

**计算机网络**

**课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 可靠数据传输协议-停等协议的设计与实现  可靠数据传输协议-GBN协议的设计与实现 | | | | | |
| 姓名 | 张瑞 | | 院系 | 计算机科学与技术 | | |
| 班级 | 1903104 | | 学号 | 1190201421 | | |
| 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 | 2021年11月7日 | | |
| 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： |
| 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。  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： |
| 停等协议的设计与实现：  (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。  (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。  GBN协议的设计与实现：  (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。  (4)将所设计的GBN协议改进为SR协议。 |
| 实验过程： |
| 本次实验虽然看上去要实现三个协议，但其实三者相似性较高：只要把GBN协议中的发送方窗口大小改为1即可得到停等协议，将GBN协议中的接收方增添缓存功能即可得到SR协议，于是本次实验可以从GBN协议下手。  (1)数据分组格式、确认分组格式及各个域作用：  (a)数据分组格式：    Seq：为 1 个字节，取值为 0~255，（故序列号最多为 256 个）；  Data：小于等于1024个字节，为传输的数据；  0：最后1个字节放入EOF，表示结尾。  (b)确认分组格式：    ACK：为1个字节，表示所确认数据包的序列号数值；  0：末尾1个字节放入0，表示数据结束。  (2)两端程序流程图:  (a)停等协议与GBN协议：  可以将停等协议视为特殊的GBN协议，当发送方窗口数量N=1时，GBN协议就变成了停等协议。  发送端：    接收端：    (b)SR协议：    (3)典型交互过程：  (a)停等协议：    (b)GBN协议：    (c)SR协议：    (4)分组丢失验证模拟方法：  为了模拟数据分组和确认分组的丢失，统一在接收方接收到分组之后（因为在发送方和接收方丢失的效果实际是等价的，实验中我选择的是在接收方丢弃）按照预设的丢包率随机丢弃掉一部分分组。具体的实现方式是每收到一个分组便产生一个随机数，比较该随机数与事先设定的丢包率的大小，若随机数更大，则正确接收该分组；反之丢弃分组，达到分组丢失的效果。  (5)程序实现的主要类（或函数）及其主要作用：  (a)BOOL lossInLossRatio(float lossRatio)：根据丢失率随机生成一个数字，判断分组是否丢失。  (b)bool seqIsAvailable()：判断当前序号是否在发送窗口内（是否可用）。  (c)void timeoutHandler()：超时重传处理函数。若为停等协议，重传前一个分组；若为GBN协议，滑动窗口内的分组都要重传；若为SR协议，仅重传超时的分组。  (d)void ackHandler(char c)：分组确认函数。若为停等协议，则确认前一个分组；若为GBN协议，累计确认；若为SR协议，逐个确认。  (e)int main(int argc, char\* argv[])：主函数，实现客户端与服务器端的数据传输。  (6)UDP编程的主要特点：  (a)无连接，主机之间不需要维持复杂的链接状态，减少延迟；  (b)不可靠，不能保证数据一定到达对方，也不能保证数据的按序到达；  (c)头部开销小；  (d)无拥塞控制；  (e)需要额外的机制实现可靠数据传输，如在应用层增加可靠性机制或应用特定的错误恢复机制。  (7)双向数据传输：  实现双向数据传输其实就是将服务器端的发送功能加给客户端，再将客户端的接收功能加给服务器端。所以，这一功能很大程度上可以通过对彼此代码的复用而迅速实现。但是需要注意Socket套接字阻塞和非阻塞模式的转换，作为数据的发送方时，套接字要设为非阻塞模式，但作为接收方时，套接字需要设为阻塞模式。  (8)C/S结构的文件传输应用：  发送方首先将需要发送的数据从文件中读入内存，在收到请求后，将这些数据按事先设定好的数据包的大小分组后，发送给接收方，并按协议（停等协议、GBN协议或SR协议）对分组计时与确认，最后一个分组确认后，发送“good bye\0”表示数据传输结束。  接收方将收到的期望的分组交付给上层（如果是SR协议，对乱序到达的分组要先缓存，等期望的基序号分组到达后，再将与其连续的分组一起上传），将内容输出到文件中。  (9)SR协议：  SR协议相对GBN协议有一些变化。首先是接收方窗口的变化，接收方不再是一个窗口，而是多个窗口，这样一来就能对乱序到达的分组实现缓存，等期望的基序号分组到达后，再将与其连续的分组一起上传，并进行窗口的滑动。为了实现这一功能，我给接收方添加了recv\_buffer来缓存乱序到达的分组。然后就是对各个分组的计时应独立进行，每当有分组超时时，只重发该超时分组，其他窗口内未收到确认的分组需要继续计时至超时才重发，而不是像GBN中那样一起重发。为了实现这一功能，我给ack数组，用于标记各分组是否被确认：未确认的是0，确认的是1，重发的是-1。一旦发生超时事件，将发送方窗口基序号定位至currentAck，并按序遍历窗口内序号对应的ack数组值，若为1，代表已确认；若为-1，代表是超时重传；若为0，代表首次发送，需要在发送后将值改为-1，等下次在确认前再遍历到该值时，表明分组要重传。由于遍历时是对窗口内序号按序遍历，于是可以将一次遍历处理和一次计时等价看待，计时就简化为对第一个超时分组的计时，但最终效果仍然实现的是对各个分组的分别计时。  (10)详细注释源程序：  详见随报告一同上传的源代码文件夹 |
| 实验结果： |
| (1)停等协议：  (a)服务器到客户的数据传输（默认丢包率和ACK丢失率均为0.2）：      (b)客户到服务器的数据传输（默认丢包率和ACK丢失率均为0.2）：      (c)C/S结构的文件传输应用：  (a)中命令执行的同时，客户即得到了receive.txt，打开与send.txt对比，发现完全一致，说明文件传输应用已成功实现。    (2)GBN协议：  (a)服务器到客户的数据传输（默认丢包率和ACK丢失率均为0.2）：        (b)客户到服务器的数据传输（默认丢包率和ACK丢失率均为0.2）：      (c)改进为SR协议： |
| 问题讨论： |
| (1)实现双向传播时，若只是简单复用对方代码，会有错误出现，后来发现是未修改套接字的阻塞与非阻塞模式导致的问题。  (2)GBN协议和SR协议收发双方的窗口数量之和不得大于序列号总数，否则会出现歧义，难以判别数据包的先后顺序，导致数据传输出错，一定要正确设置。  (3)实现SR协议时，将各分组的分别计时简化为对第一个超时分组计时，对后续超时分组在遍历重发中同步计时，也能实现相同的效果。 |
| 心得体会： |
| 通过本次实验我更加深入地理解了停等协议、GBN协议和SR协议的实现细节，对其流程、逻辑和注意事项（窗口加载、发送、重发、ack确认、窗口滑动、计时器创建与超时，需要的包的判定等等）有了更深的理解和体会。  同时我也对阻塞和非阻塞有了进一步的认识，也练习了文件读写操作。在调试代码的过程中也再次锻炼了自己的debug能力。 |