Announcement

• Lectures moved to
  • 150 GSPP, public policy building, right opposite Cory Hall on Hearst.
  • Effective Jan 31 i.e. next Tuesday

Outline

• APIs – Motivation
• Sockets
• Java Socket classes
• Tips for programming

What is an API?

• API – stands for Application Programming Interface.
• Interface to what? – In our case, it is an interface to use the network.

Socket Programming

Nikhil Shetty
GSI, EECS122
Spring 2006

What is an API?

• API – stands for Application Programming Interface.
• Interface to what? – In our case, it is an interface to use the network.
• A connection to the transport layer.
**What is an API?**

- API – stands for Application Programming Interface.
- Interface to what? – In our case, it is an interface to use the network.
- A connection to the transport layer.

**WHY DO WE NEED IT?**

**Need for API**

- One Word - Layering
- Functions at transport layer and below very complex.
- E.g. Imagine having to worry about errors on the wireless link and signals to be sent on the radio.

**Layering Diagramatically**

```
APPLICATION
  API
TRANSPORT
NETWORK
LINK
PHYSICAL
```

**What is a socket then?**

- What is a socket?

**Introduction**

- What is a socket?
- It is an abstraction that is provided to an application programmer to send or receive data to another process.
Introduction

• What is a socket?
• It is an abstraction that is provided to an application programmer to send or receive data to another process.
• Data can be sent to or received from another process running on the same machine or a different machine.

Socket – An Abstraction

• It is like an endpoint of a connection
• Exists on either side of connection
• Identified by IP Address and Port number
• E.g. Berkeley Sockets in C
  • Released in 1983
  • Similar implementations in other languages

Sockets

Engineers working on Sockets!!!

Client – Server Architecture

Flow in client-server model
Java Sockets

- Part of the java.net package
  - import java.net.*;

- Provides two classes of sockets for TCP
  - Socket – client side of socket
  - ServerSocket – server side of socket

- Provides one socket type for UDP
  - DatagramSocket

Java TCP Sockets

- ServerSocket performs functions bind and listen
  - Bind – fix to a certain port number
  - Listen – wait for incoming requests on the port

- Socket performs function connect
  - Connect – begin TCP session

TCP sockets

- TCP as a byte-stream
  - During data packet transmission, no packetization and addressing required by application.
  - Formatting has to be provided by application.
  - Two or more successive data sends on the pipe connected to socket may be combined together by TCP in a single packet.
  - E.g. Send “Hi” then send “Hello Nikhil” is combined by TCP to send as “HiHello Nikhil”

UDP sockets

- UDP is packet-oriented
  - Info sent in packet format as needed by app.
  - Every packet requires address information.
  - Lightweight, no connection required.
  - Overhead of adding destination address with each packet.

Java Quiz

Q. A constructor is used to...

A. Free memory.
B. Initialize a newly created object.
C. Import packages.
D. Create a JVM for applets.

Java Quiz

Q. A constructor is used to...

A. Free memory.
B. Initialize a newly created object.
C. Import packages.
D. Create a JVM for applets.
Socket Class

- Socket
  - `nameSocket = null;`
  - `nameSocket = new Socket("hostname", portno);`
- ServerSocket
  - `ServerSocket nameSocket = new ServerSocket(portno);`
  - Causes it to listen until there is a connection.

Accept

- `nameSocket = nameSocket.accept();`
- Creates a new socket to connect to the client.
- Waits till a new connection request appears.

Read or write from socket

- Associated with classes DataOutputStream and BufferedReader which create input and output streams.
- `nameSocket.getInputStream()` and `nameSocket.getOutputStream()` return input and output streams respectively.
- These streams assigned to local stream classes and byte stream can be input or output.

DatagramSocket Class

- `DatagramSocket nameSocket = new DatagramSocket();`
- `DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, portno);`
- `DatagramPacket recvPacket = new DatagramPacket(recvData, recvData.length);`
- `nameSocket.send(sendPacket);`
- `nameSocket.receive(recvPacket)`

Programming Tips

- Good programming techniques
  - Enclose all socket creations in `try{...} catch() {...}` to get the error conditions
  - 
  ```java
  try {
      clientSocket = serverSocket.accept();
  } catch (IOException e) {
      System.out.println("Accept failed: portno");
      System.exit(-1);
  }
  ```
  - Use tcpdump/Ethereal to see what is being transmitted on the link.
  - Check online guides to Java and Network Programming.
Network Programming Tips (contd)

- How to check if particular port is listening
  - Windows – use netstat
  - netstat -au
  - Linux – use nmap
  - nmap -sT -O localhost
- Tip: Use port numbers greater than 1024.
- Tip: InetAddress IPAddress = InetAddress.getByName("hostname");
- Check RFCs if in doubt about protocols.
  - http://www.ietf.org/rfc
- Lots of System.out.println("present_condition");
  - http://java.sun.com/docs/books/tutorial/networking/