南开大学

Nankai University

南　　开　　大　　学

计 算 机 学 院

计算机网络第一次实验报告

第一次作业

沙璇 1911562

年级：2019 级

专业：信息安全

2022 年 9 月 19 日

摘要

这是我的摘要

关键字：TCP,socket

目录

一、 实验要求 1

二、 协议设计 1

三、 程序设计 2

四、 程序实现 3

五、 程序运行说明 6

六、 总结 10

一、 实验要求

利用 Socket，设计和编写一个聊天程序。

基本要求如下：

1）设计一个两人聊天协议，要求聊天信息带有时间标签。

2）对聊天程序进行设计。

3）在 Windows 系统下，利用 C/C++ 中的流式 Socket 对设计的程序进行实现。程序界面可以采用命令行方式，但需要给出使用方法。

4）对实现的程序进行测试。

5）撰写实验报告，并将实验报告和源码提交至本网站。

二、 协议设计

该聊天室采用 TCP 协议的 socket 编程。

SOCKET:

SOCK STREAM 表示面向连接的数据传输方式。数据可以准确无误地到达另一台计算机，如果损坏或丢失，可以重新发送，但效率相对较慢。常用的 HTTP 协议就使用 SOCK STREAM 传输数据，因为要确保数据的正确性，否则网页不能正常解析。

TCP/IP:

在 TCP/IP 网络应用中，通信的两个进程相互作用的主要模式是客户/服务器模式，即客户端向服务器发出请求，服务器接收请求后，提供相应的服务。客户/服务器模式的建立基于以下两点：

（1）建立网络的起因是网络中软硬件资源、运算能力和信息不均等，需要共享，从而就让拥有众多资源的主机提供服务，资源较少的客户请求服务这一非对等作用。

（2）网间进程通信完全是异步的，相互通信的进程间既不存在父子关系，又不共享内存缓冲区。

因此需要一种机制为希望通信的进程间建立联系，为二者的数据交换提供同步，这就是基于客户/服务端模式的 TCP/IP。

服务端：建立 socket，声明自身的端口号和地址并绑定到 socket，使用 listen 打开监听，然后不断用 accept 去查看是否有连接，如果有，捕获 socket，并通过 recv 获取消息的内容，通信完成后调用 closeSocket 关闭这个对应 accept 到的 socket，如果不再需要等待任何客户端连接，那么用 closeSocket 关闭掉自身的 socket。

客户端：建立 socket，通过端口号和地址确定目标服务器，使用 Connect 连接到服务器，send 发送消息，等待处理，通信完成后调用 closeSocket 关闭 socket。

tcp 协议三次握手如图1所示

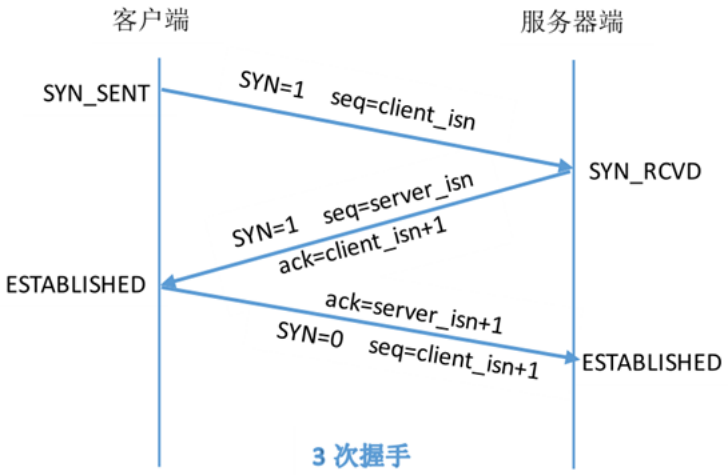

图 1: 3 次握手

四次挥手如图2所示

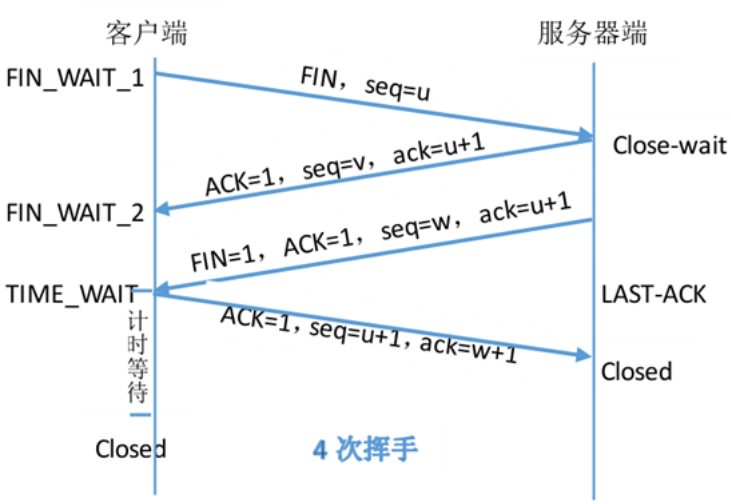

图 2: 4 次挥手

三、 程序设计

该程序的运行需要两部分：服务器端 server 和客户端 client

（1）服务端

1、加载套接字库，创建套接字（WSAStartup()/socket()）；

2、绑定套接字到一个 IP 地址和一个端口上（bind()）；

3、将套接字设置为监听模式等待连接请求（listen()）；

4、请求到来后，接受连接请求，返回一个新的对应于此次连接的套接字（accept()）；

5、用返回的套接字和客户端进行通信（send()/recv()）；

6、返回，等待另一个连接请求；

7、关闭套接字，关闭加载的套接字库（closesocket()/WSACleanup()）；

（2）客户端

1、加载套接字库，创建套接字（WSAStartup()/socket()）;
2、向服务器发出连接请求（connect()）;
3、和服务器进行通信（send()/recv()）;
4、关闭套接字，关闭加载的套接字库（closesocket()/WSACleanup()）;

四、 程序实现

服务器端代码如下

server

```
#include <stdio.h>
#include <winsock2.h>
#include <iostream>
#include <string>
#include<time.h>
using namespace std;

#pragma comment(lib,"ws2_32.lib")

int main(int argc, char* argv[])
{
    //初始化WSA
    WORD sockVersion = MAKEWORD(2,2);
    WSADATA wsaData;
    if(WSAStartup(sockVersion, &wsaData)!=0)
    {
        return 0;
    }

    //创建套接字
    SOCKET slisten = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if(slisten == INVALID_SOCKET)
    {
        printf("socket error !");
        return 0;
    }

    //绑定IP和端口
    sockaddr_in sin;
    sin.sin_family = AF_INET;
    sin.sin_port = htons(8888);
    sin.sin_addr.S_un.S_addr = INADDR_ANY;
    if(bind(slisten, (LPSOCKADDR)&sin, sizeof(sin)) == SOCKET_ERROR)
    {
        printf("bind error !");
    }

```

```
    //开始监听
    if(listen(slisten, 5) == SOCKET_ERROR)
    {
        printf("listen error !");
        return 0;
    }

    //循环接收数据
    SOCKET sClient;
    sockaddr_in remoteAddr;
    int nAddrlen = sizeof(remoteAddr);

        printf("等待连接...\n");
    sClient = accept(slisten, (SOCKADDR *)&remoteAddr, &nAddrlen);

        if(sClient == INVALID_SOCKET)
        {
            printf("accept error !");
        }
        printf("接受到一个连接: %s \r\n", inet_ntoa(remoteAddr.sin_addr));
    while (true)
    {


         char revData[255];
        //接收数据
        int ret = recv(sClient, revData, 255, 0);
        if(ret > 0)
        {
            revData[ret] = 0x00;
            printf(revData);
        }

        //发送数据
                string data;
                cin >> data;
        char sendData[256];
                strcpy(sendData, data.c_str());

                struct tm* timeinfo;
                time_t rawtime;
                time(&rawtime);
                timeinfo = localtime(&rawtime);
                string time_str;
                time_str = asctime(timeinfo);
                char time_char[255];
                strcpy(time_char, time_str.c_str());
                strcat(sendData," ");
```

```
87                  strcat(sendData, time_char);
88          send(sClient, sendData, strlen(sendData), 0);
89
90      }
91      closesocket(sClient);
92      closesocket(slisten);
93      WSACleanup();
94      return 0;
95  }
```

客户端代码如下:

client

```
1
2   #include<WINSOCK2.H>
3   #include<STDIO.H>
4   #include<iostream>
5   #include<string>
6   #include<time.h>
7   using namespace std;
8   #pragma comment(lib, "ws2_32.lib")
9
10  int main()
11  {
12          WORD sockVersion = MAKEWORD(2, 2);
13          WSADATA data;
14          if(WSAStartup(sockVersion, &data)!=0)
15          {
16                  return 0;
17          }
18          SOCKET sclient = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
19                  if(sclient == INVALID_SOCKET)
20                  {
21                          printf("invalid socket!");
22                          return 0;
23                  }
24
25                  sockaddr_in serAddr;
26                  serAddr.sin_family = AF_INET;
27                  serAddr.sin_port = htons(8888);
28                  serAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
29                  if(connect(sclient, (sockaddr *)&serAddr, sizeof(serAddr)) ==
                        SOCKET_ERROR)
30                  {  //连接失败
31                          printf("connect error !");
32                          closesocket(sclient);
33                          return 0;
34                  }
35          while(true){
```

```
36
37
38                  string data;
39                  cin >> data;
40                  char sendData[256];
41                  strcpy(sendData, data.c_str());     //string转const char*
42
43                  struct tm* timeinfo;
44                  time_t rawtime;
45                  time(&rawtime);
46                  timeinfo = localtime(&rawtime);
47                  char * sendmsg;
48                  sendmsg = asctime(timeinfo);
49                  strcat(sendData," ");
50                  strcat(sendData,sendmsg);
51                  send(sclient, sendData, strlen(sendData), 0);
52                  //send()用来将数据由指定的socket传给对方主机
53                  //int send(int s, const void * msg, int len, unsigned int
                        flags)
54                  //s为已建立好连接的socket，msg指向数据内容，len则为数据长度，
                        参数flags一般设0
55                  //成功则返回实际传送出去的字符数，失败返回-1，错误原因存于
                        error
56
57                  char recData[255];
58                  int ret = recv(sclient, recData, 255, 0);
59                  if(ret>0){
60                          recData[ret] = 0x00;
61                          printf(recData);
62                  }
63          }
64
65
66          WSACleanup();
67          return 0;
68
69  }
```

五、 程序运行说明

客户端与服务器端连接成功后服务器端如图3所示

图 3: server

客户端如图4所示

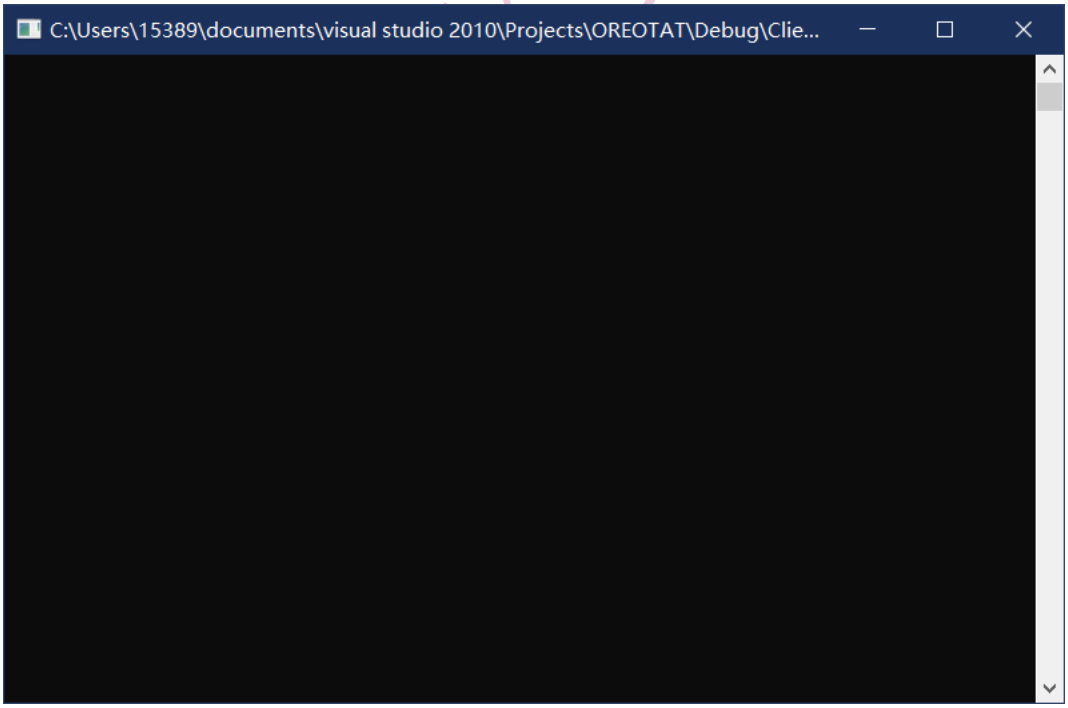

图 4: client

在客户端键入 message 如图5所示

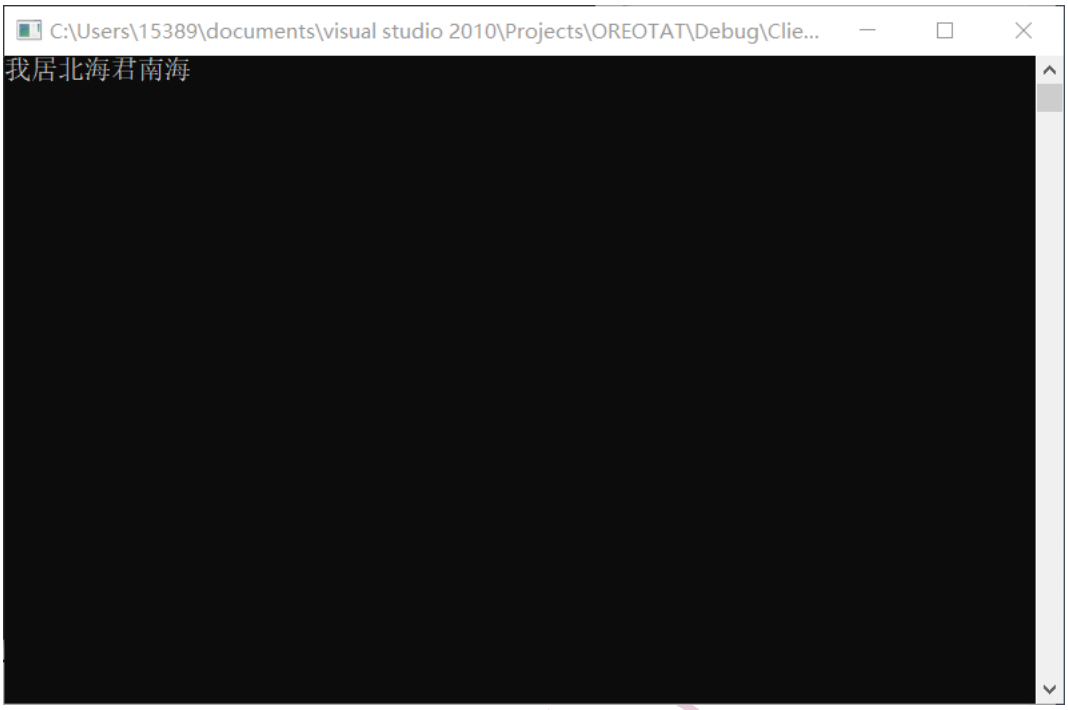

图 5: server

收到 message 后在服务器端键入新 message 如图6所示

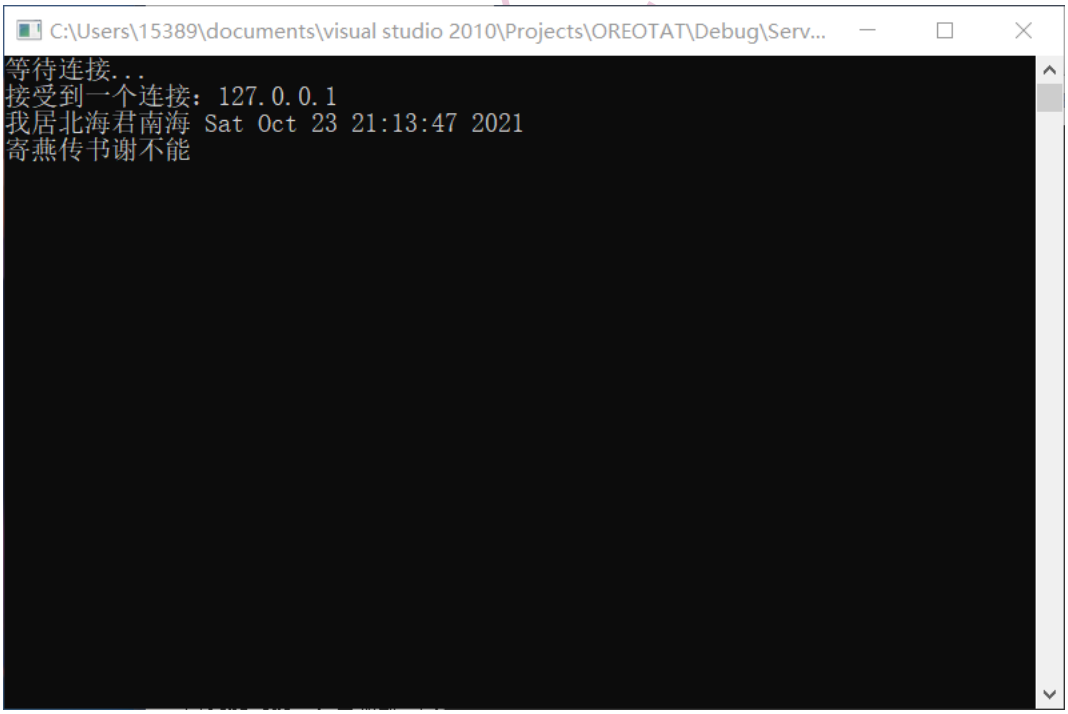

图 6: client

一系列对话后客户端如图7所示

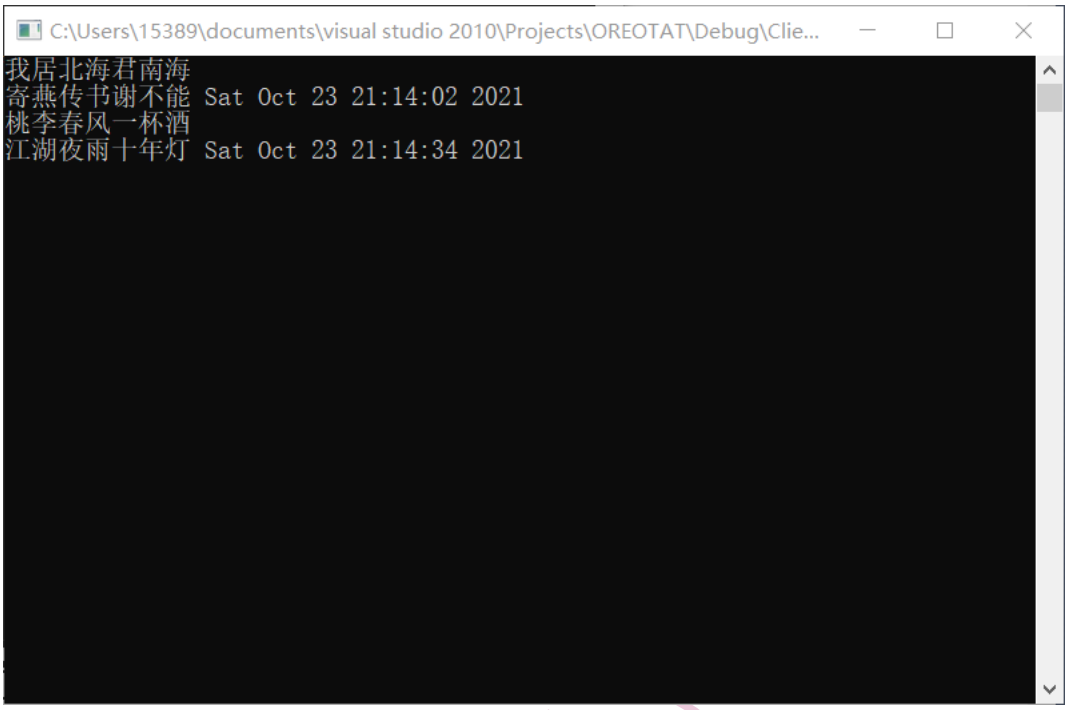

图 7: client

服务器端如图8所示

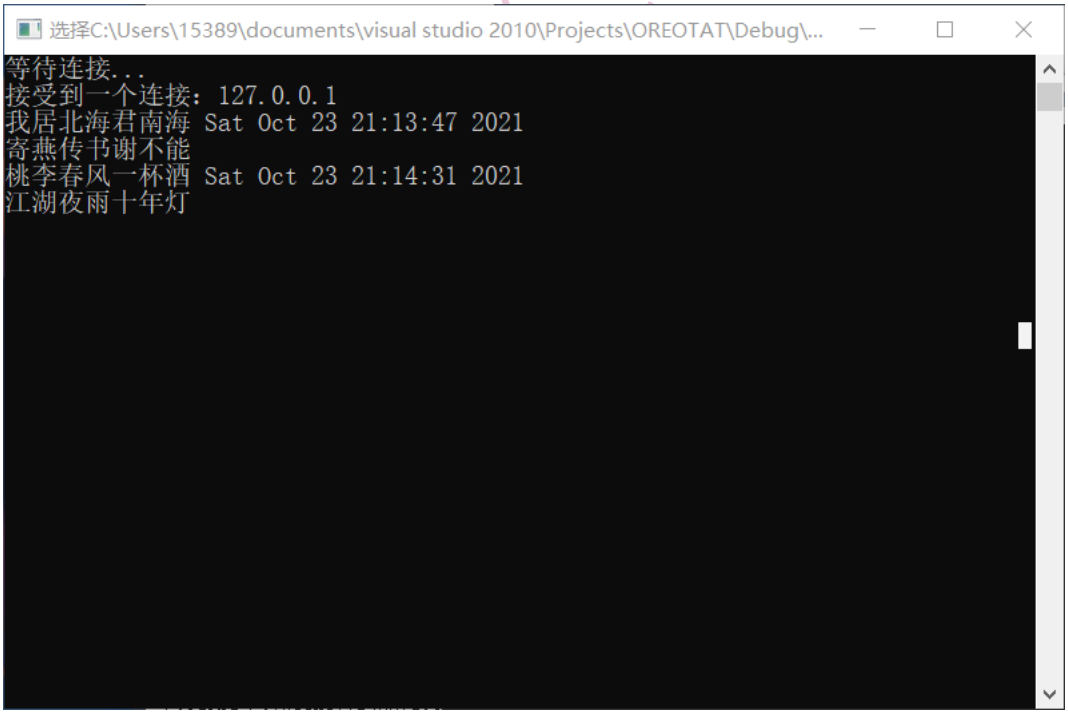

图 8: server

六、 总结

通过本次实验，我对 TCP 协议这一概念有了更深刻的体会，并且对于 C 的流式 Socket 编程的实现所包含的内容进行了初步的学习。