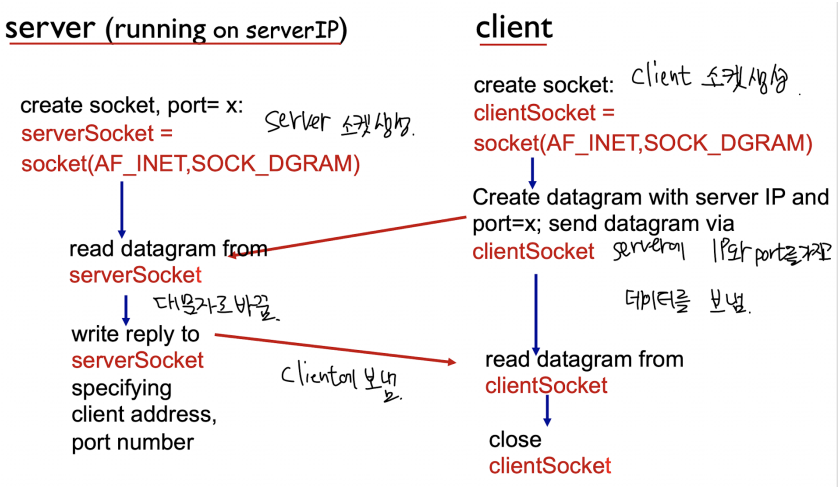
컴네 중간 정리 socket part

1. **socket programming with UDP and TCP**
    a. **Socket programming**
        i. 목표: client/servr가 socket을 이용하여 communicate을 할때 어떻게 이것을 만드느냐?
        ii. socket: trasport layer과 application process의 layer의 어떠한 data search(door느낌)
        iii. Two socket types for two transport services: 2개의 type중 1개를 골라야함
            1. UDP: unrelialbe datagram
            2. TCP: relialbe, byte stream-oriented
        iv. Application Example:
            1. client가 사용자로부터 character들을 입력받음, 이를 server에 보냄
            2. server는 그 입력받은 stream을 uppercase로 바꿔서 다시 client로 보냄
            3. 변경된 data를 받아서 화면에 띄움
    b. **Socket programming with UDP**
        i. UDP: client& server사이에 connection이 만들어지지 않음
            1. 데이터를 보내기전에 handshaking이 없음
            2. sender는 IP destination address와 port를 각각 packet에 붙여줘야함
            3. recevier는 sender에게 IP address와 port를 표시해야한다.
        ii. UDP: data가 lost될 수 있다. sender에서 순서대로 보냈지만 순서가 바뀔수도 있다.
        iii. UDP는 unreliable transfer를 제공한다.
    c. **Client/server socket interaction: UDP**



1. UDP를 이용한 socekt 통신 방법
2. UDP의 port number는 16bit

d. **Example app: UDP client**

Python UDPClient

include Python's socket library → `from socket import *`

portnumber는 16bit.

`serverName = 'hostname'`

`serverPort = 12000`

create UDP socket for server → `clientSocket = socket(AF_INET, SOCK_DGRAM)`

get user keyboard input → `message = raw_input('Input lowercase sentence:')`

Attach server name, port to message; send into socket → `clientSocket.sendto(message.encode(), (serverName, serverPort))`

함수이용하여 message를 보냄.

read reply characters from socket into string → `modifiedMessage, serverAddress = clientSocket.recvfrom(2048)`

print out received string and close socket → `print modifiedMessage.decode()`

`clientSocket.close()`

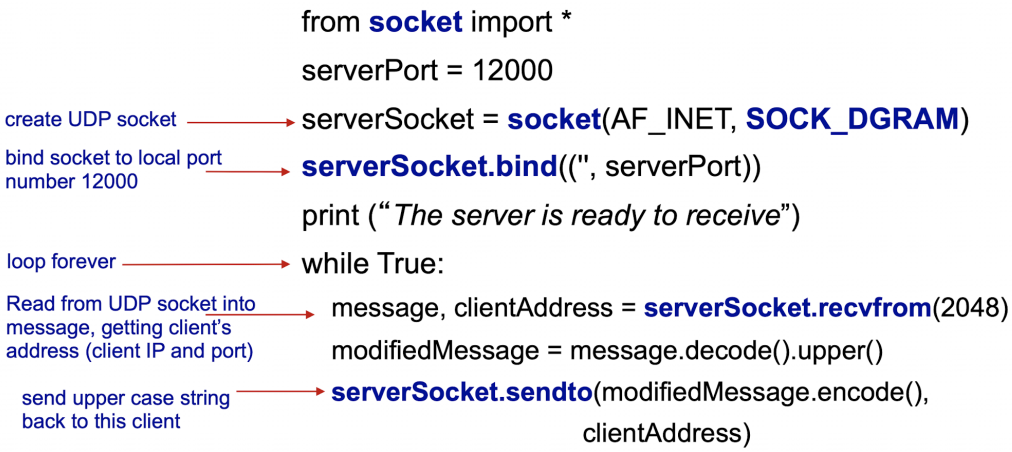
i.

ii. sendto 함수를 이용하여 message를 보냄

iii. 서버 주소와 portnumber 를 미리 알아야함

e. **Example app: UDP server**

Python UDPServer



i.

f. **Socket programming with TCP**

i. client must contact server

ii. client TCP socekt, 분명만 IP address, 서버 프로세스의 port number를 만든다

iii. client는 server TCP에 연결할때 TCP를 설립한다.(이때 socket을 만든다)

iv. TCP는 reliable을 제공하며, byte-stream(pipe 형태)을 순서에 맞게 client와 server에 제공한다

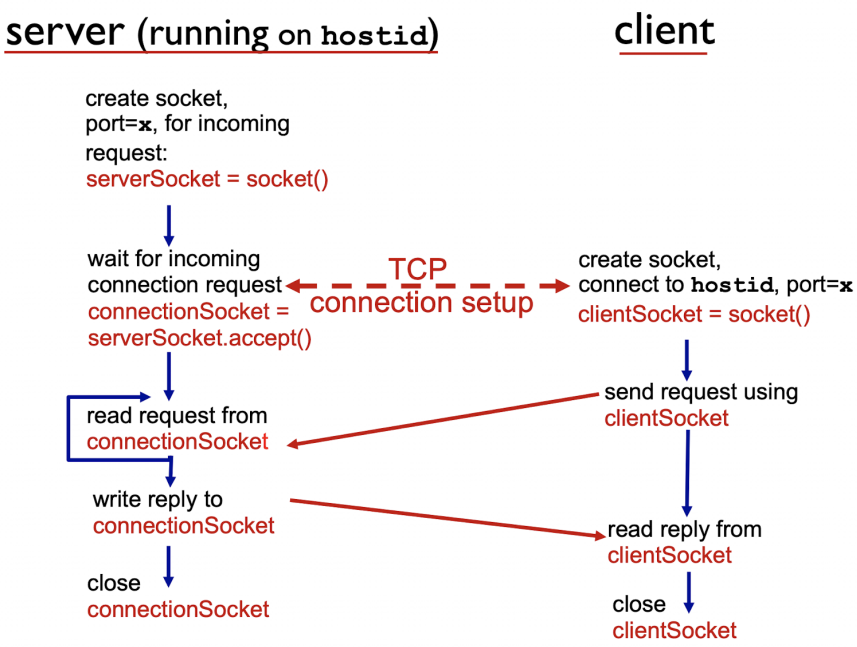
v. recvfrom으로 받는다.

vi. server가 client와 contact할때 새로운 socket을 생성한다.

vii. UDP는 1번에 1번씩 chunk를 보낸다.(Group of byte)

viii. TCP는 pipe를 만들어 byte-stream, 즉 byte를 쭉 보냄

g. **Client/server socket interaction : TCP**



i.

h. **Example app: TCP client**

Python TCPClient

```
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()
```

create TCP socket for server, remote port 12000

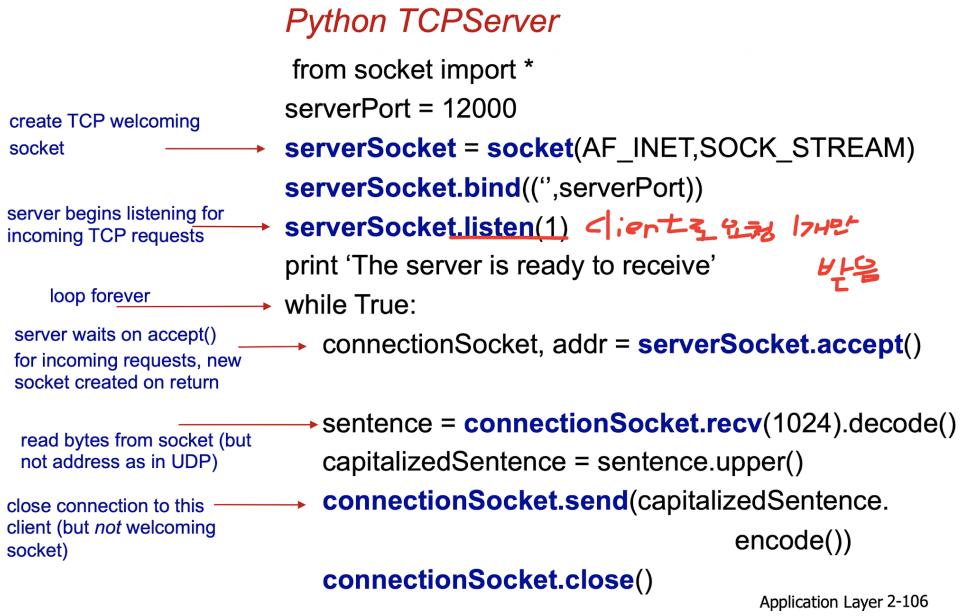
No need to attach server name, port

i.

ii. TCP에선 그냥 send만하고 서버주소와 port number를 명시하지않음

iii. recvfrom이 아닌 recv임

i. **Example app: TCP server**



i.

1. TCP server에 있는 listen(1)은 client로 요청을 1개만 받는다는 뜻임

j. **Summary**

i. application architectures
   1. client-server
   2. P2P

ii. application server의 필요사항
   1. reliability, bandwidth, delay

iii. Internet transport service model
   1. connection-oriented, reliable: TCP
   2. unreliable, datagrams: UDP

iv. specific protocols:
   1. HTPP
   2. SMTP,POP,IMAP
   3. DNS
   4. P2P:BitTorrent

v. video streaming, CDNs

vi. socket programming: TCP, UDP sockets

2. **유용한 몇가지 명령어**
    a. traceroute
    b. ping
    c. telent
    d. ifconfig
        i. 호스트의 interface 정보를 보여줌
        ii. IP Address를 알 수 있음
    e. ssh
    f. sco
    g. nc
        i. TCP 또는 UDP 에서 client또는 server 중 1개만 구현했을 때 둘 중 1개의 역할을 대신하여 connection을 할 수 있음
        ii. message를 받았는지 못받았는지 확인 가능
3. **Multiplexing File Descriptors**
    a. FD_SET: 어떤 bit를 세팅하기 위한 함수
    b. FD_CLR:어떤 bit를 0으로 만들어라
    c. FD_ISSET:그 n번 bit가 세팅되어있는지 check하는 함수
    d. FD_zero: 모든 bit를 다 0으로 만듬