Socket Programming Report

File Share P2P for Computer Networking 2016 Spring

Name:ShenJiyuan St.No.:5130309194 E-mail:sheryalyuan@126.com

***Abstract*—This report describes a socket programming about a simple file share application using TCP Socket APIs in detail. Here, the Java Socket API is chosen as the implementing tool.**

***Index Terms*—Socket, C/S, TCP, file share, P2P, client, server.**

I. Recall Problem

Implement C/S model:

1) Server listens to a given port(>1024);

2) The client initiates a TCP connection to the server (hostname or IP address of the server as the input,default port numbers);

3) The client send a request to download a file/text;

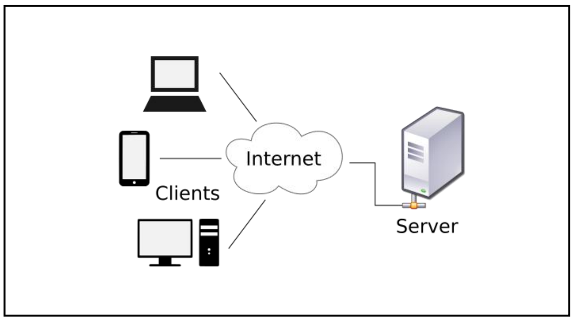
4) The server respond with the file/text;

5) The client save the file to local director.

6) Repeat step 3) 4) 5) until 'Esc' is pressed, client tear down the TCP connection.

II. Introduction of Basic Concepts

Network Socket: An ndpoint of a connection across a computer network. More precisely, a socket is a handle(abstract reference) that a local program can pass to the networking application programming interface to use the connection. Internally often integers, which identify which connection to use. Socket API: An application programming interface, usually provided by the operating system, that allow application programs to control and use network standard, sockets are a form of file handle. A socket address is the combination of an IP address and a port number.



C/S Model(client-server model): A distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function.

III. On Java API Functions

'java.io' includes all functions provided for input and output stream, for UI and file.'java.net' includes that for socket.

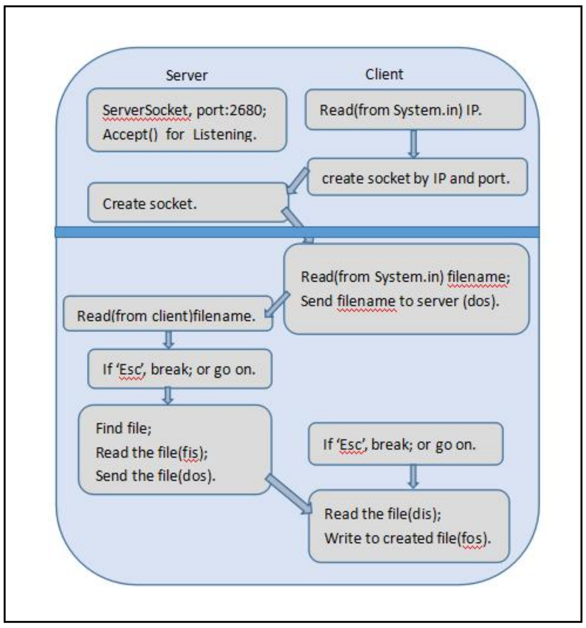
In the following table, I list all the relating functions in the two packages related to socket programming for file sharing and used in my program.

java.io	
BufferedReader	change 'InputStreamReader' into buffer
readLine()	read the buffer content(better efficiency)
dataInputStream	file data stream, for client read(here)
read()	client read data from stream
dataOutputStream	file data stream, for server write(here)
writeBytes()	server write data to stream
fileInputStream	file data stream, for server read(here)
fileOutputStream	file data stream, for client write(here)
write()	client write data to file
java.net	
Socket	socket for both server and client
connect()	client connect to server by socket
ServerSocket	listening and controls in server
accept()	server listen to a port and get connection

IV. Programming

In this part, I will show you the whole programming process and detailed relations between the socket process.

A. OverView (important)


Server
Client
ServerSocket, port:2680; Accept() for Listening.
Read(from System.in) IP.
create socket by IP and port.
Create socket.
Read(from System.in) filename; Send filename to server (dos).
Read(from client)filename.
If 'Esc', break; or go on.
Find file; Read the file(fis); Send the file(dos).
If 'Esc', break; or go on.
Read the file(dis); Write to created file(fos).

Since for this problem, step 3) 4) 5) are exactly loop steps. And in the above figure, I draw only one round, leaving the loop steps separated by a coarse blue line(the latter steps).

B. Detailed Description on Key Codes

a) Environment: Eclipse, JavaSE-1.8

(1) New a Java Program in Eclipse, named 'tcp'.

(2) New 2 classes, named 'TCPClient', 'TCPServer'.

b) TCPClient

- Read(from System.in) IP & create socket

```
BufferedReader inFromUser=new
BufferedReader(new
InputStreamReader(System.in));
//输入 ip,如 59.78.29.78
String hostname=inFromUser.readLine();
socket = new Socket();
socket.connect(new    InetSocketAddress
(hostname, 2680),10 * 1000);
```

- Read(from System.in) filename & send to server

```
dos   =  new  DataOutputStream  (socket.
getOutputStream ());

String filename=inFromUser.readLine();
dos.writeBytes(filename+"\n");
```

- Read(from DataInputStream) file & Write to file

```
if ((length = dis.read(inputByte, 0,
inputByte.length)) > 0)
{
    System.out.println(length);
    fos.write(inputByte, 0, length);
}
```

c) TCPServer

- Create ServerSocket & Listening & create socket

```
final   ServerSocket   server   =   new
ServerSocket(2680);

try {
Socket socket = server.accept();
```

- Read(from client) filename

```
BufferedReader          inFromClient=new
BufferedReader(new     InputStreamReader
(socket.getInputStream()));

String
fileName=inFromClient.readLine();
```

- Find file (by filename)

```
File file = new File(fileName);
FileInputStream fis = null;
fis = new FileInputStream(file);
```

- Read the file & Write the file

```
while ((length = fis.read(sendBytes, 0,
sendBytes.length)) > 0) {
dos.write(sendBytes, 0, length);
```

V. FILELIST AND RUN

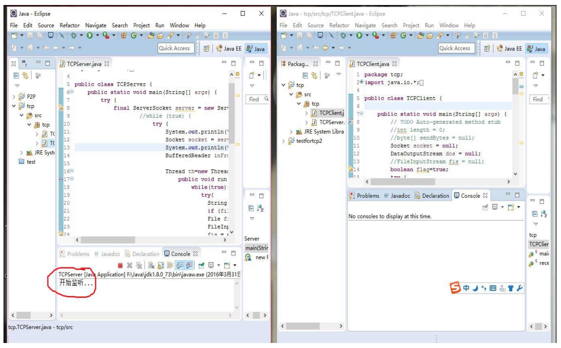
A. FileList and Run

My program is written in Java, and all the codes are exactly the wo files 'TCPClient.java' and 'TCPServer.java'. And they are all in the 'src' filefold.

Press the 'run' in your java_program_platform.

B. ShotScreens when Running after immediate compiling

- Open Server (Server Listening)



```
public class TCPServer {
    public static void main(String[] args) {
        try {
            final ServerSocket server = new Ser
            //while (true) {
                try {
                    System.out.println('
                    Socket socket = ser
                    System.out.println('
                    BufferedReader inFr

                    Thread th=new Threa
                        public void run
                            while(true)
                                try{
                                    String
                                    if (fi
                                    File f
                                    FileIn
```

```
TCPServer [Java Application] F:\Java\jdk1.8.0_73\bin\javaw.exe (2016年3月31E
开始监听...
```

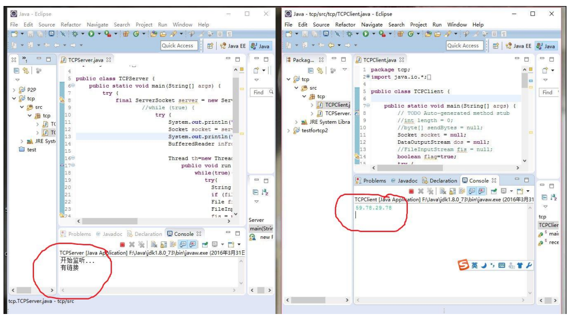
```
package tcp;
import java.io.*;

public class TCPClient {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //int length = 0;
        //byte[] sendBytes = null;
        Socket socket = null;
        DataOutputStream dos = null;
        //FileInputStream fis = null;
        boolean flag=true;
```

```
No consoles to display at this time.
```

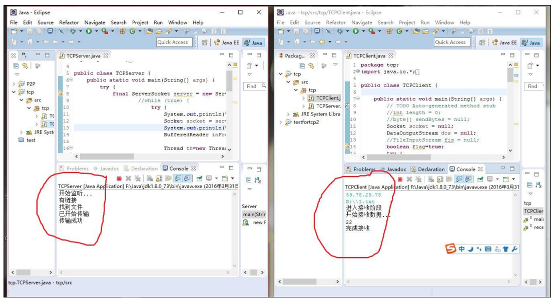
- Open Client , User input IP (connect)



```
TCPServer [Java Application] F:\Java\jdk1.8.0_73\bin\javaw.exe (2016年3月31E
开始监听...
有链接
```

```
TCPClient [Java Application] F:\Java\jdk1.8.0_73\bin\javaw.exe (2016年3月31
59.78.29.78
```

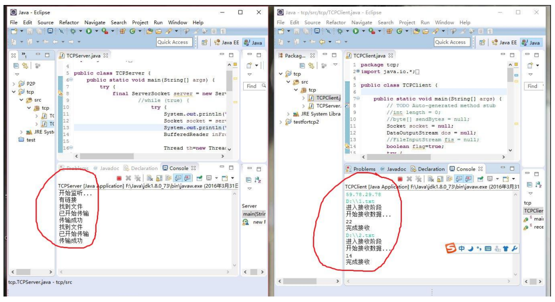
- User input filePath (get the file)



```
TCPServer [Java Application] F:\Java\jdk1.8.0_73\bin\javaw.exe (2016年3月31E
开始监听...
有链接
找到文件
已开始传输
传输成功
```

```
TCPClient [Java Application] F:\Java\jdk1.8.0_73\bin\javaw.exe (2016年3月31
59.78.29.78
D:\\1.txt
进入接收阶段
开始接收数据...
22
完成接收
```

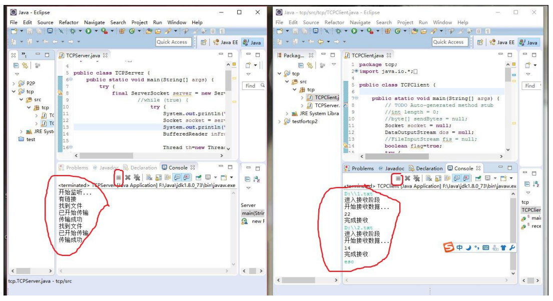
- User input another filePath (get the file)



```
TCPServer [Java Application] F:\Java\jdk1.8.0_73\bin\javaw.exe (2016年3月31E
开始监听...
有链接
找到文件
已开始传输
传输成功
找到文件
已开始传输
传输成功
```

```
TCPClient [Java Application] F:\Java\jdk1.8.0_73\bin\javaw.exe (2016年3月31
59.78.29.78
D:\\1.txt
进入接收阶段
开始接收数据...
22
完成接收
D:\\2.txt
进入接收阶段
开始接收数据...
14
完成接收
```

- User input esc (both server and client exit)

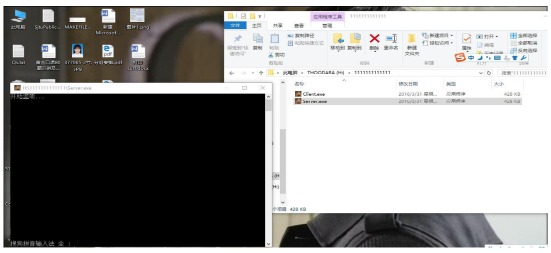


```
<terminated> TCPServer [Java Application] F:\Java\jdk1.8.0_73\bin\javaw.exe
开始监听...
有链接
找到文件
已开始传输
传输成功
找到文件
已开始传输
传输成功
```

```
<terminated> TCPClient [Java Application] F:\Java\jdk1.8.0_73\bin\javaw.exe
D:\\1.txt
进入接收阶段
开始接收数据...
22
完成接收
D:\\2.txt
进入接收阶段
开始接收数据...
14
完成接收
esc
```

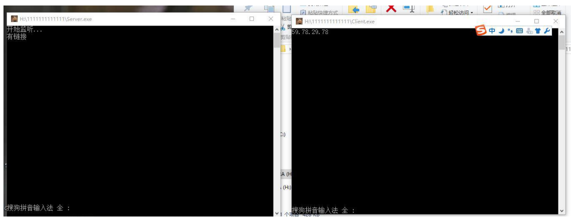
C. ShotScreens when Running exetable files

- Open Server (Server Listening)



```
H:\1111111111111\Server.exe
开始监听...
```

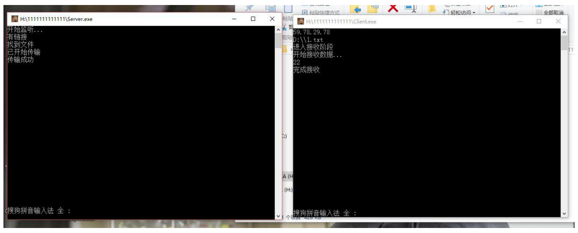
- Open Client , User input IP (connect)



```
H:\1111111111111\Server.exe
开始监听...
有链接
```

```
H:\1111111111111\Client.exe
59.78.29.78
```

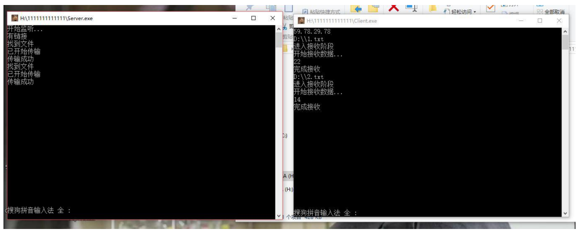
- User input filePath (get the file)



```
H:\1111111111111\Server.exe
开始监听...
有链接
找到文件
已开始传输
传输成功
```

```
H:\1111111111111\Client.exe
59.78.29.78
D:\\1.txt
进入接收阶段
开始接收数据...
22
完成接收
```

- User input another filePath (get the file)



```
H:\1111111111111\Server.exe
开始监听...
有链接
找到文件
已开始传输
传输成功
找到文件
已开始传输
传输成功
```

```
H:\1111111111111\Client.exe
59.78.29.78
D:\\1.txt
进入接收阶段
开始接收数据...
22
完成接收
D:\\2.txt
进入接收阶段
开始接收数据...
14
完成接收
```

VI. Problems and Experience

A. Convert a java program into executable files

Not like C++, there are some tricks to convert a Java program into corresponding executable files. And After searching in the Internet, I take the following method:

1. Use the eclipse to transfer the objected main function into file '.jar' by...
   ① Export your program as 'Java-> Runnable Jar File'.
   ② Set your specifications: pay attention to that the library handling should select the 'Package required libraries Into generated JAR'.
2. Use the software 'JSmooth'
   ① Download the 'JSmooth' in http://jsmooth.sourceforge.net/
   ② By using your generated 'jar' to get the executable file.
   Notice in this program, we will generate two 'jar' file, for Client and Server; and two executable files finally.

B. Multiple threads for Server

Since the project require us to have a loop for the C/S model, and the loop only exits when ‘Esc’, we have to consider threads when implementing.

At first, I write a while-loop to check whether ‘Esc’, and I thought it can well deal with the loop for server. However, the program for server always executes once and automatically exits out.After searching for the problem, I finally figue out that for server, we have to use threads instead of while-loop where the program only needs small changes.

C. Bubbles on savePath for Client

Actually the savePath can also be defined by user. We just need to adjust like this—In the main function, read user input by ‘String sPath=inFromUser.readLine();’. In the main function, transfer the string as a parameter for function receiveFile. In the receiveFile function, use ‘sPath+=arr[l-1]’ to get client’s final savePath.