

Fastcampus

Web Programming & Frontend Dev SCHOOL

Intro to Network Computing

Network

Network

우리는 어떻게 다른 컴퓨터와 통신하고, 웹서핑을 할 수 있을까?

Network

A computer network or data network is a telecommunications network which allows nodes to share resources.

--> 컴퓨터간 리소스를 공유 가능하게 만드는 통신망

Charcteristics of Network

- 컴퓨터사이의 리소스를 공유
- 네트워크로 연결된 다른 컴퓨터에 접근하여 파일을 생성, 수정, 삭제할 수 있음
- 프린터와 스캐너, 팩스 등의 출력장치에 네트워크를 연결하여 여러 컴퓨터가 동시에 접근 가능

Requirements of Network

- Network Cable
- Distributor(Switch Hub)
- Router
- Network card

커버 범위에 따른 네트워크 구분

LAN

- Local Area Network(근거리 통신망)
- 학교, 회사 등 가까운 지역의 좁은 범위

WAN

- Wide Area Network(광역 통신망)
- 국가, 대륙 등 넓은 지역을 커버

MAN

- Metropolitan Area Network(도시권 통신망)
- LAN < MAN < WAN

WLAN

- Wireless Local Area Network(무선 근거리 통신망)
- IEEE 802.11 표준을 기반
- <http://standards.ieee.org/about/get/802/802.11.html>

802.11 == wifi ????

802.11 != wifi

802.11: IEEE에서 개발된 표준 무선통신기술

wifi: 와이파이 얼라이언스의 상표. 802.11 기술을 사용하는 무선 근거리 통신망 제품

wifi



wifi a, b, g, n, ac

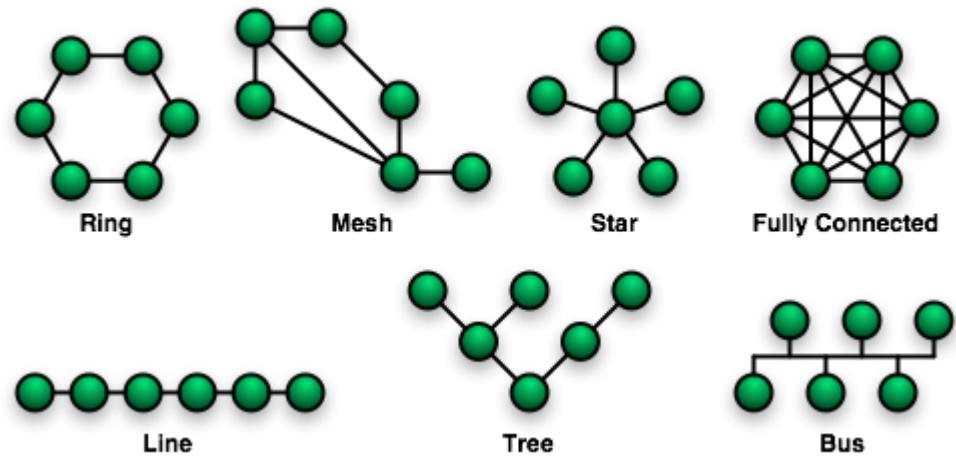


Another way of Networking

- Lifi(IEEE 802.15.7r1)
- Power line Networking(IEEE 1901)

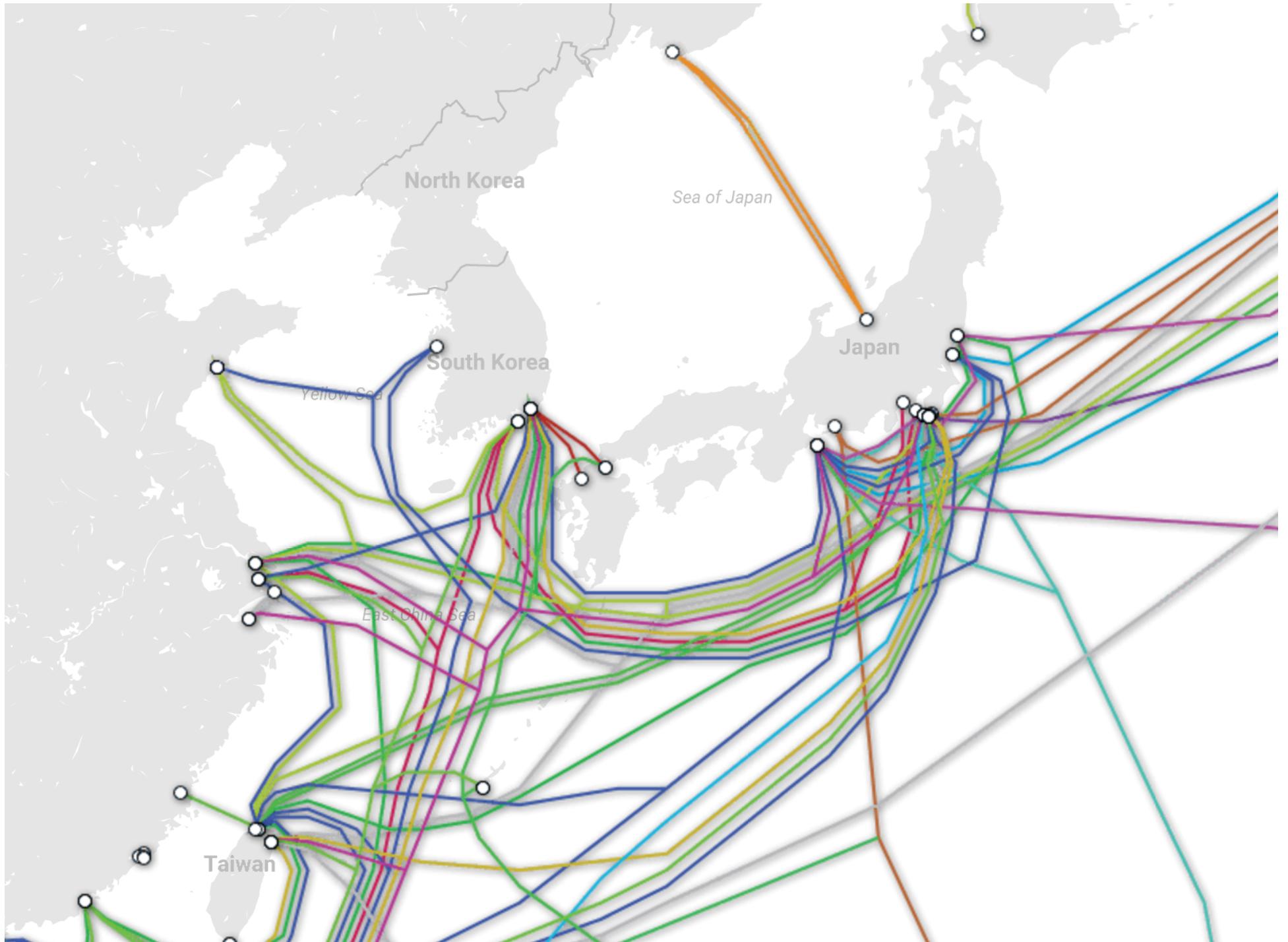
Star, Ring, Bus

Network Topology



Submarine Cable Map

<https://www.submarinecablemap.com/>



Ethernet

- 전세계의 사무실이나 가정에서 일반적으로 사용되는 유선 LAN에서 가장 많이 활용되는 기술 규격
- ether == 에테르 == 빛의 매질
- IEEE 802.3 규약 기반
- OSI 7 Layer에서 Data-link Layer에 위치
- <http://standards.ieee.org/about/get/802/802.3.html>

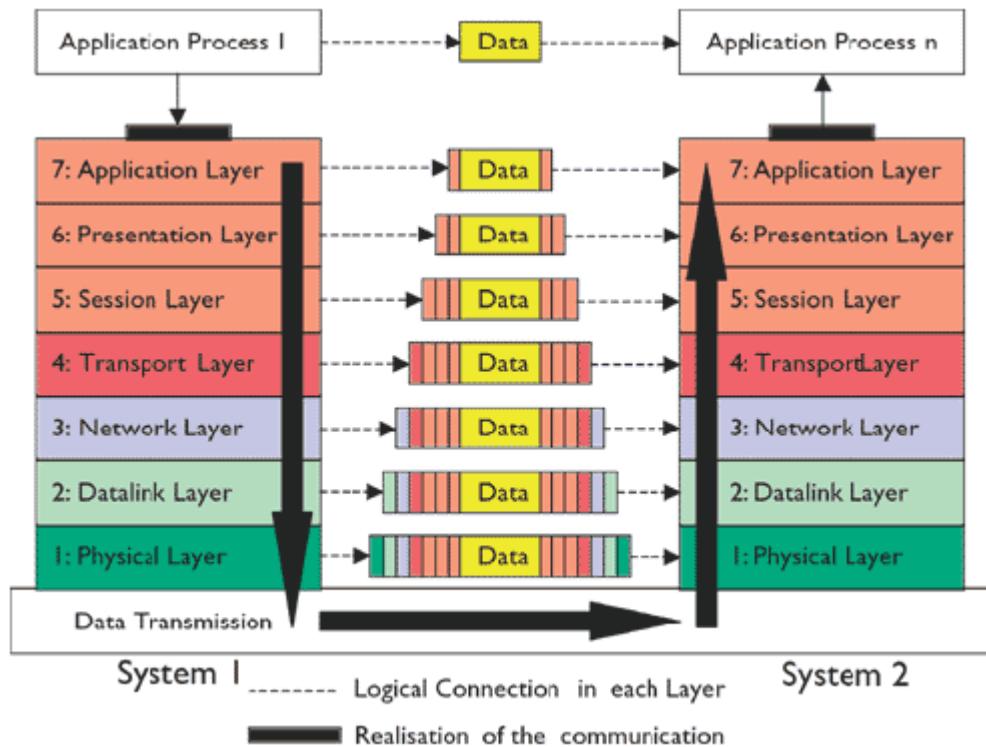
Network OSI 7 layer

- Open Systems Interconnection Reference Model
- 국제 표준화기구에서 개발한 컴퓨터 네트워크 프로토콜 디자인과 통신을 계층으로 나누어 설명한 것

Packet

- 데이터를 한번에 전송할 단위로 자른 데이터의 묶음 혹은 그 크기
- 1492 ~ 1500 bytes(프로토콜에 따라 다름)
- 네트워크에서는 바이트(byte)라는 표현 대신 옥텟(octet)으로 표현

Network OSI 7 layer



Application Layer

- 사용자에게 네트워크 자원에 대한 접근을 제공
- 네트워크 활동들에 대한 모든 기본적인 인터페이스를 제공
- 사용자에게 보이는 유일한 계층

Presentation Layer

- 응용 계층으로 부터 전송 받거나 전달되는 데이터의 인코딩과 디코딩
- 안전하게 데이터를 사용하기 위해 몇 가지 암호화와 복호화 형식 보유

Session Layer

- 두 대의 컴퓨터 사이의 세션이나 대화(Dialogue)를 관리
- 모든 통신 장비를 연결하고 관리하며 종료
- 순간적으로 연결이 끊어지는 것을 막고 호스트 사이의 연결을 적절하게 종료시키기 위한 기능과 연결이 단방향인지 양방향인지에 대한 것을 담당

Transport Layer

- 아래 계층에 신뢰성 있는 데이터를 전송할 수 있게 함
- 흐름 제어, 분할, 재조립, 오류 관리를 포함하지만 전송 계층은 지점과 지점 간의 오류가 없음을 보장
- 연결 지향적인 프로토콜과 비연결 지향적인 프로토콜을 제공하며, 방화벽과 프록시 서버가 이 계층에서 동작

Network Layer

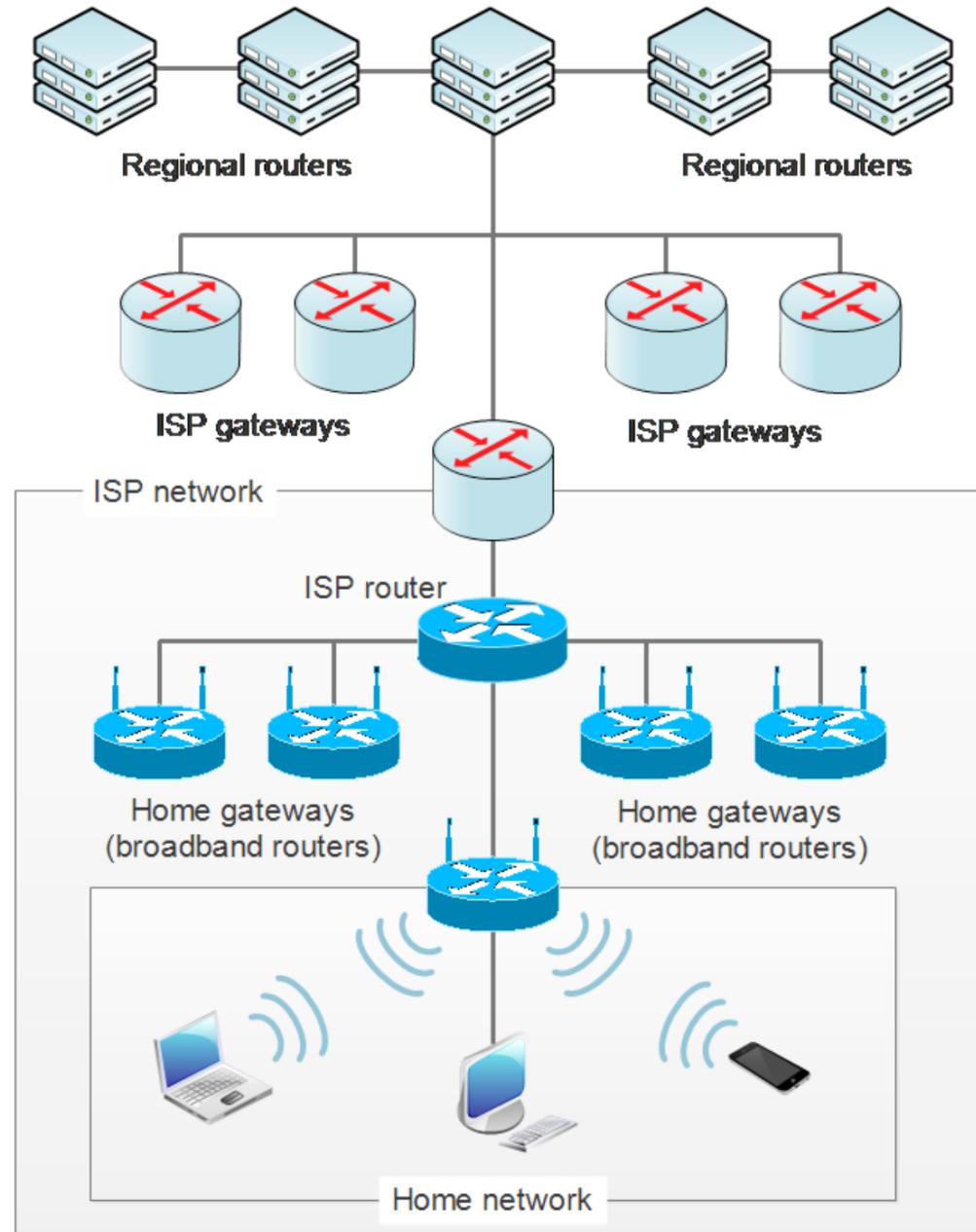
- 가장 복잡한 OSI 계층 중 하나로, 물리적인 네트워크 사이의 라우팅을 담당 하며, 라우터가 이 계층에서 동작
- 네트워크 호스트의 논리적인 주소(IP 주소같은)를 관리하고 패킷을 분할해 프로토콜을 식별하는 기능, 오류 탐지 같은 몇 가지 경우를 담당

Datalink Layer

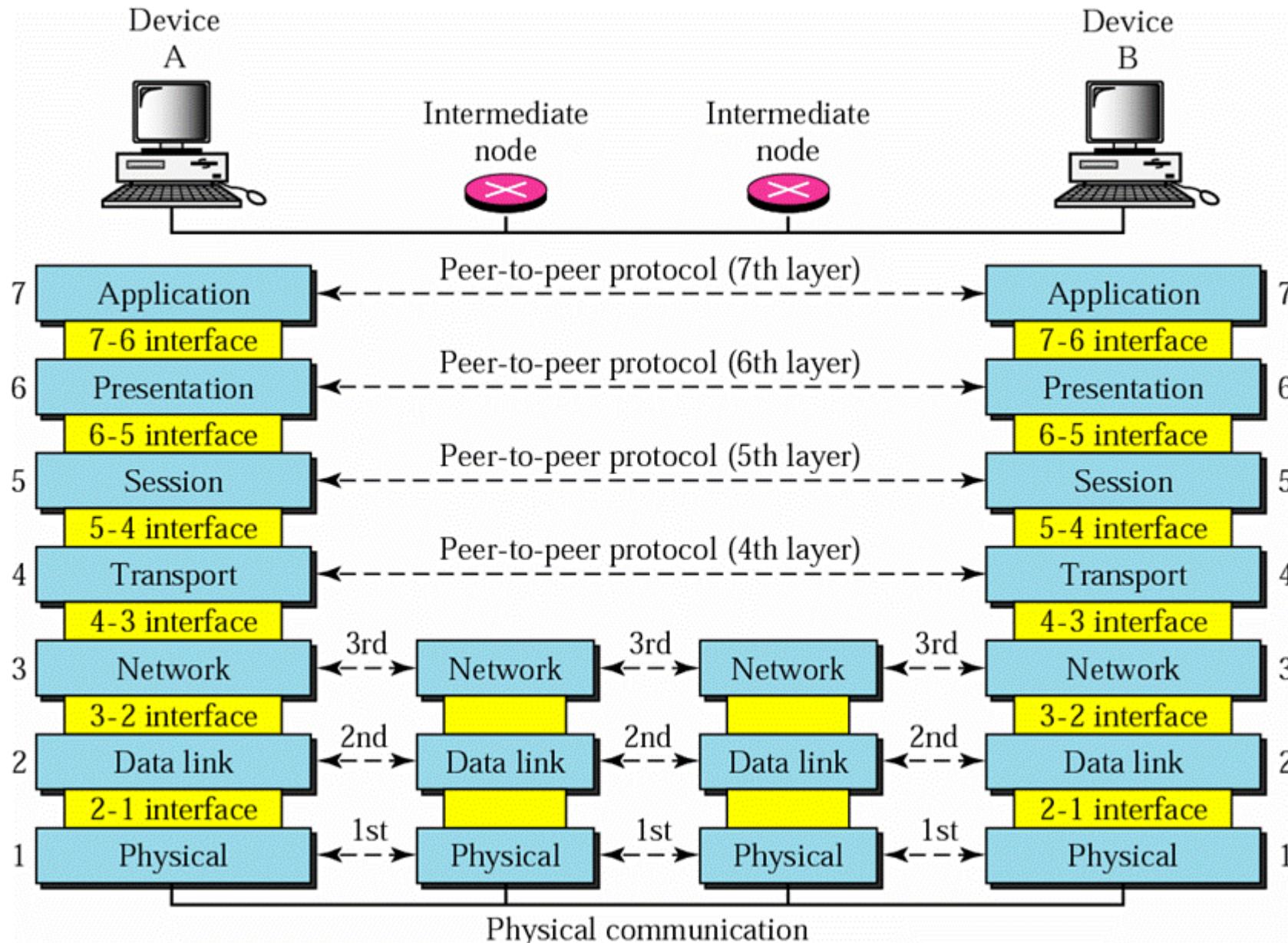
- 물리적인 네트워크 사이의 데이터 전송을 담당
- 물리적인 장비를 식별하는 데 사용되는 주소 지정 체계(Addressing Schema)와 데이터가 변조되지 않았음을 확증하기 위한 오류 확인을 제공
- 브리지와 스위치가 이 계층에서 동작하는 물리적인 장비

Physical Layer

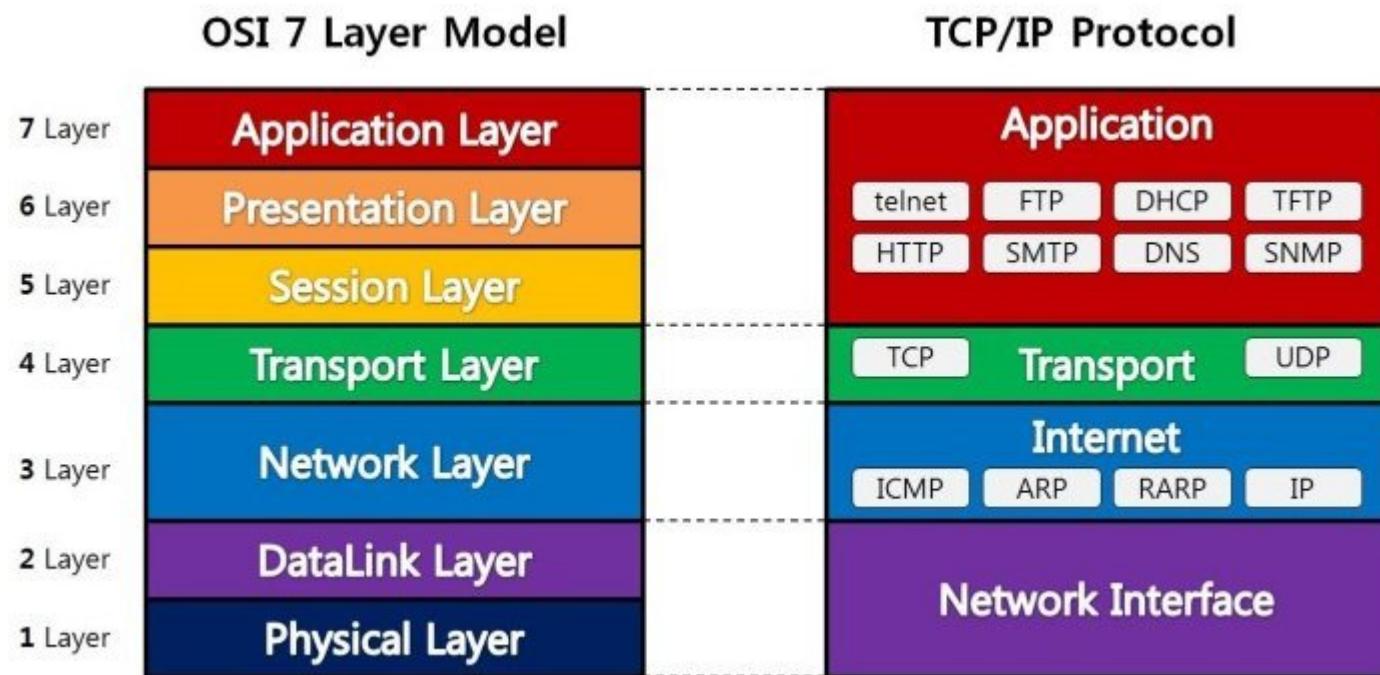
- 네트워크 데이터가 전송될 때 사용되는 물리적 매개체
- 전압, 허브, 네트워크 어댑터, 리피터, 케이블 명세서를 포함해 모든 하드웨어의 물리적이고 전자적인 특성을 정의
- 연결을 설정하고 종료하며, 공유된 통신 자원을 제공하고, 아날로그를 디지털로, 디지털을 아날로그로 변환



Network OSI 7 layer



Network OSI 7 layer



HTTP

HyperText Transfer Protocol

- www상에서 정보를 주고받는 프로토콜
- TCP, UDP를 활용함
- HTTP method: GET, POST, PUT, DELETE

```
> from urllib.request import urlopen  
> response = urlopen('https://www.google.com/')  
> response  
> response.readline()  
> response.url  
> response.status
```

```
> from urllib.request import urlopen
> import urllib.error
> try:
>     urlopen('https://www.onoffmix.com/')
> except urllib.error.HTTPError as e:
>     print('status', e.code)
>     print('reason', e.reason)
>     print('url', e.url)

> response.getheaders()
```

FTP

File Transfer Protocol

- 서버와 클라이언트 사이에 파일전송을 위한 프로토콜
- but, 보안에 매우 취약(패킷 가로채기, 무차별 대입, ...)
- 현재는 FTPS(FTP-SSL), SFTP(simple FTP), SSH(Secure SHell) 등을 사용

SMTP

Simple Mail Transfer Protocol

- Internet에서 메일을 보내기 위한 프로토콜

TCP/IP

Transmission Control Protocol / Internet Protocol

- 전송제어 프로토콜 + 송수신 호스트의 패킷교환을 위한 프로토콜

TCP

- 전송제어프로토콜 / Transmission(Transfer) Control Protocol
- 근거리 통신망이나 인트라넷, 인터넷에 연결된 컴퓨터에서 실행되는 프로그램 간에 일련의 옥텟(==byte)을 안정적으로, 순서대로, 에러없이 교환할 수 있게 함

STREAM

- 문자형식의 데이터가 열의 형태로 연속성을 띄게 됨

DATAGRAM

- 하나의 패킷이 발신지와 수신지 정보를 모두 담고 있는 독립적인 패킷

STREAM socket

- 연결형 스트림 소켓은 두개의 시스템이 연결된 후 데이터를 교환
- 패킷 순서 신경쓰지 않아도 되어 안정적인 데이터 전송 가능

DATAGRAM socket

- 비연결형 데이터그램 소켓은 명시적으로 연결되지 않은 상태로 데이터를 주고 받음
- 연결과 해제 과정이 없어 빠른 데이터 교환이 가능

IP

IPv4, IPv6

- Internet Protocol version 4
 - 32bit로 구성
 - 0.0.0.0 ~ 255.255.255.255
 - 0000 0000.0000 0000. 0000 0000. 0000 0000
 - $2^{32} = 42.9\text{억}$
 - 5개의 클래스를 가지며, 상위 3개의 클래스를 가짐
 - A(1~126)
 - B(128~191.XXX)
 - C(192~223.XXX.XXX)
 - D()
 - E()

| IP | 용도 |
|----------------|-----------------------|
| 0.0.0.0/8 | 자체 네트워크 |
| 10.0.0.0/8 | 사설 네트워크 |
| 127.0.0.0/8 | 루프백(loopback) 즉, 자기자신 |
| 169.254.0.0/16 | 링크 로컬(link local) |
| 172.16.0.0/12 | 사설 네트워크 |
| 192.0.2.0/24 | 예제 등 문서에서 사용 |
| 192.88.99.0/24 | 6to4 릴레이 애니캐스트 |
| 192.168.0.0/16 | 사설 네트워크 |
| 198.18.0.0/15 | 네트워크 장비 벤치마킹 테스트 |
| 224.0.0.0/4 | 멀티캐스트 |
| 240.0.0.0/4 | 미래 사용 용도로 예약 |

127.0.0.1 vs 192.168.0.x

127.0.0.1

- Loopback: 컴퓨터가 가지고 있는 무조건 반대신호를 반환하는 대역
- Localhost

192.168.0.x

- LAN에서 라우터가 할당한 내컴퓨터의 IP address

Global IPv4 depletion



Let's Count Number

$10^4 =$ 만

$10^8 =$ 억

$10^{12} =$ 조

$10^{16} =$ 경

$10^{20} =$ 해

$10^{24} =$ 자

$10^{28} =$ 양

$10^{32} =$ 구

$10^{36} =$ 간

IPv4, IPv6

- Internet Protocol version 6
 - 128bit로 구성
 - 0000:0000:0000:0000:0000:0000:0000:0000 ~
FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF
 - $2^{128} = 16*16*16*16*16^8 = 340,282,366,920,938,463,463,374,607,431,768,211,456 = 3.4*10^{38}$

Public, Private

Public IP address(Internet IP address)

- Globally Unique

Private IP address(Local IP address)

- Private network 내에서 유효

DNS

- Domain Name System
- 외우기 힘들며, 더 힘들어질 ip address를 사람이 판별하기 쉬운 url을 매팅하는 시스템

ipconfig / ifconfig

현재 컴퓨터와 연결된 네트워크 정보를 확인할 수 있음

MAC

- Media Access Control address
- 12개의 16진수로 구성

ARP

- Address Resolution Protocol
- IP address -> MAC address

Subnetmask

- 커다란 네트워크를 효율적으로 분배하여 사용하기 위한 방법
- 할당받은 하나의 IP주소를 네트워크 환경에 맞춰 적절히 나누어줌
- IPv4 기준 2진수로 구성
- 255.255.255.255
- 1111 1111 . 1111 1111 . 1111 1111 . 1111 1111

Default Subnetmask

- 네트워크를 서브넷으로 나누지 않아도 기본적으로 할당
- Class C
- 255.255.255.0
- 1111 1111.1111 1111.1111 1111.0000 0000
- Class B
- 255.255.0.0
- 1111 1111.1111 1111.0000 0000.0000 0000
- Class A
- 255.0.0.0
- 1111 1111.0000 0000.0000 0000.0000 0000

\$ ping

```
$ ping www.google.com  
$ ping www.naver.com  
$ ping www.fastcampus.co.kr
```

UDP

User(Universal) Datagram Protocol

- 데이터그램을 전송하기 위한 프로토콜
- 메시지 수신확인x, 도착순서 예측x
- 빠른 속도, 적은 오버헤드

TCP vs UDP segment

TCP Segment Header Format

| Bit # | 0 | 7 | 8 | 15 | 16 | 23 | 24 | 31 |
|--------|-------------|-----|--------------------------|-----------------------|----|------------------|----|----|
| 0 | Source Port | | | | | Destination Port | | |
| 32 | | | | Sequence Number | | | | |
| 64 | | | | Acknowledgment Number | | | | |
| 96 | Data Offset | Res | | Flags | | Window Size | | |
| 128 | | | Header and Data Checksum | | | Urgent Pointer | | |
| 160... | | | | Options | | | | |

UDP Datagram Header Format

| Bit # | 0 | 7 | 8 | 15 | 16 | 23 | 24 | 31 |
|-------|-------------|---|---|----|--------------------------|------------------|----|----|
| 0 | Source Port | | | | | Destination Port | | |
| 32 | Length | | | | Header and Data Checksum | | | |

Packet Sniffer

- 프로토콜 간 오고가는 메시지를 관찰하는 도구

Wireshark

[download](#)

ip.addr != [xxx.xxx.xxx.xxx](#) && tcp

intranet vs Internet vs internet

- intranet: internet의 www기술을 활용하여 특정 단체의 내부 정보시스템을 구축하는 것 혹은 그 네트워크
- Internet(International Network): TCP/IP를 활용하여 정보를 주고 받는 통신 네트워크(www)
- internet(internetwork): 패킷을 교환하는 방식으로 기기간의 정보를 주고 받는 방식

BUT!! AP가 Internet을 internet으로 표현하는 style guide를 발표!!

- e-mail --> email
- Web site --> website

Civilization



data

- 컴퓨터가 처리할 수 있는 문자, 숫자, 소리, 그림 따위의 형태로 된 정보.
- Latin "Datum"의 복수형 "Data"에서 유래

Database

- 체계화된 데이터의 모임
- 여러 응용 시스템들의 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 묶음

DB?? DBMS??

DBMS(DataBase Management System)

- 데이터의 모임인 Database를 만들고, 저장, 관리 할 수 있는 기능을 제공하는 응용프로그램
- Oracle, Mysql, MariaDB, DB2, MS SQL Server, ..

DBMS의 조상님



DBMS의 조상님

dBASE

- 마이크로컴퓨터용 최초의 DBMS
- 1979년 Ashton이 개발
- SQL이 아닌 독자 스크립트 언어로 실행 -> dbf 파일 생성

Differences between DataBase & File System

자기기술성

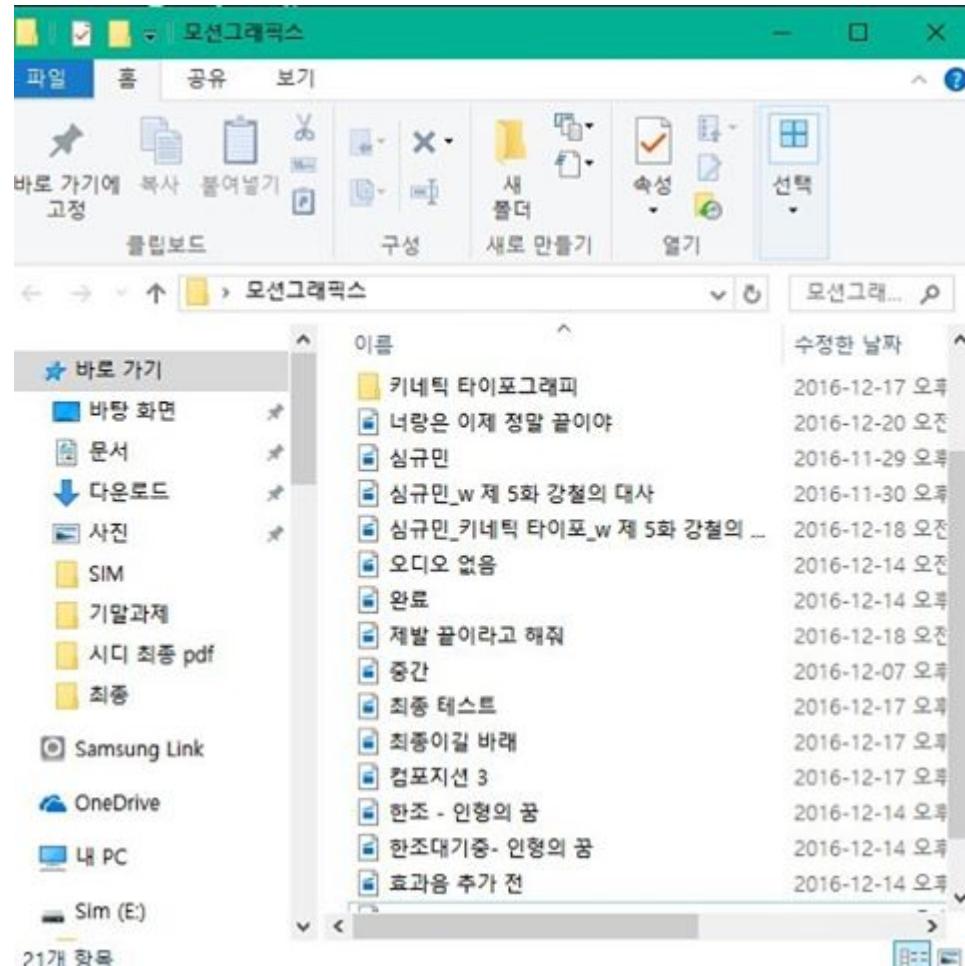
File System

- .hwp -> 한글
- .doc -> Microsoft Word
- .xls -> Microsoft Excel

DB

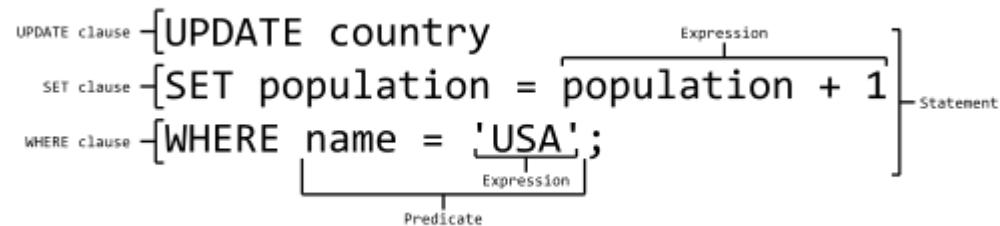
- Only SQL(RDBMS)





SQL(Structured Query Language)

데이터 관리를 위해 설계된 특수 목적의 프로그래밍 언어



RDBMS vs NoSQL

| 구분 | RDBMS | NoSQL |
|-----|-------------------------------|---------------------------------------|
| 형태 | Table | Key-value, Document, Column |
| 데이터 | 정형 데이터 | 비정형 데이터 |
| 성능 | 대용량 처리시 저하 | 잦은 수정시 저하 |
| 스키마 | 고정 | Schemaless |
| 장점 | 안정적 | 확장성, 높은 성능 |
| 유명 | Mysql, MariaDB, PostgreSQL | MongoDB, CouchDB, Redis, Cassandra |

RDBMS

[PostgreSQL Docs](#)

[MariaDB Docs](#)

| name | age |
|------|-----|
| John | 17 |
| Mary | 21 |

```
rdb =  
{  
    name: [John, Mary],  
    age: [17, 21]  
}
```

Table == Relation

| Primary Key | Attribute1 | Attr2 | Attr3 | Attr4 |
|-------------|------------|-------|-------|-------|
| Tuple1 | | | | |
| Tuple2 | | | | |
| Tuple3 | | | | |
| Tuple4 | | | | |

NoSQL

MongoDB Docs

```
nosql =  
[  
    {  
        name:John,  
        age:17  
    },  
    {  
        name:Mary,  
        age:21  
    },  
    ...  
]
```

Document vs Key-value

```
document
{
    key: value,
    key: {
        key: value,
        key: value
    }
}
```

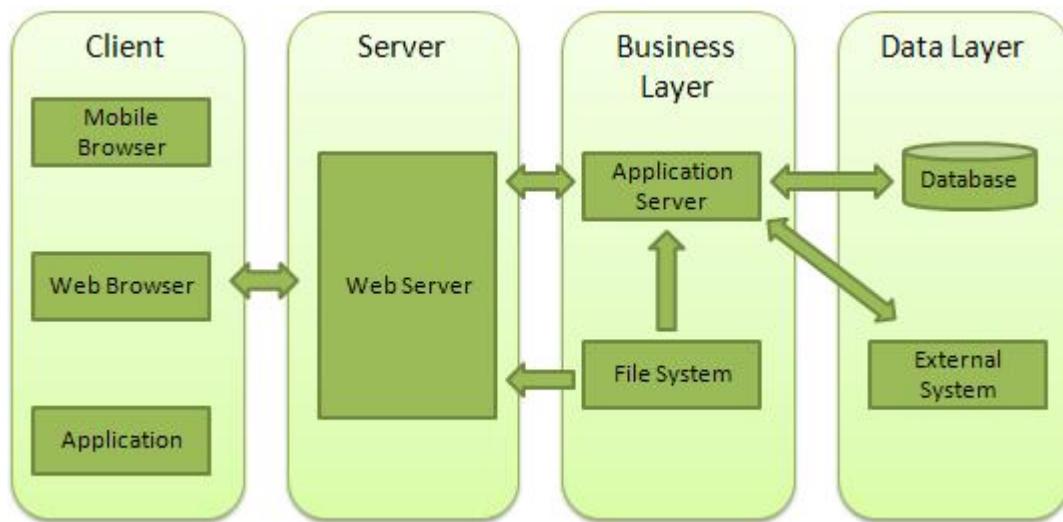
```
key-value
{
    key: value,
    key: value,
    key: value
}
```

Civilization



Web Programming

Web architecture

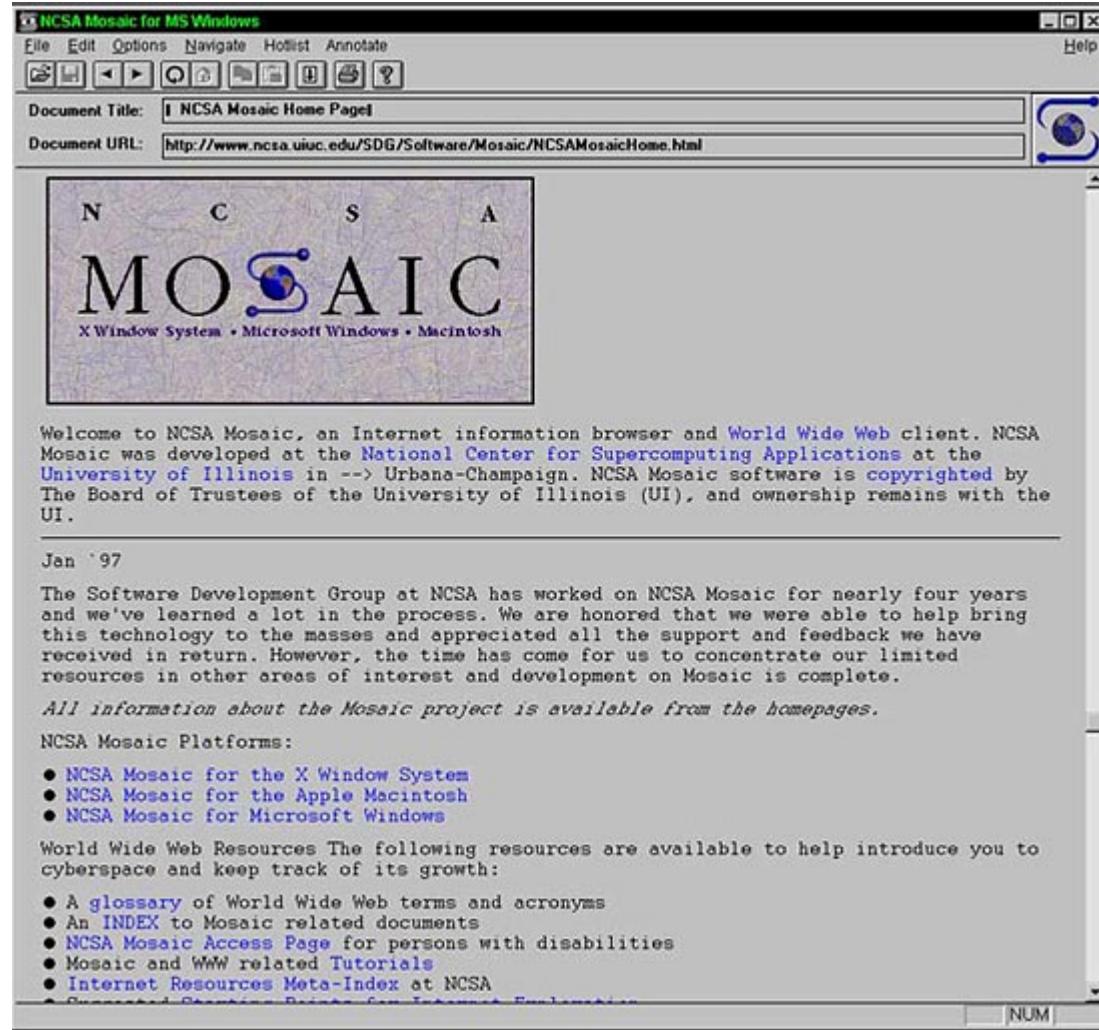


웹 개발 패턴의 변화

- 1991 ~ 1999: Sir Timothy John "Tim" Berners-Lee가 하이퍼텍스트 기반의 프로젝트를 제안한 이후 정적인 컨텐츠들을 중심으로 한 웹 기술이 발달
- 1999 ~ 2009: Linux, Apache, Mysql, Php 중심의 동적인 서버, 정적인 클라이언트 모델이 지속됨
- 2010 ~ 현재: javaScript!! (Dynamic Web Client)

Web Browser

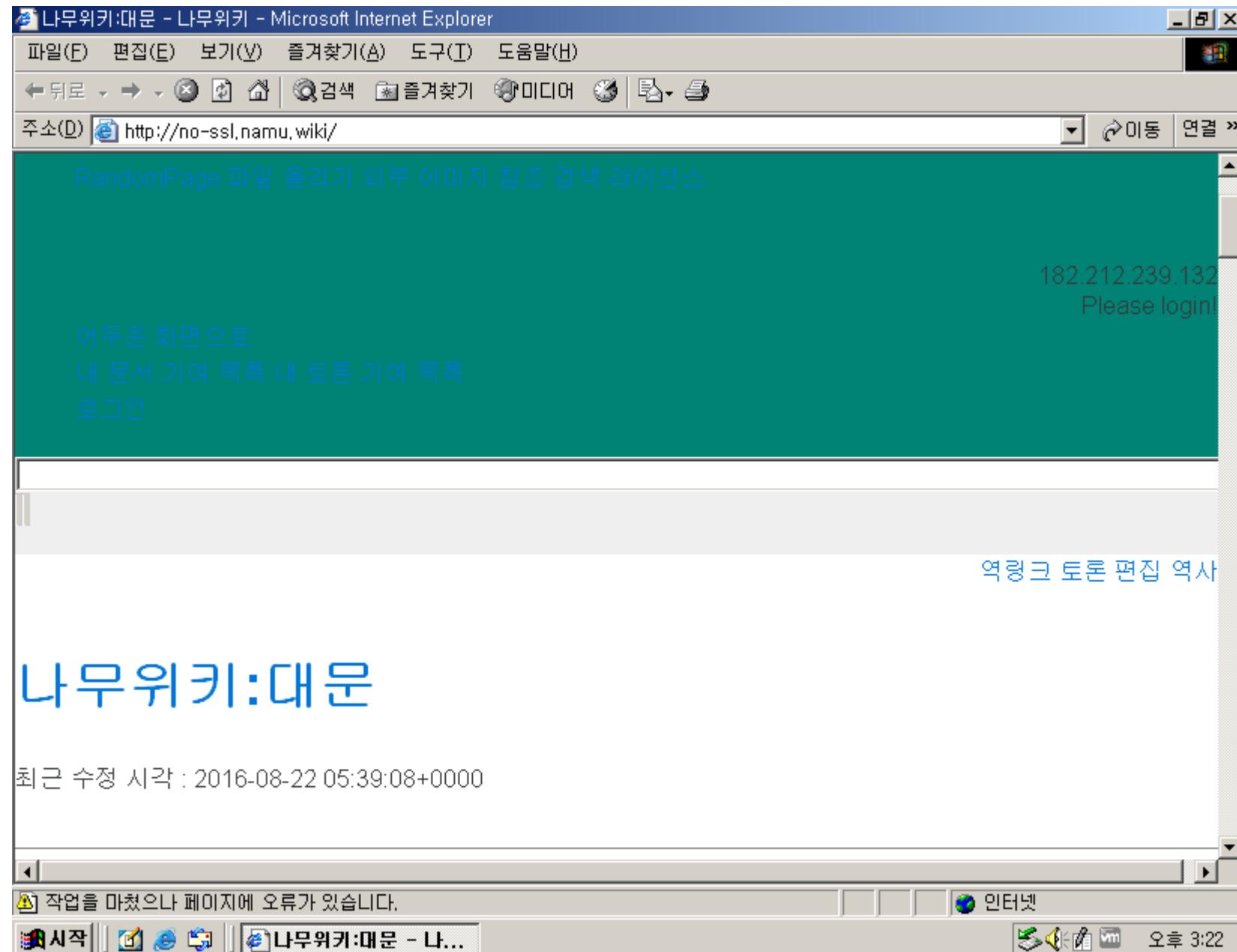
Mosaic(1993)



Netscape(1994)



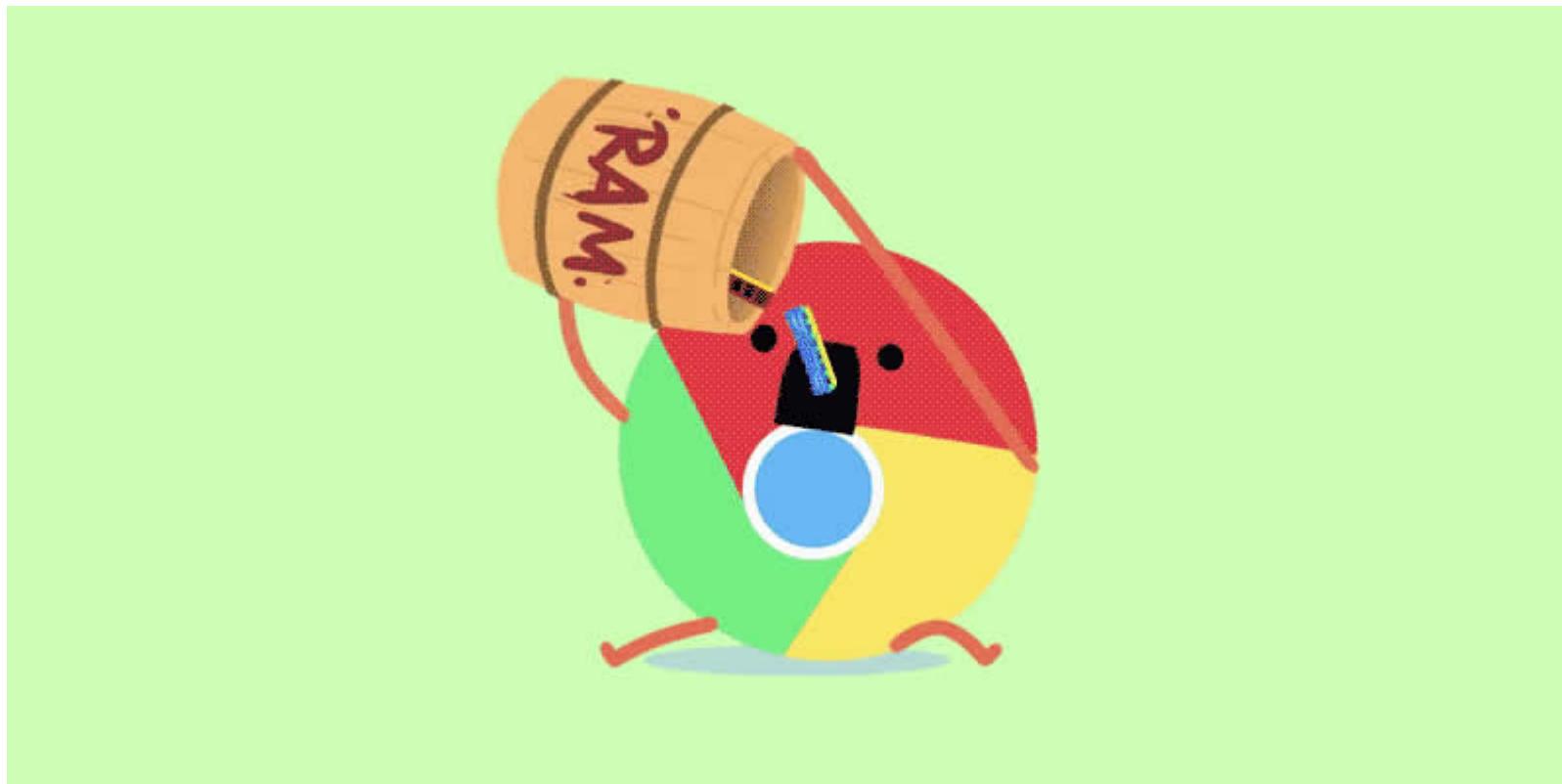
Internet Explorer(1995)



FireFox(2004)



Chrome(2008)



웹 개발의 현재

javaScript

Client-side

- HTML/CSS, javaScript
- jQuery, AJAX
- Front-end Web Framework
 - AngularJS
 - React.js
 - Vue.js
- CSS Framework
 - Bootstrap
 - Foundation

Server-side

- Depends on Language
 - PHP: Laravel
 - javaScript: Node.js(Express.js)
 - Java: Spring
 - C++, C#: [ASP.net](#)
 - Python: Django, Flask
 - Golang: itself
 - Ruby: Ruby on Rails

Database

- RDBMS
 - MySQL
 - PostgreSQL
 - MariaDB
- noSQL
 - MongoDB
 - CouchDB
 - Redis

etc

- celery (for Distributed Task Queue)
- github, Bitbucket, gitlab (for SCM)
- travis CI or jenkins (for Continuous Integration)
- slack, trello

URI, URL, URN

URI

- Uniform Resource Information
- `https://www.example.com/post/how-to-make-url`

URL

- Uniform Resource Locator
- `https://www.example.com/post/`

URN

- Uniform Resource Name
- `www.example.com/post/how-to-make-url`