

**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 | 贺宗磊 | | 院系 | 计算机科学与技术 | | |
| 班级 | 1703202 | | 学号 | 170300421 | | |
| 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 | 2019.11.2 | | |
| 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： |
| 本次实验的主要目的。  理解可靠数据传输和滑动窗口协议的基本原理；掌握停等协议和GBN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；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： |
| 概述本次实验的主要内容，包含的实验项等。  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： |
| 以文字描述、实验结果截图等形式阐述实验过程，必要时可附相应的代码截图或以附件形式提交。   1. GBN的原理与实现   原理:  GBN是属于传输层的协议,它负责接收应用层传来的数据,将应用层的数据报发送到目标IP和端口  滑动窗口: 假设在序号空间内，划分一个长度为N的子区间，这个区间内包含了已经被发送但未收到确认的分组的序号以及可以被立即发送的分组的序号，这个区间的长度就被称为窗口长度。（随着发送方方对ACK的接收，窗口不断的向前移动，并且窗口的大小是可变的）  GBN一个分组的发送格式是 Base + seq + data  GBN协议的传送流程是: 从上层应用层获得到一个完整的数据报,将这个数据报进行拆分(一个GBN数据帧最大传输的数据大小限制为1024B,因为在以太网中，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）),如果发送方的滑动窗口中,如果窗口内已经被发送但未收到确认的分组数目未达到窗口长度,就将窗口剩余的分组全部用来发送新构造好的数据,剩余未能发送的数据进行缓存。发送完窗口大小的数据分组后,开始等待接收从接收方发来的确定信息(ACK),GBN协议采取了累积确认，当发送方收到一个对分组n的ACK的时候，即表明接收方对于分组n以及分组n之前的分组全部都收到了。对于已经确认的分组,就将窗口滑动到未确认的分组位置(窗口又有空闲位置,可以发送剩余分组了),对于未确认的分组,如果计时器超时,就需要重新发送,直到收到接收方的ACK为止。  图形模拟：  对于超时的触发，GBN协议会将当前所有已发送但未被确认的分组重传，即如果当前窗口内都是已发送但未被确认的分组，一旦定时器发现窗口内的第一个分组超时，则窗口内所有分组都要被重传。每次当发送方收到一个ACK的时候，定时器都会被重置。  接收方只需要按序接收分组，对于比当前分组序号还要大的分组则直接丢弃。假设接收方正在等待接收分组n，而分组n＋1却已经到达了，于是，分组n＋1被直接丢弃，所以发送方并不会出现在连续发送分组n，分组n＋1之后，而分组n＋1的ACK却比分组n的ACK更早到达发送方的情况。（其中，丢包的情况，按照超时处理）  具体实现:  发送方:  首先定义窗口大小,起始 base 的值, 窗口采用链表的数据结构存储  private int WindowSize = 16;（注：当窗口大小设为1时，即为停等协议的实现）  private long base = 0;（初始设为0）  进入一个循环,循环结束条件是所有需要传送的数据都已经发送完成,并且窗口中的分组都已经全部确认。  在这个循环中,如果窗口内有空余,就开始发送分组,直到窗口被占满,计时器开始计时,之后进入接收ACK的状态,收到ACK之后,更新滑动窗口的位置,之后如果计时器超时,就将窗口内所有的分组全部重发一次。之后开始下一次循环。  接收方:  不需要有缓存,只需要记录一个seq值,每成功接收一个数据帧,seq+1,开始循环顺序接收数据帧,对于seq不是目标值得数据帧直接丢弃,如果是符合要求的数据帧,就给发送方发送一个ACK=seq的确认数据帧,直到发送方没有数据传来为止。  GBN的实现就完成了。  二、SR协议的实现：  发送方:  在GBN发送方的基础上,增加一个基于链表数据结构的计时器,对每一个未被确认的分组进行计时。在每次判断是否超时时,需要对链表中所有的计时进行判断,与GBN重传不同的是,SR只对超时的那一个分组进行重传。  如图：Untitled Diagram (3)  模拟过程：    接收方：  需要增加一个同发送方的对分组的缓存,用于缓存乱序到达的分组,同样使用链表数据结构。  List<ByteArrayOutputStream> datagramBuffer = new LinkedList<>();  首先进入一个循环, 一次循环需要进行如下工作:  接收分组,将分组的数据缓存到datagramBuffer对应的位置(因为到达的数据可能是乱序的)  然后发送数据分组对应seq的ACK,告知发送方自己已经成功接收。 之后更新滑动窗口的位置,更新的规则同发送方一样。之后进行下一次循环。  直到发送方没有新的数据传来,超过接收方设定的最大时间,就结束循环,将接收到的数据拼接成一个完整的Byte数组,传给应用层，即实现SR协议  流程图如下：  Untitled Diagram (5)  三、双向传输  发送方发送数据需要占用一个固定的端口,而接收方也需要一个固定的端口来向发送方发送 ACK,所以就可以封装一个完整的协议类,类似于TCP的有连接传输一样,发送方和接收方之间在两个固定的ip和端口之间进行数据的传输,直到双方的传输结束。发送方在使用send()函数进行发送时,也可以同时使用receive()函数进行接收,两个过程并不冲突,可以同时进行。如果要同时收发,就需要同时开一个发送线程和一个接收线程,两个线程独立运行,没有冲突,这样就可以实现双向数据传输了。  所以我构造了一个SR class,其中包含的成员变量有:  private InetAddress host;  private int targetPort, ownPort;  private int WindowSize = 16;  private final int sendMaxTime = 2, receiveMaxTime = 4; // max time for one datagram  private long base = 0;  private final int virtualLossRemainder = 17; // this value is used to simulate the loss of the datagram as a remainder  包含的函数有两个:  void send(byte[] content) 负责数据的发送  ByteArrayOutputStream receive() 负责数据的接收  private ByteArrayOutputStream getBytes(List<ByteArrayOutputStream> buffer, long max) 负责将接收到的数据分组拼接成一个完整的数据报  private boolean checkWindow(List<Integer> timers) 负责判断当前的窗口是否可以移动  详细的代码见附件中SR.java  在Client 主函数中先使用SR协议发送一张图片, 在Server 主函数中使用SR协议接收这张图片,并保存。然后向Client发送一个文件, Client由发送变成接收。  这有就可以实现双向文件的发送和接收了。  四、模拟丢包  在接收端,设立一个计数变量count, 然后每次收到数据帧就加一,如果count 对一个数取余=0就不发送ACK,模拟这一分组丢失的情况,然后测试发送方会不会重新发送丢失的分组。  这一部分的代码实现详见附录中 SR.java中 receive中 count这个变量。  五、文件传输  在传输内容时，引入缓冲文件流，即引入传输函数send()中加入文件缓冲流，代码如下：  datagramBuffer.add(**new** ByteArrayOutputStream());  length = content.length - sendIndex < MAX\_LENGTH ? content.length - sendIndex : MAX\_LENGTH;  ByteArrayOutputStream oneSend = **new** ByteArrayOutputStream();//发送方数据缓冲区  **byte**[] temp = **new** **byte**[1];  temp[0] = **new** ~~Long~~(base).byteValue();  oneSend.write(temp, 0, 1);//字节写入缓冲区  同时在接收函数中也加入相应文件缓冲流，最后将缓冲流合成完整的文件，如下： |
| 实验结果： |
| 采用演示截图、文字说明等方式，给出本次实验的实验结果。   1. GBN的模拟实现：   由函数自动生成要传输的数字，实现GBN协议的模拟传输，并模拟超市情况：  数据发送端：      数据接收端运行如下：      由图提示可知发送接收情况以及超市丢包模拟情况   1. SR协议实现、文件传输、丢包模拟，双向传输的模拟   程序运行前：      （1.txt，3.png是要传输的，其中1.txt由client端到Server端，3.png由Server端到Client端）  程序运行后：      由结果可知文件传输、双向传输成功  其中控制台截图：  发送端    接收端：    其中：    由此可知，当发生重传时只重传超时的部分，由此可知实现了SR协议  综上，实现了GBN协议、停等协议、SR协议、文件传输、双向传输以及超时丢包模拟功能1. |
| 问题讨论： |
| 问题：  1.实验中，当发送数据到达窗口边界时，有的时候，会出现一些问题。  解决方案：  增加一些判断数据传输是否完成添加或修改的语句即可解决问题。  2.SR协议的协议实现出现问题，无法实现只对超时部分重传  解决方案：  经过查询后，尝试了数组和链表两种方式，因为中间会有重传，故数据需要不断插入，而使用链表,增删元素操作就很方便而且开销小,使用数组就不能够删除元素,所以最终选择了使用链表实现队列的功能  3.在实现文件传输时传输失败：  解决方案：  通过询问同学并查询文件流的使用方法，经过调试后解决，可以成功传输照片文件和txt文件等。 |
| 心得体会： |
| 结合实验过程和结果给出实验的体会和收获。  通过这次试验，对GBN协议和SR协议有了进一步认识，也对Socket变成进一步了解，在实验中遇到的问题通过不断解决，也将进一步加深了对于java中实现文件的操作过程进一步了解和加深印象，通过对SR、GBN的模拟，也对两者实际运行的过程加深了了解。也同时对java中相关类的调用，正确实现有了进一步的熟悉，受益很多 |