

Networking Support In Java 2



Nelson Padua-Perez
Chau-Wen Tseng

Department of Computer Science
University of Maryland, College Park

Overview

■ Networking

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

} Last
lecture

} This
lecture

Client / Server Model

- Relationship between two computer programs

- Client

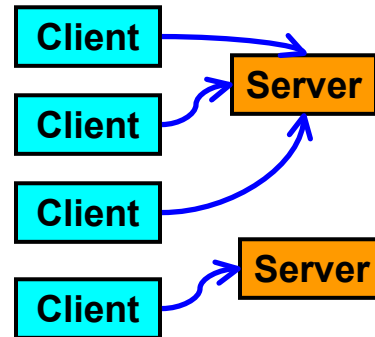
- Initiates communication
- Requests services

- Server

- Receives communication
- Provides services

- Other models

- Master / worker
- Peer-to-peer (P2P)



Client 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

Server Programming

- **Basic steps**
 1. Determine server location - port (& IP address)
 2. Create server to listen for connections
 3. Open network connection to client
 4. Read data from client (request)
 5. Write data to client (response)
 6. Close network connection to client
 7. Stop server

Server Programming

- **Can support multiple connections / clients**
- **Loop**
 - Handles multiple connections in order
- **Multithreading**
 - Allows multiple simultaneous connections

Client / Server Model Examples

Application	Client	Server
Web Browsing	Internet Explorer, Mozilla Firefox	Apache
Email	MS Outlook, Thunderbird	POP, IMAP, SMTP, Exchange
Streaming Music	Windows Media Player, iTunes	Internet Radio
Online Gaming	Half-Life, Everquest, PartyPoker	Game / Realm Servers

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

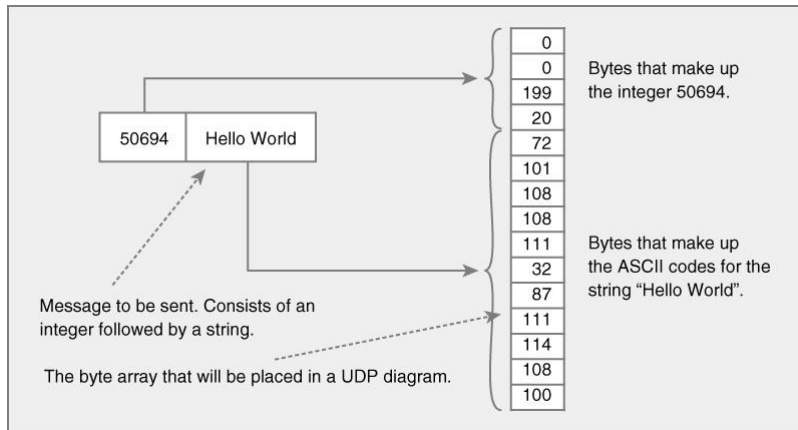
- Represents an IP address
- Can convert domain name to IP address
 - Performs DNS lookup
- Getting an InetAddress object
 - `getLocalHost()`
 - `getByName(String host)`
 - `getByAddress(byte[] addr)`

DatagramPacket Class

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

DatagramPacket Class

- Data in packet represented as byte array



DatagramPacket Methods

- getAddress()
- getData()
- getLength()
- getPort()
- setAddress()
- setData()
- setLength()
- setPort()

Socket Classes

- Provides interface to TCP, UDP sockets
- Socket
 - TCP client sockets
- ServerSocket
 - TCP server sockets
- 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**
 - Similar to standard Java I/O streams

Socket Methods

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

ServerSocket Class

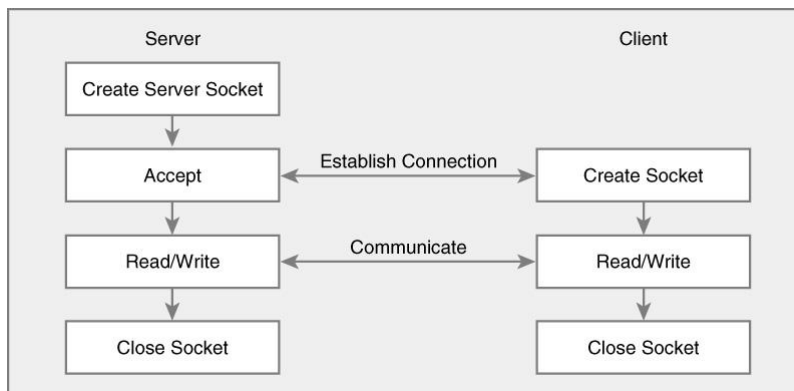
- Create socket on server
- Constructor specifies local port
 - Server listens to port
- Usage
 - Begin waiting after invoking `accept()`
 - Listen for connection (from client socket)
 - Returns **Socket** for connection

ServerSocket Methods

- `accept()`
- `close()`
- `getInetAddress()`
- `getLocalPort()`

Connection Oriented

■ TCP Protocol



DatagramSocket Class

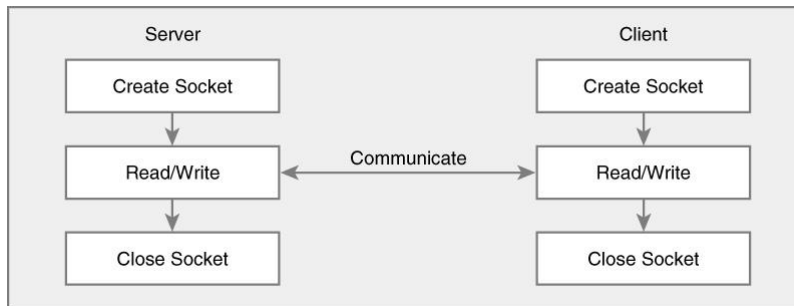
- Create UDP socket
 - Does not distinguish server / client sockets
- Constructor specifies InetAddress, port
- Set up UDP socket connection
- Send / receive DatagramPacket

DatagramSocket Methods

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

Packet Oriented

■ UDP Protocol



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("http", www.cs.umd.edu, "/class/index.html")`
- **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`**

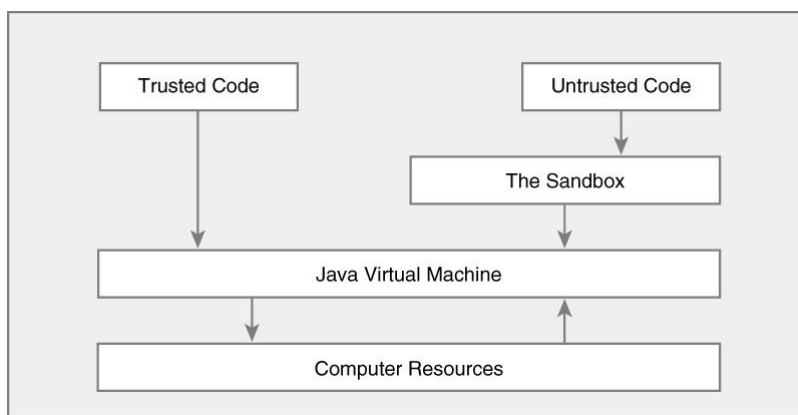
Applet Sandbox

■ 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



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

Scanner

■ Process input stream

- Provides methods for treating input as String, Int, ...
- Supports String `nextLine()`, int `nextInt()`...
- Throws `InputMismatchException` if wrong format

■ Example

```
// old approach to scanning input
BufferedReader br = new BufferedReader( new
    InputStreamReader(System.in));
String name = br.readLine();
// new approach using scanner
Scanner in = new Scanner(System.in);
String name = in.nextLine(); int x = in.nextInt();
```