CS601: Principles of Software Development

Sockets. Basic Client-Server Programming.

Olga A. Karpenko

Parts of this presentation is based on the materials of Prof. Engle.

Announcements

- Lab 3 part 2 due tonight

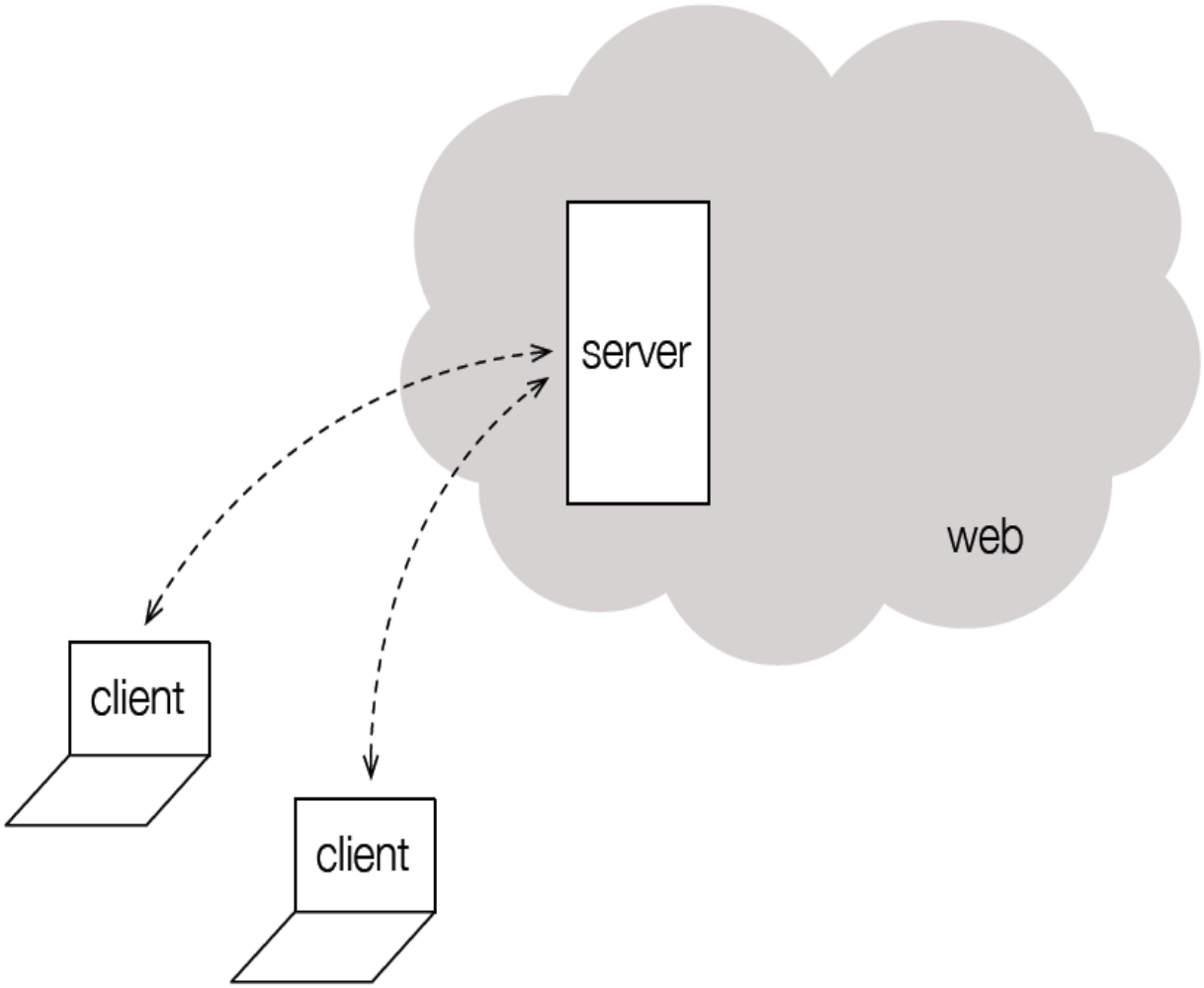
Announcements

- Lab 3 part 2 due tonight

Client-server model

- Standard model for developing network applications
- Server: provides some service to clients
  - Waits for requests from clients
- Client: requests some service
- May run on different machines

Basic Client-Server Architecture



Client-Server communication

1. Connection-oriented with stream sockets
   - Covered in this lecture
2. Connectionless transmission with datagrams

Sockets

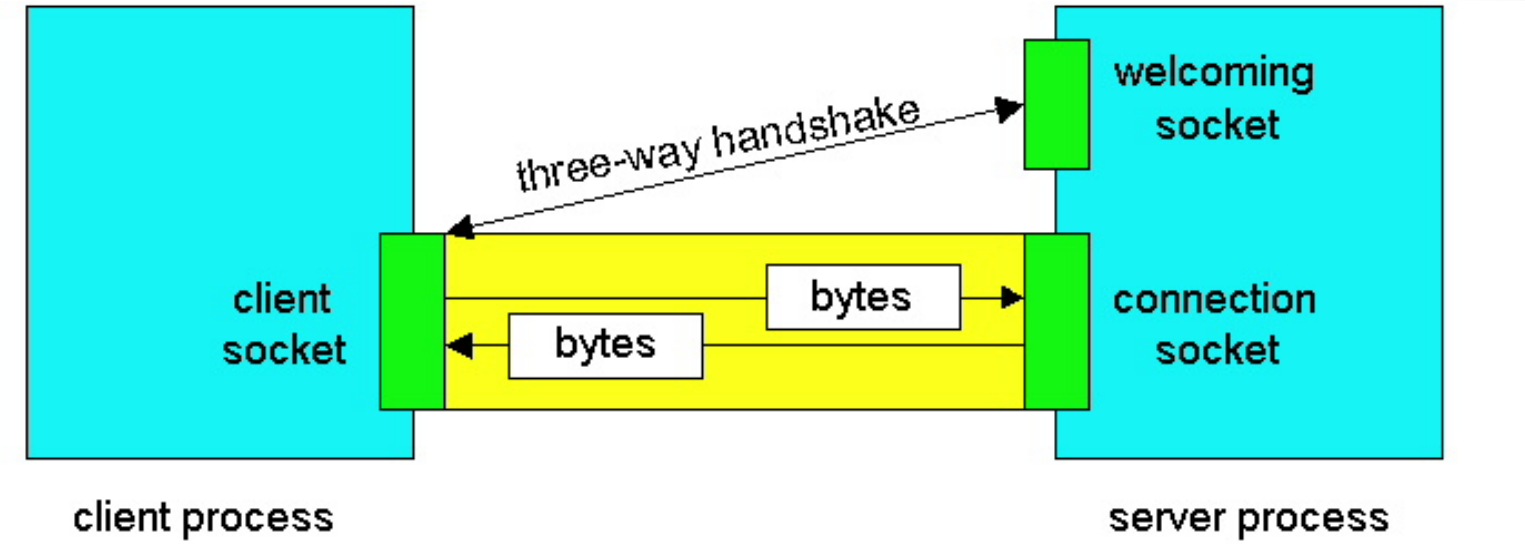
- “End-points for communication”
- Imagine client and server hosts being connected by a "pipe" through which the data flows
  - Each end of the pipe - endpoint

Sockets

- A socket connection between machines:
  - Create input and output streams
  - The programs can then talk to each other
- The lowest-level form of communication
  - From application developer's view
  - Programmer responsible for managing the flow of bytes between computers

From: people.cs.uchicago.edu/~asiegel/courses/cspp51036/Notes/**Sockets.ppt**

Sockets



- Welcoming socket listens for client's contact requests
- When contacted by client, the server creates a new connection socket to communicate

http://jpkc.ncwu.edu.cn/jsjwl/net/ross/book/apps/sockettcp.htm

Sockets in Java

- Easy!
- java.net.Socket
- java.net.ServerSocket

Client

1. Open a socket
2. Open an input stream and an output stream to the socket
3. Read from the input stream, Write to the output stream
   - according to the server's protocol
4. Close the streams
5. Close the socket

Port: Motivation

- Usually one physical connection to the network
- All the data from the network comes through this connection
- How does the computer redirect to specific applications?
  - The **port** address is used to identify the specific application to send the packet to

Port

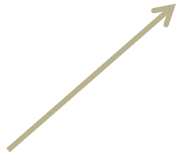
- Port numbers range from 0 to 65,535
  - Represented by 16-bit numbers
- The port numbers 0 - 1023 are restricted
  - Reserved for use by well-known services
  - Ex: Port 80 reserved for HTTP

Class Socket

- java.net.Socket
  - Socket(String *addr*, int *port*);
    - create a Socket connection to address *addr* on port *port*
  - InputStream getInputStream();
    - returns InputStream for getting info from socket
  - OutputStream getOutputStream();
    - returns OutputStream for sending info to socket
  - close();
    - close connection to implicit socket object, cleaning up resources.

Opening a socket: Client

```
Socket socket = new Socket("g1212.cs.usfca.edu", 5301);
```



Name of the machine

Port number

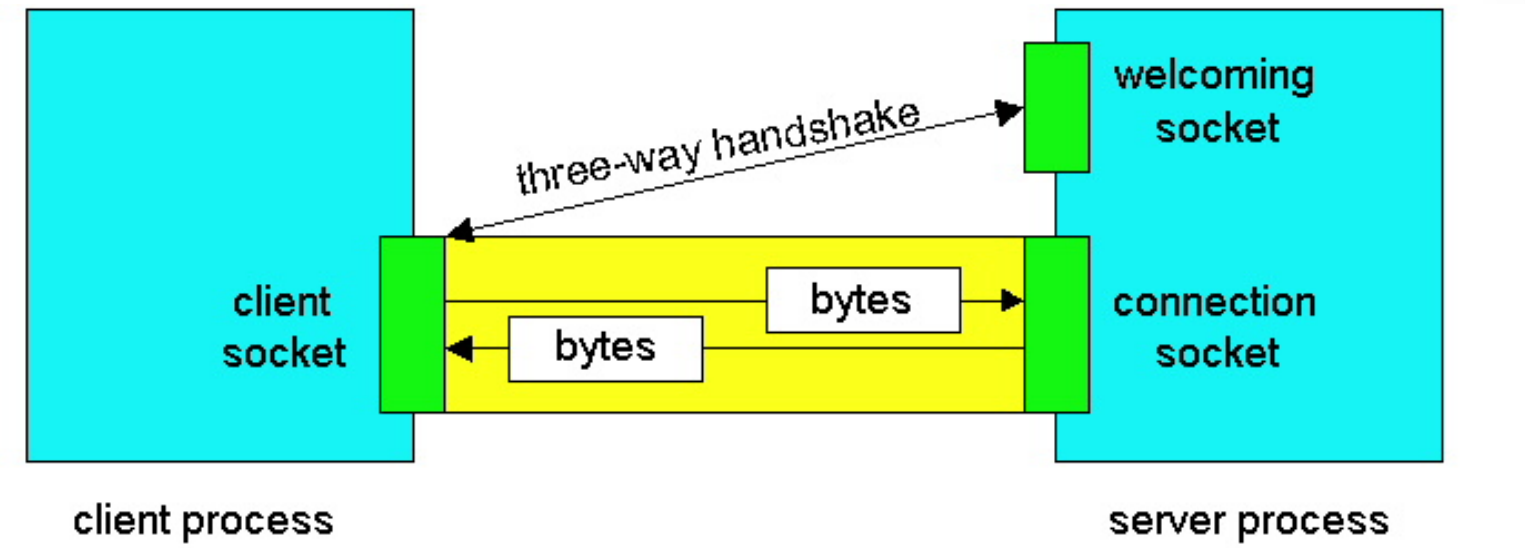
Opening a socket: Client

```
Socket socket = new Socket("g1212.cs.usfca.edu", 5301);

BufferedReader reader =  new BufferedReader(new
InputStreamReader(socket.getInputStream()));

PrintWriter writer = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));
```

Server

1. Create a ServerSocket object ("welcoming" socket)
2. Wait indefinitely for connections from clients
   - accept() method of ServerSocket
   - Once a client asks to connect, create a new "connection" socket for this client
3. Open input and output streams to the "connection" socket
4. Communicate with the client via streams
5. Close the socket and streams

Sockets



http://jpkc.ncwu.edu.cn/jsjwl/net/ross/book/apps/sockettcp.htm

Server

```
// listener is a “welcoming” socket
// listens for connection requests from clients
ServerSocket listener = new ServerSocket(9090);

// A new connection socket is created for each client
Socket connectionSocket = listener.accept();

//can read from reader to get messages from the client
BufferedReader reader =  new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));

// can write to out to send messages to the client
PrintWriter out = new
PrintWriter(connectionSocket.getOutputStream());
```

Examples

- Basic Server & Client
- ReverseEcho Server & Client
- DateServer & DateClient