利用中间件 Socket 实现多客户机端底层网络通信

Application of Low Network Communication for Multi – Client Based on Middleware Socket

朱 蓉 （嘉兴学院 信息工程学院 314001）

摘要：本文介绍了底层网络通信的含义，给出了基于 TCP/IP 协议的中间件 Socket 的运行机制，利用 Java 语言提供的套接字 Socket 类设计了面向客户机/服务器模式的网络通信过程，通过引入线程的手段解决服务器端与多客户端之间互相信息传输问题，并编程实现这一过程。

关键词：底层网络通信 TCP/IP 协议 中间件 Socket 多客户机端

1 引言

随着互联网技术日益迅速发展，需要提供一种能使存在于网络中不同平台上的计算机间进行信息传输的标准约定，TCP/IP 协议具有的强大异种机联网功能正好满足这种需求。TCP/IP 是一组协议群，其核心为传输层 TCP 协议和网络层 IP 协议，其中，TCP 协议提供一种可靠的、有连接的数据流服务；IP 协议能准确标识网络中任意计算机的 IP 地址，由于这两种协议均对应于 ISO/OSI 网络参考模型 7 层协议的底层部分，因此，常把利用 TCP/IP 协议解决的网络信息传输问题称为底层网络通信。

2 中间件 Socket 运行机制

TCP/IP 协议不是一种应用程序，它不提供直接的用户服务，是通过中间件 Socket 来实现的。Socket 指套接字，是一种网络进程间的通信机制，它由两部分组成，即：IP 地址和端口号。IP 地址用于确定应用程序所在主机的网络地址，一般由四个 8 位的二进制数组成，每个数字的范围在 0 到 255 之间，中间以小数点分隔，如：127.0.0.1 为本地主机 IP 地址。端口号是应用程序进行信息传输的端口地址，可用于区分不同的服务进程，如：HTTP 服务端口号为 80、FTP 服务端口号为 21，一般 1—1024 为系统保留的端口号。

在基于 TCP/IP 协议的网络通信中，套接字 Socket 面向客户机/服务器模式按照 4 个步骤运行，即：建立 Socket、创建连接到 Socket 的输入流和输出流、按照指定协议对 Socket 进行读写操作、关闭 Socket。在开始网络通信前，套接字 Socket 首先在服务器端建立 Socket 对象，并在指定端口上进行监听，当接收到来自客户机端的连接请求时，由 Socket 为相互通信的两端建立一条可靠的专用虚拟传输通道，同时在 Socket 两端分别建立输入流和输出流以进行双向信息交流，当两端通信结束后拆除套接字 Socket 连接通道，具体如图 1 所示：

可见，当网络上计算机间进行信息传输时，每建立一个连接通道对应两个套接字 Socket，一个套接字 Socket 位于通道的一端。套接字 Socket 独特而便捷的运行机制使得网络间位于不同地域、不同机型上的计算机能进行信息传输和资源共享，是一种非常有效的网络应用程序接口。

3 Java 语言中的 Socket API 类

Java 语言是一种目前广为流行的网络编程语言，它提供的与网络操作有关的类大多存放在 java.net 包中，其中，以 Socket 类与 ServerSocket 类最为有用。

3.1 Socket 类和 ServerSocket 类

(1) Socket 类

构造方法：

Socket (String host, int port) throws UnknownHostException, IOException

建立套接字，连接到指定名称主机 (host) 的指定端口 (port)。

Socket (InetAddress address, int port) throws IOException

建立套接字，连接到指定 IP 地址(address)的指定端口(port)。

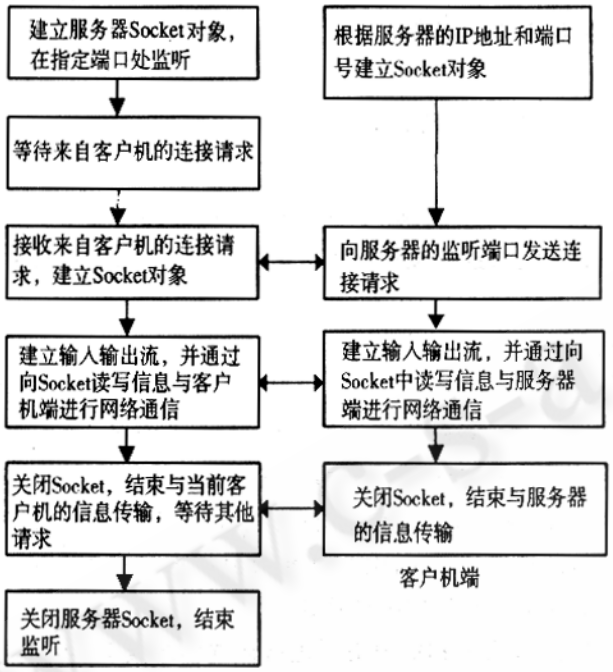

图 1　基于 TCP/IP 协议的中间件-Socket 运行机制

输入流、输出流建立方法：

public InputStream getInputStream( ) throws IOException

返回从套接字读入数据的输入流。

public OutputStream getOutputStream ( ) throws IOException

返回向套接字进行写数据操作的输出流。

关闭方法：

public void close( ) throws IOException

关闭套接字

(2) ServerSocket 类

构造方法：

ServerSocket( ) throws IOException

建立服务器套接字，未绑定端口。

ServerSocket(int port) throws IOException

建立服务器套接字，绑定在指定端口(port)。

等待连接请求方法：

public Socket accept( ) throws IOException

监听来自客户机的连接请求，方法阻塞，直到连接成功。

关闭方法：

public void close( ) throws IOEception

关闭服务器套接字。

3.2 服务器端 Socket 使用

(1) 调用 socket( )建立一个套接字；

(2) 调用 bind( )将 IP 地址和端口号绑定到监听套接字；

(3) 在指定端口处调用 listen( )监听来自客户机的连接请求；

(4) 调用 accept( )接收客户机的连接请求；

(5) 调用 getInputStream( )输入流接收信息；

(6) 调用 getOutputStream( )输出流发送信息；

(7) 调用 close( )关闭套接字。

3.3 客户机端 Socket 使用

(1) 调用 socket( )建立一个套接字；

(2) 调用 connect( )建立一个连接；

(3) 调用 getOutputStream( )输出流发送信息；

(4) 调用 getInputStream( )输入流接收信息；

(5) 调用 close( )关闭套接字。

4 服务器与多客户机间网络通信

客户机/服务器模式采用“请求－回应”方式进行网络间信息的传输，其中，提出请求的一端为客户机，提供回应服务的另一端为服务器。客户机是主动方，首先向服务器发出连接请求；服务器是被动方，一直处于监听状态，等待被来自客户机的请求服务所触发并做出响应。客户机和服务器都是应用程序，可以位于同一物理机器上，也可以位于不同物理机器上。一个客户机可以对应多个服务器进程，如：服务器同时调用 FTP 服务进程和 HTTP 服务进程；同样，一个服务器进程也可以对应多个客户机，如：能同时在线连接多个客户机的网上论坛，后者可参照图 2 所示。

4.1 服务器端并发服务实现

为解决一个服务器与多个客户机间进行网络通信的问题，可以利用 Java 语言提供的多线程来实现，即对每个客户机都通过生成一个独立的线程来处理。服务器在无连接请求到来时，一直处于等待状态，并随时

准备为多个客户机提供服务，当某一客户机向服务器发出连接请求时，服务器主线程立即建立一个新的套接字连接，同时产生一个子线程来处理该客户机的请求，当服务器启动完子线程后马上又回到监听状态，等待下一个客户机的连接请求，服务器的这种方式称为并发服务。

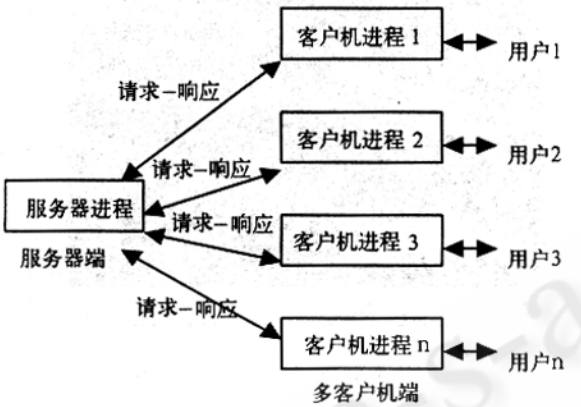

图 2　服务器与多客户机间网络通信模式

(1) 服务器主线程程序实现

```
import java.io.*;
import java.net.*;
public class MServer {
    public static void main(String args[]) throws IOException
    {ServerSocket sersoc=null; //服务器套接字对象 sersoc 初始化
     int cn=0; //客户机计数器清零
     boolean listened=true; //测试是否监听
     sersoc=new ServerSocket(6868);//绑定端口号为 6868
     while(listened) //循环监听
     { new Mthread(sersoc.accept(),cn).start();
     //监听到来自客户机的请求，启动子线程
       cn++; //客户机计数器加 1
       System.out.println("客户机"+cn+"号已连接.");}
     sersoc.close();}} //关闭服务器套接字
```

(2) 子线程程序实现

```
import java.io.*;
import java.net.*;
public class Mthread extends Thread{
    Socket fsoc=null; //套接字对象 fsoc 初始化
    BufferedReader fin=null; //套接字输入流对象 fin 初始化
    BufferedReader fsin=null; //标准输入流对象 fsin 初始化
    PrintWriter fout=null; //套接字输出流对象 fout 初始化
    String fcin=null; //由 fin 生成的字符串对象 fcin 初始化
    String fcsin=null; //由 fsin 生成的字符串对象 fcsin 初始化
    int cn=0; //客户机计数器初始化
    Mthread(Socket fsoc,int num)
     { this.fsoc=fsoc; //当前客户机 Socket 对象赋值
       cn=num+1;} //当前客户机个数
    public void run(){
    try{ fin=new BufferedReader(new InputStreamReader(fsoc.getInputStream()));
     //由 Socket 对象创建输入流，并得到 BufferedReader 对象
     fsin=new BufferedReader(new InputStreamReader(System.in));
     //由系统标准输入设备得到 BufferedReader 对象
     fout=new PrintWriter(fsoc.getOutputStream());
     //由 Socket 对象创建输出流，并得到 BufferedReader 对象
     fout.println("欢迎您!...");fout.flush();
     //向客户机写信息，并刷新输出流，使客户机马上接收到信息
     fcin=fin.readLine();
     //从客户机读入信息
     while(!fcin.equals("结束通信"))
     //判断客户机是否结束网络通信
     {System.out.println("客户机"+cn+"号发出请求:"+fcin);
     System.out.println("请向客户机"+cn+"
```

```
        号回应信息:");
        fcsin = fsin.readLine();
        //从标准输入设备读入信息
        fout.println(fcsin);fout.flush();
        //向客户机写信息,并刷新输出流,使客户机
        马上接收到信息
        fcin = fin.readLine();}
        //继续从客户机读入信息
        if(fcin.equals("结束通信"))
          {System.out.println("客户机 "+cn+" 号
           结束网络通信.");
           fout.close(); //关闭输出流
           fin.close(); //关闭输入流
           fsoc.close();}} //关闭套接字
      catch(Exception e){System.out.println("出错
信息:"+e);}
        //捕捉异常,将异常结果输出到标准输出设备
上
        }}
```

4.2 客户机端程序实现

与服务器端相比,客户机端程序不必考虑已连接客户机数量的多少,只要给出一个客户机与服务器建立套接字相关步骤的程序实现即可。代码略。

4.3 运行结果

(1) 服务器端程序运行结果(如图 3 所示)

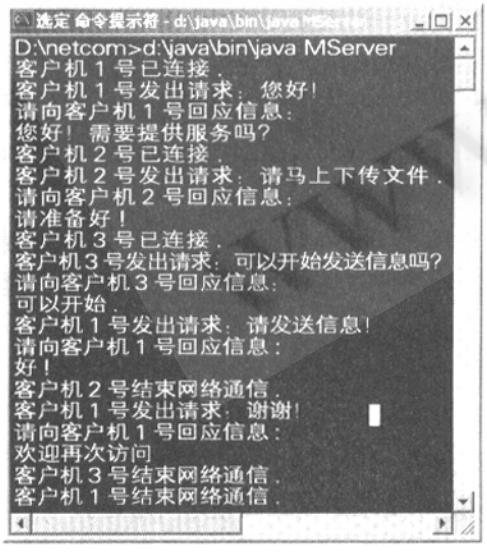

图 3　服务器端程序运行结果

(2) 多客户机端程序运行结果(如图 4、图 5、图 6 所示)

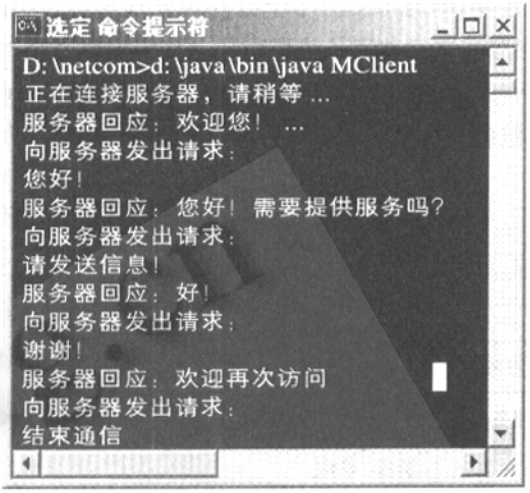

图 4　多客户机端(客户机 1 号)运行结果

图 5　多客户机端(客户机 2 号)运行结果

5　结束语

套接字 Socket 是 TCP/IP 协议与应用程序联系的中间件,同时也是 Java 语言提供的网络应用程序接口(API),它将有关底层网络通信的各种技术封装在一起,提供给用户快速、高效的开发手段,广泛应用于面向客户机/服务器的分布式模式中。当然,这种套接字 Socket 运行机制同样也可实现基于浏览器/服务器模式的网络通信。

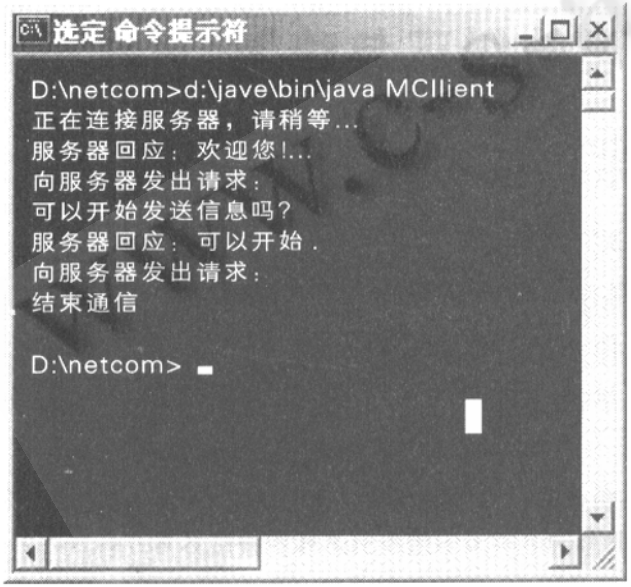

图 6 多客户机端(客户机 3 号)运行结果

参考文献

1 Timothy Budd. 面向对象 Java 编程思想[M]，清华大学出版社，2002. 322 – 330。

2 殷兆麟等，Java 网络编程[M]，北京：国防工业出版社，2001. 105 – 135。

3 《电脑编程与维护》杂志社，Java 编程精选集锦[M]，北京：科技出版社，2003. 140 – 147。

4 谷岩等，利用 Java 的 Socket 编程机制实现在线交谈[J]，计算机工程与设计，2004(6). 944 – 947。

5 何进等，基于 Socket 的 TCP/IP 网络通讯模式研究[J]，计算机应用研究，2001(8). 134 – 136。

6 谢晓芹等，基于 Socket 的网络编程[J]，南昌大学学报，1997(12). 337 – 340。

7 牛立华等，浅析用 Socket API 实现的网络通信[J]，计算机时代，2002(8). 37 – 40。

8 刘奕等，基于 Socket 的 Java 语言网络通讯机制和程序