



# 学院：软件学院

# 班级：软件工程18-4班

# 学号：2018214937

# 姓名：刘铭源

# 计算机网络实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课程名称** | 网络及其计算 | **班级** | 软件工程18-4班 | **实验日期** |  |
| **姓名** | 刘铭源 | **学号** | 2018214937 |  |  |
| **实验名称** | TCP实验 | | | | |
| **实验目的及要求** | 掌握TCP协议的原理，深入理解TCP协议中的连接管理、可靠传输机制、流量控制机制和拥塞控制 | | | | |
| **实验环境** | Wireshark | | | | |
| **实验内容** | （1）TCP协议基础  传输层源地址结构  传输层目的地址结构  （2）分析TCP连接管理的机制  建立连接机制  释放连接机制  连接过程中的异常处理  （3）分析TCP可靠传输原理  确认机制  重传机制  分析其传输模型  （4）分析TCP的流量控制原理  流量控制机制  零窗口处理机制  小窗口、傻瓜窗口问题  （5）分析TCP的拥塞控制原理 | | | | |
| **实验步骤** | 1.TCP协议基础  1.1分析传输层源地址结构  1.2传输层目的地址结构：16位目的地址端口号  2. TCP连接管理机制  2.1建立连接机制  2.2释放连接机制  2.3连接过程中的异常处理  3.TCP可靠传输原理  3.1确认机制  3.2重传机制  3.3分析传输模型  4.流量重传机制  4.1流量控制机制  4.2零窗口处理机制  4.3小窗口，傻瓜窗口问题  5.分析TCP的拥塞控制原理 | | | | |
| **调试过程及实验结果** | 1. TCP协议基础   （1）在wireshark中，与百度[www.baidu.com](http://www.baidu.com) 建立连接，来验证整个实验，通过ping [www.baidu.com](http://www.baidu.com) 得知百度 ip 为182.61.200.7  （2）停止捕捉，使用wireshark查看tcp包的详细信息，使用filter命令ip.addr == 182.61.200.7 and tcp  查询结果如下： 这里写图片描述  在每一个tcp包中，wireshark给出了具体内容    Frame: 物理层的数据帧概况 Ethernet II: 数据链路层以太网帧头部信息 Internet Protocol Version 4: 互联网层IP包头部信息 Transmission Control Protocol: 传输层T的数据段头部信息，此处是TCP  **至此准备工作已经做好，开始实验**  1.1 传输层目的地址源地址结构：  IP地址和端口号用来唯一的确定网络上数据的地址。所以，每个TCP传输连接有两个端点，即源地址加端口号，目的地址加端口号   TCP包的具体内容如下图所示，我截取了上课的ppt以及wireshark截取到的TCP包:  传输地址唯一地标识主机进程，传输地址=网络号+主机号+端口号，端口号即传输服务访问点 （TSAP)，用来标识应用进程。  在IP网络，传输地址= IP地址+端口号  TCP连接用四元组<源IP地址，源端口号，目的IP地址，目的端口号>表示   1. 分析TCP连接管理的机制 2.1. TCP建立连接时，会有三次握手过程，如下图所示：   在wireshark中，找到与服务器端建立链接的三次握手 第一次握手：SYN=1,Seq=0,ACK=0  第二次握手：SYN=1,ACK=1,Seq=0  第三次握手：SYN=0,ACK=1,Seq=1    2.2 TCP释放连接时，会有“三次握手”过程，如下图所示：    由于我在本例实验数据中没有找到四次挥手，只有三次，所以实验结果如下  第一次Fin=1,ACK=1,Seq为X=2，ack为1  第二次FIN=1，ACK=1,Seq=X=1,ack=X+1=3  第三次FIN=0,ACK=1,Seq=3,ack=X+1=2  三次握手后，连接被释放。  2.3. 连接过程中的异常处理  1.［TCP Previous segment not captured］丢包  在TCP传输过程中，同一台主机发出的数据段应该是连续的，即后一个包的Seq号等于前一个包的Seq + Len（三次握手和四次挥手是例外）当后Seq>前Seq +Len，就知道中间缺失了一段数据  2.[TCP Dup ACK x#y]  当乱序或者丢包发生时，接收方会收到一些Seq号比期望值大的包。此时Ack表示，想获取seq数据包而给了其他的数据包  3、3.[TCP Spurious Retransmission] 丢包重传    丢包异常是我这个数据见到的最多的异常，经常出现  3．分析TCP可靠传输原理 3.1 确认机制  确认号ACK会告诉发送端哪些数据段已经成功接收，并且确认号会向发送端指出接收端希望收到的下一个序列号。即确实号ACK为上个数据序列号的下个数据序列号。  此时序列号Seq=4403  此时确认号ACK=4403，说明上个数据报已成功接收，并发送期望接收的下个数据报。    3.2. 重传机制  当发送端在给定时间间隔内收不到那个数据段的应答时，发送端就会重传那个数据段。三个情况下，会发生数据重传  情况1：网络延时/环路，数据段丢失  情况2：网络延时，数据段推迟到达  情况3：数据段成功到达，应答不能达到  TCP重传过程在实验数据中发生如下图：  在通常情况下，第一个报文发送之后很快会收到TCP ACK报文，意外情况发生时，紧接着的是重传报文：    可以观察到，在RST前后都是154字节，字节数一样，可以看出的确是重传了一次  3.3．分析传输模型 这个问题我查阅了有关资料，《*[Linux](http://www.douban.com/link2?type=buy&subject=4212924&vendor=beifa&url=http%3A//book.beifabook.com/Product/BookDetail.aspx%3FPlucode%3D711521626%26extra%3D0_s4212924" \t "_blank)*[内核网络栈源代码情景分析](http://www.douban.com/link2?type=buy&subject=4212924&vendor=beifa&url=http%3A//book.beifabook.com/Product/BookDetail.aspx%3FPlucode%3D711521626%26extra%3D0_s4212924" \t "_blank)》在我的理解中，TCP 协议可靠性数据传输实现是依靠数据重传和数据确认应答机制来完成TCP 协议的可靠性数据传输。  而数据超时重传和数据应答机制的基本前提是对每个传输的字节进行编号，即序列号。  数据超时重传是发送端在某个数据包发送出去，在一段固定时间后如果没有收到对该数据包的确认应答，则（假定该数据包在传输过程中丢失）重新发送该数据包。  数据确认应答是指接收端在成功接收到一个有效数据包后，发送一个确认应答数据包给发送端主机，该确认应答数据包中所包含的应答序列号即指已接收到的数据中最后一个字节的序列号加1，加1 的目的在于指出此时接收端期望接收的下一个数据包中第一个字节的序列号。  数据超时重传和数据确认应答以及对每个传输的字节分配序列号是TCP 协议提供可靠性数据传输的核心本质。重传应答机制与序列号结合能解决四个问题：  1> 能够处理数据在传输过程中被破坏的问题。  通过对所接收数据包的校验，确认该数据包中数据是否存在错误。如果有，则简单丢弃或者发送一个应答数据包重新对这些数据进行请求。发送端在等待一段时间后，则会重新发送这些数据。  2> 能够处理接收重复数据问题。  首先利用序列号可以发现数据重复问题。因为每个传输的数据均被赋予一个唯一的序列号，如果到达的两份数据具有重叠的序列号（如由发送端数据包重传造成），则表示出现数据重复问题。  3> 能够发现数据丢失以及进行有效解决。  数据丢失的判断是猜测性的，无法确定一个数据包一定丢失在传输过程中，大多是被延迟在网络中，即实质的问题只是数据包乱序到达。可能的数据丢失显然的结果是在接收端接收的数据出现序列号不连续现象。  4> 能够处理接收端数据乱序到达问题。  如果通信双方存在多条传输路径， 则有可能出现数据乱序问题，即序列号较大的数据先于序列号较小的数据到达，而发送端确实是按序列号由小到大的顺序发送的。这个问题的解决只需对这些数据进行重新排序即可。  4．分析TCP流量控制原理 4.1 流量控制机制  窗口数决定了当前传输的最大流量。当我们在传输过程中，通信双方可以根据网络条件动态协商窗口大小，调整窗口大小时，即可实现流量控制。（在TCP的每个确认中，除了ACK外，还包括一个窗口通知）TCP通过滑动窗口机制检测丢包，并在丢包发生时调整数据传输速率。滑动窗口机制利用数据接收端的接收窗口来控制数据流。并且调整窗口大小在两个方向都是可行的。当服务器能够更加快速的处理报文时，它会发送一个较大窗口的ACK报文。  在wireshark中，可以很清楚的看到，在传输过程中经常会协商窗口的大小  如上图，正在沟通自己的win窗口还剩多少个  4．2零窗口机制  可能是由于内存不足，处理能力不够，或其他原因导致服务器无法再处理从客户端发送的数据。这可能会造成数据被丢弃以及传输暂停，但接收窗口能够帮助减小负面影响。  当上述情况发生时，服务器会发送窗口为0的报文。    当客户端接收到此报文时，它会暂停所有数据传输，但会保持与服务器的连接以传输探测报文。一旦服务器能够再次处理数据，将会返回非零值窗口大小，传输会恢复。如下图所示，win窗口变为71168  4.3小窗口，傻瓜窗口问题  如下图，当发送零窗口后，将小包转化为大包传送    5. 分析TCP的拥塞控制原理  发送端设置拥塞窗口来反映网络容量，通过拥塞窗口来限制发送方向网络注入数据的速度，即发送端允许发送的数据既不能超过接收窗口的大 小，也不能超过拥塞窗口的大小；前者是为了流量控制，后者是为了拥塞控制。通过慢启动和拥塞避免策略来控制拥塞窗口的大小  **慢启动：**  设: ssthresh为慢启动阈值，MSS为最大TCP段长度，cwnd为拥塞窗口大小。 发送端初始cwnd一般设置为1个MSS，ssthresh设置为接收窗口大小。 每收到一个确认ACK，则cwnd = cwnd + 1个MSS。 初始cwnd=1，发送端发送1个报文段M0 ，接收端收到后返回ACK1 。发送端 收到ACK1后，将cwnd从1增大到2，于是接着发送M1和M2两个报文段。接收端 收到后返回ACK2和ACK3 。发送端收到后，将cwnd从2增大到4，于是接着发送 M3 ～ M6共4个报文段。当cwnd ＞ssthresh时，进入拥塞避免阶段    **拥塞避免策略：**  每经过一个往返传输时间RTT（不管在RTT 时间内收到几个ACK ），则cwnd = cwnd +1个MSS；RTT是动态变化的。如果超过一段时间（TCP重传超时）没有收到TCP报文段，则认为网络拥塞。不管是慢启动阶段，还是拥塞避免阶段，如果TCP检测到拥塞，则将ssthresh缩 减成cwnd的一半，且cwnd恢复为初始大小，即1个MSS | | | | |
| **总结** | 实验之前，我重温了TCP协议的定义及特点。 TCP是一种面向连接（连接导向）的、可靠的基于字节流的传输层通信协议。TCP将用户数据打包成报文段，它发送后启动一个定时器，另一端收到的数据进行确认、对失序的数据重新排序、丢弃重复数据。  TCP的特点有：  TCP是面向连接的运输层协议，每一条TCP连接只能有两个端点，每一条TCP连接只能是点对点的 TCP提供可靠交付的服务 TCP提供全双工通信。数据在两个方向上独立的进行传输。因此，连接的每一端必须保持每个方向上的传输数据序号。 TCP面向字节流。虽然应用程序和TCP交互是一次一个数据块，但TCP把应用程序交下来的数据仅仅是一连串的无结构的字节流。  在本次实验中，我掌握了上课老师所教授的TCP内容，通过实验了解了TCP的原理，什么情况下会重传报文，什么时候会因网络堵塞进行流量控制。这次实验整体坐下来，让我对课本的理解更加深入了。 | | | | |
| **附录** | 无 | | | | |