

**计算机网络**

**课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 实验2：可靠数据传输协议-GBN协议的设计与实现 | | | | | |
| 姓名 | 陈翔 | | 院系 | 计算机科学与技术学院 | | |
| 班级 | 1603109 | | 学号 | 1161800218 | | |
| 任课教师 | 李全龙 | | 指导教师 | 李全龙 | | |
| 实验地点 | 格物213 | | 实验时间 | 2018.11.2 | | |
| 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： |
| 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. GBN 协议实现原理  (1) 数据分组格式    seq：序列号，取值范围 0~255；  flag：传输结束标志，若是最后一个数据分组则为 1，不是则为 0；  checksum：数据校验和，长度为 8bit。  data：传输的数据，长度为 2048 字节。  (2) 确认分组格式    ack\_seq：最近一次确认的数据分组的序列号，取值范围 0~255；  expect\_seq：接收端期望收到的数据分组的序列号，取值 0~255。  (3) 协议两端程序流程图  发送端(client)：    接收端(server)：    (4) 典型交互过程    2. SR 协议实现原理  (1) 数据分组格式    seq：序列号，取值范围 0~255；  flag：传输结束标志，若是最后一个数据分组则为 1，不是则为 0；  checksum：数据校验和，长度为 8bit。  data：传输的数据，长度为 2048 字节。  (2) 确认分组格式  ack\_seq：最近一次确认的数据分组的序列号，取值范围 0~255；  expect\_seq：接收端期望收到的数据分组的序列号，取值 0~255。  (3) 协议两端程序流程图      (4) 典型交互过程    3. 分组丢失模拟方法  首先，设置分组丢失概率 LOSS\_RATE，取值范围为[0, 1)，在每次使用 UDP 发送数据分组之前，产生一个取值范围为[1, 1/LOSS\_RATE]的随机整数，若该整 数恰好等于 1，则该分组丢失，不再发送；若该整数不等于 1，则正常发送。确 认分组 ACK 丢失模拟方法同理。  4. GBN 协议双向数据传输的实现原理  在客户端(client)与服务器端(server)分别建立一个 GBN 发送端(sender)和一个 GBN 接收端(receiver)，使用两对会话模拟双向数据传输。  5. 程序中的主要类及函数  (1) gbn.py 此文件中主要实现了 GBN 协议的发送端和接收端类。  ①函数 getChecksum：计算数据的校验和  ②类 GBNSender：GBN 协议的发送端  属性：  self.sender\_socket：发送端套接字  self.timeout：超时时间  self.address：目的地址（IP 地址，端口号）  self.window\_size：窗口大小  self.loss\_rate：数据分组丢失概率  self.send\_base：窗口头部序列号  self.next\_seq：下一个可用序列号  self.packets：数据分组  方法：  self.udp\_send：使用 udp 发送数据分组  self.wait\_ack：收到确认分组后进行的处理操作  self.make\_pkt：打包生成数据分组  self.analyse\_pkt：分析收到的确认分组  ③类 GBNReceiver：GBN 协议的接收端  属性：  self.receiver\_socket：接收端套接字  self.timeout：超时时间  self.loss\_rate：确认分组丢失概率  self.expect\_seq：期望接收到的数据分组序列号  self.target：确认分组的发送目标地址（即发送端地址）  方法：  self.udp\_send：使用 udp 发送确认分组  self.wait\_data：收到数据分组后进行的操作  self.make\_pkt：打包生成确认分组  self.analyse\_pkt：分析收到的数据分组  (2) sr.py 此文件中主要实现了 SR 协议的发送端和接收端类。  ①函数 getChecksum：计算数据的校验和  ②类 SRSender：SR 协议的发送端  属性：  self.sender\_socket：发送端套接字  self.timeout：超时时间  self.address：目的地址（IP 地址，端口号）  self.window\_size：发送窗口大小  self.loss\_rate：数据分组丢失概率  self.send\_base：窗口头部序列号  self.next\_seq：下一个可用序列号  self.packets：数据分组  self.acks：标记数据分组是否已被确认  方法：  self.udp\_send：使用 udp 发送数据分组  self.wait\_ack：收到确认分组后进行的处理操作  self.make\_pkt：打包生成数据分组  self.analyse\_pkt：分析收到的确认分组  ③ 类 SRReceiver：SR 协议的接收端  属性：  self.receiver\_socket：接收端套接字  self.timeout：超时时间  self.window\_size：接收窗口大小  self.loss\_rate：确认分组丢失概率  self.recv\_base：接收窗口头部序列号  self.recvs：收到的数据分组  self.target：确认分组的发送目标地址（即发送端地址）  方法：  self.udp\_send：使用 udp 发送确认分组  self.wait\_data：收到数据分组后进行的操作  self.make\_pkt：打包生成确认分组  self.analyse\_pkt：分析收到的数据分组 |
| 实验结果：  (1) GBN 协议测试：  首先，将分组丢失概率设置为 0.1。    运行 gbn\_server.py，使接收端处于监听状态：    接着，运行 gbn\_client.py，通过发送端向接收端发送数据文件 client/data.jpg  大小为333,523 字节，共需333523/2048=163分组            传输结果（server/1541748088.jpg，文件使用时间戳命名）：    可见数据传输成功，接收端正确地接收到了发送端发送的数据文件。  (2) SR 协议测试  首先，将分组丢失概率设置为 0.1。    运行 sr\_server.py，使得接收端处于监听状态    接着，运行 sr\_client.py，通过发送端向接收端发送数据文件 client/data.jpg        传输结果（server/ 1541748641.jpg，文件使用时间戳命名）：    可见数据传输成功，接收端正确地接收到了发送端发送的数据文件。  附：数据文件未传输完毕时的图像，证明传输过程的有效性    (3) GBN 协议双向数据传输测试  运行 gbn\_biconnect.py，进行双向数据传输测试。  其中，client 向 server 发送 client/data.jpg 文件  server 向 client 发送 server/data.jpg 文件  传输过程：        传输结果：  server 收到数据后写入 server/1541749447.jpg 文件：    client 收到数据后写入 client/1541749447.jpg 文件：    可见双向数据传输成功。  (4) SR 协议双向数据传输测试  运行 sr\_biconnect.py，进行双向数据传输测试。  其中，client 向 server 发送 client/data.jpg 文件  server 向 client 发送 server/data.jpg 文件  传输过程：        传输结果：  server 收到数据后写入 server/1541750176.jpg 文件：    client 收到数据后写入 client/1541750176.jpg 文件：    可见双向数据传输成功。 |
| 问题讨论：  对实验过程中的思考问题进行讨论或回答。  SR协议和GBN协议的区别主要在哪里？  GBN特点：  因为网络中流量控制的原因，它需要对这些被发送的、未被确认的分组的数目N，否则就会造成网络的拥塞。在GBN协议中，发送方可以再窗口大小N的限制内发送足够多的分组，接收方接收到分组后就发送ACK给发送方（例如：接受到分组0，发送ACK 0），当如果发送方接收到连续的ACK（例如ACK 0和ACK 1）时，该窗口就向前滑动，发送方方便传输新的分组。在接收方，分组丢失了话，就必须从那个分组起再重新传那个丢失的分组号码之后的所有的分组（例如分组2丢失，因此分组3、4、5被认为是失序的分组被丢弃，必须再重新传）但是，这样的话无需接收方准备一定空间的缓存来储存分组。  SR特点：  SR协议相比GBN协议而言，其在接收方增加了接收窗口，对于接收窗口内乱序到达的分组进行缓存，当有一定数量的分组确认后将接收窗口向前滑动；在发送方，增加针对于每个数据包的计时器，不采取累计确认机制，对于每个数据包超时单独进行重传。 |
| 心得体会： |
| 通过本次实验，我有以下几点收获：  ①理解了滑动窗口协议的基本原理；  ②掌握了 GBN 协议的工作原理；  ③掌握了 SR 协议的工作原理；  ④掌握了基于 UDP 设计并实现一个可靠数据传输协议的过程与技术；  ⑤进一步掌握了使用 Python 语言进行 socket 编程的方法和技术 |