Socket Programming
Socket programming

- Goal:
  - learn how to build client/server applications that communicate using sockets
- Socket:
  - door between application process and end-end-transport protocol

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)**

```python
create socket, port= x:
serverSocket =
socket(AF_INET,SOCK_DGRAM)
```

```
read datagram from
serverSocket
```

```python
write reply to
serverSocket
specifying
client address,
port number
```

**client**

```python
create socket:
clientSocket =
socket(AF_INET,SOCK_DGRAM)
```

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

```
read datagram from
clientSocket
```

```
close
clientSocket
```
**Example app: UDP client**

**Python UDPClIENT**

```python
from socket import *

serverName = 'hostname'
serverPort = 12000

clientSocket = 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)
```
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)                      client

| server: create socket, port=x, for incoming request: serverSocket = socket() |
| wait for incoming connection request: connectionSocket = serverSocket.accept() |
| read request from connectionSocket |
| write reply to connectionSocket |
| close connectionSocket |

| client: create socket, connect to hostid, port=x clientSocket = socket() |
| send request using clientSocket |
| read reply from clientSocket |
| close clientSocket |
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)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

Example app: TCP server

**Python TCPServer**

```python
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()
```