Network Programming

"Understanding network fundamentals, and socket programming"

Advanced Programming

Shakirullah Waseeb
shakir.waseeb@gmail.com

Nangarhar University

April 12, 2017

Agenda

1. Introduction
   - Network Fundamentals
   - Client Server Model
2. Java Socket Programming
   - Understanding Sockets
   - Simple Socket Porgram
3. Questions and Discussion

Agenda

Introduction

- Socket programming was pioneered by Berkely Software Distribution (BSD) at University of California
- First TCP/IP communication standards
- Network Sockets
  - Similar to electrical sockets
  - Various clients can plugs around the network through sockets
  - Sockets present standard way of delivering payloads
  - IP, TCP, UDP, HTTP are some examples

Agenda

Server/Client

- Server: share services to server their clients
  - Compute server
  - Print server
  - Storage server
  - Web server
- Clients: gain access to a particular server and request for resources

Agenda

Sockets Introduction

- Allows to server many different clients at once
- Serving many different types of information
- Ports: these are numbered sockets on a particular machine
  - **Server** listening to a port until clients connects
  - **Client** connects to a particular machine on a port
  - **Session** each session is kept unique
- Reserved Sockets: TCP/IP lower than 10,24 ports are reserved for specific protocols as:
  - **21** for **FTP**
  - **23** for **telnet**
  - **25** for **email**
  - **80** for **HTTP**

Java TCP sockets

- There are two kinds of TCP sockets in Java
- ServerSocket: class is designed to be a "listener," which waits for clients to connect before doing anything
  - **Requires** a port number
  - **Listen** to clients and accepts connection to given port
  - **Returns** client socket connection object through which communicate with client
- Socket: The Socket class is designed to connect to server sockets and initiate protocol exchange
  - **Requires** a port number and address of the machine listening to given port
  - **Communicate** with server through this object once connection is established

An Example

- Here's an example [Figure 1] of a client requesting a single file, /index.html, and the server replying that it has successfully found the file and is sending it to the client:

Server	Client
Listens to port 80.	Connects to port 80.
Accepts the connection.	Writes "GET /index.html HTTP/1.0\n\n".
Reads up until the second end-of-line (\n).	
Sees that GET is a known command and that HTTP/1.0 is a valid protocol version.	
Reads a local file called /index.html.	
Writes "HTTP/1.0 200 OK\n\n".	"200" means "here comes the file."
Copies the contents of the file into the socket.	Reads the contents of the file and displays it.
Hangs up.	Hangs up.

Figure: http example [3, Page 590]

Agenda

Server Example Code

```
import java.net.*;
import java.io.*;
0 references
public class Server{
    static ServerSocket listener;
    static Socket soc;
    static PrintWriter out;
    static BufferedReader in;
    0 references
    public static void main(String args[]){
        try{
                listener = new ServerSocket(9060);
                soc = listener.accept();
                out = new PrintWriter(soc.getOutputStream(),true);
                out.println("This is server khan");
        } catch (IOException ex){
            System.out.println(ex);
        } finally{
             try{
               soc.close();
                } catch (IOException ex){
                        System.out.println(ex);
                        }
                }
        }
    }
```

Figure: Server Example Code

Client Example Code

```
import java.net.*;
import java.io.*;
0 references
public class Client{
    static Socket soc;
    static BufferedReader in;
    0 references
    public static void main(String args[]){
        try{
                soc = new Socket(args[0],9060);
                in = new BufferedReader(new InputStreamReader(soc.getInputStream()));
                System.out.println("Message from server " + in.readLine());
                }catch(IOException ex){
                        System.out.println(ex);
                }finally{
                        try{
                                soc.close();
                                }catch(IOException ex){
                                        System.out.println(ex);
                                }
                        }
                }
        }
```

Figure: Client Example Code

Your Turn: Time to hear from you!

[1]

[1] https://fensafitters.files.wordpress.com/2013/07/3d095.jpg

References

- P.J. Deitel, H.M. Deitel
  Java How to program, 10th Edition .
  Prentice Hall, 2015.
- P.J. Deitel, H.M. Deitel
  Java How to program, 9th Edition .
  Prentice Hall, 2012.
- Herbert Schildt
  The complete reference Java2, 5th Edition .
  McGraw-Hill/Osborne, 2002.