

**计算机网络**

**课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 可靠数据传输协议的设计与实现 | | | | | |
| 姓名 | 张智雄 | | 院系 | 计算学部 | | |
| 班级 | 2103601 | | 学号 | 2021112845 | | |
| 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；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： |
| 1. 基于UDP设计一个简单的停等和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. 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。 4. 改进所设计的GBN协议，支持**双向**数据传输； 5. 将所设计的GBN协议**改进为SR协议**。 |
| 实验过程： |
| 1. **GBN协议可靠数据传输**   当设置服务器端发送窗口的大小为 1 时， GBN 协议就是停-等协议。  **1.服务器端Server**  使用UDP协议传输数据（比如传输一个文件），等待客户端的请求，接收并处理来自客户端的消息（如数据传输请求）。  在服务端运行之后，首先会初始化套接字并绑定端口地址监听（**非阻塞接收**），监听到客户端发送的命令执行函数，当接收到“-time”或者“-quit”报文时，则执行返回时间并打包发给客户端和退出程序。  图形用户界面, 文本, 应用程序  描述已自动生成  当客户端开始请求数据，即服务器端接收到“-testgbn”报文的话，就会进入握手阶段，建立连接状态后（并不是真正的连接，只是一种类似连接的数据发送的状态），将数据打包成数据报文发送，然后等待客户端的ACK信息，同时启动计时器。  **握手阶段：**首先服务器端（服务器端处于0状态）向客户端发送205状态码（服务器端进入1状态），而后服务器端调用recvfrom()等待客户端回复200状态码。  如果收到（服务器端进入2状态），初始化现在的序列号curSeq，现在已确认报文序列号curAck，计时器waitCount，开始传输文件；否则延时等待直至超时，超时则放弃此次“连接”，等待从第一步开始。  图形用户界面, 文本, 应用程序  描述已自动生成  **传输阶段：**服务器端首先将数据打包成数据报文，调用函数seqIsAvailable()函数判断序列号是否在当前发送窗口之内，然后按数据帧格式将数据包发送给客户端。发完之后curSeq加1并对SEQ\_SIZE取模得到新的curSeq。  文本  描述已自动生成  而后调用recvfrom()非阻塞等待接受ACK。当收到ACK时，调用ackHandler()函数进行窗口滑动（**累积确认**），正常发送下一个数据报，计时器重新计时；若在计时器超时前没有收到ACK，则调用超时重传函数timeoutHandler()**全部重传窗口内**的所有已发送的数据报。  计时器的实现方式为：设置套接字为非阻塞模式，服务器端在recvfrom()方法上不会阻塞。如果正确接收到ACK消息，那么计时器将被清零。但如果从客户端接收到数据长度为-1，这表示没有接收到任何数据，那么计时器将增加。对计时器进行判断，如果计时器超过了阈值，就被认为发生了超时，此时会执行超时重传操作。  文本  描述已自动生成  文本  描述已自动生成  最后当序列号curSeq等于数据发送的总包数totalPacket时停止发送新的报文，而当收到的ACK等于总包数totalPacket时传输结束，并向客户端发送状态码同步传输结束。  **2.客户端Client**  使用UDP协议向服务器端请求数据，接收服务器端发送的数据报并返回确认信息ACK（注意GBN为**累积确认**，即客户端返回当前接受**有序最新**数据包的确认ACK）。  客户端首先初始化套接字并绑定端口地址监听（**阻塞接收**），使用命令行输入指令，当输入为“-time”或者“-quit”则直接作为数据包发送给服务端（查询时间以及退出）。  文本  描述已自动生成  如果输入为“-testgbn\_recv [X][Y]”，则进入等待握手阶段（客户端处于0状态），当收到来自服务器的205状态码后，向服务器发送200状态码，同时初始化已收到的序列号recvSeq为0和等待收到的序列号waitSeq为1，进入等待传输阶段（1状态）  图形用户界面, 文本, 应用程序, 电子邮件  描述已自动生成  而后调用recvfrom()等待来自服务器的数据报文，如果是期待的包，正确接收，正常确认即可；否则就**丢弃此数据包**，返回上一个正确接收的包的序列号recvSeq的ACK。如果当前一个包都没有收到，则等待Seq为1的数据包，不是则不返回ACK（因为并没有上一个正确的ACK）。  文本  描述已自动生成  最后当接收到服务器245状态码时（表示传输完毕）结束接收，等待下一条用户指令的输入。同时清除接收缓存RecvMessage和报文缓存buffer，避免下次传输出错。    **二、模拟数据包丢失**  在客户端中引入默认包丢失率packetLossRatio和默认 ACK 丢失率ackLossRatio来模拟引入数据包的丢失和ACK丢失的情况。  模拟的思路是：引入b作为随机变量，其值对应lossInLossRatio()函数的返回值，当接受到报文和发送ACK之前均对b进行判断，如若为TRUE，则跳过当前接受数据循环，直接进入下一次循环，在阻塞态下，即为接受下一个数据包；反之则正常进行数据接受和反馈。  如若报文丢失，recvSeq和waitSeq不会变化；ACK丢失，recvSeq和waitSeq正常移动加一。但两种情况下，后续服务器会进行超时重传处理。  lossInLossRatio()函数根据丢失率随机生成一个数字，判断是否丢失，如若丢失则返回TRUE，否则返回FALSE。    **三、基于C/S结构的文件传输**  为实现基于C/S结构的文件传输，**服务端**首先要将测试文件test.txt内容读入拷贝到缓存，计算出要发送的总包数totalPacket，而后将序号组的ACK设为TURE（表明该分组还没发送成功即没有收到相应ACK）。    客户端在收到服务端发送的数据包之后，如果是期望收到的，则将收到的数据缓存到recvMessage中。在接收到245状态码结束传输之后写入文本文件output.txt。  **文本  描述已自动生成**  **四、双向数据传输**  为实现双向数据传输，在客户端界面上增加一个选项为“-testgbn\_send”，输入后客户端会向服务器端发送“-receive”报文，而后将扮演服务器的角色项服务器端发送数据。经历握手、数据传输等阶段。    客户端收到“-receive”报文后，则进入接收信息的状态，经历等待握手、等待数据传输、数据传输完毕的各个阶段。  总体上看，就是合并了客户端和服务端的代码。需要注意的即是**不用换用两个套接字，但是需要在调换角色的同时更改套接字的阻塞状态**（即作为服务器端需要非阻塞态，客户端采用阻塞态）。  **五、SR协议的改进**  **1.服务器端Server**  SR协议的服务端和GBN协议的服务端的最主要区别在于，GBN只有当收到对应当前发送窗口最左侧send\_base数据包的ACK时才会处理并进行窗口滑动。  而SR可以收到并处理对应发送窗口（send\_base到send\_base+SEQ\_SIZE）内所有数据包的ACK，而当收到对应当前发送窗口最左侧send\_base数据包的ACK时进行窗口滑动直至第一个未确认接收ACK的位置。  在接受ACK阶段，如果发生了丢包或者其他错误，程序会遍历期望收到的ACK，查看哪些ACK是还没有收到（即没有被确认），然后对应分组i的计时器++，超过8次则重新传送分组。    而如果收到ACK，则会对第i个分组的ACK位置标志设为TRUE（在读入数据时将所有ACK设为FALSE以区分是否接收），同时窗口基址send\_base向右移动到当前未确认的第一个位置。    而在结束传输上，与GBN类似，当发送包序列号超过总包数totalPacket时就拒绝发送，而当发送窗口移动到totalPacket右侧时即传输结束。  **2.客户端Client**  SR协议的客户端和GBN协议的客户端的区别在于GBN只能接受当前等待接收的数据包（**接收一定有序**）同时接受窗口向右移动，否则就丢弃。  而SR客户端可以接受在接收窗口内任意的数据包（**接收不一定有序**），同样只有当收到对应当前发送窗口最左侧recv\_base数据包时才会进行窗口滑动，滑动到当前未接收的第一个序列位置。  具体实现上，同样设置一个接收序列数组recvack[]，如果收到预期消息，接收窗口右移；否则就将接受位置对应标志设为TRUE，缓存到recvMessage对应的位置中。同样采用默认包丢失率packetLossRatio和默认 ACK 丢失率ackLossRatio来模拟引入数据包的丢失和ACK丢失的情况。  图形用户界面, 文本, 应用程序, 电子邮件  描述已自动生成  **3.窗口大小**  滑动窗口协议中发送/接收窗口大小的设置：设序列号位数为，发送窗口大小，接收窗口为，则满足。SR协议中一般，原因是发送窗口大于接收窗口会导致溢出，小于则没有意义。  本实验中SR协议窗口大小均设置为8。 |
| 实验结果： |
| 1. **单向可靠数据传输—**GBN**协议**   同时运行客户端和服务器端代码，初始化套接字并对端口12340进行监听。   1. 查询时间测试连接   客户端向服务器端发送“-time”查询当前时间，返回结果如下：  D:\study\computernetwork\lab2\GBN\Client\Debug\Client.exe   1. 数据传输   客户端向服务器端发送“-testgbn\_recv”**请求**数据，建立连接后，传输结果如下：    可以看到数据发送过程中：   1. 序列号为4和8的数据包丢失了，在此过程中，客户端成功接收到了数据包5-7和9-12，但由于GBN协议不缓存乱序到达的数据包，这些数据包被直接丢弃，并且持续返回send a ack of 3。 2. 随后，服务器端触发了超时重传机制，重新发送窗口内所有数据（也就是数据包4-12）。此时，客户端重新接收了数据4-9，并进行了缓存，并发送了ACK4-9。这个过程发送和接收窗口正常移动。 3. 此时，数据包10再次丢失并再次触发超时，导致重发数据包10-12。这个重传过程会一直持续，直到接收方正确收到了ACK 12（ACK的序号达到TotalSeqNum）。   同时，由于GBN采用累计确认机制，客户端发送的ACK相当于确认此序号之前的所有数据都已正确接收，因此ACK丢失一般不影响数据传输（除非ACK连续多次丢失）。当ACK的序号达到TotalSeqNum时，表示已经成功完成了全部数据传输，连接随之解除。     1. 退出传输   客户端向服务器端发送“-quit”退出程序，返回结果如下：  Microsoft Visual Studio 调试控制台   1. **双向可靠数据传输—**GBN**协议**   客户端向服务器端发送“-testgbn\_send”**上传**数据，建立连接后，传输结果如下：    超时重传、数据和ACK的丢失与上述相同，此处不再分析。   1. **基于C/S结构的文件传输**   上述传输建立在文件读取的基础之上，传输结束后会将内容写入本地文件。即服务器端读入文件test.txt并传输，客户端正确接收并写入为output.txt。     1. **可靠数据传输—**SR**协议**   同时运行客户端和服务器端代码，初始化套接字并对端口12340进行监听。客户端向服务器端发送“-testsr”请求数据，建立连接后，传输部分结果如下：    由于原理相同，报告中只对截取部分进行分析。可以看到数据发送过程中：   1. 序列号为3的数据包丢失了，在此过程中，客户端成功接收到了数据包4-10，同时正常返回对应序列号的ACK（并不像GBN一样直接舍弃）。 2. 随后，服务器端触发了超时重传机制，重新发送数据包3。此时数据包4-10已被接收，但只有数据包4-6已确认，因而接收窗口滑动至11位置，而发送窗口滑动至7位置。 3. 此时，数据包7、8、9会再次触发超时重传。这个重传和窗口滑动过程会一直持续，直到发送窗口移到21右侧（即所有数据包均已确认收到）。   同时，由于SR不采用累计确认机制，客户端发送的ACK仅仅确认此序号数据被正确接收，因此数据包和ACK丢失都会导致数据传输。**但当ACK丢失时，接收窗口正常滑动，而发送窗口则会停留在该位置直至被确认。** |
| 问题讨论： |
| 停等协议是GBN协议的特殊情况，数据分组格式、确认分组格式、程序流程图等基本一致因而报告主要针对GBN协议作讨论（**停等协议只需将窗口的大小设置为1**）  1. GBN（停等）协议数据分组格式、确认分组格式、各个域的作用  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 | Data | 0 |   其中，为1个字节，取值为，（故序列号最多为个）；个字节，为传输的数据；最后一个字节放入，表示结尾。  **ACK数据帧**定义：   |  |  | | --- | --- | | ACK | 0 |   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。  2. GBN（停等）协议两端程序流程图    3. GBN（停等）协议典型交互过程   |  |  | | --- | --- | | **客户端（Client）** | **服务器端（Server）** | | 1. 根据目标服务器IP地址与端口号创建套接字，并连接服务器； | 1. 对到来的请求创建套接字，绑定套接字的IP地址和端口号，对端口进行监听； | | 1. 向服务端发送请求，然后等待服务端反馈； | 1. 收到客户请求消息后，读取请求数据的内容，构造数据包，向客户端进行发送数据报同时开启计时器； | | 1. 接收到服务端数据报后进行判断是否为当前期望的数据报，是则接收并返回此数据的ACK确认，否则丢弃，返回上一确认的数据对应的ACK。 | 1. 接受客户端返回的ACK确认数据报文，根据序列号判断是否滑动窗口发送帧，同时根据计时器对未确认数据段执行超时重传操作。 |     4. 数据分组丢失验证模拟方法  模拟的思路是：引入b作为随机变量，其值对应lossInLossRatio()函数的返回值，当接受到报文和发送ACK之前均对b进行判断，如若为TRUE，则跳过当前接受数据循环，直接进入下一次循环，在阻塞态下，即为接受下一个数据包；反之则正常进行数据接受和反馈。  如若报文丢失，recvSeq和waitSeq不会变化；ACK丢失，recvSeq和waitSeq正常移动加一。但两种情况下，后续服务器会进行超时重传处理。  lossInLossRatio()函数根据丢失率随机生成一个数字，判断是否丢失，如若丢失则返回TRUE，否则返回FALSE。具体代码见附件或实验过程部分。  5. 程序实现的主要类（或函数）及其主要作用   |  |  | | --- | --- | | 函数 | 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) | 收到累积确认ACK，取数据帧的第一个字节 | | bool seqIsAvailable() | 检查当前序列号curSeq是否在发送窗口内 | | void printTips() | 打印提示信息 | | BOOL lossInLossRatio(float lossRatio) | 随机模拟分组丢失 | | void getCurTime(char\* ptime) | 获取当前系统时间 |   6. UDP编程主要特点   1. UDP是无连接的，发送数据之前不需要建立连接，因此减少了开销和发送数据之前的时延。 2. UDP使用尽最大努力交付，不保证可靠交付，即可能丢失和乱序。需要引入可靠数据传输手段来确保数据传输正确。 3. UDP是面向报文的，UDP对应用层交下来的报文，既不合并，也不拆分，而是保留这些报文的边界，一次交付一个完整的报文。   7. 实验验证结果和详细注释源程序  见上文实验结果部分和附件。 |
| 心得体会： |
| 1. 本次对可靠数据传输协议的实现，更加深入的理解了滑动窗口协议，对GBN协议和SR协议有了更深的理解； 2. 对Socket编程有了进一步的学习和理解。 3. 对UDP协议有了进一步的了解，同时也更加理解了从UDP到TCP的一步步的探索过程，对运输层有了更为深刻的理解。 |