Networking Support In Java

Networking II

Department of Computer Science

University of Maryland, College Park

Overview

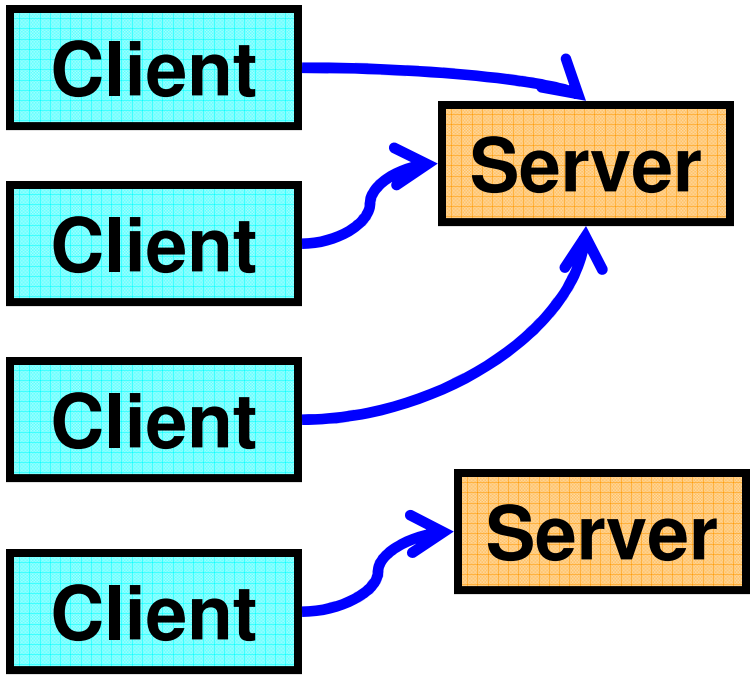
- Client/Server Model
- Client/Server Programming
- Networking in Java
- Java Networking API
- Java Networking Classes
- I/O
- Design Patterns
- Datagram Sockets
- URL class
- Great Reference:
  - http://java.sun.com/docs/books/tutorial/networking/index.html

Testing (old topic)

- You must write test cases
- Release tests are designed so that they don't give you enough information; you *have* to write test cases
- In many cases, people were failing release tests and if they had written any test for their failing methods, they would have found the error
-  We will beat on you if you don't write test cases

Client / Server Model

- Relationship between two computer programs
- Client
  - Initiates communication
  - Requests services
- Server
  - Receives communication
  - Provides services
- Other models
  - 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

Simple Server Programming

- **Basic steps**

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

```
While (true) {
    accept network connection to client
    Read data from client (request)
    Write data to client (response)
    Close network connection to client
 }
```

Supporting multiple connections / clients

- Can support multiple connections / clients
- Loop
  - Handles multiple connections in order
  - Limits on how much traffic we can handle
  - Not resilient in face of slow/stopped clients
- 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)

InetAddress is a factory

- You can't create InetAddress objects by invoking the InetAddress constructor
- Instead, invoke static methods of the InetAddress class
- These methods return either a Inet4Address or a Inet6Address
  - both of which extend InetAddress
- The factory design pattern delegates responsibility for design what class of object to create to the factory

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()

Socket Class

- One end of a TCP connection
- Client uses Constructor, providing:
  - Machine name or IP address
  - Port number
- Socket also returned by accept method
- Transfer data via streams
  - standard Java I/O streams

Socket Methods

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

I/O

- 4 fundamental interfaces
  - InputStream - incoming stream of bytes
  - Reader - incoming stream of characters
  - OutputStream - outgoing stream of bytes
  - Writer - outgoing stream of characters

Starting points

- FileInputStream/FileOutputStream
- FileReader/FileWriter
- ByteArrayInputStream/ByteArrayOutputStream
- StringReader/StringWriter
- Socket inputStream / outputStream

Decorator design pattern

- The java I/O libraries demonstrate the decorator design pattern
- You have some interface X, and a bunch of class that both implement X and take an X object as an argument
- A decorate intercepts / monitors / modifiers calls that are usually delegated to the object specified at construction time
- For example, a LineNumberReader
  - used to decorate any Reader, gives the ability to find out what line number you are currently at

Adapter design pattern

- The I/O libraries also demonstrate the adapter design pattern
  - Given two interfaces, an X-to-Y adapter takes an X object when constructed and implements the Y interface
    - Y call's are translated into corresponding X calls on the contained object
- For example, an InputStreamReader takes any InputStream and produces a Reader

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

Some Important Facts

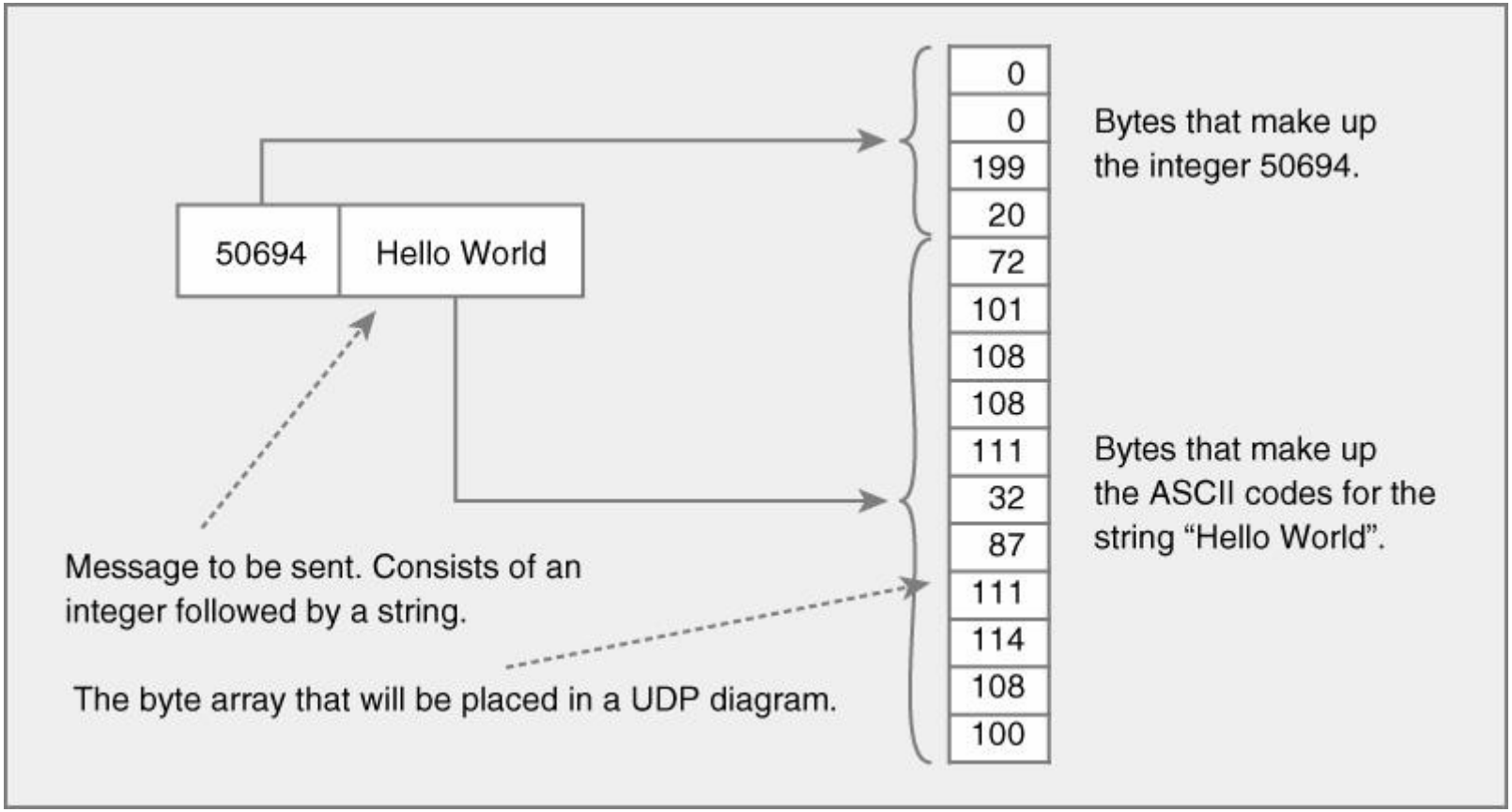
- Once you have written to a writer or outputStream, it may not be immediately sent
  - flush or close the output stream to ensure the data is sent
- A socket, it's inputStream and it's outputStream are all linked
  - closing any one closes all three
- Have to carefully choreograph who reads and who writes when
  - if both ends are reading, waiting for the other to send something, they deadlock

UDP/Datagram Sockets

- Create DatagramSockets
  - bound to a port
    - for clients, can bind to “arbitrary, unspecified but free port”
- Create DatagramPacket
  - Stores data to be sent/received, address and port
    - destination for packets being sent
    - source for received packets
  - can reuse the byte arrays associated with a Datagram
    - avoid allocating arrays

DatagramPacket Class

- **Data in packet represented as byte array**



DatagramPacket Methods

- getAddress()
- getData()
- getLength()
- getPort()

DatagramSocket Class

- Create UDP socket
- Constructor specifies port
  - can let system find arbitrary free port
- Send / receive DatagramPacket

DatagramSocket Methods

- close()
- getLocalAddress() - only interesting if you have multiple IP addresses on the local machine
- getLocalPort()
- receive(DatagramPacket p)
- send(DatagramPacket p)
- setSoTimeout(int t) - timeout
- getSoTimeout()

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( )
- openConnection() returns URLConnection

- URLConnection
  - getContentType()
  - getInputStream()

URL Reader

```
URLConnection connection = url.openConnection();
System.out.println("Type : " +
    connection.getContentType());
BufferedReader in = new BufferedReader(
    new InputStreamReader(
        connection.getInputStream()));
String inputLine;
while ((inputLine = in.readLine()) != null)
        System.out.println(inputLine);
in.close();
```