实验三　Socket 网络编程实验

1.实验目的

两人合作设计一个 TCP 服务器，可以从客户端获得一个字符串，把字符串全部转为大写字母之后返回给客户端。

2.实验内容

2.1 实验思路

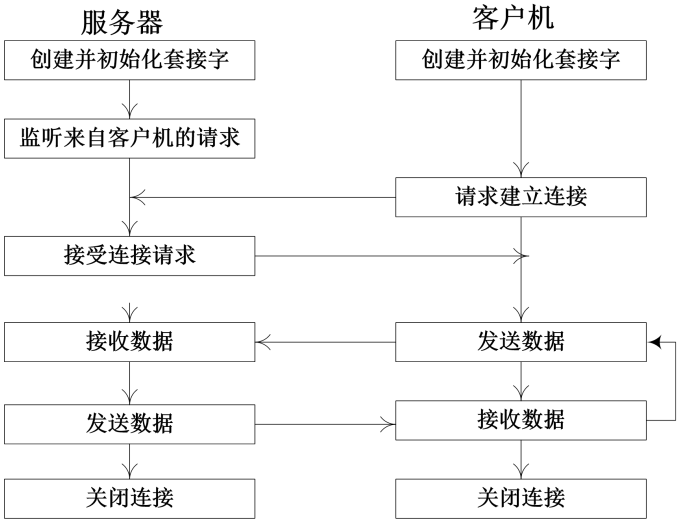


图3 流式套接字（有连接通信)编程

此次实验分为客户端和服务器的编写。我主要负责编写客户端代码，周文斌同学负责服务器端的代码编写。

主要思路是两个主机处于同一局域网内，然后设置服务器的端口号和 ip 地址。首先开启服务器，创建并且初始化套接字，然后就阻塞等待来自客户端的请求。打开客户端，创建并且初始化套接字，然后请求与服务器建立连接，服务器响应请求，客户端就发送字符串，服务器接收字符串进行大小写转化，然后向客

户端发送数据，客户端接收数据，完成数据的传输，客户端关闭连接。服务器在等待一段时间没有客户端连接的时候就关闭连接。在以上思路的基础上，客户端和服务器端都进行了多线程的优化。

2.2 实验代码

客户端 ChatClient.java 具体代码如下所示：

```
import java.io.*;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;
import java.util.Date;

// 客户端类
public class ChatClient implements Runnable {//创建类
    private String host = "127.0.0.1";
    private int port = 4088;

    @Override
    public void run() {
        try {
            //System.out.println("客户端准备连接");//输出来自客户端的信息
            // 连接到服务器
            Socket socket = new Socket(host, port);//创建 Socket 类对
象


            try {

                DataInputStream in = new DataInputStream(socket
                        .getInputStream());// 读取服务器端传过来信息的
DataInputStream
                DataOutputStream out = new DataOutputStream(socket
                        .getOutputStream());// 向服务器端发送信息的
DataOutputStream
                long AverageTime = 0;
                for(int i = 50 ; i > 0;i--) {//每个线程会发送 50 次字符串
                    String send = "auewebvnwqiwhvwecsjooooooohuhuffffjgu
gugfitfycxh";
                   // System.out.println("客户端: " + send);
```

```
                    // 把从控制台得到的信息传送给服务器

                    long begin = new Date().getTime();
                    //System.out.println("发送时间是："+ begin);
                    out.writeUTF(send);//将客户端的信息传递给服务
器
                    String accpet = in.readUTF();// 读取来自服务器的信息

                    long end = new Date().getTime();
                   // System.out.println("接收时间是："+ end);

                    System.out.println("服务端："+ accpet);
                    long time = end - begin;
                    //System.out.println("往返时间是："+ time);
                    AverageTime += time;
                }
                AverageTime /= 50;
                System.out.println(Thread.currentThread().getName()+"平
均往返时间是："+AverageTime );



            } finally {
                socket.close();//关闭 Socket 监听
            }
        } catch (IOException e) {//捕获异常
            e.printStackTrace();
        }
    }

}
```

客户端 Client.java 具体代码如下所示：

```
public class Client {

    public static void main(String[] args) {
        int count = 1000;//线程数是 1000
        while (count!=0) {
            ChatClient thread = new  ChatClient();
            new Thread(thread).start();//开启线程
            count--;
        }
    }
}
```

服务器端 MyThread. java 具体代码如下所示：

```
package com.nethw;

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;

public class MyThread implements Runnable{
    private Socket clientSocket;
    /**
     * 通过构造方法来传递客户的信息
     */
    public MyThread(Socket clientSocket) {
        this.clientSocket = clientSocket;
    }
    @Override
    public void run() {
        DataInputStream inputStream = null;
        DataOutputStream outputStream = null;
        try {
            System.out.println("连接成功");
            outputStream = new DataOutputStream(clientSocket.getOutputS
tream());
            inputStream = new DataInputStream(clientSocket.getInputStre
am());

            for(int i = 50; i>0 ; i--) {
                String accpet = inputStream.readUTF();// 读取来自客户端的
信息
                System.out.println(accpet);//输出来自客户端的信息
                String NewAccpet = Thread.currentThread().getName() + ":
" + accpet.toUpperCase() ;
                System.out.println("服务器回答"+ NewAccpet);//输出提示信
息
                outputStream.writeUTF(NewAccpet);//把服务器端的输入发给客
户端
            }
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            try {
                inputStream.close();
                outputStream.close();
                clientSocket.close();
```

```
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

    }
}
```

服务器端 Server. java 具体代码如下所示：

```
package com.nethw;
import jdk.management.resource.internal.inst.SocketOutputStreamRMHooks;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    public static void main(String[] args) throws IOException {
        try {
            ServerSocket serverSocket = new ServerSocket(4088);
            while (true) {
                Socket clientSocket = serverSocket.accept();
                MyThread thread = new MyThread(clientSocket);
                new Thread(thread).start();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

2.3 实验测试和结果

2.3.1 往返时间测试

```
                long AverageTime = 0;
                for(int i = 5 ; i > 0;i--) {
                    String send = "auewebvnwqiwhvwecsjooooooohuhuffffj";
                    System.out.println("客户端： " + send);
                    // 把从控制台得到的信息传送给服务器

                    long begin = new Date().getTime();
                    System.out.println("发送时间是："+ begin);
                    out.writeUTF(send);//将客户端的信息传递给服务
器
                    String accpet = in.readUTF();// 读取来自服务器的信息

                    long end = new Date().getTime();
                    System.out.println("接收时间是："+ end);

                    System.out.println("服务端："+ accpet);
                    long time = end - begin;
                    System.out.println("往返时间是："+ time);
                    AverageTime += time;
                }
                AverageTime /= 5;
                      System.out.println("平均往返时间是：
"+AverageTime );
```

在发送数据前记录下此时的系统时间 begin,然后接收到服务器端的数据后再次记录系统时间 end,两数相减就是一次数据传送的往返时间，这里每个线程发送五次字符串，取往返时间的平均数。由测试得知，只有一个客户端时的平均往返时间是 3ms。

```
PS D:\java_src\.vscode>  cd 'd:\java_src\.vscode'; & 'c:\Users\86188\.vscode\extensions\vscjava.vscode-j
88\AppData\Roaming\Code\User\workspaceStorage\8fdc4965de25742edb140cc5396d814d\redhat.java\jdt_ws\jdt.ls
\bin' 'Client'
客户端准备连接
客户端：auewebvnwqiwhvwecsjooooooohuhuffffjgugugfitfycxh
发送时间是：1604491767418
接收时间是：1604491767418
服务端：AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXHThread-1
往返时间是：0
客户端：auewebvnwqiwhvwecsjooooooohuhuffffjgugugfitfycxh
发送时间是：1604491767422
接收时间是：1604491767422
服务端：AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXHThread-1
往返时间是：0
客户端：auewebvnwqiwhvwecsjooooooohuhuffffjgugugfitfycxh
发送时间是：1604491767426
接收时间是：1604491767434
服务端：AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXHThread-1
往返时间是：8
客户端：auewebvnwqiwhvwecsjooooooohuhuffffjgugugfitfycxh
发送时间是：1604491767434
接收时间是：1604491767438
服务端：AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXHThread-1
往返时间是：4
客户端：auewebvnwqiwhvwecsjooooooohuhuffffjgugugfitfycxh
发送时间是：1604491767438
接收时间是：1604491767442
服务端：AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXHThread-1
往返时间是：4
平均往返时间是：3
```

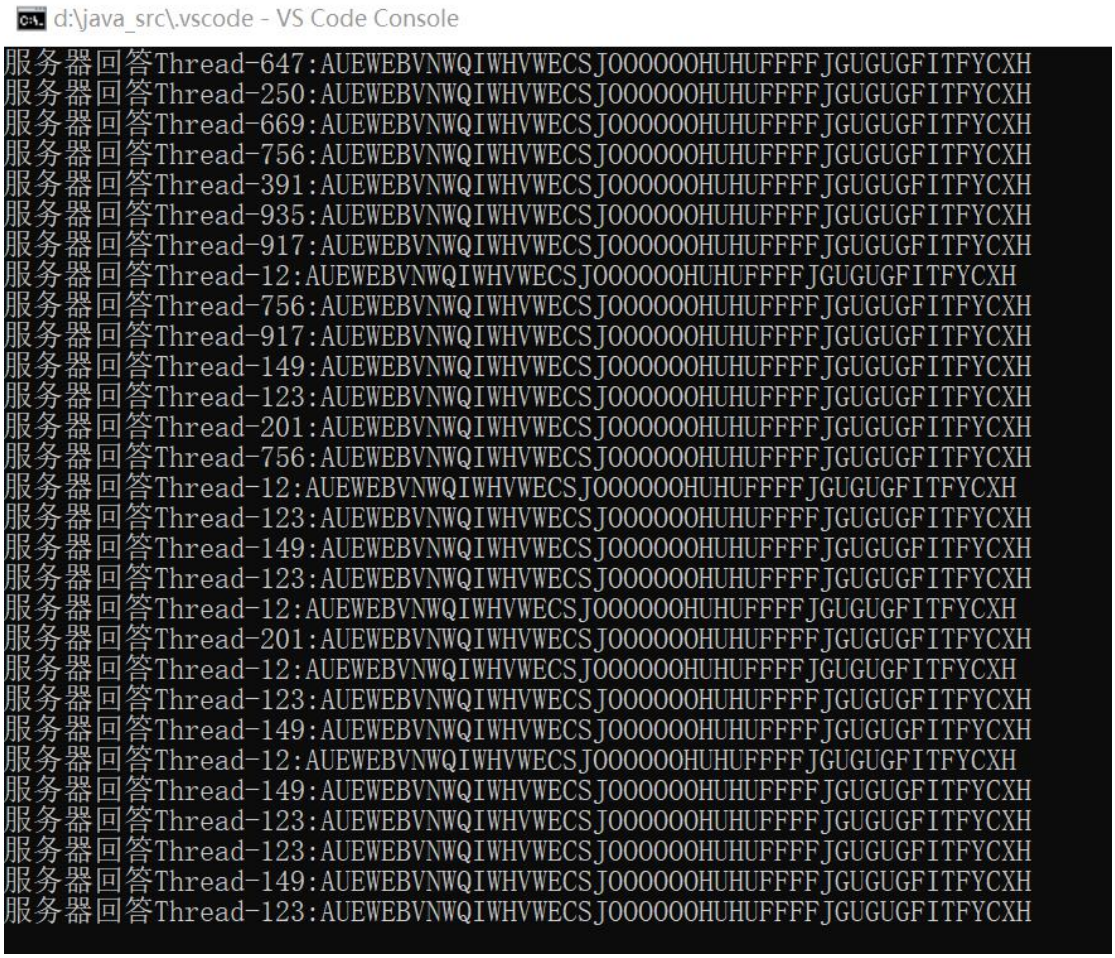
2.3.2 压力测试

开启 1000 个客户端线程，每个线程向服务器发送 100 个字符串。可以看到服务器依然可以正常运行，有一定的抗压能力。

客户端运行结果如下：

```
d:\java_src\.vscode - VS Code Console
Thread-842平均往返时间是：50
服务端：Thread-756:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-917:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-12:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-123:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
Thread-680平均往返时间是：51
服务端：Thread-149:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-123:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-12:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-756:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-201:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-12:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
Thread-599平均往返时间是：55
服务端：Thread-201:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-123:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-149:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
Thread-687平均往返时间是：57
服务端：Thread-12:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-149:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-123:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-12:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-123:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-149:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
Thread-2平均往返时间是：58
服务端：Thread-123:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
服务端：Thread-149:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
Thread-377平均往返时间是：59
服务端：Thread-123:AUEWEBVNWQIWHVWECSJOOOOOOHUHUFFFFJGUGUGFITFYCXH
Thread-313平均往返时间是：60
Press any key to continue . . .
```

服务器端运行结果如下：



3.实验总结

3.1 socket 的概念

Socket，又叫套接字，是一种软件形式的抽象，用于表达两台机器间一个连接的“终端”。服务端的 socket 在服务端机器上特定的端口进行等待，客户端的 socket 绑定了服务端的 IP 地址以及它正在监听着的端口，所以两者通过 IP 和端口连接起来，就如同一条虚拟的通道。

Socket 的类别有：

（1）TCP 流式套接字(SOCK_STREAM)提供了一个面向连接、可靠的数据传输服

务，数据无差错、无重复地发送，且按发送顺序接收。内设流量控制，避免数据流超限；数据被看作是字节流，无长度限制。文件传送协议（FTP）即使用流式套接字。

（2）数据报式套接字(SOCK_DGRAM)提供了一个无连接服务。数据包以独立包形式被发送，不提供无错保证，数据可能丢失或重复，并且接收顺序混乱。网络文件系统（NFS）使用数据报式套接字。

（3）原始式套接字(SOCK_RAW)该接口允许对较低层协议，如 IP、ICMP 直接访问。常用于检验新的协议实现或访问现有服务中配置的新设备。

3.2 客户端和服务器交互的流程

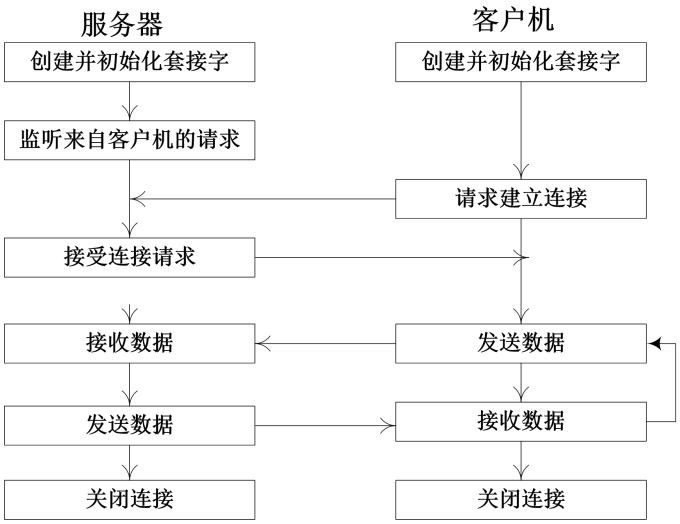


图3 流式套接字（有连接通信)编程

3.3 数据存储的方式

计算机的数据存储一般分为大端存储和小端存储。计算机内部一般是小端存储，即低地址存放低字节，高地址存放高字节。而 TCP 传输的过程一般是大端存储数据。

3.4 同一个服务器和多个客户端建立连接时 socket 是一个还是多个?

这样的情况是多个 socket,内核是以一个五元信息组来标识不同的 socket,即（源地址、源端口、目的地址、目的端口、协议号）。任何一个不同都不能算同一个 socket,因此每次建立连接都是一个新的 socket。