Socket programming

goal: learn how to build client/server applications that communicate using sockets

socket: dropbox between application process and end-end-transport protocol
Socket programming

Two socket types for two transport services:
- **UDP:** unreliable datagram
- **TCP:** reliable, byte stream-oriented

Application Example:
1. Client reads a line of characters (data) from its keyboard and sends the data to the server.
2. The server receives the data and converts characters to uppercase.
3. The server sends the modified data to the client.
4. The client receives the modified data and displays the line on its screen.
Socket programming **with UDP**

**UDP: no “connection” between client & server**
- no handshaking before sending data
- sender explicitly attaches IP destination address and port # to each packet
- rcvr extracts sender IP address and port# from received packet

**UDP: transmitted data may be lost or received out-of-order**

**Application viewpoint:**
- UDP provides *unreliable* transfer of groups of bytes (“datagrams”) between client and server
Client/server socket interaction: UDP

**server** (running on serverIP)

create socket, port= x:
serverSocket = DatagramSocket(x)

read datagram from serverSocket

write reply to serverSocket specifying client address, port number

**client**

create socket:
clientSocket = DatagramSocket()

Create datagram with server IP and port=x; send datagram via clientSocket

read datagram from clientSocket specifying client address, port number

close clientSocket
import java.io.*;
import java.net.*;

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

- Create datagram with data-to-send, length, IP addr, port:
  ```java
  DatagramPacket sendPacket =
  new DatagramPacket(sendData, sendData.length,
                      IPAddress, 9876);
  clientSocket.send(sendPacket);
  ```

- Send datagram to server:
  ```java
  DatagramPacket receivePacket =
  new DatagramPacket(receiveData, receiveData.length);
  clientSocket.receive(receivePacket);
  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);
            // Create datagram socket at port 9876
            // Create space for received datagram
            // Receive datagram
        }
    }
}
```java
String sentence = new String(receivePacket.getData());
InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();
String capitalizedSentence = sentence.toUpperCase();
sendKeysData = capitalizedSentence.getBytes();
DatagramPacket sendPacket =
    new DatagramPacket(sendData, sendData.length, IPAddress, port);
serverSocket.send(sendPacket);
```

Example: Java server (UDP)
Example app: UDP client

Python UDPClient

```
import socket
serverName = 'hostname'
serverPort = 12000

clientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```
Example app: UDP server

**Python UDPServer**

```python
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

- create UDP socket
- bind socket to local port number 12000
- loop forever
- Read from UDP socket into message, getting client’s address (client IP and port)
- send upper case string back to this client
**Socket programming with TCP**

**Client must contact server**
- Server must be first running
- Server must have created socket (dropbox) that welcomes client’s contact

**Client connects to server by:**
- Creating TCP socket, specifying IP address, port number of server process
- Client socket is now bound to that specific server

**Server accepts connect by:**
- Creating new connection-specific socket
- Allows server to talk with multiple clients

---

**Application viewpoint:**
TCP provides reliable, in-order byte-stream transfer (“pipe”) between client and server
Client/server socket interaction: TCP

server (running on hostid)  client

create socket, port=x, for incoming request:
serverSocket = ServerSocket()

wait for incoming connection request
connectionSocket = serverSocket.accept()

TCP connection setup

read request from connectionSocket
write reply to connectionSocket
close connectionSocket

send request using clientSocket
read reply from clientSocket
close clientSocket
Example: Java client (TCP)

```java
import java.io.*;
import java.net.*;

class TCPClient {
    public static void main(String argv[]) throws Exception {
        String sentence;
        String modifiedSentence;
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));
        Socket clientSocket = new Socket("hostname", 6789);
        DataOutputStream outToServer =
            new DataOutputStream(clientSocket.getOutputStream());

        create input stream
        create clientSocket object of type Socket, connect to server
        create output stream attached to socket

        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));
        Socket clientSocket = new Socket("hostname", 6789);
        DataOutputStream outToServer =
            new DataOutputStream(clientSocket.getOutputStream());
```

This package defines Socket() and ServerSocket() classes.

Server name, e.g., www.umass.edu
Server port #
**Example: Java client (TCP)**

```java
BufferedReader inFromServer =
    new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();

outToServer.writeBytes(sentence + '
');

modifiedSentence = inFromServer.readLine();

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

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

class TCPServer {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
        String capitalizedSentence;

        ServerSocket welcomeSocket = new ServerSocket(6789);

        while(true) {
            Socket connectionSocket = welcomeSocket.accept();

            BufferedReader inFromClient =
                new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));

            String clientSentence = inFromClient.readLine();
            String capitalizedSentence = clientSentence.toUpperCase();

            System.out.println(capitalizedSentence);
        }
    }
}
Example: Java server (TCP)

```java
DataOutputStream outToClient =
   new DataOutputStream(connectionSocket.getOutputStream());

clientSentence = inFromClient.readLine();

capitalizedSentence = clientSentence.toUpperCase() + '\n';

outToClient.writeBytes(capitalizedSentence);
```
Example app: TCP client

Python TCPClient

```python
import socket
serverName = 'servername'
serverPort = 12000
clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

create TCP socket for server, remote port 12000

No need to attach server name, port
Example app: TCP server

Python TCPServer

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()