소켓프로그래밍

라즈베리파이로 배우는 소켓 통신 프로그래밍

동양미래대학교

컴퓨터공학부 정석용

동양미래대학교

[실습과제 4]

TCP 소켓프로그램 프로그래밍 - chatting

```
import socket, threading
class ClientThread(threading.Thread):
    def __init__(self,addr,c):
        threading.Thread.__init__(self)
        self.csocket = c

    def run(self):
        print ("connection from : ", addr)
        msg = ''
        while True:
            data = self.csocket.recv(2048)
            msg = data.decode()
            if msg=='bye':
                break
            for client in client_sockets :
                client.send(bytes(msg,'UTF-8'))
        print ("Client at ", addr , " disconnected...")
        if self.csocket in client_sockets :
            client_sockets.remove(self.csocket)
        self.csocket.close()
```

```
client_sockets = []
host = '0.0.0.0'
port = 9999
s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET,
             socket.SO_REUSEADDR, 1)
s.bind((host, port))
s.listen(10)
print("Chat Server start")
while True:
    c, addr = s.accept()
    client_sockets.append(c)
    n = ClientThread(addr, c)
    n.start()
    print('# of client:', len(client_socks))
```

```
import socket, threading
class ChatThread(threading.Thread):
    def __init__(self,s):
        threading.Thread.__init__(self)
        self.csocket = s

    def run(self):
        while True:
            data = self.csocket.recv(2048)
            print('receive : ', data.decode())
```

```
host = "127.0.0.1" # 서버 IP 주소
port = 9999
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
Print('chat client start')

n = ChatThread(s)
n.daemon = True
n.start()

while True:
        message = input()
        s.send(message.encode())
        if message =='bye':
                break
s.close()
```

키보드 입력 메시지를 전송하고 "bye"를 입력하면 연결 종료

실행

Terminal 1 (Chat Server)

```
> python server.py
```

1. 자신의 컴퓨터에서 4개 터미널을 생성해서 실습
2. 다른 컴퓨터에서 서버에 접속하여 채팅해 보기

> 자신의 채팅 메시지도 자신에게 보임 ??

Terminal 2 (Chat Client)

```
> python client.py
Hi I am Jung
[chat message] Hi I am Jung
```

Terminal 3 (Chat Client)

```
> python client.py
[chat message] Hi I am Jung
```

Terminal 4 (Chat Client)

```
> python client.py
[chat message] Hi I am Jung
```

[도전 과제 4-1]

TCP 소켓프로그램 프로그래밍 – chating

실습 과제 6에서 자신의 메시지는 보이지 않도록 처리하시오.

```
import socket, threading
class ClientThread(threading.Thread):
    def __init__(self,addr,c):
        threading.Thread.__init__(self)
        self.csocket = c

    def run(self):
        print ("connection from : ", addr)
        msg = ''
        while True:
            data = self.csocket.recv(2048)
            msg = data.decode()
            if msg=='bye':
              break
            for client in client_sockets :
               if client != self.csocket :
                  client.send(bytes(msg,'UTF-8'))
        print ("Client at ", addr , " disconnected...")
        if self.csocket in client_sockets :
               client_sockets.remove(self.csocket)
        self.csocket.close()
```

```
client_sockets = []
host = '0.0.0.0'
port = 9999
s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET,
               socket.SO_REUSEADDR, 1)
s.bind((host, port))
s.listen(10)
print("Chat Server start")
while True:
    c, addr = s.accept()
    client_sockets.append(c)
    n = ClientThread(addr, c)
    n.start()
    print('# of client:', len(client_socks))
```

Chat - 클라이언트로 부터 수신한 메시지를 모든 클라이언트로 전송

```
import socket, threading
class ChatThread(threading.Thread):
    def __init__(self,s):
        threading.Thread.__init__(self)
        self.csocket = s

    def run(self):
        while True:
            data = self.csocket.recv(2048)
            print('receive : ', data.decode())
```

```
host = "127.0.0.1" # 서버 IP 주소
port = 9999
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
Print('chat client start')

n = ChatThread(s)
n.daemon = True
n.start()

while True:
        message = input()
        if message =='bye':
                s.send(message.encode())
                break
        s.send(message.encode())
s.close()
```

키보드 입력 메시지를 전송하고 "bye"를 입력하면 연결 종료

[추가 과제 4-2]

TCP 소켓프로그램 프로그래밍 – chating

클라이언트 실행시 서버 IP 주소와 port 번호를 입력 받자…

```
import socket, threading
Import sys

class ChatThread(threading.Thread):
    def __init__(self,s):
        threading.Thread.__init__(self)
        self.csocket = s

    def run(self):
        while True:
            data = self.csocket.recv(2048)
            print('receive : ', data.decode())
```

```
Print(len(sys.argv), sys.argv)
host = sys.argv[1]
port = int(sys.argv[2])

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
Print('chat client start')

n = ChatThread(s)
n.daemon = True
n.start()

while True:
        message = input()
        if message =='bye':
                s.send(message.encode())
                break
        s.send(message.encode())
s.close()
```

실행

```
> python chat_client.py 127.0.0.1 9999
```

키보드 입력 메시지를 전송하고 "bye"를 입력하면 연결 종료

```
import socket, threading
Import sys

class ChatThread(threading.Thread):
    def __init__(self,s):
        threading.Thread.__init__(self)
        self.csocket = s

    def run(self):
        while True:
            data = self.csocket.recv(2048)
            print('receive : ', data.decode())
```

```
if len(argv) != 3 :
        print(f'usage : python {sys.argv[0]} ip# port#')
        quit()
host = sys.argv[1]
port = int(sys.argv[2])

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
Print('chat client start')

n = ChatThread(s)
n.daemon = True
n.start()

while True:
        message = input()
        if message =='bye':
                s.send(message.encode())
                break
        s.send(message.encode())
s.close()
```

실행 : 오류 처리 기능 추가

```
> python chat_client.py 127.0.0.1
```

키보드 입력 메시지를 전송하고 "bye"를 입력하면 연결 종료

[도전 과제 4-3]

채팅 클라이언트에서 자신의 이름(별명)을 입력 받고
채팅 메시지에 붙여서 보낸다.

- **name : syjung**
- **[syjung] hello**

[더 알아보기 과제 4-4]

딕셔너리(dictionary) 데이터 전송

JSON (JavaScript Object Notation)

- XML, YAML 과 함께 **데이터를 저장하고 교환(exchange data)하는데 사용하는 텍스트 데이터 포맷** 중의 하나

문자열

```
{ "name" : "Jones" }
```

숫자

```
{ "number_1" : 210,
  "number_2" : 215,
  "number_3" : 21.05,
  "number_4" : 10.05
}
```

객체

```
{ "Influencer" : { "name" : "Jaxon" , "age" : "42"} }
```

배열

```
{ "Influencers" : [
    { "name" : "Jaxon",
      "age" : 42
    }
    { "name" : "Miller",
      "age" : 35
    }
  ]
}
```

Dictionary in Python

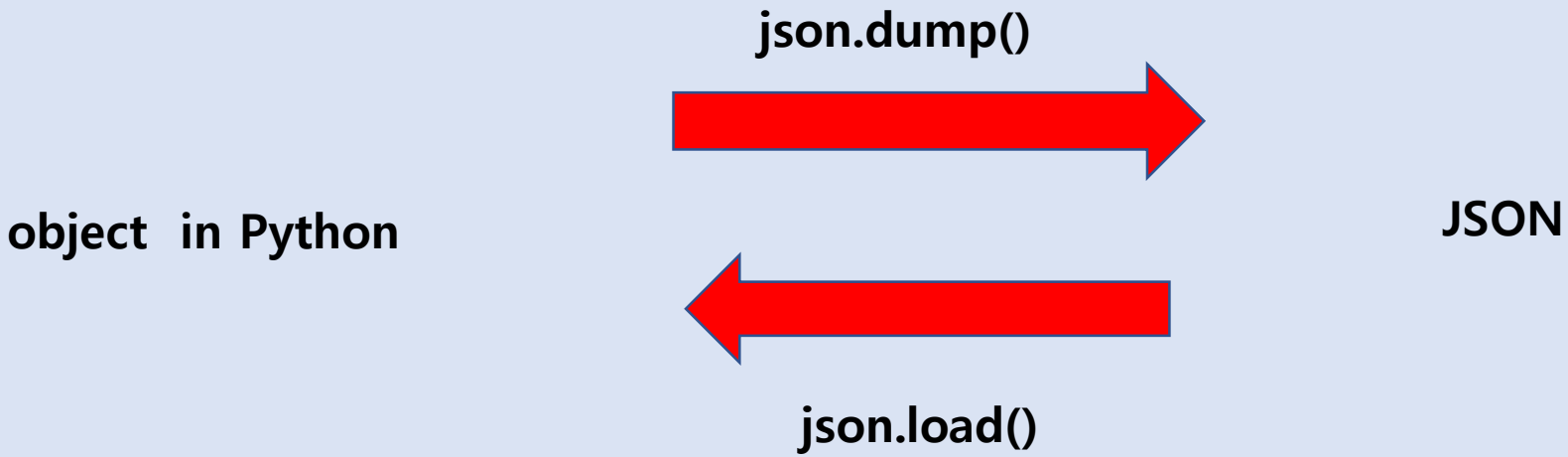
- 딕셔너리 타입은 키(key)와 값(value)으로 맵핑되어 있는 순서가 없는 집합입니다.
- 순서가 없기 때문에 인덱스로 접근할 수 없고, 키로 접근 가능

Dictionary 사용 예

```
>>> a = {1: 5, 2: 3} # int 사용
>>> a
{1: 5, 2: 3}
>>> a = {(1,5): 5, (3,3): 3} # tuple사용
>>> a
{(1, 5): 5, (3, 3): 3}
>>> a = { 3.6: 5, "abc": 3} # float 사용
>>> a
{3.6: 5, 'abc': 3}
>>> a = { True: 5, "abc": 3} # bool 사용
>>> a
{True: 5, 'abc': 3}
```

```
>>> d = {'abc' : 1, 'def' : 2}
>>> d[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 0
>>> d['abc']
1
```

import json



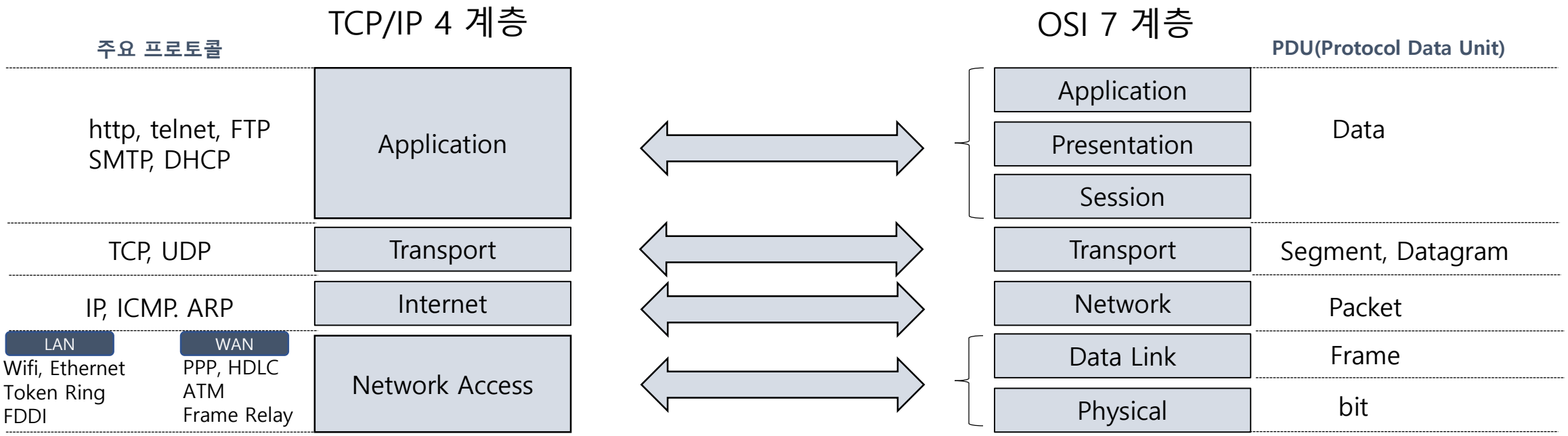
```
import socket
import json

s = socket.socket()                                          # 통신용 소켓 생성(TCP 소켓)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)  # port가 다른 사용 중일 때도 강제 점유
host ='0.0.0.0'    # server IP address                       # 서버 프로그램이 연결될 IP 주소(컴퓨터 모든 IP)
Port = 9000                                                  # 서버 프로그램이 연결될 포트 번호 9000번
s.bind((host, port))                                         # 소켓을 (host, port)에 연결
s.listen(10)                                               # 클라이언트 연결 요청 대기(동시 요청 수락 수)

while True:
    c, addr = s.accept()                                   # 클라이언트 연결 요청 수락
    print('Got connection from', addr)

    for i in range(10)
            data = c.recv(2048)                              # json 파일 수신(byte type)
            r_data = json.loads(data.decode())               #byte 타입의 json 데이터를 str으로 변환 후, dictionary 로 변환
            print('r_data :', r_data)

            s_data = json.dumps(r_data)                      # dictionary를 Json으로 변환
            c.sendall(bytes(s_data, 'utf-8'))
      print('client at', addr, 'disconnected...')
s.close()
```

```
import socket
import json
import time
s = socket.socket()                          # 통신용 소켓 생성
Host = '127.0.0.1'                           # 연결할 서버의 IP 주소
Port = 9000                                  # 연결할 서버의 포트 번호

s.connect((host, port))                      # 소켓으로 서버(host, port)연결 요청
data = {}
data['name'] = 'jung'
for count in range(10):
    data['count'] = count                    # 키보드 데이터를 읽어옴
    body = json.dumps(data)                  # dic to json
    print('send message: ', bytes(body, 'utf-8')
    s.sendall(bytes(out_data, 'UTF-8'))      # 문자열을 Unicode(UTF-8) Byte 객체로 변환 전송

    in_data = s.recv(1024)                   # 서버가 전송한 데이터 수신(최대 1024 바이트)
    r_data = json.loads(in_data.decode())
    print('From Server: ', r_data)
    time.sleep(3)

s.close
```

```
> python server.py
> python client.py
```

[실습과제 5]

UDP 소켓프로그램 프로그래밍



- **1960년대 후반 미국 국방성의 ARPANET에서 유래**
- **1982년 미 국방성이 TCP/IP를 국방 전산망 표준으로 선언**
- **이후 IBM, AT&T, DEC 등 대규모 컴퓨터 회사에서 TCP/IP채택**

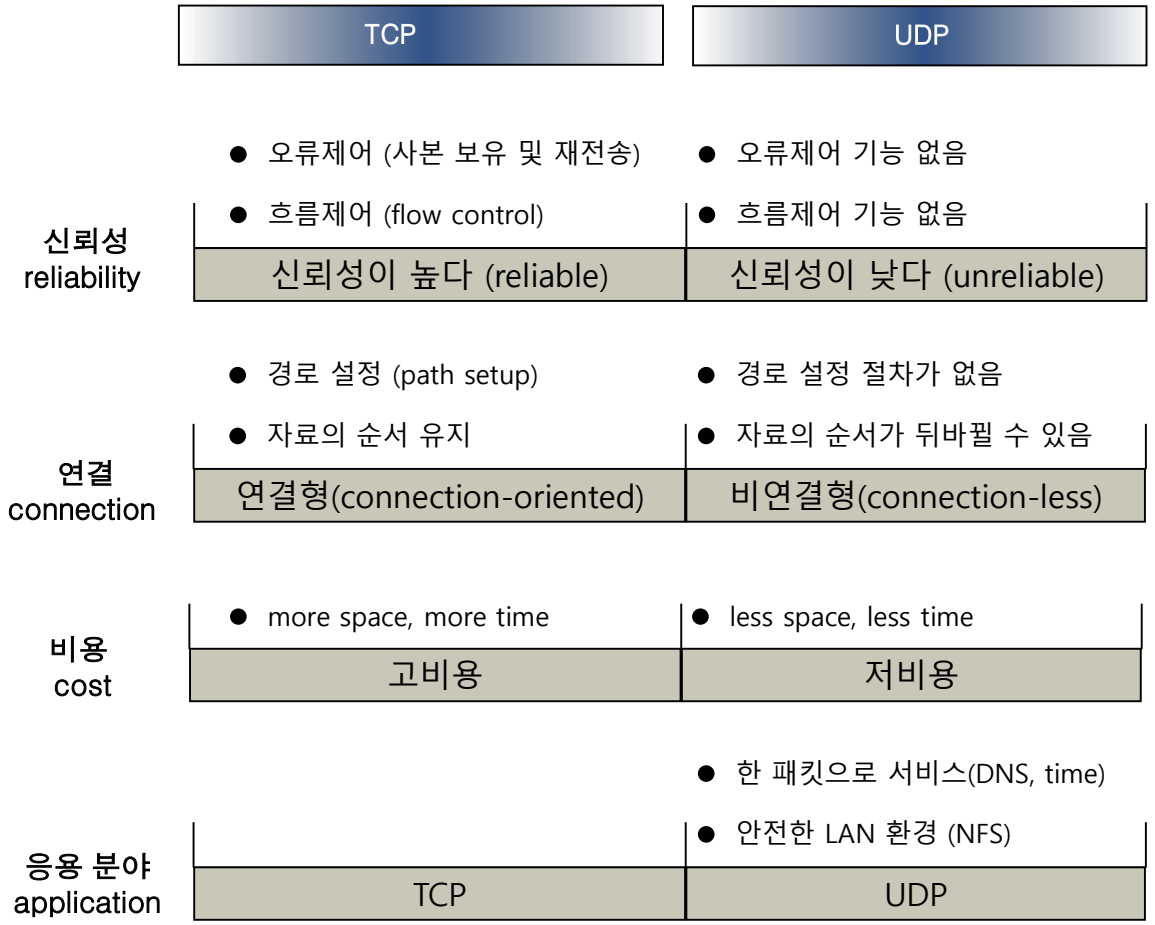
De Facto(사실상의 표준) → 현재 사용하는 인터넷

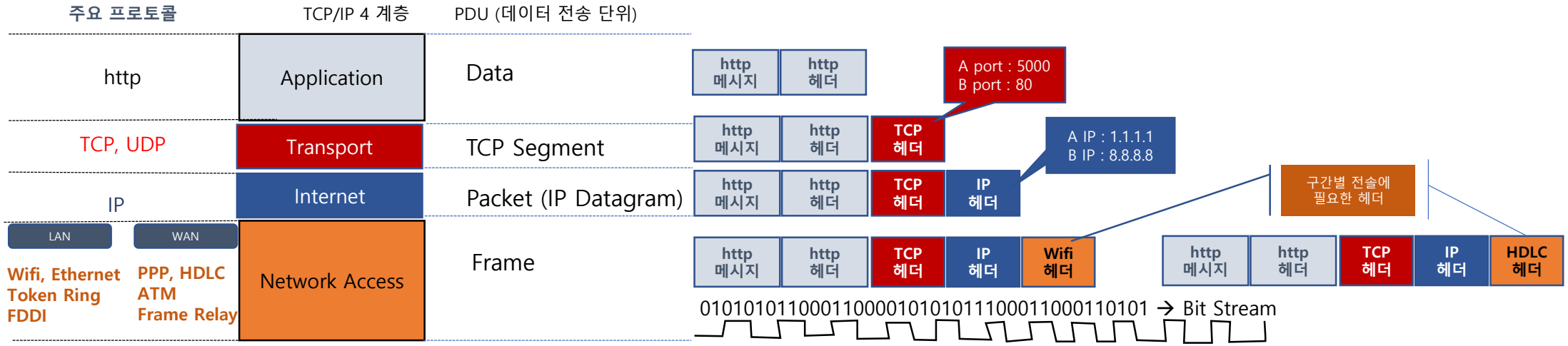
- **1983년 국제 표준화 기구인 ISO에서**
- **서로 다른 규약을 따르던 통신망 간에 서로 통신할 수 있도록 표준 규약을 정의**

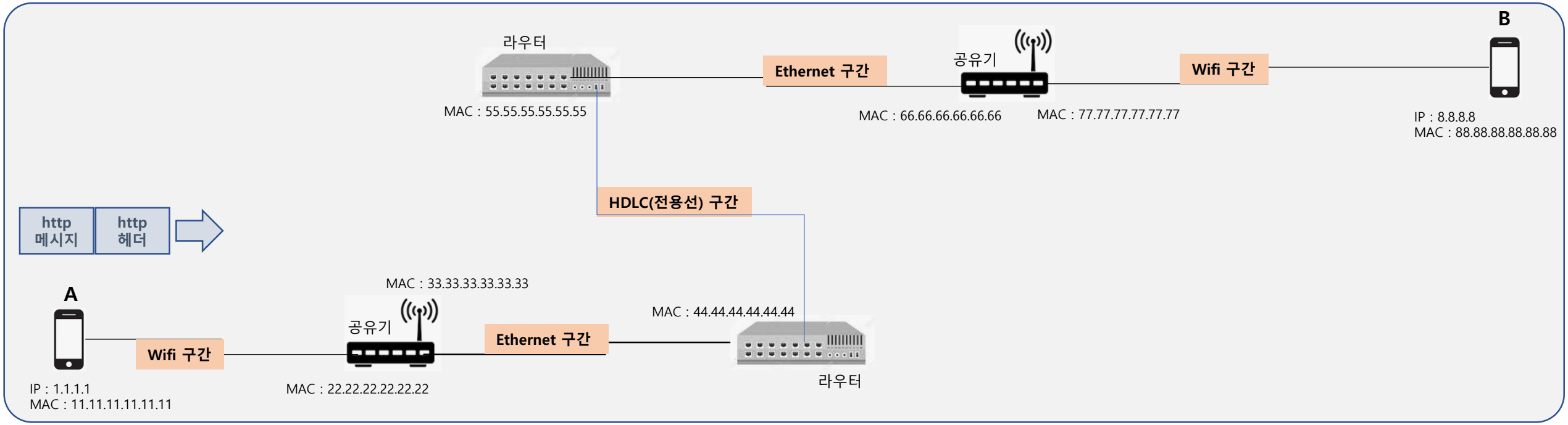
De Jure(명목상의 표준)→ 구현되지 않음. 기준을 제시

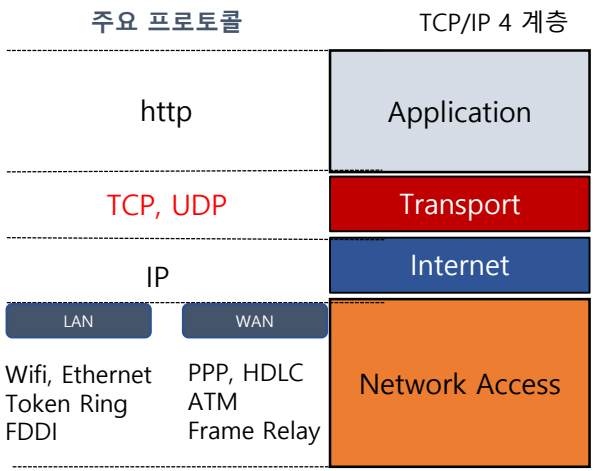
서로 다른 회사가 만든 컴퓨터, 네트워크 장비 간에 통신이 가능하도록 표준 규약 정의

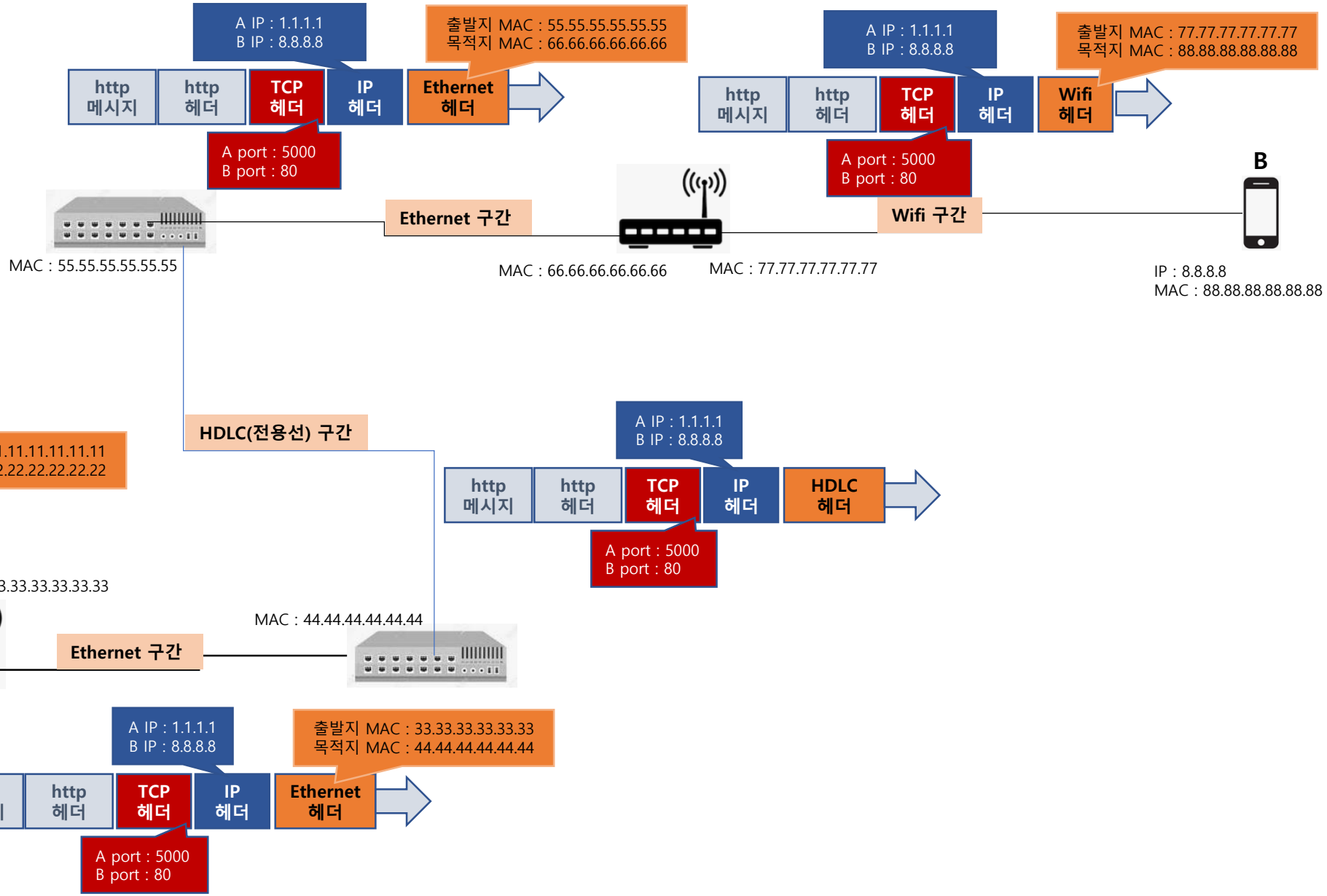
TCP v.s. UDP


TCP
UDP
신뢰성
reliability
● 오류제어 (사본 보유 및 재전송)
● 흐름제어 (flow control)
신뢰성이 높다 (reliable)
● 오류제어 기능 없음
● 흐름제어 기능 없음
신뢰성이 낮다 (unreliable)
연결
connection
● 경로 설정 (path setup)
● 자료의 순서 유지
연결형(connection-oriented)
● 경로 설정 절차가 없음
● 자료의 순서가 뒤바뀔 수 있음
비연결형(connection-less)
비용
cost
● more space, more time
고비용
● less space, less time
저비용
응용 분야
application
TCP
● 한 패킷으로 서비스(DNS, time)
● 안전한 LAN 환경 (NFS)
UDP


주요 프로토콜
TCP/IP 4 계층
PDU (데이터 전송 단위)
http
Application
Data
TCP, UDP
Transport
TCP Segment
IP
Internet
Packet (IP Datagram)
LAN
WAN
Wifi, Ethernet
Token Ring
FDDI
PPP, HDLC
ATM
Frame Relay
Network Access
Frame
http 메시지
http 헤더
TCP 헤더
IP 헤더
Wifi 헤더
HDLC 헤더
A port : 5000
B port : 80
A IP : 1.1.1.1
B IP : 8.8.8.8
구간별 전송에 필요한 헤더
01010101100011000010101011100011000110101 → Bit Stream

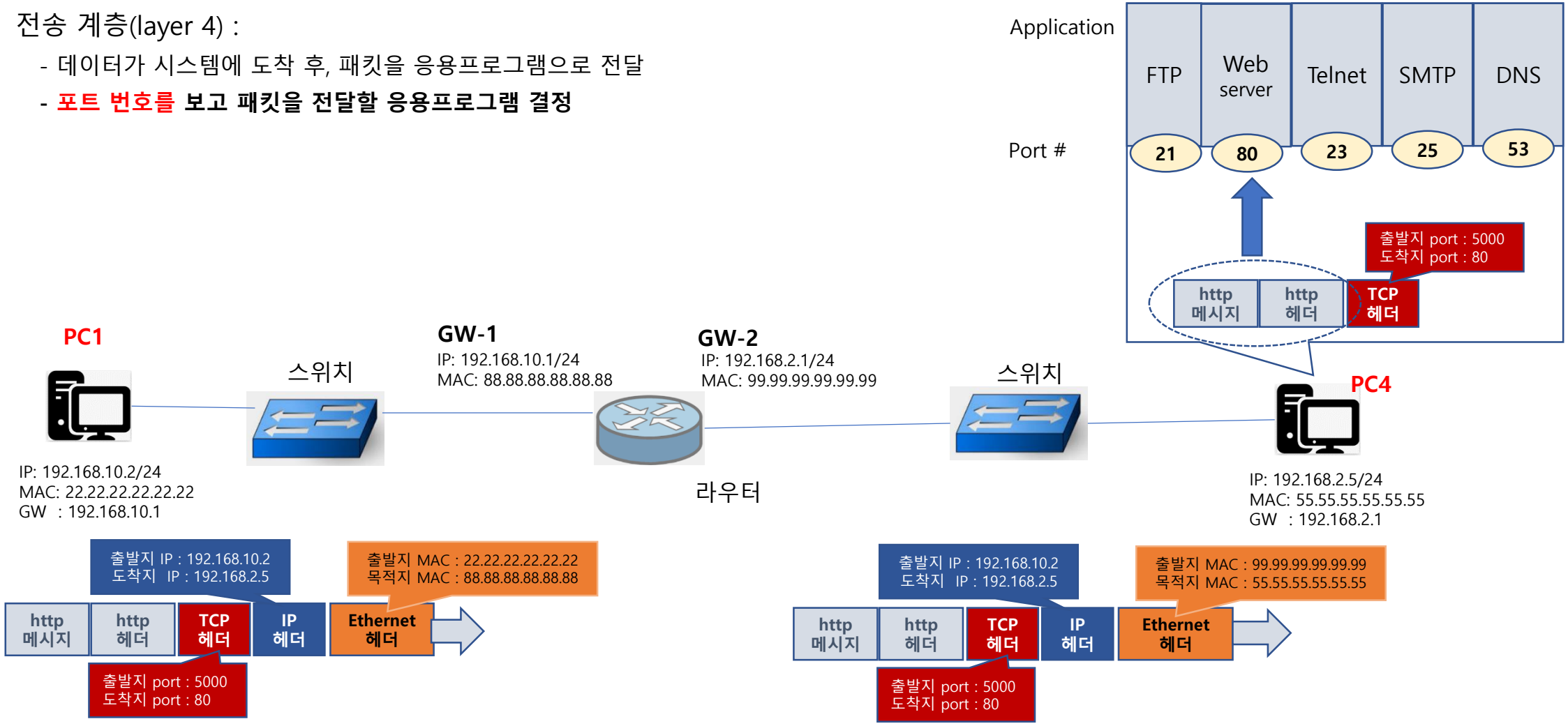

라우터
MAC : 55.55.55.55.55.55
Ethernet 구간
공유기
MAC : 66.66.66.66.66.66
MAC : 77.77.77.77.77.77
Wifi 구간
B
IP : 8.8.8.8
MAC : 88.88.88.88.88.88
HDLC(전용선) 구간
http 메시지
http 헤더
A
MAC : 33.33.33.33.33.33
공유기
Wifi 구간
Ethernet 구간
MAC : 44.44.44.44.44.44
라우터
IP : 1.1.1.1
MAC : 11.11.11.11.11.11
MAC : 22.22.22.22.22.22


주요 프로토콜
TCP/IP 4 계층
http
Application
TCP, UDP
Transport
IP
Internet
LAN
WAN
Wifi, Ethernet
Token Ring
FDDI
PPP, HDLC
ATM
Frame Relay
Network Access


A IP : 1.1.1.1
B IP : 8.8.8.8
출발지 MAC : 55.55.55.55.55.55
목적지 MAC : 66.66.66.66.66.66
http 메시지
http 헤더
TCP 헤더
IP 헤더
Ethernet 헤더
A port : 5000
B port : 80
MAC : 55.55.55.55.55.55
Ethernet 구간
MAC : 66.66.66.66.66.66
A IP : 1.1.1.1
B IP : 8.8.8.8
출발지 MAC : 77.77.77.77.77.77
목적지 MAC : 88.88.88.88.88.88
Wifi 헤더
A port : 5000
B port : 80
Wifi 구간
MAC : 77.77.77.77.77.77
B
IP : 8.8.8.8
MAC : 88.88.88.88.88.88
HDLC(전용선) 구간
A IP : 1.1.1.1
B IP : 8.8.8.8
출발지 MAC : 11.11.11.11.11.11
목적지 MAC : 22.22.22.22.22.22
Wifi 헤더
A port : 5000
B port : 80
A
MAC : 33.33.33.33.33.33
MAC : 44.44.44.44.44.44
Ethernet 구간
Wifi 구간
IP : 1.1.1.1
MAC : 11.11.11.11.11.11
MAC : 22.22.22.22.22.22
A IP : 1.1.1.1
B IP : 8.8.8.8
HDLC 헤더
A port : 5000
B port : 80
A IP : 1.1.1.1
B IP : 8.8.8.8
출발지 MAC : 33.33.33.33.33.33
목적지 MAC : 44.44.44.44.44.44
Ethernet 헤더
A port : 5000
B port : 80

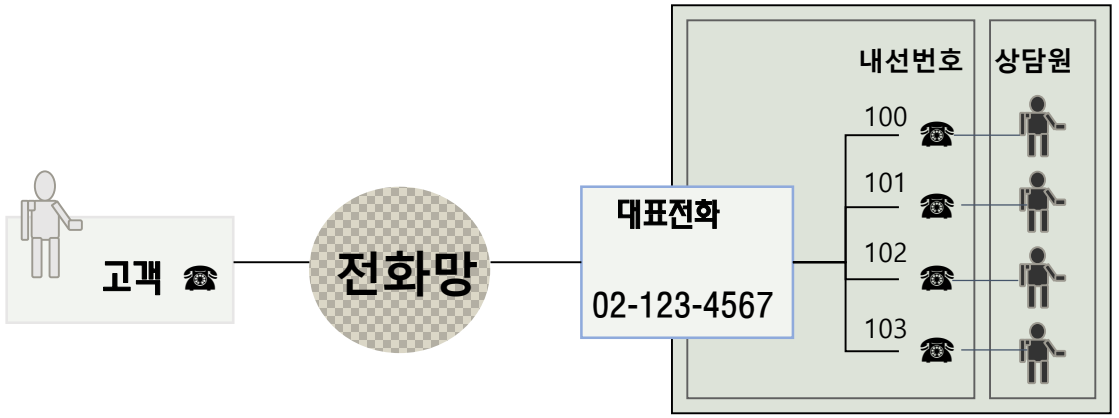
전송 계층(layer 4) :

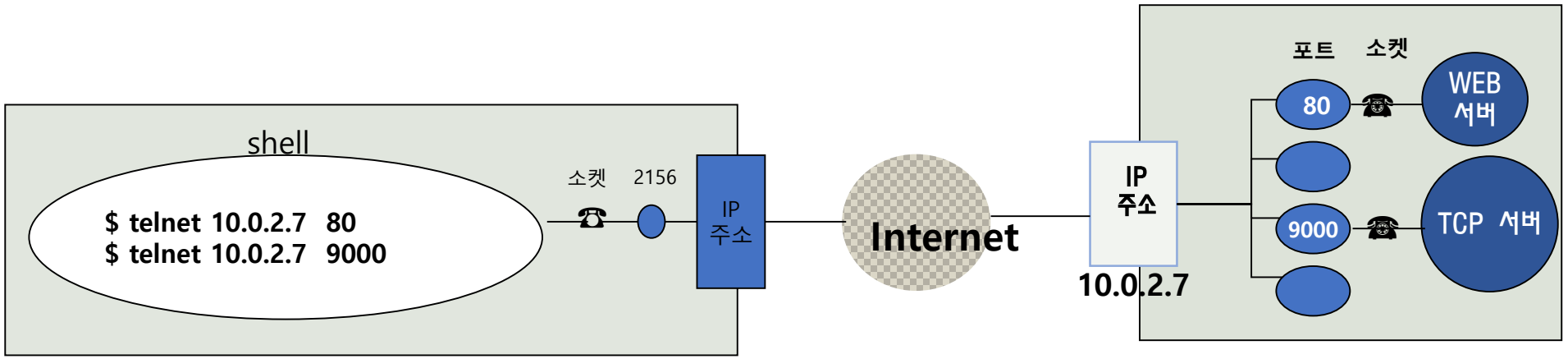
- 데이터가 시스템에 도착 후, 패킷을 응용프로그램으로 전달
- **포트 번호를 보고 패킷을 전달할 응용프로그램 결정**



Port 번호 | 2Byte(16bit) | 0000000000000000 (0) ~ 1111111111111111 ($2^{16} - 1$)
0 ~ 65535

포트 번호	설명	
0 ~ 1023	Well known port	일반적으로 서비스를 위해 사용되면 잘 알려진 포트 FTP – 21 telnet – 23 SMTP – 25 HTTP – 80 HTTPS - 443
1024 ~ 49151	Registered port	일반 사용자에 의해 사용되는 포트
49152 ~	Dynamic & Private port	일반적으로 사용되지 않고 학술, 연구 목적으로 사용


내선번호
상담원
100
101
102
103
고객
전화망
대표전화
02-123-4567


shell
$ telnet 10.0.2.7 80
$ telnet 10.0.2.7 9000
소켓
2156
IP
주소
Internet
IP
주소
10.0.2.7
포트
소켓
80
9000
WEB
서버
TCP 서버

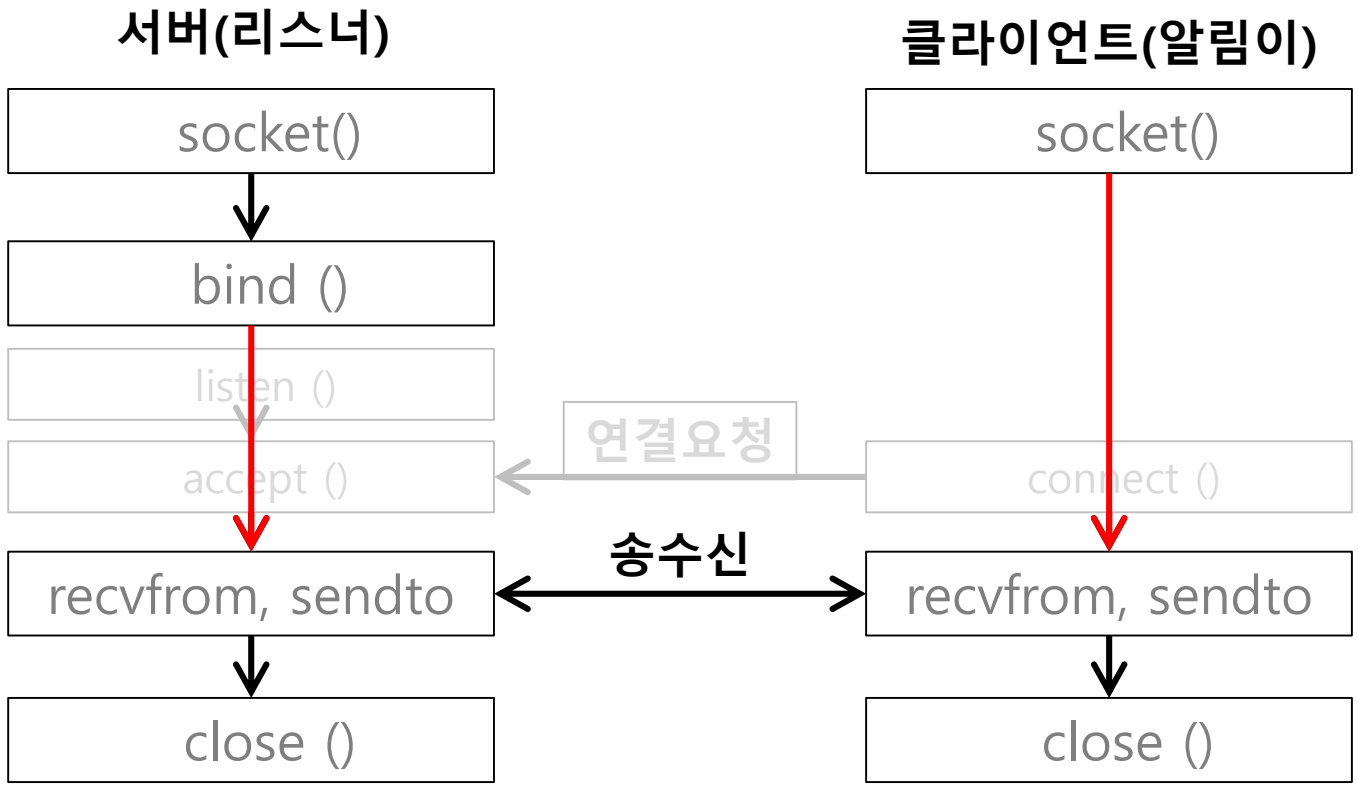
TCP v.s. UDP

TCP 소켓 생성

```
t1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM, socket.IPPROTO_TCP)
t2 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

UDP 소켓 생성

```
u1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
u2 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```



TCP 통신으로 1:1 통신 지원

파일명 : listener.py

```
import socket
host = "0.0.0.0"
port = 20001

u = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
u.bind((host, port))
print("UDP server up and listening")
while(True):
    c, addr = u.recvfrom(1024)
    print(f'Message [{c}] from {addr}')
    u.sendto(b'Hello UDP Client', addr)
```

실행
```
> python listener.py
```

파일명 : notify.py

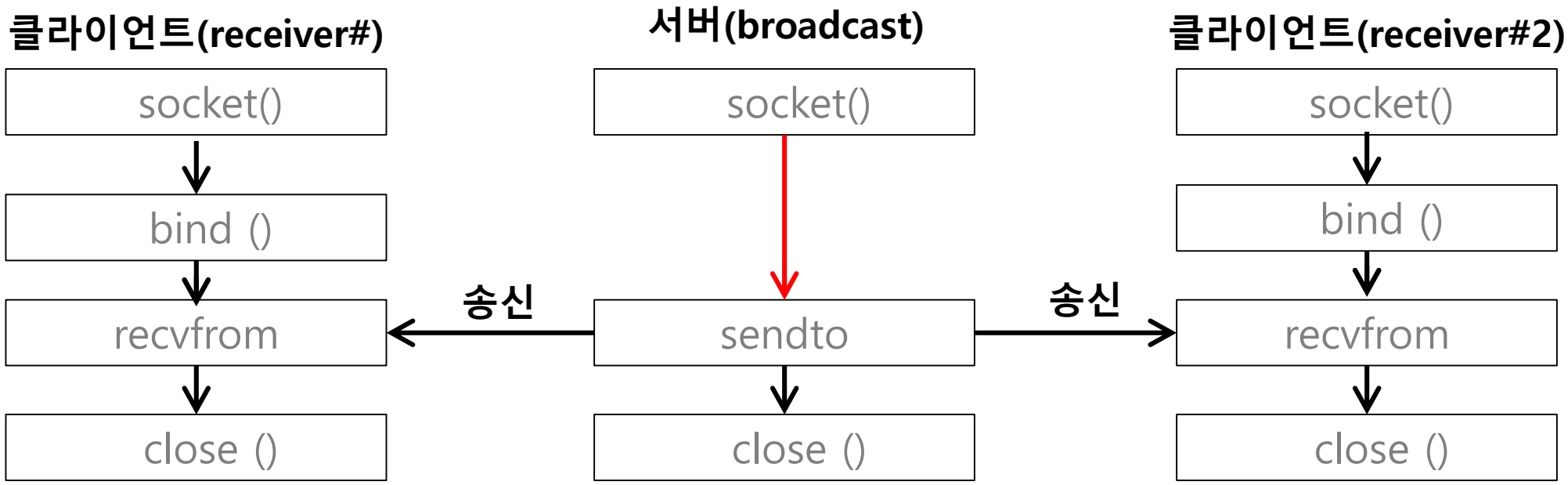
실행
```
> python notify.py
```

```
import socket
name = socket.gethostname()
host = "127.0.0.1"
port = 20001

u = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
u.settimeout(5.0)
u.sendto("Hello UDP Listener".encode('UTF-8'), (host, port))
c, addr = u.recvfrom(1024)
print(f'Message from Server {addr}')
print(f'[Recv Data] {c.decode()}')
```

[실습과제 6]

UDP 브로드캐스팅 소켓프로그램 프로그래밍

UDP 브로드캐스트 - 메시지 방송



UDP 통신으로 다수 컴퓨터에 메시지 전송 (응답 기다리지 않음)

UDP 브로드캐스트 소켓 프로그램 - 서버

파일명 : broadcaster.py

```
import socket
import time
host = '<broadcast>'
port = 37020
u = socket.socket(socket.AF_INET,socket.SOCK_DGRAM, socket.IPPROTO_UDP)
u.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
u.settimeout(0.2)
print("UDP Broadcast server up and announce message")
while(True):
    u.sendto(b"broadcast message", (host, port))
    print("Braodcast message")
    time.sleep(1)
```

실행

```
> python broadcaster.py
```

UDP 브로드캐스트 - 클라이언트

파일명 : receiver.py

실행
```
> python receiver.py
```

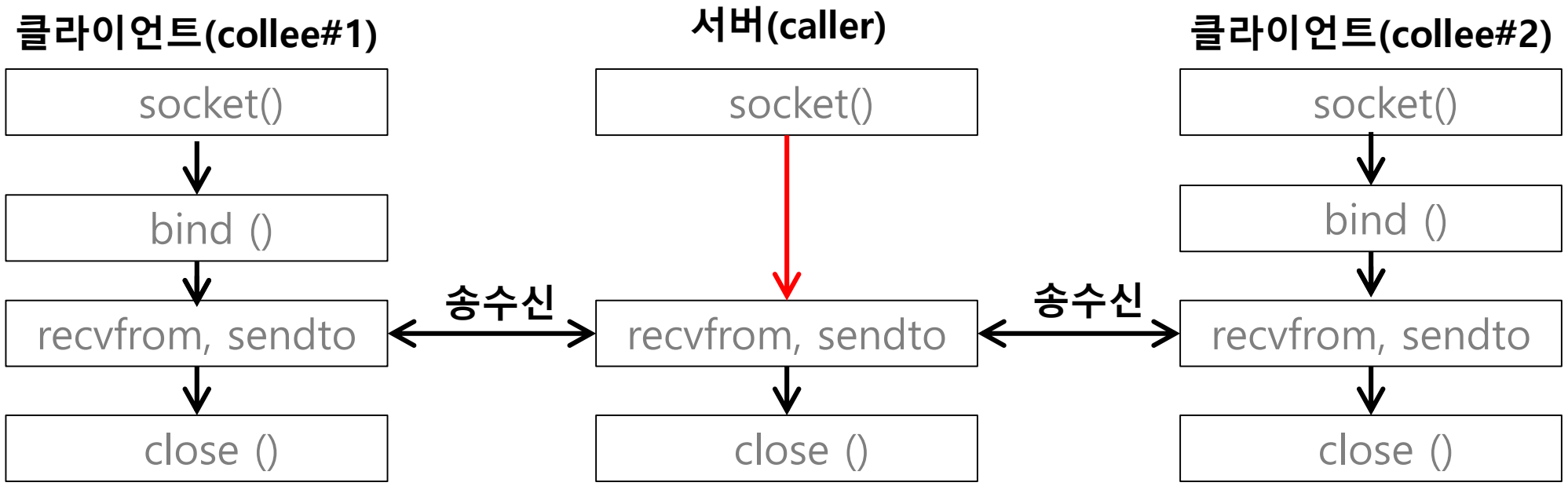
```
import socket

host = ""
port = 37020
u = socket.socket(family=socket.AF_INET,type=socket.SOCK_DGRAM,proto=socket.IPPROTO_UDP)
u.setsockopt(socket.SOL_SOCKET,
             socket.SO_BROADCAST, 1)
u.bind((host, port))
print("Broadcast receiver start")
while True:
    c, addr = u.recvfrom(1024)
    print(f'Message [{c}] from Server {addr}')
```

[실습과제 7]

UDP 브로드캐스팅 소켓프로그램 프로그래밍 확장

UDP broadcast 확장

UDP 1:N caller, collee



UDP 브로드캐스트 소켓 프로그램 - 서버

파일명 : caller.py

실행
```
> python caller.py
```

```
import socket
import time
host = '<broadcast>'
port = 37021
u = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
u.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
u.settimeout(3)
print("UDP Broadcast to check devices")
while(True):
    u.sendto(b"broadcast message", (host, port))
    print("Braodcast message")
    try:
        while True:
            c, addr = u.recvfrom(1024)
            print(f'Message from client {addr}')
    except:
        pass
    time.sleep(1)
```

UDP 브로드캐스트 - 클라이언트

파일명 : callee.py

```
import socket
host = ""
port = 37021
u = socket.socket(family=socket.AF_INET,
                  type=socket.SOCK_DGRAM,
                  proto=socket.IPPROTO_UDP)

u.setsockopt(socket.SOL_SOCKET,
             socket.SO_BROADCAST, 1)
u.bind((host, port))
print("Answer to caller")
while True:
    c, addr = u.recvfrom(1024)
    print(f'Call from Server {addr}')
    u.sendto(b"receive message", addr)
```

실행

```
> python callee.py
```