Socket Programming

• Socket Programming Overview
  - Socket Programming with UDP
  - Socket Programming with TCP

• Python Socket Programming

Readings: Chapter 2: Sections 2.7
Recap: Client-Server Communication Paradigm

Typical network app has two pieces: client and server.

Client:
- initiates contact with server ("speaks first")
- typically requests service from server

Server:
- provides requested service to client e.g., Web server sends requested Web page, mail server delivers e-mail
Recap: Addresses and Ports

- Host (NIC card) identified by unique IP address
- Network application/process identified by port number
- Network connection identified by a 5-tuple (src ip, src port, dst ip, dst port, protocol)

- Two kinds of Internet transport services provided to applications
  - Connection-oriented TCP
  - Connectionless UDP
Goal: learn how to build client/server applications that communicate using sockets

Socket: door 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 data to server
2. server receives the data and converts characters to uppercase
3. server sends modified data to client
4. client receives modified data and displays 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
• receiver 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 = socket(AF_INET,SOCK_DGRAM)

read datagram from serverSocket

write reply to serverSocket

specifying client address, port number

client

create socket:
clientSocket = socket(AF_INET,SOCK_DGRAM)

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

read datagram from clientSocket

specifying client address, port number

close clientSocket
Example app: UDP client

Python UDPCClient

```python
from socket import *

serverName = 'hostname'
serverPort = 12000

clientSocket = socket(AF_INET, SOCK_DGRAM)

message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message.encode(), (serverName, serverPort))

modifiedMessage, serverAddress = clientSocket.recvfrom(2048)

print modifiedMessage.decode()
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 True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

- create UDP socket
- bind socket to local port number 12000
- 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 process must first be running
• server must have created socket (door) that welcomes client’s contact

client contacts server by:
• Creating TCP socket, specifying IP address, port number of server process
• when client creates socket: client TCP establishes connection to server TCP

• when contacted by client, server TCP creates new socket for server process to communicate with that particular client
  - allows server to talk with multiple clients
  - source port numbers used to distinguish clients
    (more in Chap 3)

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

**server (running on hostid)**

1. Create socket, port=x, for incoming request:
   ```python
   serverSocket = socket()
   ```

2. Wait for incoming connection request:
   ```java
   connectionSocket = serverSocket.accept()
   ```

3. Read request from connectionSocket:

4. Write reply to connectionSocket:

5. Close connectionSocket:

**client**

1. Create socket, connect to hostid, port=x:
   ```java
   clientSocket = socket()
   ```

2. Send request using clientSocket:

3. Read reply from clientSocket:

4. Write reply to connectionSocket:

5. Close connectionSocket:
Example app: TCP client

**Python TCPClient**

```python
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print ('From Server:', modifiedSentence.decode())
clientSocket.close()
```
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 True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
capitalizedSentence = sentence.upper()
connectionSocket.send(capitalizedSentence.encode())
connectionSocket.close()
Socket Programming: Basics

- The server application must be running before the client can send anything.
- The server must have a socket through which it sends and receives messages. The client also need a socket.
- Locally, a socket is identified by a port number.
- In order to send messages to the server, the client needs to know the IP address and the port number of the server.

Port number is analogous to an apartment number. All doors (sockets) lead into the building, but the client only has access to one of them, located at the provided number.
Helpful Resources

• Python Socket Tutorial
  - https://docs.python.org/2/library/socket.html
  - https://docs.python.org/3.4/library/socket.html

• Java Socket Tutorial
  - http://download.oracle.com/javase/tutorial/networking/sockets/