CSCD 330  
Network Programming  

Winter 2020  

Lecture 7  
Application Layer –  
Socket Programming in Java  

Reading: Chapter 2, Java links Relevant Links page  

Some Material in these slides from J.F Kurose and K.W. Ross  
All material copyright 1996-2007
• So far,
  • Host has IP Address  
    146.187.134.22
    • Network Layer identifier
    • Every network “device” has this identifier
      • Phone, Toaster, Laptop etc.
  • Processes running on Hosts
    • Assigned a port number
    • Port numbers identifiers for processes
    • Some port numbers reserved 1 - 1023
    • Other port numbers reserved for widely recognized processes
Review Client/Server Programming

• Communication Between Client/Server
  • Uses object, “Socket”
  • Socket is API between a program and the TCP/IP stack of the OS
  • It has an input stream and an output stream built into it
  • Both the client and server define different ends of this socket
  • Link to the Java .net Package API

https://docs.oracle.com/javase/10/docs/api/java/net/package-summary.html
TCP/IP Client/Server

• How this works in Java
  • Server Socket

1. Binds socket to specific port number
2. Listens for incoming connections on that port
3. When connection attempted, it accepts connection, creates a regular socket for communication to client
4. Port number on client side is different and selected by stack software
5. What you see on Server side is same port number for the server program
TCP/IP Client/Server

- Java Code for Server
  ```java
  ss = new ServerSocket (port);
  // Loop forever
  While (true) {
    // Get a connection
    Socket newSocket = ss.accept ();
    // Deal with the connection
    // ....
  }
  ```

What is not obvious is that the new connection is through a different port number on the client side.
TCP/IP Client/Server

- Java code for Client, Send/Receive Data

// Create a socket for communicating with server
Socket clientSocket = new Socket("hostname", 6789);

// Create data streams for communicating through the socket
BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
PrintWriter out = new PrintWriter(clientSocket.getOutputStream());
System.out.println(in.readLine()); // Print to screen

Create streams to send input and get output from server
User Datagram Protocol (UDP)
UDP Introduction

• User Datagram Protocol (UDP) is communication protocol that transmits independent packets over network with no guarantee of arrival and no guarantee of order of delivery

• Most communication over Internet takes place over Transmission Control Protocol (TCP) … why do you think that is true?
UDP Advantages

• Advantage of UDP is that it requires much less overhead than TCP because
  • No hand-shaking,
  • No retry if an acknowledge isn't received,
  • No buffering and numbering of packets,

Where do we use UDP?

Connectionless protocols used either for one-packet messages for which delivery is not crucial
  – Responses to time requests, or

• To reduce transmission overhead for time-critical data such as streaming audio/video
UDP

- Building UDP applications is very similar to building a TCP app
- Difference is that we don’t establish a point to point connection between a client and a server

- The setup is very straightforward too.
- Java ships with built-in networking support for UDP – which is part of the java.net package.
- To use UDP, we only need to import classes from java.net package:
  java.net.DatagramSocket and java.net.DatagramPacket
UDP Socket Programming

• UDP no real “connection” between client and server
  • Sender **attaches** IP address and destination port to each packet
  • Server must **extract** IP address and port of sender from received packet, so answer can be sent back!

UDP, transmitted data may be
  – Received out of order, or
  – Lost
Client/server socket interaction: UDP

**Server (running on hostid)**

- Create socket, port=50, for incoming request:
  - `serverSocket = DatagramSocket()`

- Read request from `serverSocket`

- Write reply to `serverSocket` specifying client host address, port number

**Client**

- Create socket, `clientSocket = DatagramSocket()`

- Create, address (hostid, port=50)

- Send datagram request using `clientSocket`

- Read reply from `clientSocket`

- Close `clientSocket`
UDP Summary

• No connection setup – no “pipe”

• More Differences with TCP:
  1. Each batch of bytes sent with attached address information
  2. No special ServerSocket class in java
UDP Summary

• Create a Packet
  • Push it out into network through a socket
  • Server accepts packet addressed to him

• Mail is a lot like UDP
  • Each letter needs address of destination
  • Independent letters sent to same address
Java Classes for UDP

• Datagrams for connectionless protocol
• Two classes implement datagrams in Java:
  • `java.net.DatagramPacket`
  • `java.net.DatagramSocket`

• `DatagramPacket` is actual packet of information, an array of bytes, that gets transmitted over the network.

• `DatagramSocket` is socket that sends and receives DatagramPackets across the network.

• Think of `DatagramPacket` as a letter and `DatagramSocket` as the mailbox that the mail carrier uses to pick up and drop off your letters
• Need both classes for UDP sockets !!!!
Java Classes for UDP

• **DatagramPacket** class provides programmer with two constructors.
  • First is for **DatagramPackets** that receive data

  **Constructor needs**
  
  Array to store the data
  Amount of data to receive

  public DatagramPacket(byte[ ] ibuff, int ilength);

  **ibuf** is the byte array into which the data portion of the datagram will be copied.
  **ilength** is the number of bytes to copy from the datagram into the array receiving the data
Java Classes for UDP

- Second is for `DatagramPacket` that sends data
  
  **Constructor needs**
  
  Array to store the data, Amount of data to send
  
  Plus destination address and port number

  ```java
  public DatagramPacket (byte[] ibuf, int length, InetAddress iaddr, int iport);
  ```

  - `ibuf` is array of bytes that stores data of message,
  - `length` is length of byte array being sent via this datagram
  - `iaddr` stores the IP address of recipient
  - `port` identifies port datagram should be sent to on receiving host
Java Classes for UDP

• DatagramSocket represents connectionless socket

It provides three constructors,

1. Programmer can specify a port   OR
2. Allow system to randomly use a port
3. System can also select a specific IP address

public DatagramSocket() throws IOException
public DatagramSocket(int port) throws IOException
public DatagramSocket(int port, InetAddress localAddr) throws IOException
public DatagramSocket() throws IOException

- First constructor allows you to create a socket at an unused ephemeral port, generally used for **client applications**

- Second constructor allows you to specify a port, which is useful for **server applications**

public DatagramSocket(int port) throws IOException
Java Classes for UDP

```java
public DatagramSocket(int port,
        InetAddress localAddr)   throws IOException
```

- Final constructor is useful for machines with multiple IP interfaces
- You can use this constructor to send and listen for datagrams from one of the IP addresses assigned to the machine
Java Classes for UDP Methods

- **Methods**
  - Two most important methods, `send()` and `receive()`
  - Each takes an argument of a constructed `DatagramPacket`

  - `send()` method
    - Data in packet is sent to specified host and port
  - `receive()` method
    - Will block execution until packet is received by underlying socket, then data copied into packet provided
UDP Example Program

Sentence Capitalizer
UDP Sentence Capitalizer

Client

1. Read string from keyboard, convert to bytes
2. Create DatagramSocket for communicating to UDP Server
   DatagramPacket has string to send in bytes, length, IPAddress server, port server
3. Send DatagramPacket through Socket
4. Create DatagramPacket to receive reply from Server
5. Receive reply from Server, print Capitalized string
UDP Sentence Capitalizer

Server

1. Create socket for server with port
2. Create DatagramPacket to receive sentence from client
3. Wait to receive from client
4. Convert bytes from client to sentence
5. Get IPAddr and port from received packet
6. Capitalize the sentence, convert to bytes
7. Create send DatagramPacket to send back to Client, with IPAddr and port
8. Send packet to client
import java.io.*;
import java.net.*;

class UDPClient {
    public static void main(String args[]) throws Exception {
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getByName("localhost");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
Example: Java client (UDP), cont.

Create datagram with data-to-send, length, IP addr, port

Send datagram to server

Send thru socket

DatagramPacket sendPacket =
    new DatagramPacket(sendData, sendData.length,
    IPAddress, 9876);

clientSocket.send(sendPacket);

Read datagram from server

Build Receive Packet

DatagramPacket receivePacket =
    new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

Convert bytes to characters

Recv thru socket

String modifiedSentence =
    new String(receivePacket.getData());

System.out.println("FROM SERVER:" + modifiedSentence);

clientSocket.close();
}
import java.io.*;
import java.net.*;

class UDPServer {
    public static void main(String args[]) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(9876);
        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];

        while(true) {
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);
        }
    }
}
Example: Java server (UDP), cont

String sentence = new String(receivePacket.getData());

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, port);

serverSocket.send(sendPacket);

End of while loop, back and wait for another datagram
More Java.net Package

- Always good to see the documentation on the Classes and methods
- Look up DatagramSocket and DatagramPacket

https://docs.oracle.com/javase/10/docs/api/java/net/package-summary.html
Summary

• Brief coverage of Java sockets - TCP/UDP
• Should be enough to get started
  • Examples available as links on the main class page

• Also, practice **client-server** in next lab
• Read references in RelatedLinks for tutorials and more information on Java Client Server
Assignment 3 – Due Friday