Universe Scientific Publishing
Computer System Networking and Telecommunications
计算机系统网络和电信，2019,1(2)
ISSN:2661-3719 (Print); 2661-3727 (Online)

Using Socket to Realize Connection-Oriented Network Communication

Wei ZENG Zhengde Bao Yawen TANG
School of Computer and Software, Jincheng College, Sichuan University, Chengdu, Sichuan, 611731

Abstract

Using new media platform to communicate with friends and playing games in local area network are all the applications of Socket network communication in daily life. Socket, like landline and mobile phones, can communicate with each other by defining the phone number and the prescribed constraints. It adopts C/S mode. With the help of Socket, they can communicate with each other, just as mobile phone users can communicate with landline users. Based on the analysis of it, this paper realizes the network communication of client/server mode based on connection-oriented.

Key Words

Socket, Network Programming, Client/Server Model

DOI:10.18686/jsjxt.v1i2.628

利用 Socket 实现面向连接的网络通信

曾伟 鲍正德 唐娅雯
四川大学锦城学院计算机与软件学院，四川，成都，611731

摘 要

使用新媒体平台与好友通信、在局域网中玩游戏，这些都是生活中 Socket 网络通信的应用。而 Socket 就像座机与移动电话，通过确定的电话号码与规定的约束从而能够实现通信。它采用了 C/S 模式。而在 Socket 的帮助下两者间能够进行通信，就好比移动电话使用者能与座机使用者进行通话。本文通过对其进行浅析，基于面向连接从而实现客户端/服务器模式的网络通信。

关键字

Socket；网络编程；客户/服务器模式

1.引言

随着科技的不断提高，人们的生活方式也不断趋向于快节奏，而在这快节奏的生活当中，网络通信是必不可缺的部分。要想通信能够确定并且唯一，通信的双方需要确定唯一的端口号，同时也需要本地地址以及服务器地址的 IP，只有在这样的条件下能够进行端到端之间的信息通信。

2.技术背景

要利用套接字基于 TCP 面向连接实现客户端/服务器模式的网络通信，首先要了解 TCP 及其传输机制，其次要对套接字的概念及工作原理有所了解。

2.1 Socket 套接字

Socket 是建立在传输层协议 (主要是 TCP 和 UDP) 上的一种规范[1]。通过这个规范，通信双方便有了相互之间的标准，如何去通信，如何去确定连接便有了规范。而它也可以理解为编程接口，其中封装了网络通信的协议规范，这些接口也为网络通信提供了可能。而这个通信过程类似座机和移动电话，在通话之前需要座机进行号码的绑定和申请，而通信另外一边如同移动电话同样也需要号码申请，而这两个号码都是能够唯一确定的，在这样端到端的基础之下便可建立连接，双方通话的过程实际上就是接收信息发送信息的过程。

2.2 TCP 协议

ISSN:2661-3719 (Print); 2661-3727 (Online)

它是传输层的协议。而它的特点是面向连接的，就好比两个间谍在互相传达任务的时候会再三的确认身份在进行信息交换，因此用此协议建立的连接是可靠的。TCP/IP 协议中操作端及服务器端通信系统应用可通过 Socket 技术进行开发[2]。

2.3 三次握手

有许多方法可以建立连接，而这种方法能够使得建立的连接更加的可靠。通俗的讲就是主机 A 和主机 B 不断的相互确认是否进行连接，而主机 A 与主机 B 之间有一个确认序号，如果序号是一致的则可以进行下一步连接。具体流程：

1) 第一次握手:客户端向服务器表示想要与之建立连接，而这个过程客户端发出了一个数据包，而这个包中会随机带有一个序列号 seq，以及同步序号标志 SYN，并且 SYN 为 1，代表发起一个连接。在这个过程之后客户端会等待回应，从而它会进入等待状态。

2) 第二次握手:服务器接收之前的包之后,通过数据包中的信息知道有人想要与之进行连接的意图。此时服务器同样也会向客户端发送一个数据包，这个数据包中 SYN 为 1，ACK 为 1(代表确认序号有效)，ack 确认号为第一次握手产生的序列号+1(协议规定，这样的变换使得通信双方能够互相确认)，当然同样也会随机产生一个 seq 序列号。通过这个数据包服务器也与客户端请求建立连接，同时经过这个过程之后服务器进入了等待状态。

3)第三次握手:客户端收到第二次握手中的包之后，会按照预先的规定检测 ack 是否正确，同时也检测 ACK 是否有效。在检测正确之后，与第二次握手中的机制一样产生一个数据包，将该数据包发送给服务器。这时候服务器对数据包的进行检测，会对 ack 进行判断，同时也判断 ACK 是否有效，如果检测是正确的，则连接成功，此时的两者便处于已建立状态，从而双方能够互发信息。因为有了这样再三的确认，所以两者之间的连接更加的可靠。

这三个过程简而言之就是不断的确认身份，在这样的再三确认之下建立的连接也就更加的稳定，更加的可靠，因而这样的利用这样的机制进行的通信一般情况下是稳定可靠的。

3.通信实现

Socket 通信本质上是端到端通信[3]。在此基础下进行了三次握手，如同间谍间互通消息时反复确定这样更加安全可靠。整个过程好比一个服务器座机(电话号码能唯一确定)，一直在等待客户端移动电话(号码同样能够唯一确定)使用者拨进电话，在此之前座机就一直处于待机状态，当有客户机接入请求之后，在经历了三次握手后与客户机建立连接，进而能够收发消息。

4.通信流程及具体实现

Socket 编程是基于 C/S 模式设计的网络通信接口[4]，主要的通信流程如下：

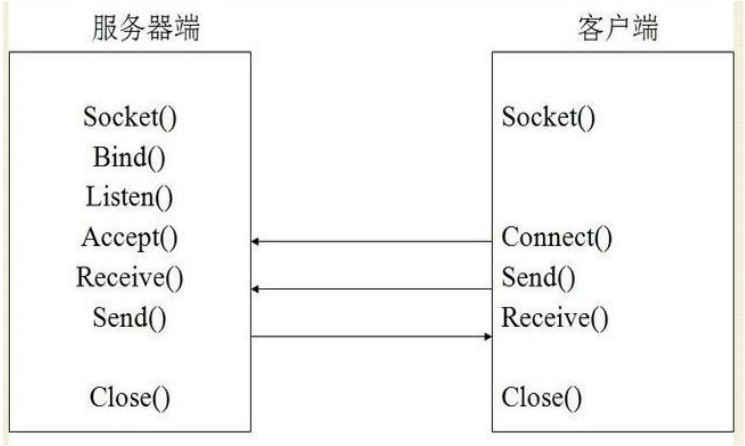

图 1 通信流程

4.1 服务器套接字创建

套接字的创建如同购买了一个座机，要想创建套接字，必不可少的是 socket()函数，函数原型:

SOCKET WSAAPI socket(

__in int af,

__in int type,

__in int protocol

);

af 代表地址类型，表示本地地址具体是 ipv4 还是 ipv6，一般情况下参数用 AF_INET。

type 表示 socke 的类型，常见的类型有:

SOCK_STREAM（流套接字，适用于 TCP 协议）;

SOCK_DGRAM（数据报套接字，适用于 UDP 协议）;

```
retVal = bind(sServer, (LPSOCKADDR)&addrServ, sizeof(SOCKADDR_IN));
if(SOCKET_ERROR == retVal)
{
    cout << "bind failed!" << endl;
    closesocket(sServer);   //关闭套接字
    WSACleanup();           //释放套接字资源;
    return -1;
}
```

图 2 代码实现

4.3 服务器 listen()

首先要能够听，其次才有接收信息的条件，而这个函数正是给予服务器听的前提，有了听的条件也才能够调用 accept 函数。而此时有两个队列，其中一个是代表未完成连接，另外一个代表已完成连接。前者是三次握手尚未完成，此时这些套接字是处于等待状态，后者代表已经完成三次握手，而队列中对应有建立连接的客户，此时的套接字处于已建立状态。这个函数是使得服务器能够监听的必要条件，就如同此时的座机能够处于监听电话的状态。

//开始监听

```
retVal = listen(sServer, 1);
if(SOCKET_ERROR == retVal)
{
    cout << "listen failed!" << endl;
    closesocket(sServer);   //关闭套接字
    WSACleanup();           //释放套接字资源;
    return -1;
}
```

图 3 代码实现

4.4 客户端创建套接字

SOCK_RAW（原始套接字）;

Protocol 代表了数据传输协议，一般情况下在前两个参数之后便能确定该参数，而这个参数一般使用如下三种协议:

IPPROTO_TCP 代表使用 TCP 协议

IPPROTO_UDP 代表使用 UDP 协议

IPPROTO_RM 代表使用实际通用组播协议

4.2 服务器 bind（）

类似于给购买的电话绑定上一个固定的电话号码，在网络通信中，必然要确定两个通信双方。而 bind()正是这个道理，他会与 IP 地址和端口号进行绑定，通过对这两者的绑定服务器便能唯一确定。当服务器唯一确定之后，用户一方便可以对其进行访问。

//绑定套接字
与服务器创建套接字一致

//套接字创建

```
sHost = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if(INVALID_SOCKET == sHost)
{
 cout << "socket failed!" << endl;
 WSACleanup();//释放套接字资源
 return  -1;
}
```

图 4 代码实现

4.5 客户端 connect()

客户端要想与服务器进行连接，这个过程中需要确定服务器的端口号以及 IP 地址[5]。如果客户端调用了这个函数之后，将会请求与服务器建立连接。也只有通过这个函数才能够使的服务器真正的发挥本质的功能，就好比如果一台座机，纵使各项硬件设施软件设施都是齐全的但如果没有其他电话去拨通的话也是形同虚设。同样的道理倘若一方不调用该函数，另外一方 accept 则会一直等待。这个过程类似于移动电话使用者向服务器座机拨打号码进行连接。

//连接服务器

```
retVal=connect(sHost,(LPSOCKADDR)&servAddr, sizeof(servAddr));
if(SOCKET_ERROR == retVal)
{
    cout << "connect failed!" << endl;
    closesocket(sHost); //关闭套接字
    WSACleanup(); //释放套接字资源
    system("pause");
    return -1;
}
```

图 5 代码实现

4.6 服务器端 accept()

这个函数会使服务器去接收客户端发出的连接，通过这个函数能够知道发出连接的客户端的信息，好比座机使用者知道了客户端移动电话打了的号码是多少。这个函数默认情况下会阻塞进程的继续执行，通俗的讲也就是说如果没有用户去唤醒它他就会一直在那里阻止后续过程的继续。因此也就意味着倘若没有了一方的请求，另一方就会一直等待。在该函数之后，服务器会回复客户端告诉客户端我也想建立连接,这个过程类似于服务器座机收到了电话并准备接通。

//接收客户端请求

```
sClient = accept(sServer,(sockaddr FAR*)&addrClient, &addrClientlen);
if(INVALID_SOCKET == sClient)
{
    cout << "accept failed!" << endl;
    closesocket(sServer);    //关闭套接字
    WSACleanup();            //释放套接字资源;
    return -1;
}
```

图 6 代码实现

4.7 连接成功，客户端 send()，服务器端 recv()

当两者连接成功后，两者便能够进行信息交流，类似与移动电话使用者向服务器座机通话，而座机的使用者接收信息，此时移动电话使用者在说，座机使用者在听。

//客户端发送消息

```
retVal = send(sHost, buf, strlen(buf), 0);
if (SOCKET_ERROR == retVal)
{
  cout << "send failed!" << endl;
  closesocket(sHost); //关闭套接字
  WSACleanup(); //释放套接字资源
  return -1;
    }
```

图 7 代码实现

//服务器端接收消息

```
retVal = recv(sClient, buf, BUF_SIZE, 0);
if (SOCKET_ERROR == retVal)
{
    cout << "recv failed!" << endl;
    closesocket(sServer);    //关闭套接字
    closesocket(sClient);    //关闭套接字
    WSACleanup();            //释放套接字资源;
    return -1;
}
```

图 8 代码实现

4.8 服务器端 send(),客户端 recv()

服务器端在接收信息后，也能够向客户端发送信息，客户端此时能够接收信息，这个过程中客户端与服务器端交换了角色，类似与座机的使用者对通话发起者进行回应，而通话发起者此时接收信息，此时移动电话使用者在听，座机使用者在说。

4.9 服务器关闭 close()，客户端关闭 close()

```
closesocket(sServer);     //关闭套接字
closesocket(sClient);     //关闭套接字
WSACleanup();             //释放套接字资源
```

5.结束语

当使用 Socket 接口对网络通信编程时, Socket 是网络通信过程中端点的抽象表示[6]。简而言之，就如同两台电话，Socket 好比为两台电话添加了能够唯一标识的电话号码，通过此唯一的电话号码，客户端电话能够唯一确定服务器电话，从而与之建立连接，进而与此进行通信。而在这个过程中，三次握手就像两个间谍之间互相地再三确认身份，直到身份真正确认才进行信息的交换。而如今使用新媒体平台与好友通信、在局域网中玩游戏，这些功能的实现都可以利用这样技术和思想。

参考文献

[1]刘骏,颜钢锋.基于 Socket 的网络编程技术及其实现[J].江南大学学报,2004(03):249-251.
[2]石磊.基于 TCP/IP 协议的管道内行走机器人设计[J].黑河学院学报,2019(01):208-209.
[3]牟综磊.采用 Socket 和多线程实现网络高并发服务器[J].电脑编程技巧与维护,2019(02):148-149+157.
[4]蒋达.基于 Socket 的网络接口编程[J].办公自动化,2018,23(23):29-30+32.
[5]梁璐. 基于网络信息内容的分析还原系统研究与实现[D].北京交通大学,2009.
[6]王晓鹏.TCP/IP 下的 Socket 及 Winsock 通信机制[J].航空计算技术,2004(02):126-128.

作者简介

第一作者：曾伟（1997-），男，汉，四川省西昌市，本科，四川大学锦城学院，研究方向：C++。
第二作者（通讯作者）：鲍正德（1989-），男，汉，黑龙江哈尔滨，研究生，四川大学锦城学院，研究方向：电子商务。
第三作者：唐娅雯（1999-），女，汉，四川省资阳市，本科，四川大学锦城学院，研究方向：信息管理、J2EE