

**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，GBN、SR协议 | | | | | |
| 姓名 | 姜昕阳 | | 院系 | 计算学部 | | |
| 班级 | 2203201 | | 学号 | 2022111742 | | |
| 任课教师 | 詹东阳 | | 指导教师 | 詹东阳 | | |
| 实验地点 | 格物207 | | 实验时间 | 2024.4.10 | | |
| 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。  5) 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）。  6) 模拟引入数据包的丢失，验证所设计协议的有效性。  7) 改进所设计的GBN 协议， 支持双向数据传输；  8）将所设计的GBN 协议改进为SR协议。 |
| 实验过程： |
| UDP编程特点：  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,既不保证可靠交付,因此主机不需要维护复杂的连接状态表  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.UDP一次交付一个完整的报文  4.UDP没有拥塞控制,因此网络出现拥塞不会使源主机的发送速率降低,这对某些实时应用很重要的,很是个多媒体通信的要求  5.UDP支持一对一,一对多,多对一和多对多的交互通信  GBN协议描述：  工作原理：  流程图：  6ecac75503e5e425c5b3744a66c9fdc  58c692a7e1a79817784bf406a24b3e3  发送端：   1. 发送数据：将当前窗口内所有数据包通过socket传输给接收方； 2. 超时重传：每一个窗口都设置一个计时器，若正确接收到ack则重置计时器，否则等待计时器超时后，重传全部窗口数据包； 3. 收到ack：如果ack序号大于窗口base，则移动窗口至ack+1，反之等待计时器超时进行重传。   接收方：  1.接收数据：通过套接字将发送方发来的报文解析出来，并对应发送ack；  2.发送ack：根据接收到的报文序号发送ack  代码描述：  客户端（GBNClient）：  1. 初始化：  创建一个UDP套接字（DatagramSocket）用于与服务器通信。  初始化变量，包括序列号、基序列号、期望序列号、窗口大小等。  2. 数据发送：  在一个循环中，发送数据包给服务器端。  如果某个数据包模3为0，即被模拟为丢失，不发送该数据包。  发送数据包时，序列号递增，同时更新基序列号，如果基序列号未发送的数据包数小于N，则继续发送。  如果发送的数据包是基序列号，启动计时器，等待服务器端确认。  3. 数据接收和处理：  接收来自服务器端的确认包或数据包。  如果收到确认包，则更新基序列号，停止计时器（如果所有未确认的数据已经收到确认）。  如果收到数据包，则检查序列号是否是期望序列号，如果是，则将数据写入文件，并发送确认包，期望序列号递增。  如果收到的数据包不是期望序列号，则发送之前期望的确认包。  4. 超时重传：  如果定时器超时，未收到服务器端的确认包，则启动超时重传机制。  重新发送大于确认ack的所有分组  服务器端（GBNServer）：  1. 初始化：  创建一个UDP套接字用于接收来自客户端的数据包和发送确认包。  2. 数据接收和处理：  在一个循环中，接收来自客户端的数据包。  如果收到数据包，则检查序列号是否是期望的序列号。  如果是期望的序列号，则发送确认包，并将数据写入文件，期望序列号递增。  如果收到的数据包不是期望的序列号，则发送之前期望的确认包。  3. 数据发送：  在一个循环中，发送数据包给客户端。  如果发送的数据包模3为0，即被模拟为丢失，不发送该数据包。  发送数据包时，序列号递增，如果发送的数据包是基序列号，则启动计时器。  4. 超时重传：  如果定时器超时，未收到客户端的确认包，则启动超时重传机制。  重新发送大于确认ack的所有分组  SR协议描述：  工作原理：  流程图：  062a253da6985c815d802d9958ec97da7351fd6beb8d5b5c8769163a8a3f59  发送方：与GBN不同，SR发送方的每个分组采用单独的计时器，超时是单独发送该分组。收到ack后如果序号在窗口内，则标记该分组已被接收，如果等于base值，则将窗口移动到最近的未标记接受的分组处。  接收方：不同之处是要增加一个接收窗口，与发送窗口大小一致。发送ack是接收到哪个包就发送对应的ack，同时收到乱序分组时进行缓存，根据序号是否等于base进行交付上层与移动窗口。  代码描述：  客户端（SRClient）：  1. 初始化：  创建一个UDP套接字（DatagramSocket）用于与服务器通信。  初始化变量，包括序列号、基序列号、标记数组、窗口大小等。  2. 数据发送：  在一个循环中，发送数据包给服务器端。  如果某个数据包模3为0，即被模拟为丢失，不发送该数据包。  发送数据包时，序列号递增，同时更新基序列号。  如果发送的数据包是基序列号，启动计时器。  3. 数据接收和处理：  接收来自服务器端的确认包。  更新标记数组中对应的位置为true，表示收到了确认。  根据收到的确认包更新基序列号。  根据基序列号滑动窗口，检查缓存队列中是否有可以被确认的数据包。  服务器端（SRServer）：  1. 初始化：  创建一个UDP套接字（DatagramSocket）用于监听客户端发送的数据。  初始化期待的序列号为1，创建一个缓存队列。  2. 数据接收和处理：  循环接收来自客户端的数据包。  提取数据包中的序列号。  如果收到的数据包序列号等于期待的序列号，即为期待的数据包，发送对应的确认。  如果收到的数据包序列号不等于期待的序列号，说明数据包乱序到达，将其添加到缓存队列中。  根据收到的确认更新期待的序列号，并检查缓存队列中是否有可以被确认的数据包，进行相应处理。  实现C/S架构双向传输：  GBN服务器存放文件，发送数据前进行文件操作，将文件转为字节数组，再通过套接字传输。接收方同样通过套接字进行解析，同时在本地创建文件进行存储。  双向传输则只需在客户和服务器同时调用接收和发送方法，即可实现双向传输。  停等协议：GBN的发送窗口设置为1，接收窗口保持1不变。  其它代码功能：  Model.java：保证线程的互斥，防止对计时器资源抢占。  1e8712b8e1fd69eaffebaf233d34dc9  Timer.java：实现计时器，用Thread.sleep（）方法模拟实现超时0d71ba935ecefe294f927d58858820a |
| 实验结果： |
| GBN：实现双向数据传输，文件传输，数字序号传输，模拟数据包丢失，设置概率为三分之一。  SR：实现单向数据传输，模拟数据包丢失，设置概率为三分之一，模拟传输数字序号。  GBNClient  实验要求全部完成  d144b78e1bfc1503c20f8afaf2aae66aa8e73879ec3b373f4000e66d1ffcd8  GBNServer  4ff977dde57ca0058656f826bf2fc2a9f0073b6a6aff2e758a7141c4f71e3f  SRClient  b894df5f1210e3efe767fe0fcdd63175d166f0ead385c71ce9a5353ad7e384  SRServer  d9c07a772def17f5a6c89ff57b75b34779379c51bde27287e3eec6448406e4  文件传输：  f604c397c6df085fae4ae6285520641  4f633c928e35e4836c28b79fa0d1a1f  b8af6839fb6563ab8078d2e5bfb6aa1  92462b7375383297f05cd11d469c528 |
| 问题讨论： |
| GBN和SR之间应该如何进行选择？  在不同的网络环境下，GBN和SR的性能可能会有所不同。在低丢包率或者稳定网络环境下，SR协议通常比GBN协议具有更好的性能，因为SR可以灵活地选择性重传丢失的数据包，而不需要重传整个窗口。  在高丢包率或者不稳定网络环境下，GBN协议可能比SR协议更适合，因为GBN采用Go-Back重传策略，可以更快地恢复丢失的数据包，而SR可能因为频繁的选择性重传而导致额外的开销和延迟。 |
| 心得体会： |
| 通过实践GBN和SR的模拟，不仅加深了对这两种协议的理解，也提升了对网络通信和UDP编程的能力，为进一步深入研究和应用网络通信协议打下了良好的基础。 |