Recitation 1

Internet Technology (Section 01)

Network Sockets

- A **software abstraction** used for creating network endpoints for applications
- Managed by the operating system
- Identified by a *socket address*
  - An *ip address* and *port number*

Ports

- There exists a **server** and **client**
  - Client makes request to server
- What port to use?
  - Client must send request to a port that's pre-agreed upon
    - Client can use any of its own ports!

Port #	Application Layer Protocol	Type	Description
20	FTP	TCP	File Transfer Protocol - data
21	FTP	TCP	File Transfer Protocol - control
22	SSH	TCP/UDP	Secure Shell for secure login
23	Telnet	TCP	Unencrypted login
25	SMTP	TCP	Simple Mail Transfer Protocol
53	DNS	TCP/UDP	Domain Name Server
67/68	DHCP	UDP	Dynamic Host
80	HTTP	TCP	HyperText Transfer Protocol
123	NTP	UDP	Network Time Protocol
161,162	SNMP	TCP/UDP	Simple Network Management Protocol
389	LDAP	TCP/UDP	Lightweight Directory Authentication Protocol
443	HTTPS	TCP/UDP	HTTP with Secure Socket Layer

Socket Workflow

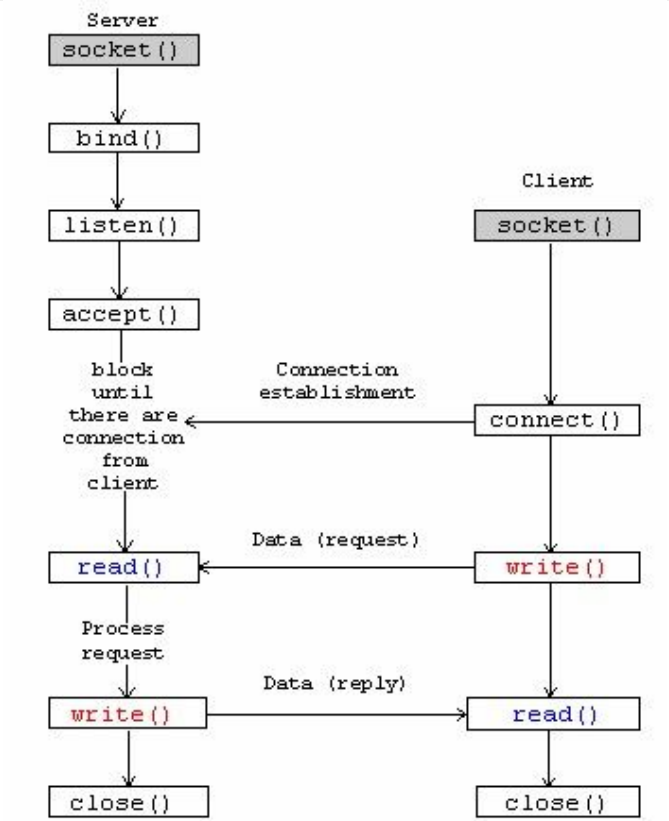
Server 

1. Bind socket to port and address
2. Wait for connection requests
3. Receive data, process it, send back data
4. Disconnect

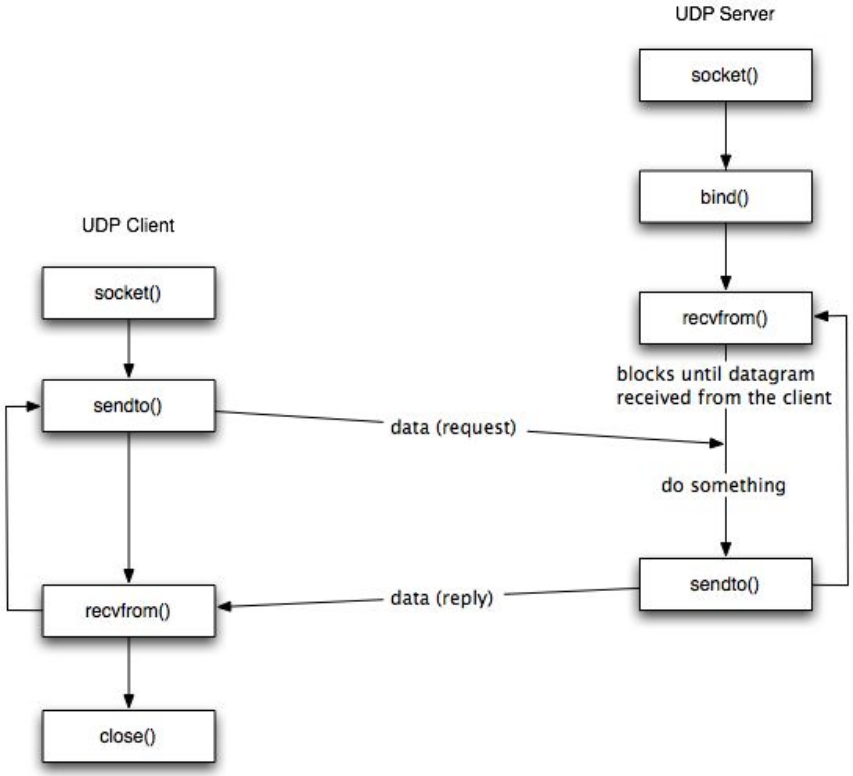
Client 

1. Connect to a remote socket
2. Send requests, receive replies
3. Disconnect

Socket Workflow (TCP)



Socket Workflow (UDP)



TCP Client/Server Python

Server

```
import socket
HOST="127.0.0.1"
PORT=1234
sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
sock.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR, 1)
sock.bind((HOST,PORT))
sock.listen()
while True:
    (conn, address) = sock.accept()
    data = conn.recv(1)
    if not data:
        continue
    data = data[0]
    if data == 0x1b:
        print("1")
        conn.sendall(b"first!")
    elif data == 0x2b:
        print("2")
        conn.sendall(b"second!")
sock.close()
```

Client

```
import socket
HOST="127.0.0.1"
PORT=1234
sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
sock.connect((HOST,PORT))
m = bytearray(1)
m[0] = 0x1b
sock.sendall(m)
data = sock.recv(1024)
print(data)
sock.close()
```

UDP Client/Server Python

Server

```
import socket
HOST="127.0.0.1"
PORT=1234
sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR, 1)
sock.bind((HOST,PORT))
while True:
    data, addr = sock.recvfrom(1)
    if not data:
        continue
    data = data[0]
    if data== 0x1b:
        print("1")
        sock.sendto(b"first!",addr)
    elif data== 0x2b:
        print("2")
        sock.sendto(b"second!",addr)
sock.close()
```

Client

```
import socket
HOST="127.0.0.1"
PORT=1234
ADDR = (HOST,PORT)
sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
m = bytearray(1)
m[0] = 0x1b
sock.sendto(m,ADDR)
data,addr = sock.recvfrom(1024)
print(data)
sock.close()
```

Berkeley Socket Specification (Extra Slide)

Address Family	Socket Type	
	SOCK_DGRAM	SOCK_STREAM
IPX/SPX	SPX	IPX
NetBIOS	NetBIOS	n/a
IPv4	UDP	TCP
AppleTalk	DDP	ADSP
IPv6	UDP	TCP
IrDA	IrLMP	IrTTP
Bluetooth	?	RFCOMM