Network programming (IT423+IT432)

Spring 2017

Dr. Islam Taj-Eddin

IT Dept., FCI, Assiut Univ.

Sockets for Servers

Server Sockets

- A host that responds to a connection.
- Instead of connecting to a remote host, a server program will wait for others to connect to it.

Accepting Connections

- A server socket binds to a particular port.
- It then listens for connection attempts.
- When it detects an attempt, it will accept the connection.

  - public Socket accept() Throws IOException
- The accept () method blocks until a connection is detected.
- It returns a java. net. Socket Object that is used to perform the communication.

Server Sockets

- Multiple clients can connect to the same port on a server at any time.
- The data are distinguished by the port numbers and the client addresses.
- However, no more than one socket can listen to a particular port at a time.
- This is why servers tend to be heavily multithreaded.
- Generally, the server socket listening to the port will accept connections and then pass the actual processing to other threads.

Server Sockets:Queue

- Incoming connections are stored in a FIFO queue until the server can accept them.
- If the queue is full, then connection attempts will be refused until space opens up.
- The default queue length for most systems is between 5 and 50.

Server Sockets:Queue

- The length of the queue can be adjusted by passing in the size of the queue in the “backlog" parameter.
  - public ServerSocket(int port,int backlog)
- Each system has a maximum queue length, so any values for the queue length longer than the maximum will be set to the maximum.

Example: Chat Server(l)

```
//This is a simple chat server written in Java

import java.io.*;
*import java.net.*;
import RcveData;
import SendData;

public class chat_server extends Thread {
```

Example: Chat Server(2)

```
public static void main(String a[]) {
   int blog = 600;
   int port = 9100;
   Socket   sock = null;

   if (a.length > 0) {
         port = Integer.parseInt(a[0]);
}//if
ServerSocket servsock - null;
```

Example: Chat Server(3)

```
System.out.println("using port “ + port);
try {
        servsock = new Serversocket(port,blog);
}// try
catch (java.io.IOException e) {
   System.out.println(e);
   System.exit(0);
}//catch
try {
        sock = servsock.accept();
} //try
```

Example: Chat Server(4)

```
            catch (java.io.IOException e) {
               System.out.println(e);
               System.exit(0); )
            }//catch

            SendData Sd = new SendData(sock);
            RcveData rd = new RcveData(sock);

            sd.start();
            rd.start();
        }//main
}//class
```

Example: Send Data(1)

```
import java.net.*;
import java.io.*;

public class SendData extends Thread {
        Socket sock;
        public SendData(Socket sock) {
            this.sock = sock;
        }//SendData constructor

public void run() {
        String line;
```

Example: Send Data(2)

```
try {
        outputStreamWriter outw = new
                        outputStreamwriter(sock.getOutputStream());
        BufferedWriter sockout=new BufferedWriter(outw);
        TnputStreamReader inr = new InputStreamReader(System.in);
        BufferedReader in = new BufferedReader(inr);
        while ((line = in.readLine()) != null) {
            sockout.write(line+"\n");
```

Example: Send Data(3)

```
                    sockout.flush();
                    yield();
                }//while
            }//try
            catch (java.io.IOException e) {
            System.out.println(e);
            System.exit(0);
        }//catch
    }//run
}//SendData
```

Example: Receive Data(1)

```
import java.net.*;
import java.io.*;

public class RcveData extends Thread {
        Socket sock;

        public RcveData(Socket sock) {
           this.sock = sock;
        }

        public void run() {
           String line;
```

Example: Receive Data(2)

```
try {
      InputStreamReader inr = new
                  InputStreamReader(sock.getInputStream());
      BufferedReader in = new BufferedReader(inr);

      while (line = in.readLine( )) != null) {

         System.out.print(“Receiving: ”);
         System.out.println(line);
         yield();
      }//while
}//try
```

Example: Receive Data(3)

```
            catch (java.io.IOException e) {
               system.out.println(e);
               System.exit(0);
            }//catch
        }//run
}//RcveData
```

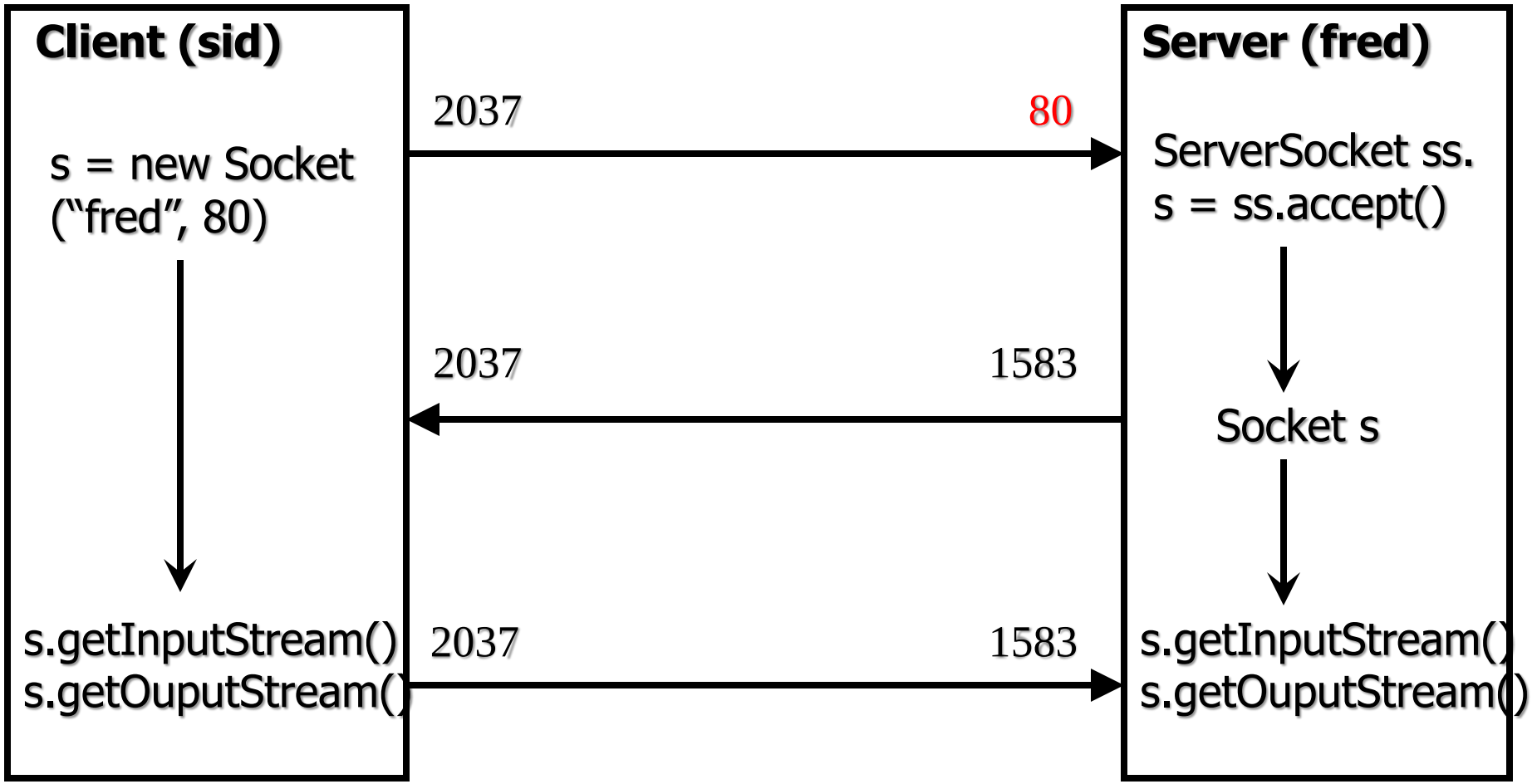
Http Server

```
ServerSocket socket = new ServerSocket(80, 5);

public void listen()
    throws IllegalAccessException,
           InstantiationException,
           IOException
    {
        for (;;) {
            System.err.println("HttpServer: waiting...");
            Socket s = socket.accept();

            FileServer f = createFileServer();
            f.dispatch(s);
        }
    }
```

How it all fits together



Socket Options

- SO_TIMEOUT
- SO_REUSEADDR
- SO_RCVBUF