 |