

# 廈門大學



## 信息学院软件工程系

### 《计算机网络》实验报告

题 目 实验五 利用 Socket API 实现许可认证软件

班 级 软件工程 2019 级 1 班

姓 名 姬颖超

学 号 22920192204218

实验时间 2021 年 4 月 30 日

2021 年 4 月 30 日

# 填写说明

- 1、本文件为 Word 模板文件，建议使用 Microsoft Word 2019 打开，在可填写的区域中如实填写；
- 2、填表时，勿破坏排版，勿修改字体字号，打印成 PDF 文件提交；
- 3、文件总大小尽量控制在 1MB 以下，勿超过 5MB；
- 4、应将材料清单上传在代码托管平台上；
- 5、在学期最后一节课前按要求打包发送至 [cni21@qq.com](mailto:cni21@qq.com)。

## 1 实验目的

通过完成实验，掌握应用层文件传输的原理。

了解传输过程中传输层协议选用、应用层协议设计和协议开发等概念。

## 2 实验环境

操作系统：Windows10，编程语言：C++。

## 3 实验结果

### 3.1 实验流程

以下是服务器端的设计流程：

(1) 创建套接字、绑定 IP 和端口（这里是和本地 127.0.0.1 建立连接）、打开监听，代码如下

```
//创建服务器Socket(监听套接字)
SOCKET sListen = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (sListen == SOCKET_ERROR) {
    printf("create socket() error: %d\n", WSAGetLastError());
    return;
}

//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 = inet_addr("127.0.0.1");
if (bind(sListen, (LPSOCKADDR)&sin, sizeof(sin)) == SOCKET_ERROR)
{
    printf("bind error !");
}

//打开监听
listen(sListen, MAX);
printf("wait connect...\n");
```

(2) 在端口监听，如果有客户机发起连接，建立一个与客户端通信的线程

```

AddrSize = sizeof(client);
//监听是否有连接请求
sClient = accept(sListen, (struct sockaddr*)&client, &AddrSize);
if (sClient == INVALID_SOCKET) {
    printf("accept() error: %d\n", WSAGetLastError());
    break;
}
printf("connect: %s:%d\n", inet_ntoa(client.sin_addr), ntohs(client.sin_port));

//创建一个线程去处理
HANDLE hThread = CreateThread(NULL, 0, ClientThread, (LPVOID)sClient, 0, &dwThread);

if (hThread == NULL) {
    printf("CreateThread() error: %d\n", GetLastError());
    break;
}
//处理完后关闭
CloseHandle(hThread);

```

(3) 与客户端进行通信的具体过程:

a. 接受消息

```

int recv(SOCKET sock, char *Buffer)
{
    int ret, nLeft, idx;
    ret = recv(sock, Buffer, DEFAULT_BUFFER, 0);
    if (ret == 0)
        return 1;
    else if (ret == SOCKET_ERROR) {
        printf("recv() error: %d\n", WSAGetLastError());
        return 1;
    }
    Buffer[ret] = '\0';
    printf("client>> %s\n", Buffer);    //打印接收到的消息
    return 0;
}

```

b. 根据消息做出回复

```
int send(SOCKET sock, const char* sendData)
{
    int ret;
    ret=send(sock, sendData, strlen(sendData), 0);
    if (ret == 0)
    {
        return 1;
    }
    else if (ret == SOCKET_ERROR) {
        printf("send() error: %d\n", WSAGetLastError());
        return 1;
    }
    printf("server>> %s\n", sendData);
    return 0;
}
```

(4) 服务器响应客户端的几条指令：

- [1]. buy 管理员购买许可证      [2]. run 客户端第一次运行程序
- [3]. exit 客户端退出              [4]. 许可之后的其他指令返回 exit

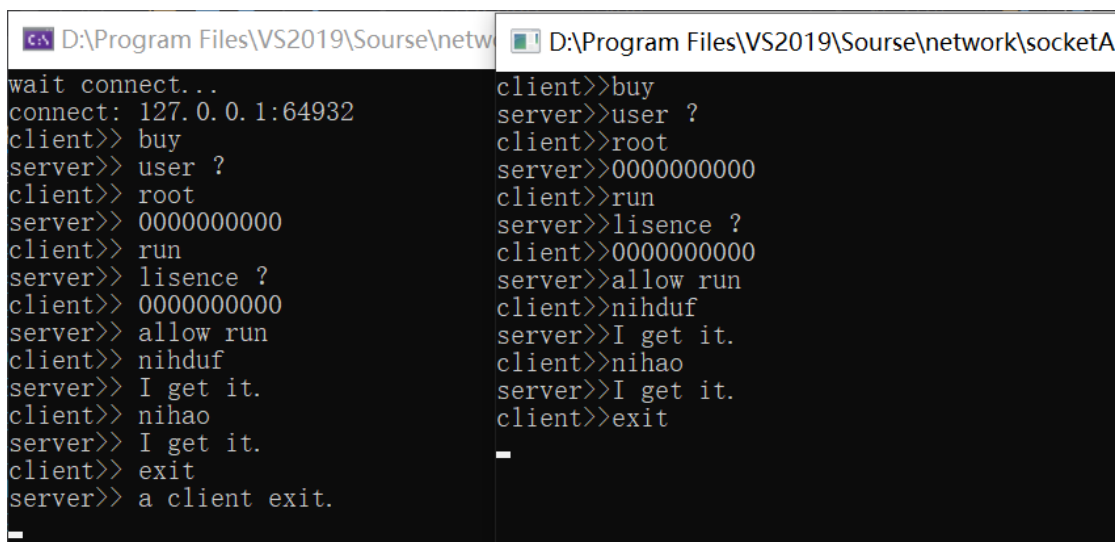
客户端的程序与服务器端类似，但只需要发送和接受数据，不需要建立多个线程。

### 3.2 运行过程

- (1) 运行许可证服务器程序，等待连接
- (2) 运行客户端程序，建立连接
- (3) 客户端购买许可证后，使用程序

### 3.3 实验结果

- (1) 客户端购买证书，服务器返回一个包含 10 位整数的字符串



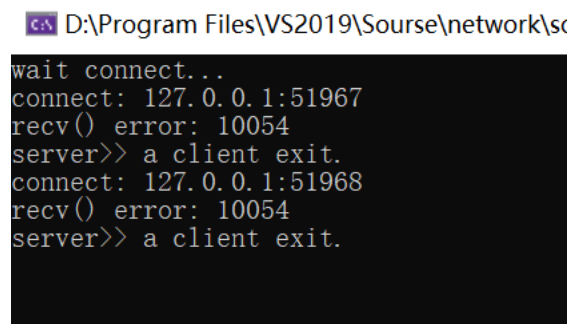
```

D:\Program Files\VS2019\Source\netw
wait connect...
connect: 127.0.0.1:64932
client>> buy
server>> user ?
client>> root
server>> 0000000000
client>> run
server>> lisenice ?
client>> 0000000000
server>> allow run
client>> nihduf
server>> I get it.
client>> nihao
server>> I get it.
client>> exit
server>> a client exit.

D:\Program Files\VS2019\Source\network\socketA
client>>buy
server>>user ?
client>>root
server>>0000000000
client>>run
server>>lisenice ?
client>>0000000000
server>>allow run
client>>nihduf
server>>I get it.
client>>nihao
server>>I get it.
client>>exit

```

(2) 只要客户端未正常发送消息（接受不到客户端的消息），则认为退出

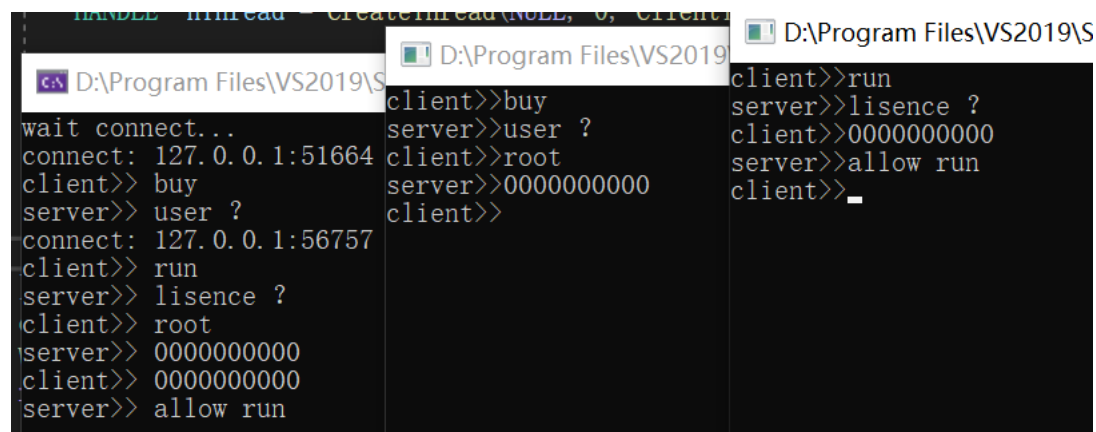


```

D:\Program Files\VS2019\Source\network\sc
wait connect...
connect: 127.0.0.1:51967
recv() error: 10054
server>> a client exit.
connect: 127.0.0.1:51968
recv() error: 10054
server>> a client exit.

```

(3) 一个客户端购买证书，另一个客户端利用证书运行



```

D:\Program Files\VS2019\Source\network\sc
wait connect...
connect: 127.0.0.1:51664
client>> buy
server>> user ?
connect: 127.0.0.1:56757
client>> run
server>> lisenice ?
client>> root
server>> 0000000000
client>> 0000000000
server>> allow run

D:\Program Files\VS2019\Source\network\socketA
client>>buy
server>>user ?
client>>root
server>>0000000000
client>>

D:\Program Files\VS2019\Source\network\sc
client>>run
server>>lisenice ?
client>>0000000000
server>>allow run
client>>

```

(4) 如果序列号错误，则拒绝运行

```

选择D:\Program Files\VS2019\Source\network\socketAPI\Debug
wait connect...
connect: 127.0.0.1:63231
client>> run
server>> lisence ?
client>> 0002
server>> do not allow run, please try a minute later

D:\Program Files\VS2019\Source\network\socketAPI_client\De
client>>run
server>>lisence ?
client>>0002
server>>do not allow run, please try a minute later
client>>_

```

(5) 现存问题：发送的数据如果有空格就会被分成两部分发送

```

D:\Program Files\VS2019\Source\network\socketAPI\Debug\socketAPI_client\De
wait connect...
connect: 127.0.0.1:54459
client>> buy
server>> user ?
client>> root
server>> 0000000000
client>> run
server>> lisence ?
client>> 0000000000
server>> allow run
client>> hello
server>> I get it.
client>> server
server>> I get it.
client>> hi
server>> I get it.
client>> server
server>> I get it.
client>> exit
server>> a client exit.

D:\Program Files\VS2019\Source\network\socketAPI_client\De
client>>buy
server>>user ?
client>>root
server>>0000000000
client>>run
server>>lisence ?
client>>0000000000
server>>allow run
client>>hello server
server>>I get it.
client>>server>>I get it.
client>>hi server
server>>I get it.
client>>server>>I get it.
client>>exit

```

## 4 实验代码

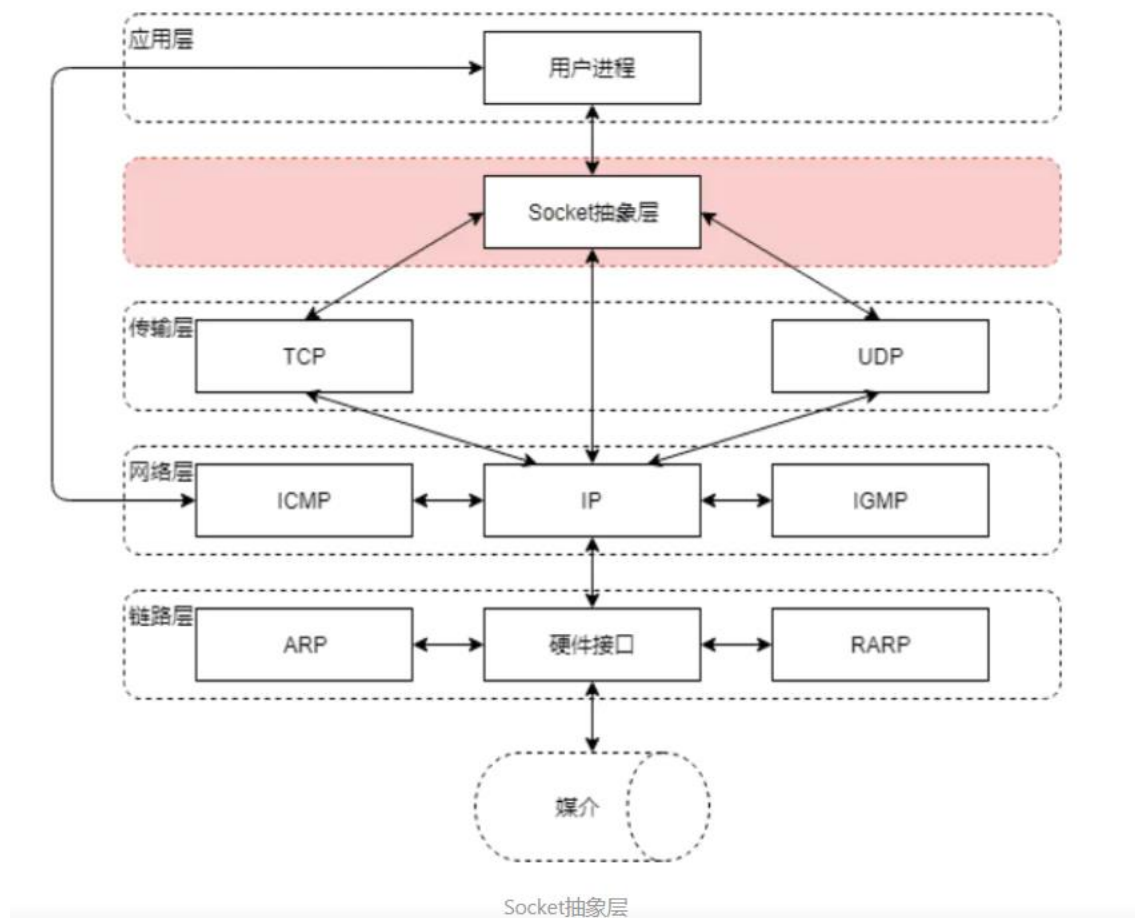
本次实验的代码已上传于以下代码仓库：<https://www.gitee.com/xxx/xxx>（注意：建议使用码云，并设置公开权限；本学期暂不推荐使用 GitHub；如使用厦门大学私有 Git 服务，应将 whuang@xmu.edu.cn 加入项目成员备查，本段话删除。）

## 5 实验总结

(1) 深入认识 socket:

Socket 作为应用层与 TCP/IP 协议簇通信的中间软件抽象层，是一组接口，用于描述 IP 地址和端口，以实现不同虚拟机或物理机之间的通信。应用程序通过 Socket

向网络发出请求或应答请求。网络中两个进程通过一个双向的通信连接实现数据的交换，建立网络通信连接至少需要一对 Socket，连接的一端称为一个 Socket。



## (2) 协议选择:

**SOCK\_STREAM:** 表示面向连接的数据传输方式。对应 TCP；是可靠的传输；

**SOCK\_DGRAM:** 表示无连接的数据传输方式。对应 UDP；是不可靠的传输。

若为保证数据的准确无误，使用 **SOCK\_STREAM**，如邮件等；若为保证通信效率，使用 **SOCK\_DGRAM**，如视频和语音适合传输数据。为保证数据的准确，此次选用 **SOCK\_STREAM**。

## (3) 实现 TCP Socket 通信流程:

服务器:



服务器根据 IP 地址类型（IPv4/IPv6）、Socket 类型和协议创建套接字；

服务端为 Socket 绑定 IP 地址和端口号；

服务端 Socket 监听端口请求，随时准备接收客户端发来的连接，此时 socket 并未被打开。

客户端：

客户端打开 Socket，根据服务器 IP 地址和端口试图连接服务端的 Socket。

服务器 Socket 接收到客户端 Socket 请求，被动打开开始接收客户端请求，直到客户端返回连接信息，此时 Socket 进入阻塞状态。

交互过程：

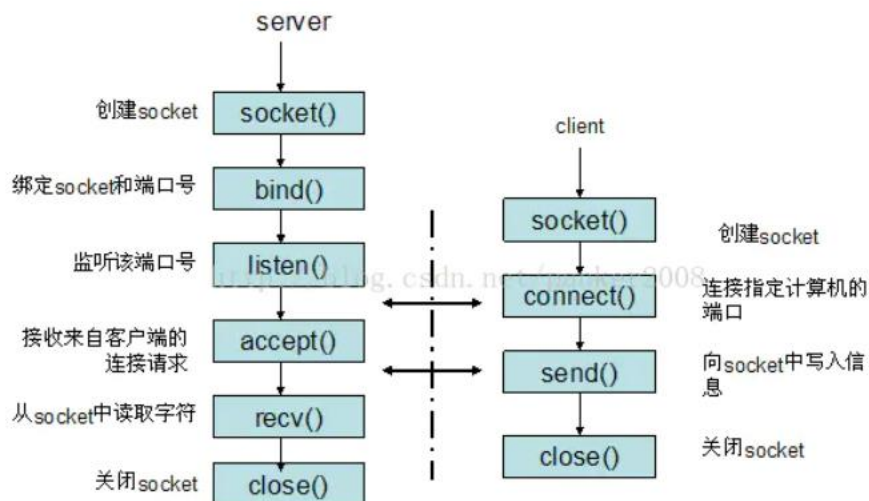
客户端连接成功向服务端发送连接状态信息；

服务端 Accept 返回连接成功；

客户端向 Socket 写入数据；

服务端读取数据；

客户端关闭。



(4) 服务器绑定端口时，要注意同一个端口不能同时被两个程序使用。

查看端口号是否被占用可以用 cmd 查看：

a. 查看所有运行的端口

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19042.985]
(c) Microsoft Corporation。保留所有权利。

C:\Users\可可儿>netstat -ano

活动连接

 协议 本地地址          外部地址          状态          PID
TCP    0.0.0.0:21         0.0.0.0:0         LISTENING      4160
TCP    0.0.0.0:80         0.0.0.0:0         LISTENING      4
TCP    0.0.0.0:135        0.0.0.0:0         LISTENING      1088
TCP    0.0.0.0:445        0.0.0.0:0         LISTENING      4
TCP    0.0.0.0:808        0.0.0.0:0         LISTENING      4244
TCP    0.0.0.0:4303       0.0.0.0:0         LISTENING      7380
TCP    0.0.0.0:5040       0.0.0.0:0         LISTENING      9904
TCP    0.0.0.0:5357       0.0.0.0:0         LISTENING      4
TCP    0.0.0.0:6646       0.0.0.0:0         LISTENING      6460
TCP    0.0.0.0:7680       0.0.0.0:0         LISTENING      3416
TCP    0.0.0.0:9001       0.0.0.0:0         LISTENING      4
TCP    0.0.0.0:49664      0.0.0.0:0         LISTENING      936
TCP    0.0.0.0:49665      0.0.0.0:0         LISTENING      844
TCP    0.0.0.0:49666      0.0.0.0:0         LISTENING      1412
TCP    0.0.0.0:49667      0.0.0.0:0         LISTENING      1656
TCP    0.0.0.0:49668      0.0.0.0:0         LISTENING      3712
TCP    0.0.0.0:49670      0.0.0.0:0         LISTENING      916
TCP    10.30.89.214:139   0.0.0.0:0         LISTENING      4
TCP    10.30.89.214:49968 124.225.166.114:443 ESTABLISHED    11112
TCP    10.30.89.214:51047 43.255.227.59:443  CLOSE_WAIT    4988
TCP    10.30.89.214:51059 52.114.75.78:443  ESTABLISHED    11112
TCP    10.30.89.214:52097 140.143.52.226:443 ESTABLISHED    11112
```

b. 查看端口是否被占用

```
C:\WINDOWS\system32\cmd.exe

UDP    [:]:5353           *: *              3264
UDP    [:]:5355           *: *              3264
UDP    [:]:49351          *: *              8136
UDP    [:1]:1900          *: *              6656
UDP    [:1]:51976         *: *              6656
UDP    [fe80::5427:a995:4073:678c%17]:1900 *: *              6656
UDP    [fe80::5427:a995:4073:678c%17]:51977 *: *              6656

C:\Users\可可儿>netstat -ano|findstr "8080"
C:\Users\可可儿>netstat -ano|findstr "8081"
C:\Users\可可儿>netstat -ano|findstr "1900"
UDP    10.30.89.214:1900   *: *              6656
UDP    127.0.0.1:1900     *: *              6656
UDP    [:1]:1900          *: *              6656
UDP    [fe80::5427:a995:4073:678c%17]:1900 *: *              6656

C:\Users\可可儿>_
```

如图，“1900”被占用，“8080”和“8081”未被占用