Distributed Applications – Session 2

Lecturer Mouhsen Ibrahim

Contents

- Client-Server Applications
- Sockets
- Stream Sockets
- Server Sockets
- Client Sockets
- Sockets Input/Output
- Example Program
- Exercise

Client-Server Applications

- Clients request services from Servers, which run on a single server.
- In Distributed Applications there are more than one server which appear to the clients as a single server (transparent).
- These application implement a protocol standard defined in an RFC.
- RFC 2161 for Hypertext Transfer-Protocol.
- RFC 5321 for Simple Mail Transfer Protocol.
- These protocols have well known port numbers defined by IANA.

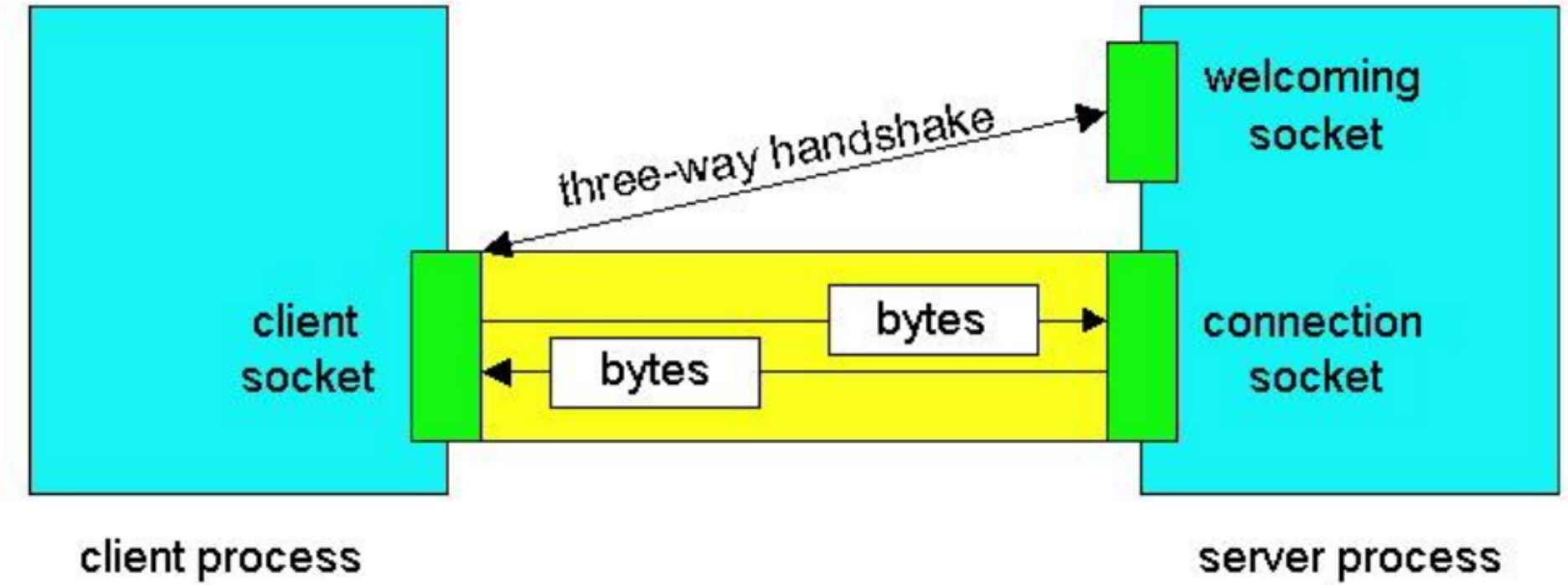
Sockets

- A socket is a pair of IP address and port.
- A connection is a pair of two sockets.
- A socket can be either:
  - Active Sockets: It is created at the client and starts the connection with another socket.
  - Passive Sockets: It is created at the server and listens on an IP address and port to accept connections from active sockets.
- Two types of sockets according to Transport Layer Protocol
  - Stream Sockets which use TCP and are reliable.
  - Datagram Sockets which use UDP and are not reliable.

Stream Sockets

- These sockets use TCP protocol for communication.
- At the server side there is the welcoming socket and connection socket.
- At the client side there is the client socket.
- In java we use java.net.* packages for socket programming. (Check the documentation for this package on devdocs.io for more information).
- The Socket class is used to create Active Sockets.
- The ServerSocket class is used to create Passive Sockets.

Stream Sockets



Client socket, welcoming socket (passive) and connection socket (active)

Server Sockets

- These sockets are created using the ServerSocket class.
- ServerSocket server = new ServerSocket(int port);
- Create a Server Socket that listens on a port.
- You can specify backlog (the number of connection requests in the wait queue) and the local address used to bind to it.
- ServerSocket server = new ServerSocket(int port, int backlog, InetAddress bindAddr);

Server Sockets

- ServerSocket methods include
- Socket accept() to accept a client connection and return a new Socket to communicate with the client.
- void close() to close the connection.
- InetAddress getInetAddress() to get the client's address.
- Int getLocalPort() to get the local port which the server is listening on it.

Client Sockets

- These sockets are created using the Socket class and they start a connection with a server socket.
- Socket client = new Socket(<hostname>, <port number>);
- hostname is the IP address or name of the host where the server is running.
- Port number is the port which the server is listening on it.
- We could use InetAddress class to specify the server address.
- We can also specify the IP address and port number to use at the client to initiate the connection.

Client Sockets

- Once the constructor is called the socket attempts to connect to the host and port number specified in it.
- If any errors happen an IOException is thrown.
- Socket methods
- void close() to close the connection
- InetAddress getInetAddress() to get remote address.
- InetAddress getLocalAddress() to get local address.
- InputStream getInputStream() to read from the socket.
- OutputStream getOutputStream() to write to the socket.

Sockets Input/Output

- We use InputStream to read data from sockets.
- The method getInputStream() is used to return the socket's input stream object.
- We can use the read method to read data from it to a byte array.
- int read(byte[] b)
- We use OutputStream to write data to sockets.
- The method getOutputStream() is used to return the socket's output stream object.
- We can use the write method to write data to it from a byte array.
- int write(byte[] b)

Example Program

- Write a TCP Server program that reads a word from the client, capitalize its letters and then return the result to the client.
- Write a TCP Client program that writes a word to the server and then reads the response back from it and display it on the screen.
- Use threads to do the processing at the server side.
- The Source code for these two programs can be found along with the lecture's file.

Exercise

- Write a TCP program that reads your a word from the client, then adds “hello” to the word and send the response back to the client, and waits for “bye server” from client to close connection.
- Write a TCP client that writes a word to the server, reads the response from the server and display it to the screen then sends “bye server” to the server.
- Using threads is optional.

Good Luck