网络编程实验报告

实验名称: 网络编程第一次作业
实验日期: 2020/5/1
学生姓名: 杨彬
学生学号: 09017423

1 实验目的

1. 实验 1-1 :使用Freebsd/Linux操作系统下的C编译器和网络程序的调试方法,掌握TCP连接建立和终止以及调整缓冲区大小的方法
2. 实验 1-2: 使用ethereal/TCPDump等抓包工具，截取TCP建立过程中产生的数据包，分析连接建立过程。

2 实验环境

ubuntu 18.04 , gcc 7.5.0

3 实验内容

3.1 实验1-1

3.1.1 实验思路

1. 修改 datetimec.c 和 datetimes.c 的源代码。因为源代码原本使用的是13端口，修改源代码将端口换为12000端口
2. 编译 datetimec.c 为client 和 datetimems,c 为 server
3. 先运行 server 再运行 client

3.2 实验1-2

3.2.1 实验思路

1. 使用命令 `sudo tcpdump -i lo port 12000 -vvv -w ./segment` 监听本地通信，并将抓到的包保存到指定目录下
2. 在实验 1.1 的基础上，先运行 server 在运行 client
3. 用 wireshark 或者 tcpdump 分析抓到的包分析连接的建立过程

4 实验结果分析及讨论

4.1 实验1-1

4.1.1 datetime程序运行

运行编译得到的 client 和 server 得到如下结果

```
chonepieceyb@chonepieceyb-VirtualBox:~/文档/network_programming/lab1$ ./client 1
27.0.0.1
Fri May  1 21:23:10 2020
chonepieceyb@chonepieceyb-VirtualBox:~/文档/network_programming/lab1$ ./server
```

图 1: 实验1-1结果

4.1.2 TCP状态图分析

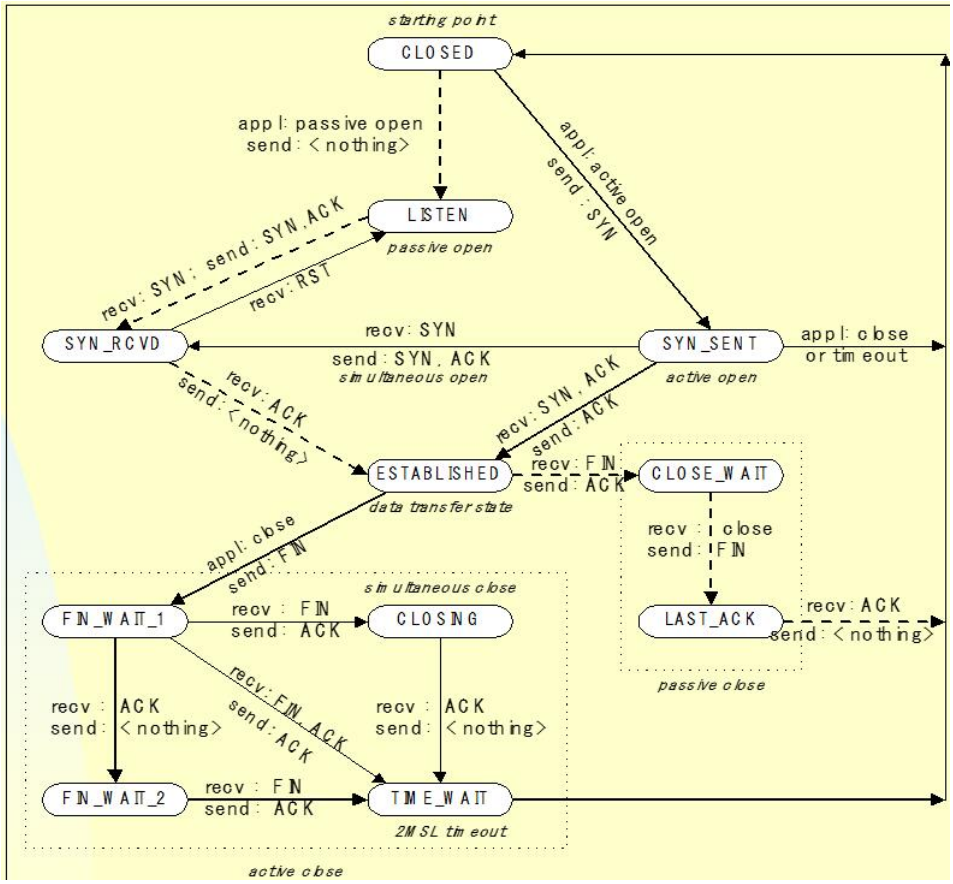

图 2: tcp连接状态图

对TCP连接状态图做简要总结，对于正常的 tcp连接一般由客户端发起连接请求(也称为active open)，服务端监听请求(passive open)。在关闭连接的时候，由客户端或者服务端某一

方先发送 fin 包请求关闭连接(先发送 fin 包的一方的连接是 active close 一般是客户端),相对的另外一方接收 fin 包的为 passive close。其建立连接和关闭连接的过程如下;

建立连接，三次握手

1. 第一次握手，client 发送 syn 包，发起连接请求，设置 client 的 ISN (initial sequence number)，由 $CLOSED$ 状态转入 SYN_SENT状态;
2. 第二次握手，server 直接由 CLOSED状态进入 LISTEN状态，监听连接请求，一旦收到client 发送的 syn包，server端发送 [syn,ack] 包，设置 server 的 ISN 同时回应(ack) client 发送的 syn 包， server 由 $LISTENED$ 状态进入到 SYN_RCVD状态
3. 第三次握手，client 收到 server 发送的 [syn,ack] 包，发送 ack 包回应server, client 由 SYN_SENT状态进入$ESTABLISHED$状态，server在收到 client 的 ack 包之后由 SYN_RCVD状态进入$ESTABLISHED$状态，三次握手完成，server client都互相知晓对方的LSN，连接建立完成

关闭连接，四次挥手,假设由 client 先发送 fin

1. 第一次挥手 client 发送 [fin] 包，表明 client 的数据已经全部传送完毕，client 由 $ESTABLISH$状态进入FIN_WAIT1状态。
2. 第二次挥手 server 收到 client 发送的 fin 包后，知道 client 的数据已经发生完毕，向客户端发送 ack 包，从 $ESTABLISH$ 状态进入到 $CLOSE_WAIT$状态，等待server自身数据发送完成， client 收到 ack之后由 FIN_WAIT1状态变为FIN_WAIT2状态。在server发送 FIN之前，client仍然可以接收server发送的数据，但client不能再发送数据，这种状态就成为TCP的半连接状态
3. 第三次挥手 server确认自己已经发送完所有的数据之后，发送 fin 包给服务端，告知client自己没有数据要发送了。server 由 $CLOSSE_WAIT$状态变为$LAST_ACK$状态。等待client的ack，最终关闭。
4. 第四次挥手 client收到 server 的 fin包之后，发送 ack，进入 $TIME_WAIT$状态，在等待两个 MLT 之后进入 $CLOSE$状态。而 server在收到 ack 之后进入 $CLOSE$状态，设置 $TIME_WAIT$状态的目的是 1 当最后一个 ack 丢包，client能然可以重传ack 2. 防止当前的socket被复用(在连接关闭之后，马上建立四元组相同的新的连接)，导致出错。

4.1.3 实验1-2

4.1.4 用tcpdump抓包结果

4.1.5 抓包分析

三次握手包

1. 1 号包, 第一次握手由 client 发送给 server ,seq为799005335，表示 ISN为该值，flag 中有 syn标志，表明这是一个同步包
2. 2 号包，第二次握手由 server 发送给 client , seq为2266925295。表示server方的ISN为该值，flag中有 syn和 ack 标记， ack的值为 client的LSN+1(syn占一个序号)。第二次握手包同时发送了 syn和ack

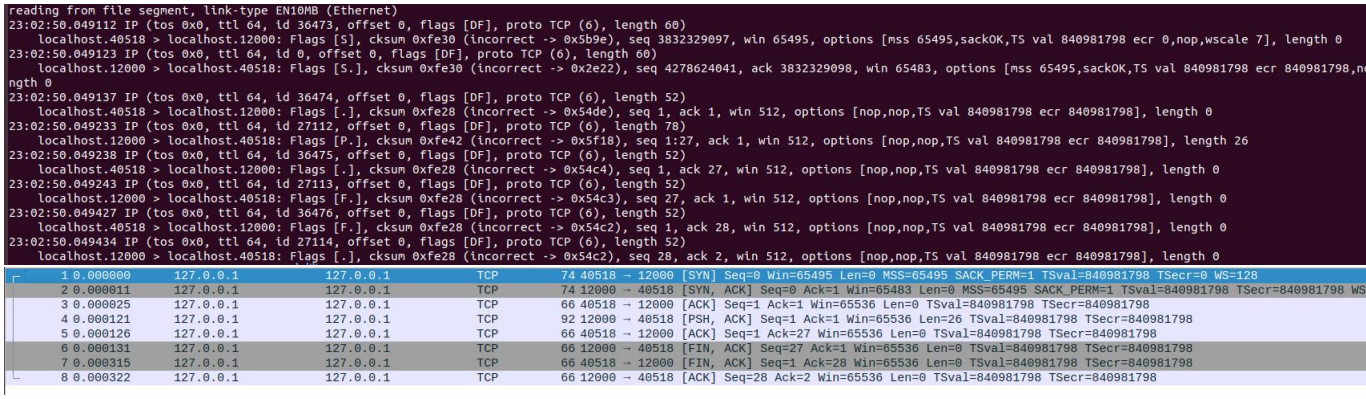

图 3: tcp抓包结果

3. 3 号包，第三次握手，由client 发送给server,对 server发送的 syn包进行回应，该包syn的大小为 server的 ISN+1。

四次挥手包

1. 6 号包，第一次挥手包，有 client 发送 fin 包给 server

2. 7 号包，第二次和第三次挥手，由于 server端发送完数据之后，直接关闭，因此直接将第二次和第三次挥手合并。发送 [ack,fin] 包给client，回应 client 的 fin包，同时发送server的fin包

3. 8 号包，第四次挥手，client收到 server发送的 fin包之后，发送 ack回复。进入time wait状态，server收到 ack之后连接关闭。