JAVA Socket Tutorial

CSCE-315 Chris Pu

What’s a socket?

- Normally, a **server** runs on a specific computer and has a socket that is bound to a specific port number. The **server** just waits, listening to the socket for a **client** to make a connection request.

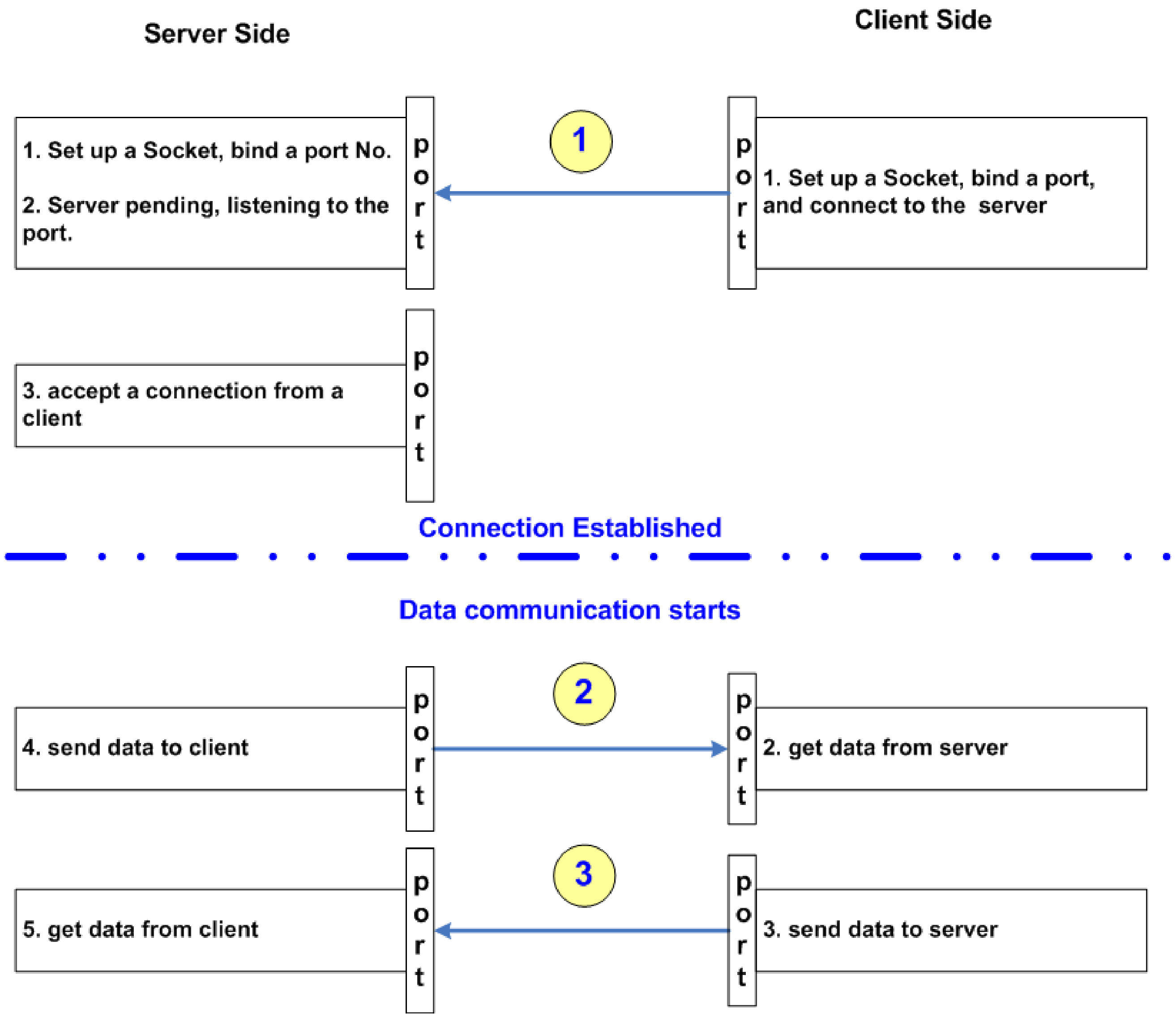
- The client specifies the server by using the server's **hostname** and **port**.
  - The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection.

Existing Implementation of Socket

- Socket can be implemented by C, C++, Java, C#, and various programming languages

- Socket is support by Windows, Linux, Unix. But socket libraries on different platforms are different.
  - For example, the socket library on windows is winSocket, and on Linux is POSIX socket.

Behavior


Server Side
Client Side
1. Set up a Socket, bind a port No.
2. Server pending, listening to the port.
port
1
port
1. Set up a Socket, bind a port, and connect to the server
3. accept a connection from a client
port
Connection Established
Data communication starts
4. send data to client
port
2
port
2. get data from server
5. get data from client
port
3
port
3. send data to server

Phase 1: Establish Connection

- Server side:
  - Port number should **>** 1024

```
try {
        serverSocket = new ServerSocket(4001);
} catch (IOException e) {
        System.out.println("Could not listen on port: 4001");
        System.exit(-1);
}
```

- Server Side:
  - Pending, and listening to port 4001.
  - the *accept* method waits until a client starts up and requests a connection on the host and port of this server

```
Socket clientSocket = null;
try {
    clientSocket = serverSocket.accept();
} catch (IOException e) {
    System.err.println("Accept failed.");
    System.exit(1);
}
```

Client Side: connect to server

```
Socket c_socket = null;
    try {
        c_socket = new Socket(“Aserver", 4001);
    } catch (UnknownHostException e) {
        System.err.println("Don't know about host: Aserver.");
        System.exit(1);
    } catch(IOException e){
         System.err.println("Couldn't get I/O for the connection
    to: Aserver.");
        System.exit(1);
    }
```

Phase 2: Data Communication

- Server side:
  - Set up stream for sending data to client:
  - Note: there are many ways to handle the socket I/O data
  - An example:

```
InputStream sockInput = null;
OutputStream sockOutput = null;
sockInput = clientSocket.getInputStream();
sockOutput = clientSocket.getOutputStream();
```

- Client Side:
  - Similarly,

  InputStream c_sockInput = c_socket
  .getInputStream();

  OutputStream c_sockOutput = c_socket
  .getOutputStream();

Example of Sending data

- Server side:

```
byte[] buf=new byte[1024];
/* buf ← certain data */
OutputStream sockOutput = clientSocket.getOutputStream();
sockOutput.write(buf, 0, buf.length);
```

Example of reading data

```
byte[] buf=new byte[1024];
int bytes_read = sockInput.read(buf, 0, buf.length);
```

- The function sockInput.read() will wait forever, until the program on the other side either sends some data, or closes the socket.

In the end

- Do not forget to close the socket:
  - Server side:
    - clientSocket.close();
    - then, serverSocket.close();
  - Client side:
    - c_socket.close();

Question?