Networking Support in Java

Department of Computer Science
University of Maryland, College Park
Overview

Networking

- Background
- Concepts
- Network applications
- Java’s object-oriented view
- Java’s networking API
  (Application Program Interface)

Last lecture

This lecture
Client / Server Model

- Relationship between two computer programs

- **Client**
  - Initiates communication
  - Requests services

- **Server**
  - Receives communication
  - Provides services

- **Other models**
  - Master / worker
  - Peer-to-peer (P2P)
## Client / Server Model Examples

<table>
<thead>
<tr>
<th>Application</th>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Browsing</td>
<td>Internet Explorer, Mozilla Firefox</td>
<td>Apache, Microsoft IIS</td>
</tr>
<tr>
<td>Email</td>
<td>MS Outlook, Thunderbird</td>
<td>POP, IMAP, Exchange</td>
</tr>
<tr>
<td>Streaming Music</td>
<td>Windows Media Player, iTunes</td>
<td>Internet Radio</td>
</tr>
<tr>
<td>Online Gaming</td>
<td>World of Warcraft, Halo 2, PartyPoker</td>
<td>Game / Realm Servers</td>
</tr>
</tbody>
</table>
Client Programming

Basic steps

1. Determine server location – IP address & port
2. Open network connection to server
3. Write data to server (request)
4. Read data from server (response)
5. Close network connection
6. Stop client
Simple Server Programming

**Basic steps**

1. **Determine server location - port (& IP address)**
2. **Create ServerSocket to listen for connections**
3. **Loop**

   ```java
   while (true) {
       Accept network connection from client
       Read data from client (request)
       Write data to client (response)
       Close network connection to client
   }
   ```
Advanced Server Programming

Server supports multiple connections / clients

Two approaches

1. **Loop**
   - Handles multiple connections in order
   - Limits on amount of network traffic
   - Not resilient in face of slow / stopped clients

2. **Multithreading**
   - Allows multiple simultaneous connections
Networking in Java

Packages

- java.net ⇒ Networking
- java.io ⇒ I/O streams & utilities
- java.rmi ⇒ Remote Method Invocation
- java.security ⇒ Security policies
- java.lang ⇒ Threading classes

Support at multiple levels

- Data transport ⇒ Socket classes
- Network services ⇒ URL classes
- Utilities & security
Java Networking API

Application Program Interface
- Set of routines, protocols, tools
- For building software applications

Java networking API
- Helps build network applications
- Interfaces to sockets, network resources
- Code implementing useful functionality
- Includes classes for
  - Sockets
  - URLs
Java Networking Classes

- IP addresses
  - InetAddress

- Packets
  - DatagramPacket

- Sockets
  - Socket
  - ServerSocket
  - DatagramSocket

- URLs
  - URL
InetAddress Class

- Represents an IP address
- Can convert domain name to IP address
  - Performs DNS lookup
- Getting an InetAddress object
  - getLocalHost()
  - getByName(String host)
  - getByAddress(byte[] addr)
DatagramPacket Class

- Each packet contains
  - InetAddress
  - Port of destination
  - Data
DatagramPacket Class

- Data in packet represented as byte array

Message to be sent. Consists of an integer followed by a string.

The byte array that will be placed in a UDP diagram.

Bytes that make up the integer 50694.

Bytes that make up the ASCII codes for the string “Hello World”.
Socket Classes

- Provides interface to TCP, UDP sockets
  1. **Socket**
     - TCP client sockets
  2. **ServerSocket**
     - TCP server sockets
  3. **DatagramSocket**
     - UDP sockets (server or client)
Socket Class

- Creates socket for client
- Constructor connects to
  - Machine name or IP address
  - Port number
- Transfer data via streams
  - Standard Java I/O streams
    - Bytes → InputStream, OutputStream
    - Characters → FileReader, PrintWriter
Socket Methods

- `getInputStream()`
- `getOutputStream()`
- `close()`
- `getInetAddress()`
- `getPort()`
- `getLocalPort()`
ServerSocket Class

- Create socket on server
- Constructor specifies local port
  - Server listens to port
- Usage
  - Begin waiting after invoking accept()
  - Listen for connection (from client socket)
  - Returns Socket for connection
ServerSocket Methods

- accept()
- close()
- getInetAddress()
- getLocalPort()
Connection Oriented

TCP Protocol

```
Server
Create Server Socket

Accept

Read/Write

Close Socket

Establish Connection

Client
Create Socket

Read/Write

Communicate

Close Socket
```
public class Server {
    public static void main(String args[]) throws Exception {
        ServerSocket ss = new ServerSocket(4242);
        while (true) {
            Socket s = ss.accept();
            BufferedReader r = new BufferedReader(
                new InputStreamReader(s.getInputStream()));
            PrintWriter out = new PrintWriter(
                new OutputStreamWriter(s.getOutputStream()));
            String name = r.readLine();
            out.println("Hello "+ name);
            out.flush();
            s.close();
        }
    }
}
public class Client {
    public static void main(String args[]) throws Exception {
        String host = "localhost";
        InetAddress server = InetAddress.getByName(host);
        Socket s = new Socket(server, 4242);
        BufferedReader r = new BufferedReader(
            new InputStreamReader(s.getInputStream()));
        PrintWriter out = new PrintWriter(
            new OutputStreamWriter(s.getOutputStream()));
        out.println("MyName");
        out.flush();
        String response = r.readLine();
        System.out.println(response);
        s.close();
    }
}
DatagramSocket Class

- Create UDP socket
  - **Does not distinguish server / client sockets**
- Constructor specifies InetAddress, port
- Set up UPD socket connection
- Send / receive DatagramPacket
DatagramSocket Methods

- close()
- getLocalAddress()
- getLocalPort()
- receive(DatagramPacket p)
- send(DatagramPacket p)
- setSoTimeout(int t)
- getSoTimeout()
Packet Oriented

UDP Protocol

```
Server
Create Socket
Read/Write
Close Socket

Client
Create Socket
Read/Write
Close Socket
```

Communicate
URL Class

- Provides high-level access to network data
- Abstracts the notion of a connection
- Constructor opens network connection
  - To resource named by URL
URL Constructors

- URL( fullURL )
  - URL( "http://www.cs.umd.edu/class/index.html" )

- URL( baseURL, relativeURL )
  - URL base = new URL("http://www.cs.umd.edu/" );
  - URL class = new URL( base, "/class/index.html " );

- URL( protocol, baseURL, relativeURL )

- URL( protocol, baseURL, port, relativeURL )
  - URL( "http", www.cs.umd.edu, 80,"/class/index.html" )
URL Methods

- getProtocol()
- getHost()
- getPort()
- getFile()
- getContent()
- openStream()
- openConnection()
URL Connection Classes

- High-level description of network service
- Access resource named by URL
- Can define own protocols

Examples

- URLConnection ⇒ Reads resource
- HttpURLConnection ⇒ Handles web page
- JarURLConnection ⇒ Manipulates Java Archives
- URLClassLoader ⇒ Loads class file into JVM
Java Applets

Applets are Java programs
- Classes downloaded from network
- Run in browser on client

Applets have special security restrictions
- Executed in applet sandbox
- Controlled by java.lang.SecurityManager
Applet Sandbox

- Prevents
  - Loading libraries
  - Defining native methods
  - Accessing local host file system
  - Running other programs (Runtime.exec())
  - Listening for connections
  - Opening sockets to new machines
    - Except for originating host
- Restricted access to system properties
Applet Sandbox

- Trusted Code
- Untrusted Code
- The Sandbox
- Java Virtual Machine
- Computer Resources
Network Summary

- Internet
  - Designed with multiple layers of abstraction
  - Underlying medium is unreliable, packet oriented
  - Provides two views
    - Reliable, connection oriented (TCP)
    - Unreliable, packet oriented (UDP)

- Java
  - Object-oriented classes & API
    - Sockets, URLs
    - Extensive networking support