통신 프로토콜의 이해

전남대학교 소프트웨어중심대학사업단
유재명

컴퓨터

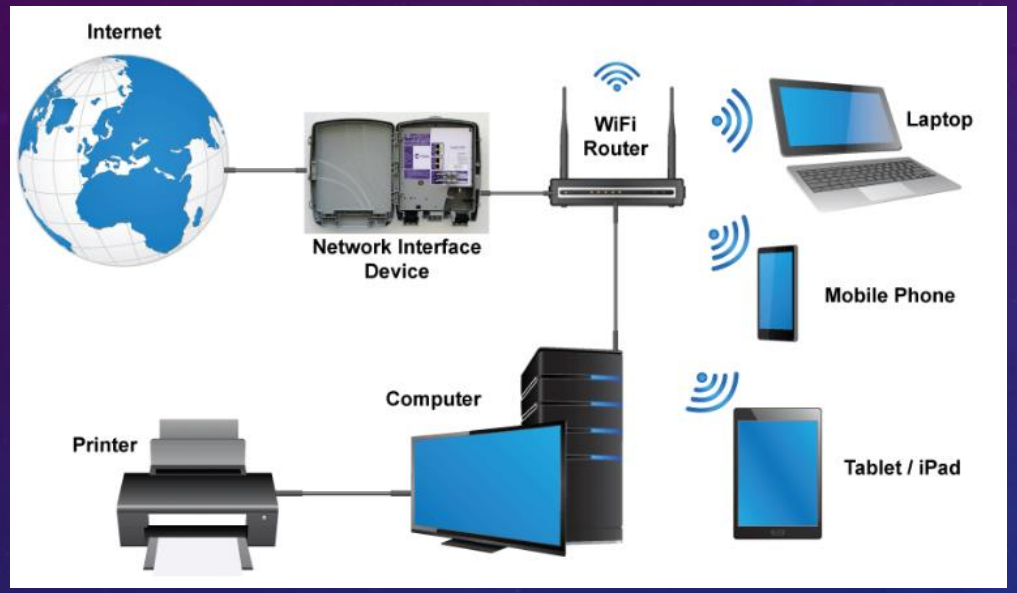
- 하드웨어
  - CPU : 32비트, 64비트 , 역할은 " 기억(레지스터,캐쉬), 연산, 제어 "
  - GPU : NVIDIA T4, V100, A100, L40S 등
  - 메모리 : ROM(비휘발성), RAM(휘발성), 플래쉬(Flash)
  - 디스크: HDD, SSD, USB, Flash, SD 등
  - 메인보드
  - 주변장치
- 소프트웨어 (운영체제, Operating System) : 자원 관리
  - Task, Scheduler, IPC
  - Memory
  - Devices
  - Networks

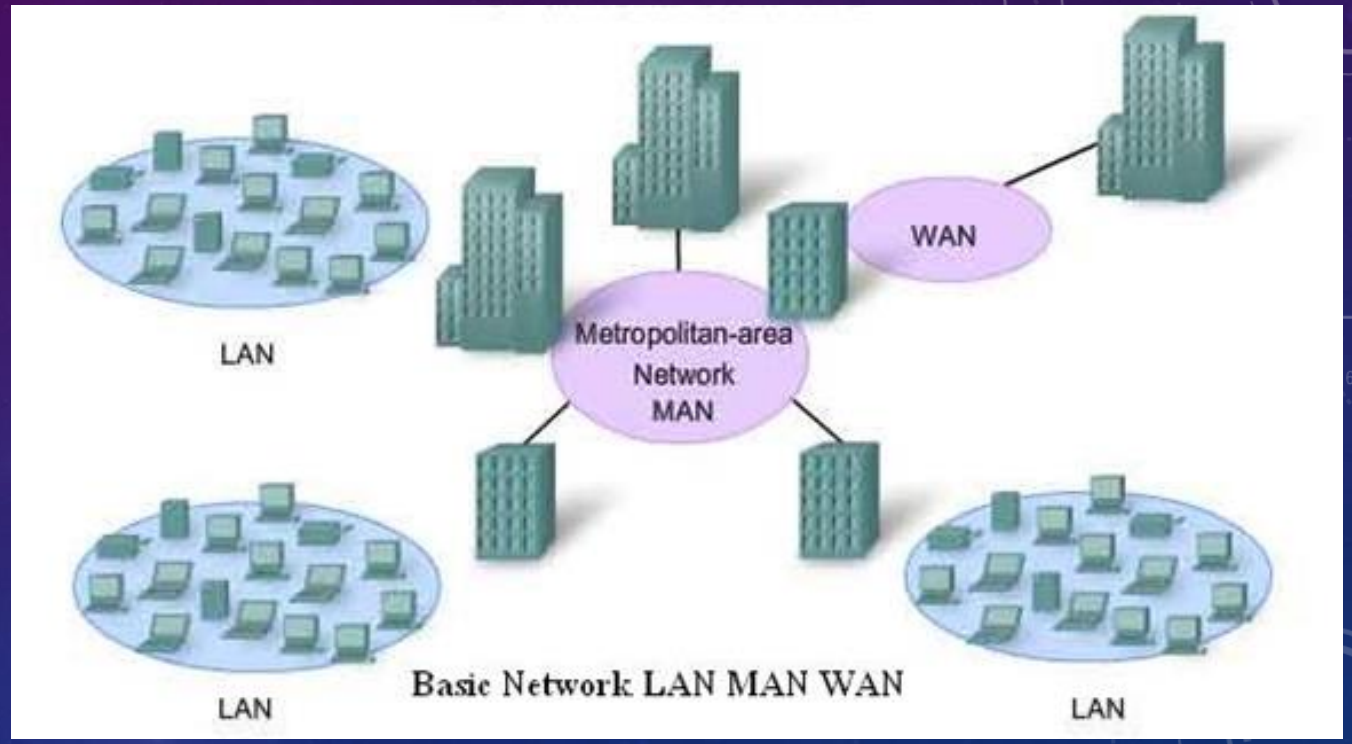
웹과 인터넷

- 인터텟 초기에는 지역내에서만 연결, LAN(Local Area Network)
- 1969년, 미국 국방성 고등 연구 계획국(ARPA, Advanced Research Project Agency)에서 미국내 여러 대학들과 연구소 간의 서로 단절되었던 LAN을 하나로 통합하는 프로젝트 진행 : ARPAnet
- 초창기 인터넷은 국방용이나 연구용으로만 사용
  - 초기 서비스 : 대부분 문자 위주로 채팅, 이메일, 게시판
- 1989년, 팀 버너스리 경(영국)이 웹 개념 제안
- 1990년, WWW(World Wide Web) 프로젝트 시작(HTTP모델, HTML언어 개발) by 팀 버너스리 경
- 1993년, 일리노이대학교 NCSA 연구소 소속 대학생 마크 앤드리슨, 에릭 비나가 모자이크(Mosaic) 개발
  - 화면에 문자, 그림 및 하이퍼링크 등 문서(Hypertext) 제공
- 1994년, 모자이크 개발팀이 NCSA를 나와서 회사를 만듬
  - 넷스케이프 내비게이터(Netscape Navigator) 웹 브라우저 출시
- 1995년, 마이크로소프트는 인터넷 익스플러러 출시
- 이후 크롬, 파이어폭스, 사파리, 오페라, 엣지 등 다양한 브라우저 출시

출처: 위키백과

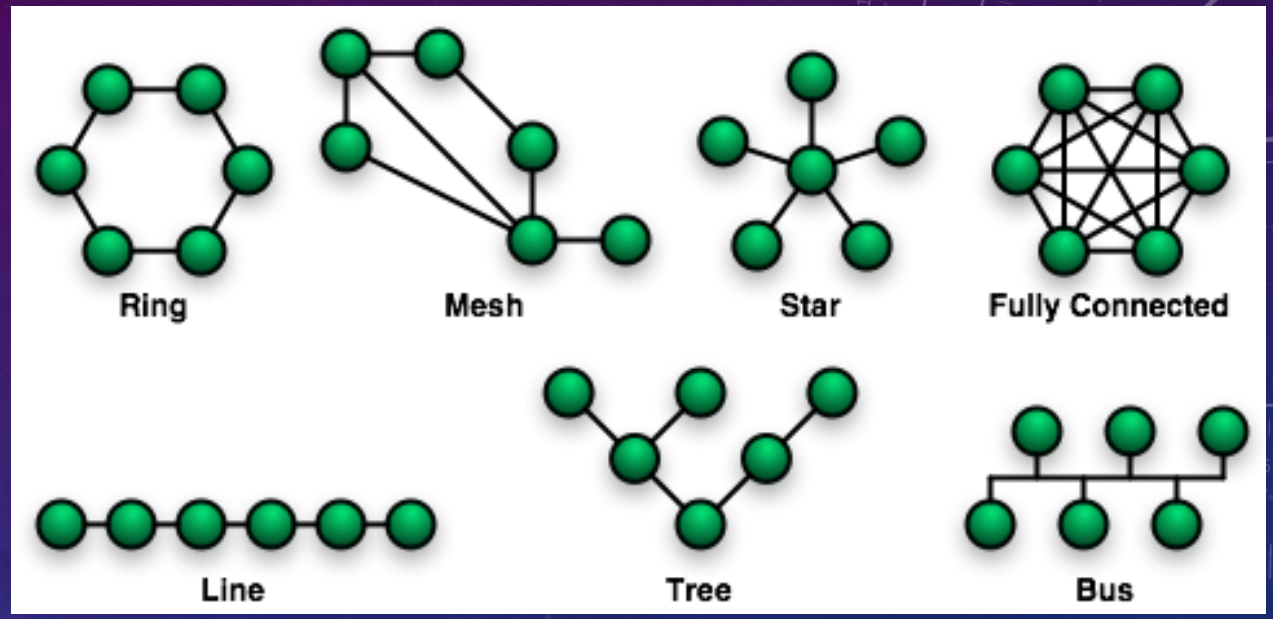
인터넷


Internet
Network Interface Device
WiFi Router
Laptop
Mobile Phone
Computer
Printer
Tablet / iPad


LAN
WAN
Metropolitan-area Network MAN
LAN
LAN
Basic Network LAN MAN WAN

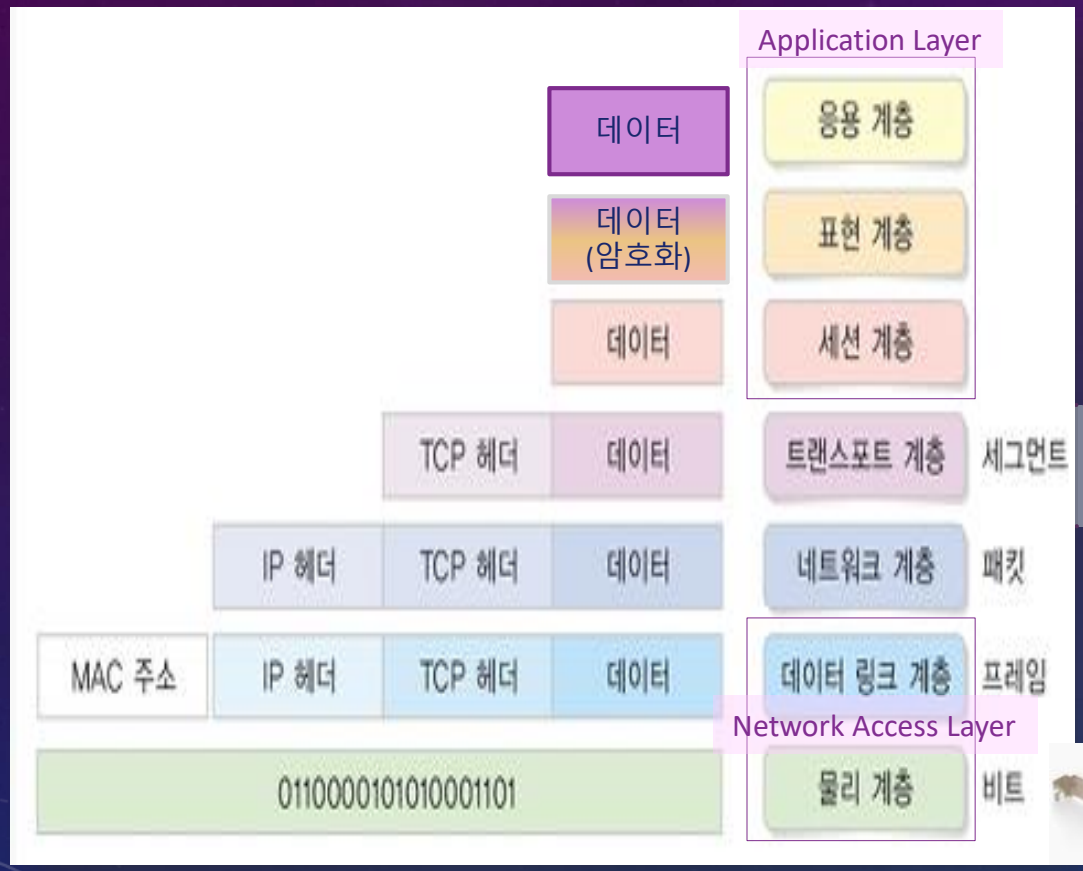
네트워크 토폴로지(NETWORK TOPOLOGY, 망구성방식)

- 버스 토폴로지
- 스타 토폴로지
  - 이더넷 LAN에서 가장 널리 사용
  - 연결지점에 허브, 스위치, 라우터 배치
- 링 토폴로지
  - 두 노드간에 오직 하나의 길 제공
  - FDDI(Fiber Distributed Data Interface 네트워크
- 트리 토폴로지
- 메시 토폴로지



출처: 위키피디아

OSI 7 계층(OPEN SYSTEMS INTERCONNECTION 7 LAYER, ISO표준 7498)



FTP(20,21),SSH(22), SMTP(25),DNS(53), HTTP(80), HTTPS(443),SNMP(161), SMB(445), MYSQL(3306), MSSQL(1433), RDP(3389),…

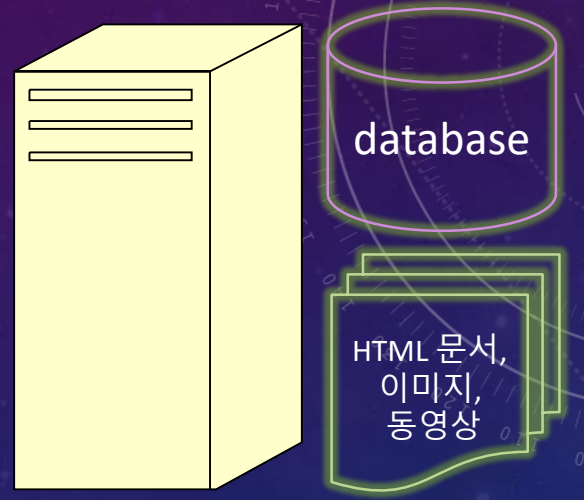
웹서비스 및 주요 용어

- 웹 브라우저를 통해 인터넷 활용 극대화
- 완성도 높은 웹사이트 제작 – 전세계에 홍보, 뛰어난 상품성, 대중의 인기
  - 웹이 폭발적으로 성장하면서 20세기초 3차 산업혁명이 시작됨
  - 쇼핑몰, SNS, 커뮤니티, 동영상 스트리밍 등 다양한 제품 및 서비스 출시
  - 전자상거래, 교육, 금융 서비스
  - 자동차, 가전제품, 센서 등이 연결되어 새로운 부가가치 창출
- 웹 관련 주요 용어
  - 하이퍼텍스트 언어: HTML(Hyper Text Markup Language)
  - 프로토콜: HTTP, HTTPS
  - DNS: IP를 이름으로 맵핑
  - URL(Uniform Resource Locator) : 프로토콜://도메인이름, 예로 https://www.naver.com:443

클라이언트-서버 구조

HTTP Request

HTML

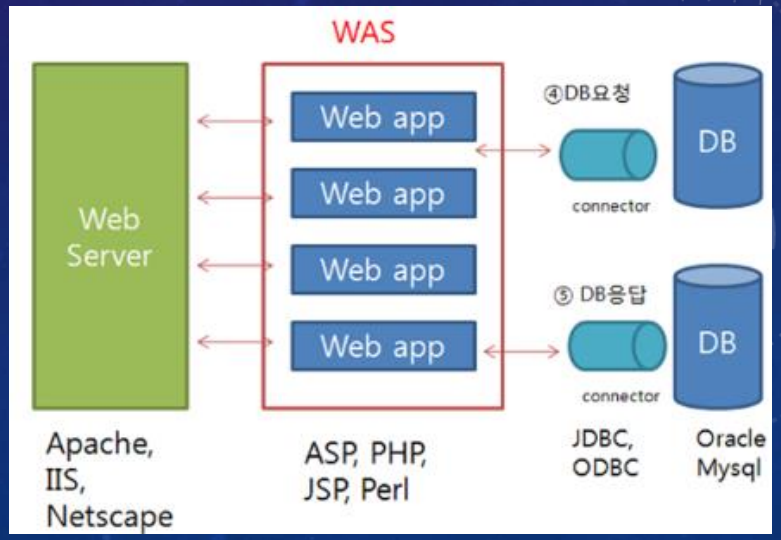


웹서버

https://gwangju.gen.hs.kr:453/sub/page.php

프로토콜 서버주소 포트번호 경로명 파일이름

HTML 페이지로 응답



HTML(HYPER TEXT MARKUP LANGUAGE)

```
<!doctype html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

HTML5
(하이퍼텍스트 마크업 언어)

파일 확장자	.html
인터넷 미디어 타입	text/html
타입 코드	TEXT
UTI	public.html[1]
개발	W3C
발표일	2014년 10월 28일 (9년 전)[2]
포맷 종류	마크업 언어
표준	HTML 5.2
오픈 포맷?	예

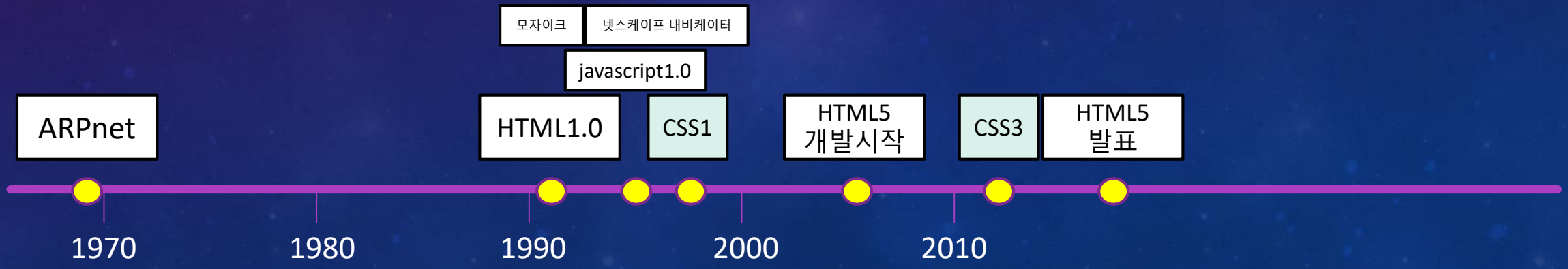
- 2007년 5월 HTML5와 Web Forums 2.0 스펙 채택 - 리뷰 기반으로
- 2007년 11월 HTML5 디자인 원칙 작업 초안
- 2010년 1월 HTML5 Last Call 작업 초안
- 2010년 8월 HTML5 Working Draft 공개

수정된 HTML 5.0 개발 일정

1. CR(Candidate Recommendation): 2012년 4분기
2. LCf: 2014년 3분기
3. PR(Proposed Recommendation): 2014년 4분기
4. Rec(Recommendation): 2014년 4분기

HTML5.1 개발 일정

1. FPWD(First Publication Working DrafT): 2012년 4분기
2. LC(Working Draft Last Call): 2014년 3분기
3. CR(Candidate Recommendation): 2015년 1분기
4. Rec(Recommendation): 2016년 4분기

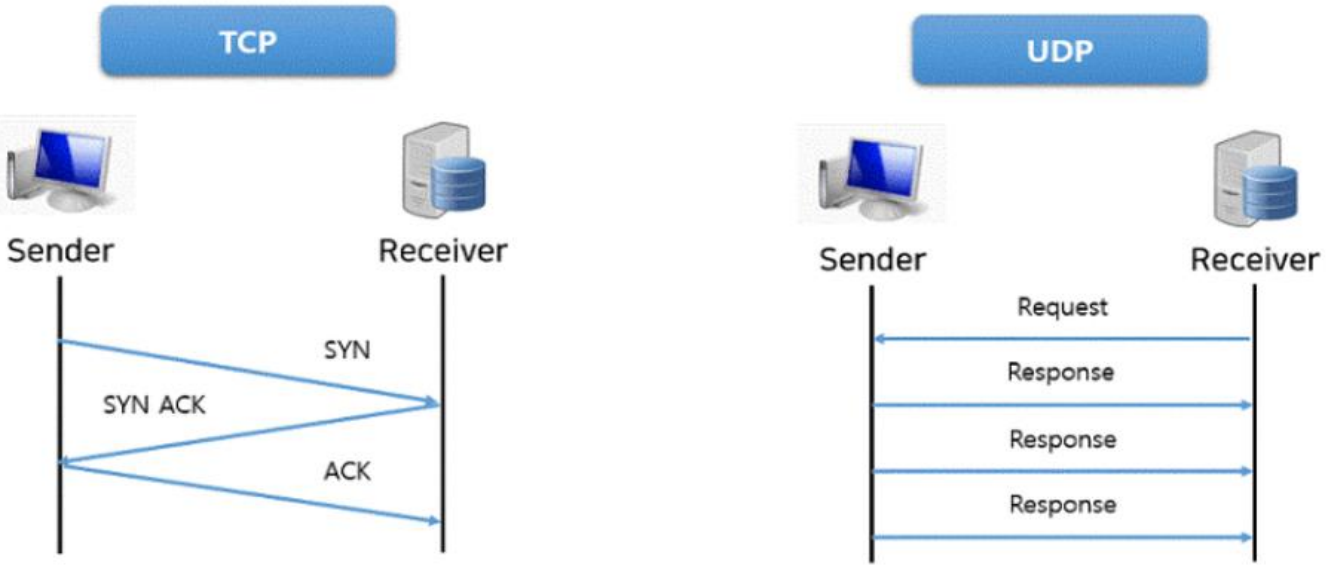


HTML5

- 웹페이지 구성요소
  - HTML : 웹페이지의 내용과 구조를 정의
  - CSS : 웹페이지의 디자인과 색상 정의
  - 자바스크립트 : 정적인 웹페이지를 동적으로 변환
- HTML5 특징
  - 멀티미디어 재생
  - 그래픽 지원
  - 양방향 통신 지원
  - 다양한 장치에 접근 가능

통신 프로토콜의 이해

통신방식 (T C P , U D P)



프로토콜 종류	TCP	UDP
연결 방식	연결형 서비스	비연결형 서비스
패킷 교환 방식	가상 회선 방식	데이터그램 방식
전송 순서	전송 순서 보장	전송 순서가 바뀔 수 있음
수신 여부 확인	수신 여부 확인	수신 여부 확인하지 않음
통신 방식	1:1 통신	1:1 or 1:N or N:N 통신
신뢰성	높다	낮다
속도	느리다	빠르다

1. 사용중인 컴퓨터의 IP 확인

```
!ipconfig

Windows IP 구성


이더넷 어댑터 이더넷 2:

   연결별 DNS 접미사. . . . :
   링크-로컬 IPv6 주소 . . . . : fe80::ae90:1ef8:65fe:2e07%19
   IPv4 주소 . . . . . . . . . . : 192.168.56.1
   서브넷 마스크 . . . . . . . . : 255.255.255.0
   기본 게이트웨이 . . . . . . . :
```

TCP 통신

1. TCP 통신
   - Socket 생성
   - 서버 바인딩
   - 서버 리스닝
2. 클라이언트 연결 수락
3. 데이터 교환
4. 연결 종료

```python
import socket
# 1. TCP 통신
host = "0.0.0.0"  # 서버의 IP 주소 또는 도메인 이름
port = 12345      # 포트 번호
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((host, port))
server_socket.listen(5)
print(f"서버가 {host}:{port}에서 대기 중입니다...")

while True:
    # 클라이언트 연결 대기
    client_socket, client_address = server_socket.accept()
    print(f"클라이언트 {client_address}가 연결되었습니다.")

    try:
        # 클라이언트로부터 요청 받기
        data = client_socket.recv(1024).decode("utf-8")
        if not data:
            continue

        # 요청 파싱
        parts = data.split("&&")
        if len(parts) != 0:
            name = parts[0]
            message = parts[1]
            response = f"어서와! {name}"

            # 클라이언트 이름과 메시지 출력
            print(f"클라이언트 이름: {name}")
            print(f"클라이언트 메시지: {message}")
        else:
            response = "유효하지 않은 요청"

        # 응답 클라이언트에게 전송
        client_socket.send(response.encode("utf-8"))

    except Exception as e:
        print(f"오류 발생: {e}")

    finally:
        # 클라이언트 소켓 닫기
        print("연결종료")
        client_socket.close()
```

```python
import socket
# TCP통신(클라이언트) 설정
server_address = "서버IP"  # 서버의 실제 IP 주소 또는 도메인 이름
server_port = 12345        # 서버 포트 번호

# 서버에 연결
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((server_address, server_port))

# 데이터 전송
name = "클라이언트"
message = "안녕, 서버!"

request = f"{name}&&{message}"
client_socket.send(request.encode("utf-8"))

# 서버로부터 응답 받기
response = client_socket.recv(1024).decode("utf-8")
print(f"{name} : {message}")
print(f"서버 : {response}\n")

# 클라이언트 소켓 닫기
client_socket.close()
```

① 서버가 0.0.0.0:12345에서 대기 중입니다...

②
```
request = f"{name}&&{message}"
client_socket.send(request.encode("utf-8"))
```

③ 클라이언트 ('168.131.49.99', 64244)가 연결되었습니다.
클라이언트 이름: 클라이언트
클라이언트 메시지: 안녕, 서버!
연결종료

④ 클라이언트 : 안녕, 서버!
서버 : 어서와! 클라이언트

UDP 통신

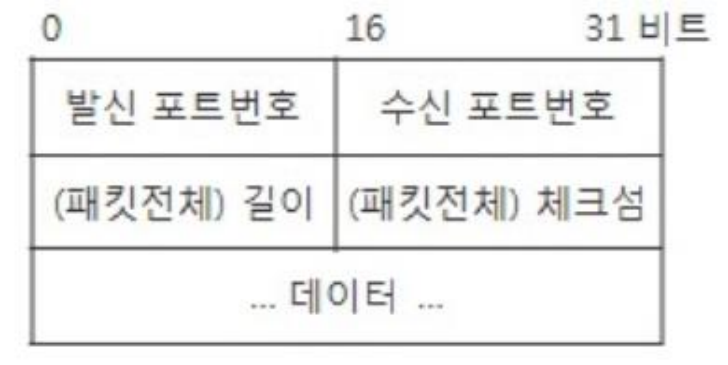
UDP통신

1. socket 이란?

> IP와 포트를 활용 : 특정 포트에 연결되어 데이터를 보내거나 받고자 할 때
> 서버 : 서비스 주체로 클라이언트로부터 메시지 받고 응답한다.
> 클라이언트 : 서비스 사용자

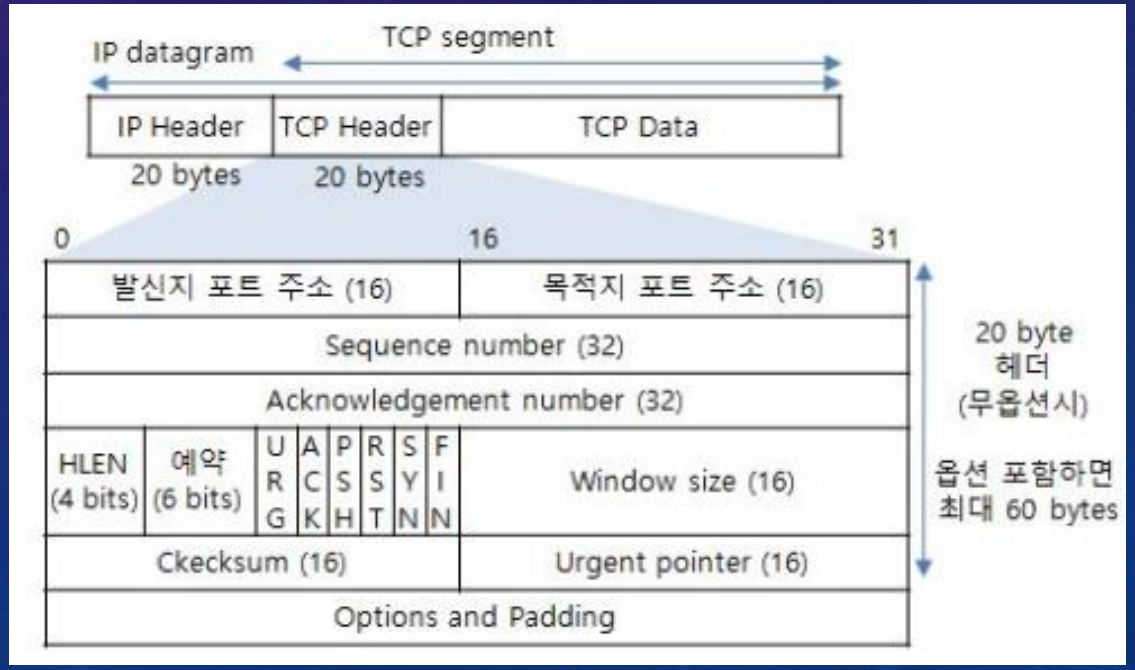
2. UDP서버와 클라이언트를 만들어 보자

UDP 패킷 헤더 구조



- 비 연결성
- 데이터가 제대로 도착했는지 확인 안함
- 흐름제어 없음
- 그러므로 송수신측에서 오류제어에 대한 기능 추가해야 함
- 빠른 요청과 응답
- 실시간 방송 등에 적합
- TFTP, SNMP, DHCP, DNS, RIP, RTP, RSTP 등

TCP 패킷 헤더 구조



UDP 통신

2. UDP 서버

1. 주소 체계(family)

   AF_INET : IPv4
   AF_INET6 : IPv6

2. 타입(type)

   SOCK_STREAM : TCP
   SOCK_DGRAM : UDP

```python
#통신 모듈
import socket
#주소 설정
local_addr = ('0.0.0.0',60080)
#udp서버 구성
server_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_sock.bind(local_addr) # socket 주소 정보 할당

print('데이터 수신 대기')
data, address = server_sock.recvfrom(1024)
print('받은메시지:'+data.decode(encoding='utf-8')+", IP:"+str(address));

#통신 종료
server_sock.close()
print('서버 프로그램 종료')
```

① 데이터 수신 대기

③ 받은메시지:반갑습니다. !!, IP:('168.131.49.99', 60963)
서버 프로그램 종료

2. UDP 클라이언트

1. 주소 체계(family)

   AF_INET : IPv4
   AF_INET6 : IPv6

2. 타입(type)

   SOCK_STREAM : TCP
   SOCK_DGRAM : UDP

```python
#통신 모듈
import socket
#주소 설정
server_ip = '192.168.0.24'
server_port = 60080

#udp클라이언트를 위한 소켓 구성
client_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

print('메시지 보내기')
msg = '반갑습니다. !!'
client_sock.sendto(msg.encode(),(server_ip,server_port))
print('보낸메시지:'+msg)

#통신 종료
client_sock.close()
print('클라이언트 프로그램 종료')
```

② 메시지 보내기
보낸메시지:반갑습니다. !!
클라이언트 프로그램 종료

UDP 통신(TELLO 드론)

```python
#통신 모듈
import socket
import time
import threading
global gmsg  # main, thread 전역 변수
global tellomsg # Tello 상태 정보
global run_flag # thread 종료 변수

#주소 설정
tello_addr = ('0.0.0.0',38889) # Tello드론 38889
notebook_addr = ('0.0.0.0',38890) # 노트북 38890

# 메시지 송수신
def send(message,delay):
    try:
        sock.sendto(message.encode(),notebook_addr) # Tello드론->노트북
        print("보낸메시지:" + message)
    except Exception as e:
        print("[tello send]에러 메시지:" + str(e))
    #지연
    time.sleep(delay)
# notebook -> tello드론 메시지 받기(recvfrom)
def receive():
    global gmsg
    global tellomsg
    global run_flag
    while run_flag:
        try:
            response, ip_addr = sock.recvfrom(1024) # 노트북 -> Tello드론
            gmsg = "[notebook]: %s, IP: %s" % (response.decode(encoding='UTF-8'), str(ip_addr))
            print(gmsg)
            # (Echo 서비스) tello정보 보내기
            send("[tello응답] %s ok" % response.decode(encoding='UTF-8'),1)
        except Exception as e:
            print("[tello receive] 에러 메시지:" + str(e))
            break
```

```python
# Tello드론(tello_addr) 통신포트 바인딩
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(tello_addr)

# 노트북으로부터 메시지 받기 쓰레드 생성 및 시작
run_flag = True
recvThread = threading.Thread(target=receive)
recvThread.daemon = True
recvThread.start()

# 통신 시작
while True:
    try:
        print('■ Tello드론서버는 대기모드')
        message = input()
        if 'quit' in message:
            print('프로그램 종료')
            run_flag = False
            recvThread.join()
            sock.close()
            break
        else:
            send(gmsg,3)
    except KeyboardInterrupt as e:
        sock.close()
        break
```

UDP 통신(NOTEBOOK)

```python
#통신 모듈
import socket
import time
import threading
global gmsg  # main, thread 전역 변수
global tellomsg # Tello 상태 정보
global run_flag # thread 종료 변수
#주소 설정
tello_addr = ('0.0.0.0',38889) # Tello드론 38889
notebook_addr = ('0.0.0.0',38890) # 노트북 38890

# 메시지 송수신
def send(message,delay):
    try:
        sock.sendto(message.encode(),tello_addr)
        print("[notebook]메시지:" + message)
    except Exception as e:
        print("local 에러 메시지:" + str(e))
    #지연
    time.sleep(delay)
# tello -> notebook 메시지 받기(recvfrom)
def receive():
    global gmsg
    global tellomsg
    global run_flag
    while run_flag:
        try:
            response, ip_addr = sock.recvfrom(1024)
            gmsg = "[tello]: %s, IP: %s" % (response.decode(encoding='UTF-8'), str(ip_addr))
            print(gmsg)
        except Exception as e:
            print("local receive 에러 메시지:" + str(e))
            break
```

```python
# 로컬(notebook) 통신포트 바인딩
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(notebook_addr)

# 노트북으로부터 메시지 받기 쓰레드 생성 및 시작
run_flag = True
recvThread = threading.Thread(target=receive)
recvThread.daemon = True
recvThread.start()

# 통신 시작
while True:
    try:
        print('■ 명령어입력: command->takeoff->land')
        message = input()
        if 'quit' in message:
            print('프로그램 종료')
            run_flag = False
            recvThread.join()
            sock.close()
            break
        else:
            send(message,1)
    except KeyboardInterrupt as e:
        run_flag = False
        recvThread.join()
        sock.close()
        break
```

JSON

- JSON, JavaScript Object Notation
  - 데이터를 저장하거나 교환할 때 사용하는 경량의 포맷
  - 포멧 형식의 예로, { " 이름":"홍길동", "직업":"의적"}
  - 웹API, 설정 파일, 데이터 교환 등 다양한 분야에 널리 사용

JSON

JSON서버

```python
import socket
import json

def start_server(host='0.0.0.0', port=12345):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((host, port))
        s.listen()
        print(f"{host}:{port} 대기상태...")

        while True:  # 무한대기
            conn, addr = s.accept()
            with conn:
                print(f"Connected by {addr}")
                while True:
                    data = conn.recv(1024)
                    if not data:
                        break  # 데이터가 없으면 클라이언트 연결 종료

                    # JSON 데이터 수신
                    received_json = json.loads(data.decode("utf-8"))
                    print(f"Received data: {received_json}")

                    # JSON 형태로 응답메시지 전송
                    response = {"status": "success", "message": "Data received"}
                    conn.sendall(json.dumps(response).encode("utf-8"))

            # 연결된 클라이언트 연결 종료
            print(f"Connection with {addr} ended")

if __name__ == "__main__":
    start_server()
```

```
0.0.0.0:12345 대기상태...
```

JSON클라이언트

```python
import socket
import json

host = '서버IP 또는 DNS명' #str
port = 12345   #int

def start_client(server_host=host, server_port=port):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((server_host, server_port))

        # JSON 데이터 전송
        data_to_send = {"name": "홍길동", "message": "Hello, JSON"}
        s.sendall(json.dumps(data_to_send).encode())

        # 서버로부터 응답 수신
        data = s.recv(1024)
        print(f"Received response: {data.decode()}")

if __name__ == "__main__":
    start_client(host,port)
```

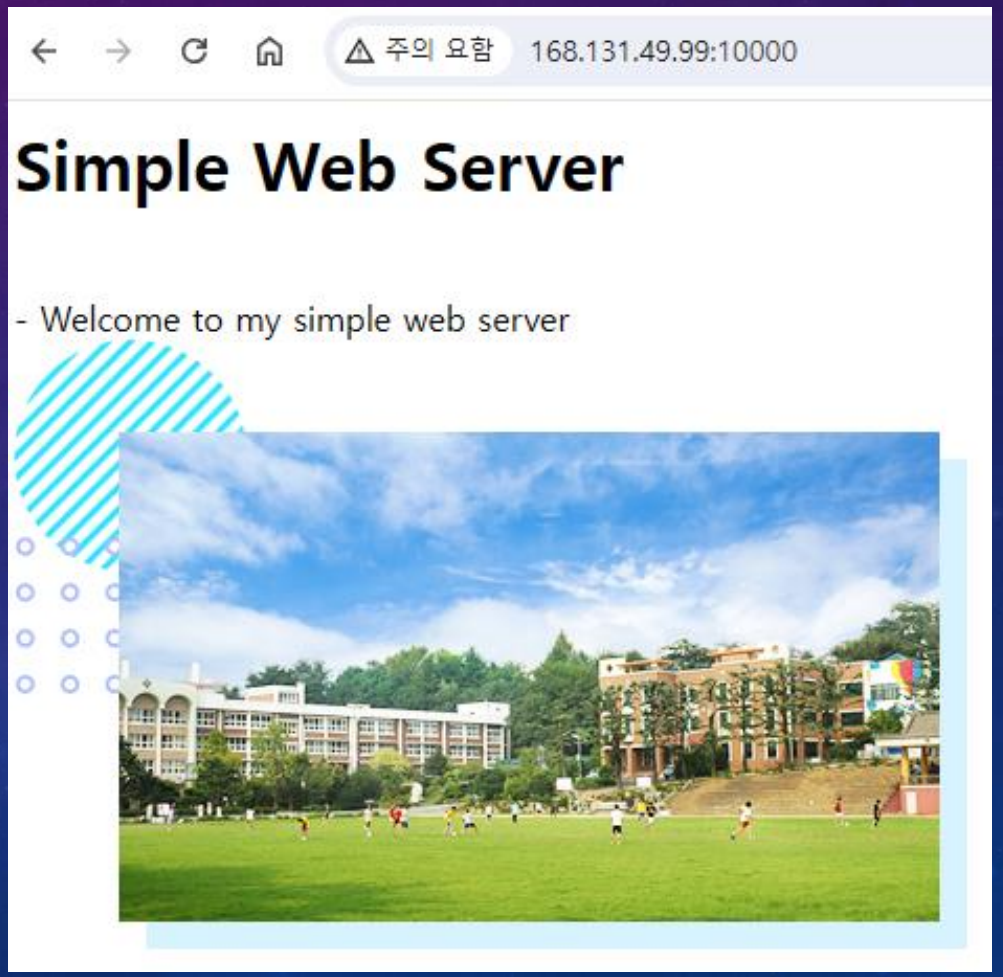
```
Received response: {"status": "success", "message": "Data received"}
```

```
Connected by ('168.131.49.99', 65185)
Received data: {'name': '홍길동', 'message': 'Hello, JSON'}
Connection with ('168.131.49.99', 65185) ended
Connected by ('168.131.49.99', 65218)
Received data: {'name': '아무개', 'message': '두번째 메시지'}
Connection with ('168.131.49.99', 65218) ended
```

HTTP서버 :: SIMPLE 서버

```python
#HTTP서버
import socket

class simple_server:
    def __init__(self):
        self.bufsize = 1024
        self.counter = 0
    def run(self,ip,port):
        #소켓 생성
        self.sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        #소켓에 ip와 포트 정보를 bind
        self.sock.bind((ip,port))
        #클라이언트(5번 등단) 대기상태 설정
        self.sock.listen(5)

        while self.counter < 5:
            #client의 request에 대해 client의 clnt_sock를 생성
            clnt_sock,req_addr = self.sock.accept()

            #clnt_sock에 적힌 정보를 read
            req_message = clnt_sock.recv(self.bufsize)
            print("client request msg: ", req_message)

            res_message = '''HTTP/1.1 200 OK
Server:simple web server
Content-length:2048
Content-type:text/html

<html lang=ko><head><title>simple web server</title></head>
<body><h1>Simple Web Server</h1><br>- Welcome to my simple web server<br></body>
</html>
'''.encode("utf-8")

            #clnt_sock를 통해서 client에 정보를 전송
            clnt_sock.send(res_message)

            #clnt_sock를 close
            clnt_sock.close()
            self.counter += 1

        #sock를 close
        print('close sock')
        self.sock.close()


server = simple_server()
server.run("0.0.0.0",10000)
```

```
netstat -anop tcp | findstr 10000
TCP    0.0.0.0:10000          0.0.0.0:0              LISTENING       2108
taskkill /f /pid 2108
```



감사합니다.