

**计算机网络**

**课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 可靠数据传输协议-GBN 协议的设计与实现 | | | | | |
| 姓名 | 汤添凝 | | 院系 | 计算机系 | | |
| 班级 | 1703201 | | 学号 | 1170300728 | | |
| 任课教师 | 李全龙 | | 指导教师 | 李全龙 | | |
| 实验地点 | 格物214 | | 实验时间 | 10/26 | | |
| 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 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： |
| 1 UDP协议  UDP 是User Datagram Protocol的简称， 中文名是用户数据报协议，是[OSI](https://baike.baidu.com/item/OSI" \t "https://baike.baidu.com/item/UDP/_blank)（Open System Interconnection，[开放式系统互联](https://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E5%BC%8F%E7%B3%BB%E7%BB%9F%E4%BA%92%E8%81%94/562749" \t "https://baike.baidu.com/item/UDP/_blank)） 参考模型中一种无连接的[传输层](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82" \t "https://baike.baidu.com/item/UDP/_blank)协议，提供面向事务的简单不可靠信息传送服务，IETF RFC 768 [1]  是UDP的正式规范。UDP在IP报文的协议号是17。  UDP协议与[TCP](https://baike.baidu.com/item/TCP" \t "https://baike.baidu.com/item/UDP/_blank)协议一样用于处理数据包，在OSI模型中，两者都位于[传输层](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82/4329536" \t "https://baike.baidu.com/item/UDP/_blank)，处于IP协议的上一层。UDP有不提供数据包分组、组装和不能对数据包进行排序的缺点，也就是说，当报文发送之后，是无法得知其是否安全完整到达的。UDP用来支持那些需要在[计算机](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA/140338" \t "https://baike.baidu.com/item/UDP/_blank)之间传输数据的网络应用。包括[网络视频会议](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E8%A7%86%E9%A2%91%E4%BC%9A%E8%AE%AE" \t "https://baike.baidu.com/item/UDP/_blank)系统在内的众多的客户/服务器模式的网络应用都需要使用UDP协议。UDP协议从问世至今已经被使用了很多年，虽然其最初的光彩已经被一些类似协议所掩盖，但即使在今天UDP仍然不失为一项非常实用和可行的网络传输层协议。  许多应用只支持UDP，如：多媒体数据流，不产生任何额外的数据，即使知道有破坏的包也不进行重发。当强调传输性能而不是传输的完整性时，如：音频和多媒体应用，UDP是最好的选择。在数据传输时间很短，以至于此前的连接过程成为整个流量主体的情况下，UDP也是一个好的选择。  2 GBN协议  2.1 现实应用中的GBN协议  GBN协议，全称后退N步协议，属于滑动窗口协议  2.1.1 滑动窗口协议的定义  传输的每个部分被分配唯一的连续序列号，接收方使用数字并以正确的顺序放置接收到的数据包，丢弃重复的数据包并识别丢失的数据。  协议中规定，对于窗口内未经确认的分组需要重传。这种分组的数量最多可以等于发送窗口的大小，即滑动窗口的大小n减去1（因为发送窗口不可能大于（n-1），起码接收窗口要大于等于1）。  滑动窗口协议必须保证数据包的按序传输，发送窗口中的序列号代表已发送但尚未收到确认的数据包，发送窗口可持续地维持一系列未经确认的数据包，因为发送方窗口内的数据包可能在传输过程中丢失或损坏，所以发送过程必须把发送窗口中的所有数据包保存起来以备重传。发送窗口一旦达到最大值，发送过程就必须停止接收新的数据包，直到有空闲缓存区。接收窗口外的数据包都要丢弃，当序列号等于接收窗口下限的数据包到达时，把它提交给应用程序并向发送端发送确认，接收窗口向前移动一位。发送窗口和接收窗口上下限无需相同，大小也无需相同，但接收窗口大小需保持固定，发送窗口大小可随着数据包而改变。  2.1.2 GBN协议的定义  GBN协议本身就是滑动窗口协议的一种，这里主要关注GBN协议中的具体特点。GBN协议中，发送方在发完一个数据帧后，连续发送若干个数据帧，即使在连续发送过程中收到了接收方发来的应答帧，也可以继续发送。且发送方在每发送完一个数据帧时都要设置超时定时器。只要在所设置的超时时间内仍收到确认帧，就要重发相应的数据帧。如：当发送方发送了N个帧后，若发现该N帧的前一个帧在计时器超时后仍未返回其确认信息，则该帧被判为出错或丢失，此时发送方就不得不重新发送出错帧及其后的N帧。  接受帧只允许按顺序接受帧。为了减少开销，累计确认允许接收端在连续收到好几个正确的确认帧后，只对最后一个数据帧发确认信息，或者可以在自己有数据要发送时才将对以前正确收到的帧加以捎带确认。这就是说，对某一数据帧的确认就表明该数据帧和这以前所有的数据帧均已正确无误地收到了。  后退N帧协议的接受窗口为1，可以保证按序接受数据帧。若采用n个比特对帧编号，则其发送窗口的尺寸应满足：1~2^(n-1)。若发送窗口的尺寸大于2^(n-1)，则会造成接受方无法分辨新帧和旧帧。  2.1.3 GBN协议交互流程图    2.2 本次实验要求的GBN协议  2.2.1 实验概述  作为只实现单向数据传输的 GBN 协议，实质上就是实现为一个 C/S应用。  服务器端：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），当客户端开始请求数据时进入“伪连接”状态（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发送，然后等待客户端的 ACK 信息，同时启动计时器。当收到 ACK 时，窗口滑动，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，则全部重传窗口内的所以已发送的数据报。  客户端：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，即若 ACK=1 和 3，表示数据帧 2 已经正确接收），必须能够模拟 ACK 丢失直至服务器端超时重传的情况。  2.2.2 本次实验的GBN数据分组格式  1、服务器端：  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），定义 UDP 数据报的数据部分格式为：   |  |  | | --- | --- | | SEQ | DATA |   Seq 为 1 个字节，取值为 1~20（因为设计的窗口大小为20）；  Data=1024 个字节，为传输的数据；   1. 客户端：   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1个字节，表示序列号数值；  末尾放入 0，表示数据结束。  2.2.3 数据分组丢失验证模拟方法  丢失模拟方式在客户端实现，如果数据包丢失，则客户端认为没有接受到；如果ACK丢失，则客户端不向服务器发送ACK数据包。  首先，用户输入“-testgbn [X] [Y]”中可以输入两个变量X和Y，分别对应数据包丢失概率和ACK丢失概率（若不输入，则认为X、Y均为0.2）。在lossInLossRatio函数，将填写的X或者Y丢失率作为参数，在该函数中在0-100中随机生成一个数，如果该数小于丢失率\*100则认为丢失。  验证方式，在服务器端将sever.txt文件作为要传输的数据，如果数据传输完成后在客户端收到的数据是准确无误的，则认为是GBN是可靠的。  2.2.4 代码核心函数及其作用  **2.2.4.1. void getCurTime(char \*ptime)**  函数功能：获取当前系统时间，结果存入 ptime 中，当客户端要求返回时间信息时使用。  **2.2.4.2. bool seqIsAvailable()**  函数功能：判断当前序列号 curSeq 是否可用，即看curseq是否在窗口内，是否已经发送过了。具体而言，函数通过检索序列号数组，判断curse的对应状态，如果是已发送或者已收到确认，则跳过，并判断窗口中下一个序列号是几。  **2.2.4.3. void timeoutHandler()**  函数功能：超时重传处理函数，滑动窗口内的数据帧都要重传。具体而言函数通过检索序列号数组，判断当前curse的状态。如果是已发送但未收到，则将其重置为待发送状态。  **2.2.4.4. void ackHandler(char c)**  函数功能：收到 ack，累积确认，取数据帧的第一个字节，并计算对应seq，在序列号数组中将对应元素修改为“已发送成功”。  **2.2.4.5 BOOL lossInLossRatio(float lossRatio)**  函数功能：根据丢失率[X][Y]随机生成一个数字，判断是否丢失。如果丢失则返回TRUE，否则返回 FALSE |
| 实验结果： |
| 双向SR协议的服务器端：  QQ图片20191105144752QQ图片20191105144809  双向SR协议的客户端界面：  QQ图片20191105144827QQ图片20191105144842  GBN协议的客户端界面  QQ图片20191105145813  GBN协议的服务器端界面  QQ图片20191105145822 |
| 问题讨论： |
| 问题：  实验中，当发送窗口到达序列号边界时，会发生停顿超时重传，例如20个序列号，而窗口中的最后一个序列号已经越过了20，到达了3，发送数据时会到20就停止而不继续发送1、2、3的数据块。  解决方案：  计算窗口时，只有加法而忘记了mod，导致窗口不继续搜索。增加一些判断数据传输是否到达序列号边界，添加判断mod语句即可解决问题。 |
| 心得体会： |
| 本次实验内容充实紧凑，且有一定难度与复杂度，但在老师、助教、同学以及个人自学的多重努力下成功完成了实验。  本次实验为“GBN协议的设计与实现”，以设计GBN协议和SR协议的形式逐步教会我们GBN协议和SR协议的含义细节，协议消息构造，滑动窗口的设计实现，效果出众。在此前，我从来没有考虑过诸如“滑动窗口的具体设计究竟如何设计数据结构？”、“GBN中到底应该设置几个状态？”这类问题，经过了这次实验后，我对这些GBN协议和SR协议的设计细节有了更深入的了解，在逐渐完成实验任务的同时，开拓了计算思维，以至于最后的测试很轻松地找到了漏洞并加以弥补。我想这大概就是身为“计算机科学有技术学院”的学生相比其他专业学生对于计算机认识更深入的地方之一了。 |