Networking Code

CSCI 201
Principles of Software Development

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu

USC Viterbi
School of Engineering

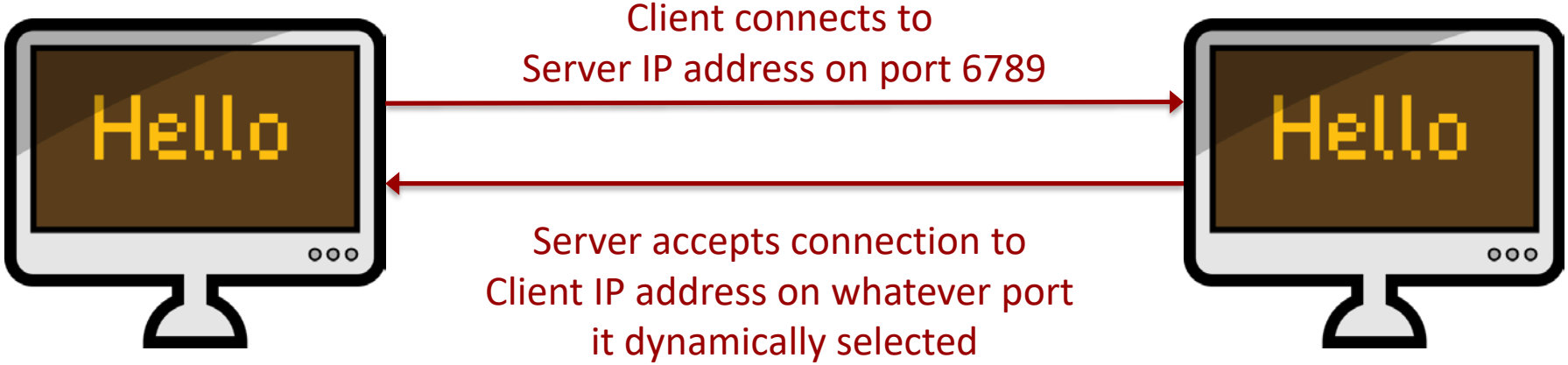
Outline

- Server Networking
- Client Networking
- Program

Server Software

- A server application is only able to serve requests that are addressed to the computer on which the server application is running
  - A server program cannot listen to a port on another physical server
- So, the only data the server application needs is the port on which to listen
  - The `ServerSocket` constructor only takes a port as a parameter
- Multiple networked applications can be running on the same computer as long as they are all using different ports

Networking Diagram



Multi-Threading with Networking

- Multi-threading is usually necessary with networking since there are two things that often are done at the same time
  - The ability to send data
  - The ability to receive data

- If sending and receiving are not performed in series, multi-threading will be needed
  - Some applications may be synchronous and only need one program to send data then wait for a response – that would *not* require multi-threading

Server Networking Example (no multi-threading)

```
1   import java.io.*; // to save space
2   import java.net.*; // to save space
3   public class NetworkingServer {
4     public NetworkingServer() {
5       ServerSocket ss = null;
6       Socket s = null;
7       PrintWriter pw = null;
8       BufferedReader br = null;
9       try {
10        System.out.println("Starting Server");
11        ss = new ServerSocket(6789);
12        s = ss.accept();
13        br = new BufferedReader(new InputStreamReader(s.getInputStream()));
14        pw = new PrintWriter(s.getOutputStream());
15        String line = br.readLine();
16        System.out.println("Line Received: " + line);
17        String str = "CSCI 201";
18        System.out.println("Sending: " + str);
19        pw.println(str);
20        pw.flush();
21      } catch (IOException ioe) {
22        System.out.println("IOE: " + ioe.getMessage());
23      } finally {
24        try {
25          if (pw != null)
26            pw.close();
27          if (br != null)
28            br.close();
29          if (s != null)
30            s.close();
31          if (ss != null)
32            ss.close();
33        } catch (IOException ioe) {
34          System.out.println("ioe: " + ioe.getMessage());
35        }
36      } // ends finally
37    }
38    public static void main(String [] args) {
39      new NetworkingServer();
40    }
41  }
```

Flushing

- Operating systems try to optimize networking similar to how they optimize file I/O
- Data is written into a buffer before it is sent along the socket
- The contents of the buffer will not be transmitted over the network until it fills up *unless* we explicitly flush the data
- Never forget to flush!

Outline

- Server Networking
- Client Networking
- Program

Client Software

- A client application is able to connect to any server application to which it has access, whether running on the same computer as the client or a different computer
- The client application needs both the IP address of the server and the port on which the server application is listening
  - Remember that multiple server applications can be running on the same computer as long as they are listening on different ports
- A Socket is the combination of the IP address and port number needed by the client

Client Networking Example (no multi-threading)

```
1   import java.io.*; // to save space
2   import java.net.Socket;
3   public class NetworkingClient {
4     public NetworkingClient() {
5       Socket s = null;
6       BufferedReader br = null;
7       PrintWriter pw = null;
8       try {
9         System.out.println("Starting Client");
10        s = new Socket("localhost", 6789);
11        br = new BufferedReader(new InputStreamReader(s.getInputStream()));
12        pw = new PrintWriter(s.getOutputStream());
13        String str = "Line being sent";
14        System.out.println("Sending: " + str);
15        pw.println(str);
16        pw.flush();
17        String line = br.readLine();
18        System.out.println("Line Received: " + line);
19      } catch (IOException ioe) {
20        System.out.println("IOE: " + ioe.getMessage());
21      } finally {
22        try {
23          if (pw != null)
24            pw.close();
25          if (br != null)
26            br.close();
27          if (s != null)
28            s.close();
29        } catch (IOException ioe) {
30          System.out.println("ioe: " + ioe.getMessage());
31        }
32      } // ends finally
33    } // ends NetworkingClient()
34    public static void main(String [] args) {
35      new NetworkingClient();
36    }
37  }
```

Outline

- Server Networking
- Client Networking
- Program

Program

- Write a single threaded chat program that allows two clients to communicate with each other in a synchronous manner – one user can only send a message after the other has sent one.

```
C:>java ChatClient localhost 6789
Hello, how are you?
Them: Fine, and you?
Good, thanks.

C:>java ChatClient localhost 6789
Them: Hello, how are you?
Fine, and you?
Them: Good, thanks.
```