Socket Programming

An Introduction using Java
Assumptions

- Familiar with network basics (i.e. TCP/IP)
- Familiar with basics of programming in Java
Objectives

- An introduction to writing and compiling a simple Java program
- Pointer on where to find tutorial on writing simple programs using IDEs
- Socket background (what is it)
- Communicating with Sockets using Java
- Scaling with threads
Using a text editor

- In a real project (school/work) an IDE would be preferable

- No auto-complete, project management tools, syntax highlighting, contextual help etc, etc, etc...

- Manual build tools (in many cases IDE take care of the whole build cycle; including makefiles)

- Using a text editor, we need to compile using a command prompt
TRY:A hello world example

- Look at HelloWorld.java

```java
/*
 * Some comment goes here
 */
import java.io.*;

public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
    }
}
```

- Make modifications to the code, re-compile
Compiling using Javac

- javac compiles java programs and produces executables which can run by the JVM (Java Virtual Machine)

- javac [ options ] [ sourcefiles ] [ @argfiles ]

- [http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/javac.html](http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/javac.html) for more details
TRY: Compiling HelloWorld

- We will compile our simple HelloWorld programs
- We will introduce errors and analyze the output
- In our case, we will simply compile using javac HelloWorld.java
Some popular Java based IDEs

- Netbeans (opensource)
- Eclipse (opensource)
- BlueJ (designed for introductory teaching, opensource)
- IntelliJ (payware, but those who use it say it is worth it)
Hello World tutorial using IDE

- Netbeans
  (http://java.sun.com/docs/books/tutorial/getStarted/cupojava/netbeans.html)
- Eclipse
  (http://cs.millersville.edu/~liffick/cs161/labs/Lab1b.html)
- BlueJ
  (http://www.cs.carleton.edu/faculty/dmusican/cs117w02/introlab/introlab.html)
What are sockets

- TCP/IP protocol suite makes the internet possible
- Sockets are an abstraction
- Connection point to IP using either TCP or UDP
- Look at:

  http://java.sun.com/docs/books/tutorial/networking/sockets/index.html
Connection vs. Connectionless

- Comes down to TCP vs. UDP
- Stream vs. Datagram
- A simplistic view: Guarantee vs. Performance
Common Client/Server pattern

- Server listens on known port
- Clients connect to that port
- The connection between the two is usually switched to another port, freeing up the initial port
Connection Initiation Diagram

1. Client: Start the application
   - Create the socket
   - Acquire streams and conduct conversation
   - Close streams and socket
   - Exit application

2. Server: Create the server socket
   - Accept new connections
   - Acquire streams and conduct conversation
   - Exchange data
   - Close streams and socket
   - Disconnect
   - Continue?

3. Decide whether to continue or exit the application.
Useful Classes

- Socket
- ServerSocket
- InputStream
- OutputStream
- BufferedReader/InputStreamReader
- DataOutputStream
In more detail

- Create new server socket and start listening to a port
- Call the accept() method to get new connections
- Obtain input & output streams from the returned socket
- “Talk” using the application protocol specified
- Close the client streams and socket
- To accept another connection go back to step 2
- Close the server socket
API Docs

- http://java.sun.com/j2se/1.4.2/docs/api/

- Look for Socket related classes in java.net
TRY: Fetch URL Google.com

- Write a simple client that will get the text for the Google homepage.

- I will provide you with a sample client that does it for www.w3c.org

- Sample code located at:
  
  http://ise.gmu.edu/~astavrou/courses/isa_656_F07/W3CClient.java