Network TCP/IP
Apache & Nginx

2019 Wheel Seminar
Cookie

Protocol

정의

-> 서로 다른 기기 / 사용자 간에 정보를 주고받을 때, 그 정보의 형식, 순서, 에러 검출 방법 등을 결정하는 규칙

예시

-> http://google.com에 접속했을 때 벌어지는 일은?

-> 메일을 보낼때는?

-> 내가 보낸 정보가 미국까지가려면 어떤 경로로 가지?

-> 내가 보낸 정보를 다른 사람이 보지 못하게 하려면?

Protocol

-> 네트워크 통신을 위해 주로 사용되는 프로토콜은 TCP / IP 이다. (Transmission Control Protocol)

-> TCP/IP는 4계층으로 이루어져있다. (뒤에서 자세히 다룸)

-> 또 다른 모형으로 OSI 7 계층 모형이 있다.

-> OSI 7 계층 모형은 컴퓨터 네트워크 프로토콜 디자인을 7개의 추상적인 계층으로 나누어 설명한 것이다.

Protocol

-> OSI 7 계층 모델은 ISO (국제 표준화 기구)와 ITTCC (국제 전신 자문 위원회)의 힘을 합쳐 만들어졌다. 이름만 들어도 벌써 재미가 없다. ~~탁상 공론임에 분명하다.~~

-> 반면 TCP/IP 4계층 모델은 미 국방성에서 만들었다. 그러니 엄청 실용적이고 잘 만들었을 수 밖에 없다.

-> 실제로 TCP / IP는 인터넷의 발전 이후에도 꾸준히 발전되어 왔지만 OSI 계층 모델은 실제로 구현된 예가 없어 표준으로서의 의미만 갖는다고 한다.

-> TCP / IP는 실제로 많이 사용되고 있는 **프로토콜**이고 OSI 모델은 통신 규약을 만드는데 참고하는 **모델**이라고 생각하면 되겠다. 새로운 통신 장비 개발에는 OSI 모델이 유용하게 활용된다고 한다.

*참고자료

Protocol

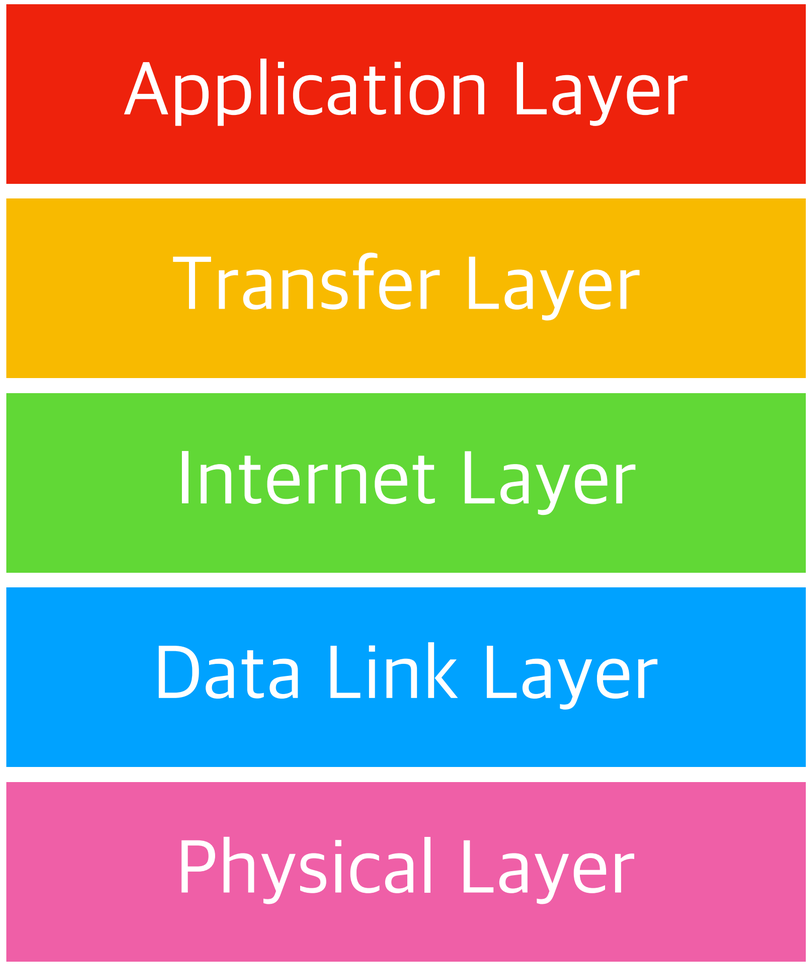
OSI 7 Layer

L7	응용 계층 (Application Layer)
L6	표현 계층 (Presentation Layer)
L5	세션 계층 (Session Layer)
L4	전송 계층 (Transport Layer)
L3	네트워크 계층 (Network Layer)
L2	데이터 링크 계층 (Data Link Layer)
L1	물리 계층 (Physical Layer)

TCP/IP 4 Layer

L4	응용 계층 (Application Layer)
L3	전송 계층 (Transport Layer)
L2	인터넷 계층 (Internet Layer)
L1	네트워크 엑세스 (Network Access Layer)

TCP / IP


Application Layer
Transfer Layer
Internet Layer
Data Link Layer
Physical Layer

왜 자꾸 계층을 나눌까

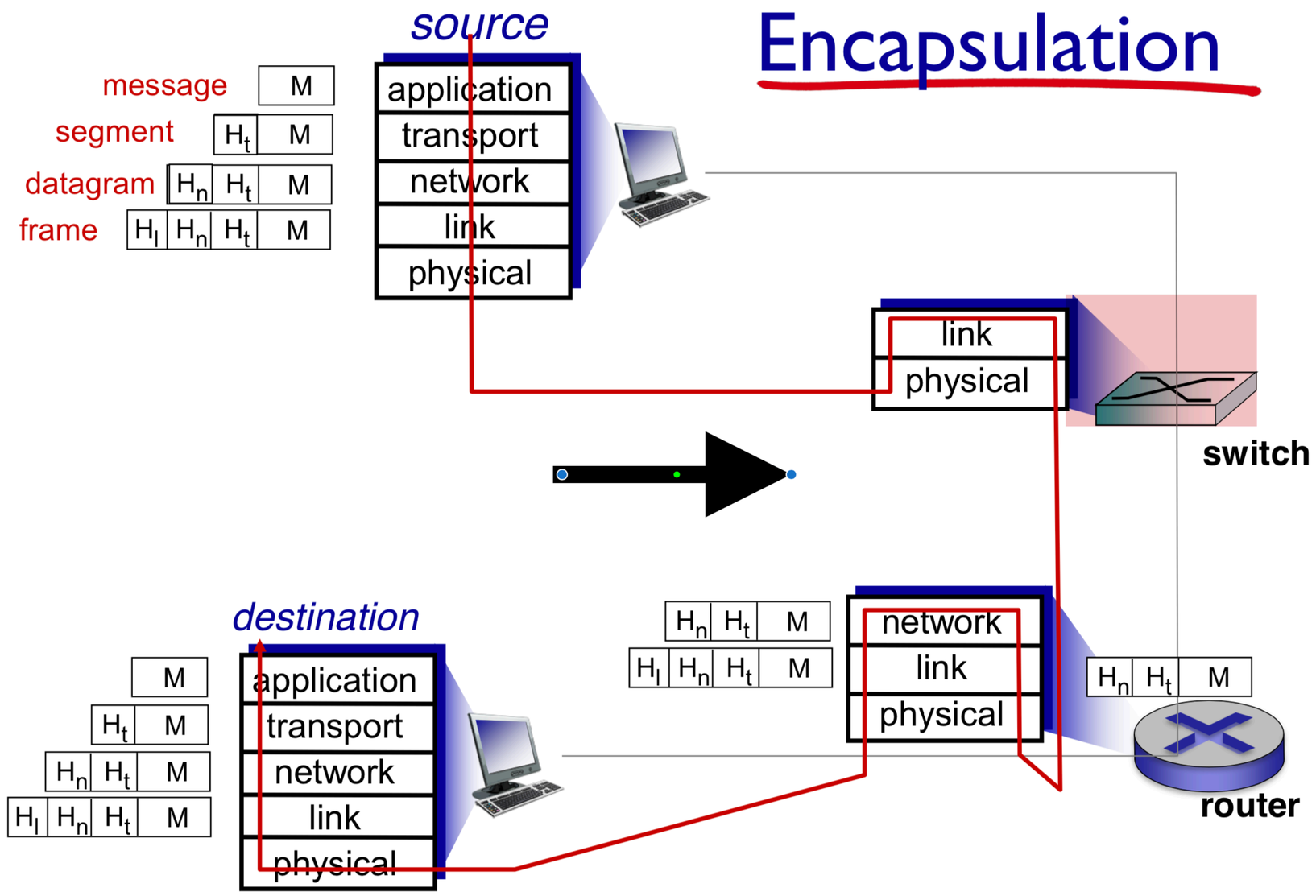
Encapsulation

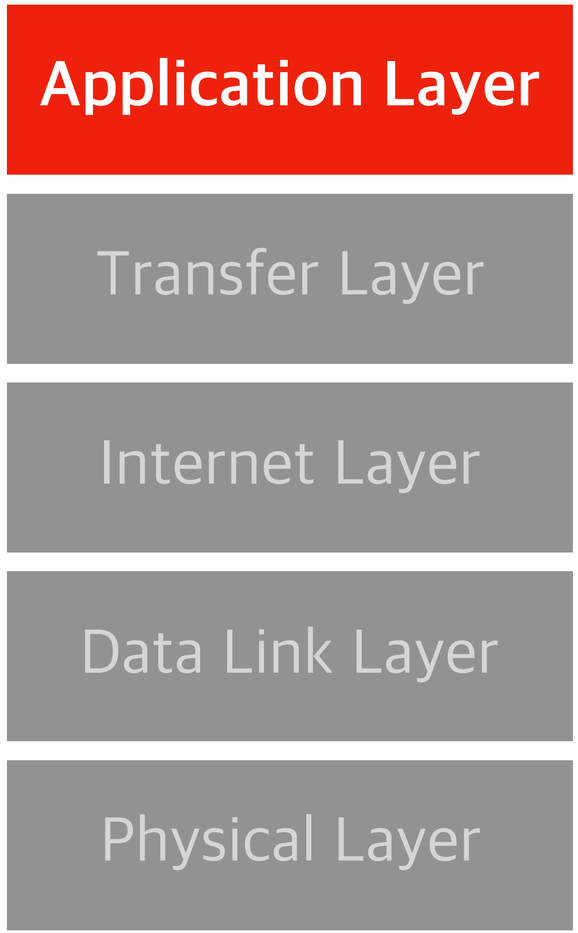
-> 1 단계 (캡슐화) - 각 단계에서 담아야 할 새로운 정보들을 기존 패킷에 추가하고(감싸는 것과 비슷) 하위 계층으로 넘겨준다.

-> 2 단계 (decapsulation) - 각 단계에서 필요한 정보만 추출하고 패킷에서 해당 정보를 삭제한 다음(벗겨내는 것과 비슷) 상위 계층으로 넘겨준다.

-> 각 계층은 다른 계층에서 일어나는 일을 신경쓰지 않아도 되므로 개발에 필요한 로드가 줄어든다.

*질문방지! 더 자세한 내용을 알고 싶다면 CS341을 수강하자





사용자와 가장 가까운 계층

-> 사용하는 **Application**에 따라 달라짐

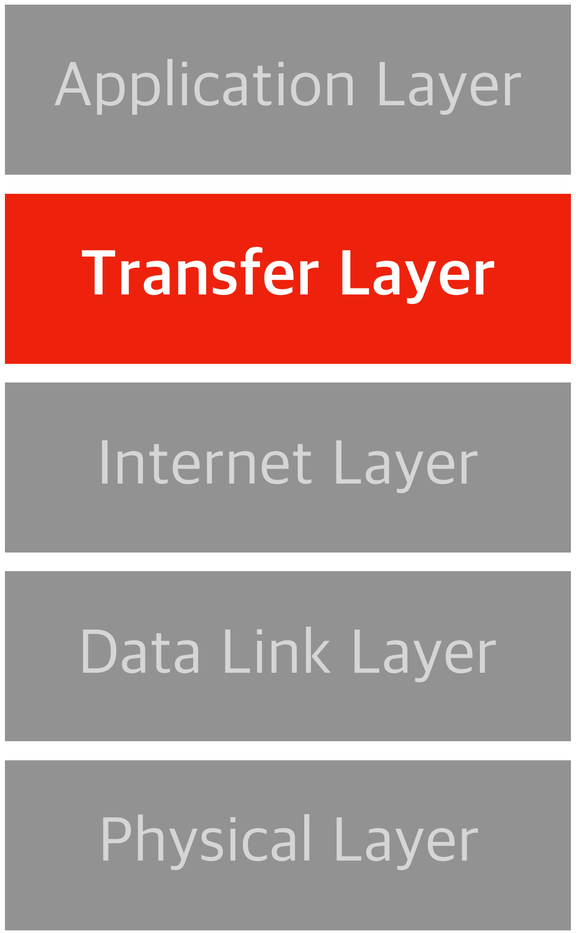
-> 웹 브라우저와 서버가 통신하기 위한 **HTTP**

-> 파일 전송을 위한 **FTP**

-> 메일 수신/발신을 위한 **SMTP, IMAP, POP**

-> 도메인 주소 해석을 위한 **DNS**

-> 암호화된 원격 **Shell** 접속을 위한 **SSH**



두 호스트 간의 통신 방식을 정의

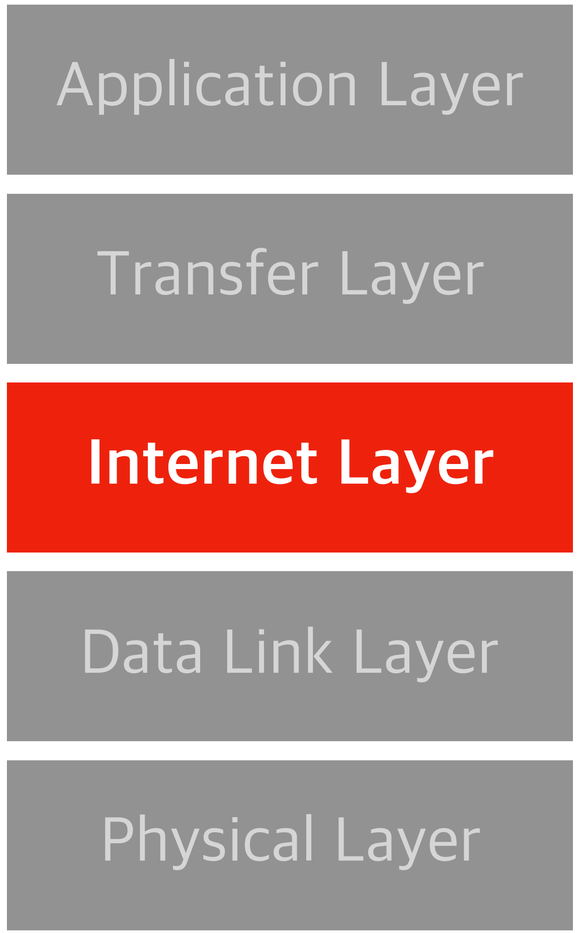
-> TCP/UDP가 여기에 속함

-> TCP는 initialization이 필요하나 신뢰성 있는 전송

-> UDP는 간단하지만 패킷이 손상/손실될 수 있음

-> TCP와 UDP의 포트는 프로세스에 할당됨

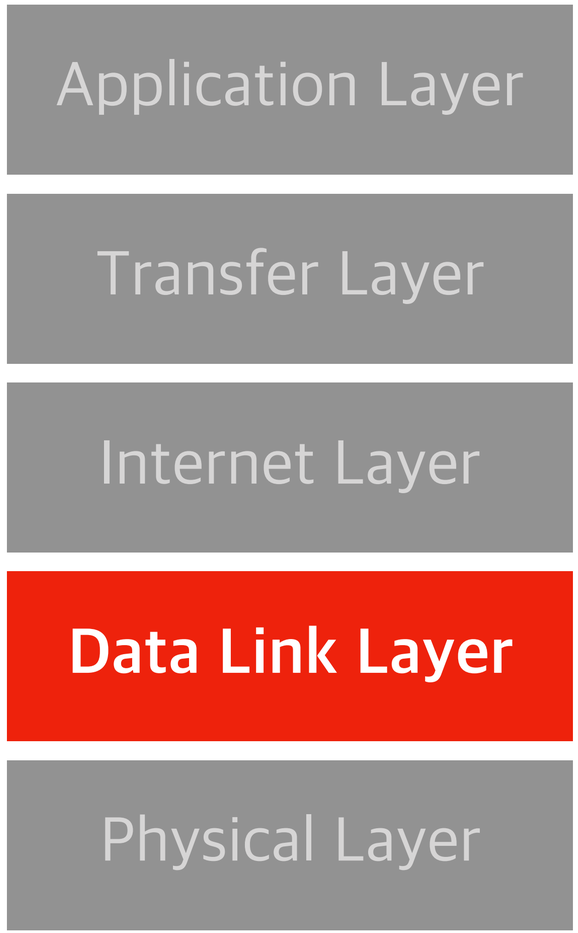
-> 포트는 16비트로 나타내므로, 범위는 0~65535



패킷을 어떻게 보낼 것인지 결정

-> IPv4, IPv6가 여기에 해당

-> 라우터는 IP(+추가적인 정보)를 기반으로 패킷을 라우팅한다

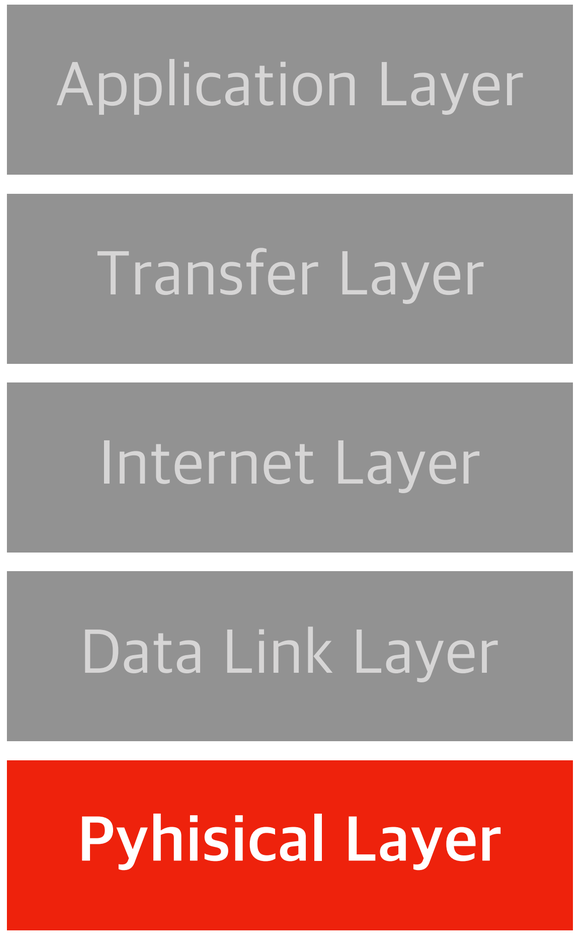


네트워크 상의 기기들을 위한 규약

-> Ethernet이 대표적

-> MAC 주소를 할당

-> 오류 감지 기능을 제공



실제 물리적인 신호를 정의

-> 딱히 우리가 공부할 필요는 없다

구글에 요청을 쏴보자

1. 크롬을 키고 HTTP로 (Hyper Text Transfer Protocol) (얘도 Protocol이다) http://google.com 에 요청을 날린다. - Application Layer

2. 이 요청은 인터넷상으로 전달되기 위해서 TCP 패킷으로 만들어진다. - Transfer Layer

3. 도착지와 출발지 정보를 추가하여 IP 패킷으로 만든다. - Internet Layer

4. 공유기, 이더넷 카드, 랜선 등을 거쳐서 Internet으로 보내진다. - Data Link / Physical Layer

5. 인터넷 상에는 IP 주소를 사용해서 원하는 곳으로 데이터를 전송해주는 역할을 하는 장치들이 있다. 이 장치들의 도움으로 구글의 서버에 패킷이 잘 도착한다.

구글이 내 요청을 받았다

6. 라우터, 스위치, 공유기, 이더넷 카드, 랜선 등을 거쳐서 구글 서버에 데이터가 도착한다. - Data Link / Physical Layer

7. 메시지가 누락된게 있는지 확인한다. - Data Link Layer

8. IP 패킷을 분해해서 누구에게서 온 패킷인지 확인한다. - Internet Layer

9. Nginx나 Apache같은 웹서버가 요청을 받아서 구글 페이지로 응답한다. - Application Layer

10. 같은 과정을 거쳐서 요청자에게 웹 페이지가 전송된다.

Network

근데 내가 보낸 요청이 어떻게 알고 구글 서버까지 가는데??

네트워크 망의 구성

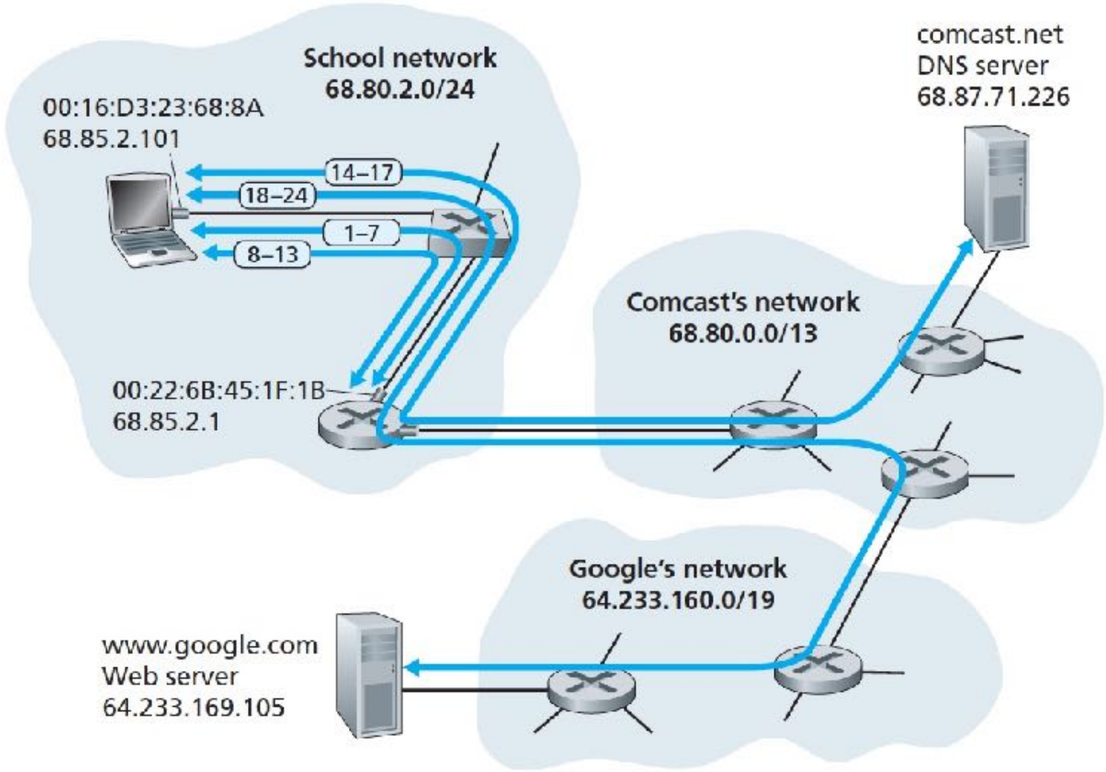


Figure 5.32 ♦ A day in the life of a Web page request: network setting and actions

*이 내용은 사실에 근거하지 않은 제 뇌피셜임을 알려드립니다.

이번엔 택배다

-> 보낼 물건을 택배 상자에 싸고 송장을 붙인다. 택배기사님이 기숙사에 와서 짐을 가져간다.

-> ~~상하차에 지친 택배아저씨에게 음료수를 한 캔 사드린다. 감동의 눈물을 흘리신다.~~

-> 기사님이 대전 유성 유체국에 도착한다.

-> 우체국에 도착한 짐은 대전 우편 집중국으로 옮겨진다.

-> 또 다시 택배가 대전 HUB로 옮겨진다.

대한통운 대전Hub는 블랙홀이다...
https://lookchang.tistory.com/161

-> ~~3일간 빠져나오지 못한다.~~

-> 택배가 군포HUB으로 옮겨진다.

-> 택배가 당동 우체국으로 옮겨진다.

-> 기사님이 택배를 우리집까지 배송한다. 택배에 적힌 내 이름을 보고 내 택밴지 안다.

주소 체계

IP 주소 192.0.2.0

-> 여러 네트워크를 따라 목적지를 찾아갈 때 쓰이는 주소로 지역적인 정보를 포함하고 있음. 계층 구조가 존재함. (경기도 -> 군포시 -> 당동 이런 느낌, 네트워크에서는 서브넷, 호스트 주소)

-> IP는 **주소**이므로, 임의로 사용할 수 없고 국제 기구인 ICANN에서 할당받아서 사용해야 함.

-> IPv4: IP 주소를 32비트로 나타내므로 약 42억 개의 주소를 할당 가능함

-> IPv6: IP 주소를 128비트로 나타내므로 2^{128}개의 주소를 할당 가능함

-> 지구상의 모든 모래알에 IPv6 주소를 부여할 수 있을 정도

-> 우리나라의 IPv6 보급률은 2%대

MAC 주소

-> 각 기기에게 부여된 고유한 주소로, hierarchy가 없음 (주민등록번호의 느낌) 로컬 네트워크에서 목적지를 찾을 때 쓰임

IPv4의 클래스

-> IP 주소의 앞부분은 **네트워크 영역(서브넷, 지역)**을, 뒷 부분은 네트워크 내에서의 **호스트 주소(상세 주소)**를 가리킨다.

서브넷 :

-> 로컬 네트워크, 우체국 관할 지역 정도로 보면 되겠다.

-> “대전광역시 유성구 대학로 291~”처럼 주소의 앞 부분을 공유. 서브넷 내에서는 라우터를 거치지 않고 전송 가능 **서브넷 마스크**를 통해 같은 서브넷인지를 확인

-> 카이스트의 네트워크는 143.248.35.xx 이에요 정도로 생각하면 되겠다.

-> 처음에는 앞 8 비트가 네트워크 영역으로, 나머지 24비트가 호스트 주소로 사용되었음.

-> 인터넷의 성장으로 더 많은 네트워크가 필요해짐. 3개의 클래스 A, B, C로 나누어 더 많은 수의 네트워크가 가능하도록 함.

-> A, B, C 클래스는 각각 도, 시/군/구, 읍/면/리 정도로 보면 된다.

IPv4의 클래스

- *n*는 네트워크 주소를 나타낸다.
- *h*는 호스트 주소를 나타낸다.
- *X*는 특별한 목적이 없는 자리를 나타낸다.

클래스	첫 비트	시작 주소	끝 주소	마스크 블록
클래스 A	0	0.0.0.0	127.255.255.255	/8
클래스 B	10	128.0.0.0	191.255.255.255	/16
클래스 C	110	192.0.0.0	223.255.255.255	/24
클래스 D (멀티캐스트)	1110	224.0.0.0	239.255.255.255	NA
클래스 E (예약됨)	1111	240.0.0.0	255.255.255.255	NA

A Class : MIT, 미 국방부 등 초창기 인터넷 멤버들이 소유
B Class : 기관
E Class : 미래에 사용하기위해 남겨둠
멀티캐스트 : 하나의 요청을 복제하여 여러 디바이스에 뿌려줌.
넷플릭스와 같은 스트리밍 서비스에서 주로 사용한다.

```
클래스 A
  0.  0.  0.  0 = 00000000.00000000.00000000.00000000
127.255.255.255 = 01111111.11111111.11111111.11111111
                  0nnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh

클래스 B
128.  0.  0.  0 = 10000000.00000000.00000000.00000000
191.255.255.255 = 10111111.11111111.11111111.11111111
                  10nnnnnn.nnnnnnnn.hhhhhhhh.hhhhhhhh

클래스 C
192.  0.  0.  0 = 11000000.00000000.00000000.00000000
223.255.255.255 = 11011111.11111111.11111111.11111111
                  110nnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh

클래스 D
224.  0.  0.  0 = 11100000.00000000.00000000.00000000
239.255.255.255 = 11101111.11111111.11111111.11111111
                  1110XXXX.XXXXXXXX.XXXXXXXX.XXXXXXXX

클래스 E
240.  0.  0.  0 = 11110000.00000000.00000000.00000000
255.255.255.255 = 11111111.11111111.11111111.11111111
                  1111XXXX.XXXXXXXX.XXXXXXXX.XXXXXXXX
```

클래스	앞선 비트	총 네트워크 수	네트워크당 주소 수
클래스 A	0	128	16,777,214
클래스 B	10	16,384	65,534
클래스 C	110	2,097,152	254

IPv4의 클래스

-> 대부분의 사이트들은 "클래스 C"에 들어가기에는 너무 컸고, 대신 "클래스 B"를 할당받았다.

-> 클래스 B를 할당받을 수 있는 네트워크의 수는 16,000개 밖에 안된다. 이는 급속도로 소진되었고 다른 해결 방법이 필요해짐.

-> **CIDR**(Classless Inter-Domain Routing, CIDR)의 도입으로 해결.

CIDR - 서브넷 마스크

네트워크 영역으로 사용될 비트의 수를 명시.

Ex: 143.248.35.11/24

-> 143.248.35.11 = 01001111 11110100 00100011 00001011

-> 255.255.255.0 = 11111111 11111111 111111111 00000000

-> 앞의 24 비트가 네트워크 영역, 뒤 8비트가 호스트 주소 영역이다.

-> 나와 같은 서브넷에 존재하는 기기들은 **143.248.35.x** 의 IP를 공유한다!

RFC 3330

- RFC 3330 – Special-Use IPv4 Addresses
- Internet Assigned Numbers Authority (IANA)에 의해서 할당됨

주소	해당 사이더	목적	RFC	클래스	전체 주소 개수
0.0.0.0 – 0.255.255.255	0.0.0.0/8	Zero 주소	RFC 1700	A	16,777,216
10.0.0.0 – 10.255.255.255	10.0.0.0/8	사설망	RFC 1918	A	16,777,216
127.0.0.0 – 127.255.255.255	127.0.0.0/8	로컬호스트 Loopback 주소	RFC 1700	A	16,777,216
169.254.0.0 – 169.254.255.255	169.254.0.0/16	Zeroconf	RFC 3330	B	65,536
172.16.0.0 – 172.31.255.255	172.16.0.0/12	사설망	RFC 1918	B	1,048,576
192.0.2.0 – 192.0.2.255	192.0.2.0/24	문서와 예제	RFC 3330	C	256
192.88.99.0 – 192.88.99.255	192.88.99.0/24	IPv6에서 IPv4로의 애니캐스트 릴레이	RFC 3068	C	256
192.168.0.0 – 192.168.255.255	192.168.0.0/16	사설망	RFC 1918	C	65,536
198.18.0.0 – 198.19.255.255	198.18.0.0/15	네트워크 장치 벤치마크	RFC 2544	C	131,072
224.0.0.0 – 239.255.255.255	224.0.0.0/4	멀티캐스트	RFC 3171	D	268,435,456
240.0.0.0 – 255.255.255.255	240.0.0.0/4	예약됨	RFC 1700	E	268,435,456

NAT

-> IP를 더더욱 효율적으로 사용하기 위한 수단

-> 라우터가 Public IP (+port) 와 Private IP (+port) 사이에서 주소를 변환하는 기능

예시

-> Public IP로는 143.248.0.7:2000과 143.248.0.7:2001 이지만 라우터가 이를 Private IP로 변환하면 10.0.0.5:5000과 10.0.0.6:5000 이 될 수 있다.

-> 외부에서 볼 때 둘은 동일한 주소의 다른 포트 같지만 실제로는 서로 다른 두 개의 기기이다. IP 주소를 더 효과적으로 사용한 셈이다.

-> TCP / UDP의 체크섬을 다시 계산해야하므로 성능에 영향을 준다.

라우터 (Router)

-> 우편 집중국 혹은 택배 허브와 같은 역할

-> Network Layer에서 작동하는 기기로 IP 헤더까지 뜯어볼 수 있음. 택배에 붙어있는 주소를 보는 것과 비슷!

-> 분리되어 있는 두 개 이상의 망 사이에서 적절한 경로를 찾아 패킷을 전송해주는 역할. 예를 들자면

  -> 인터넷 서비스 제공자 (ISP, 돈을 내면 인터넷을 연결해주는 회사들, KT나 U+ 등을 생각하면 된다.)들 사이의 네트워크를 연결해주는 라우터

  -> 중앙 네트워크와 각 지역 분점들을 연결해주는 라우터

-> 여러 개의 포트를 가지고 있음. 한 포트로 들어와서 다른 포트로 나감.

-> 들어온 패킷이 어느 포트로 나갈지 결정하는 Forwarding table을 저장하고 있음

-> 라우팅 알고리즘을 통해 패킷을 어디로 보내야 할 지 학습

-> 라우팅 알고리즘이 잘못되면…

-> 대전에서 서울로 올 택배가 부산을 거쳐 오거나

-> 옥천에서 3일간 택배가 빠져나오지 못하는 상황이 발생

스위치 (Switch)

-> 우체국 정도의 역할

-> 로컬 네트워크 내부에서 각 컴퓨터의 MAC 주소를 기억하고, 적절한 목적지로 패킷을 forwarding해 주는 역할

-> 접근 가능한 레이어에 따라 L2, L3, L4 스위치 등이 존재

-> ex. L2 스위치는 Data Link Layer까지 접근 가능 — 간단하고 빠름

-> ex. L3 스위치는 Internet Layer까지 접근 가능 — 망 분리 등의 추가적인 기능 제공 가능

Host IP

-> 로컬 네트워크 안에서 호스트 IP를 설정하는 방법은 Static IP Address 와 Dynamic IP Address가 있다.

-> Static IP Address는 호스트가 자신이 사용할 IP 주소를 미리 정해놓는 방식. 서로 다른 호스트의 IP 주소가 겹칠 위험이 있다.

-> Dynamic IP Address는 DHCP 서버 (소규모 네트워크의 경우 대부분 라우터)가 네트워크에 접속해있는 호스트들에게 겹치지 않도록 IP 주소를 할당해준다. 겹칠 위험은 없지만 IP 주소가 계속 바뀔 수 있다.

Host IP address

-> 네트워크 변경이 필요하지 않고 다른 디바이스에서 원격으로 접속해야하는 경우 static IP adress 를 설정해놓고 사용한다.

Ex) 프린터, 서버 컴퓨터

-> 노트북이나 핸드폰 등 네트워크 변경이 잦은 디바이스들에게는 Dynamic IP Address를 할당한다.

Ex) 네트워크 설정에서 지금 각자의 노트북에 할당된 주소를 볼 수 있다. 이 주소는 SPARCS-AP 라우터에게 할당받은 Dynamic IP Address!

-> 라우터(또는 DHCP 서버)가 IP 주소를 할당하기 위해서 사용하는 protocol이 DHCP (Dynamic Host Configuration Protocol)

Host IP address

-> Dynamic IP Address는 일정 기간의 수명이 있어서 계속해서 바뀔 수 있다.

-> 때문에 외부에서 계속해서 접속해야하는 서버 컴퓨터에는 Dynamic IP가 아닌 Static IP를 할당해야한다.

더 알면 좋은 내용

- ARP - MAC 주소가 어떻게 돼요?
- DNS - http://google.com 의 IP 주소가 어떻게 돼요?
- TCP - 전화
- UDP - 문자
- TCP handshake - 들려? 어 들려
- TLS, DNSSEC - 보안!

Web Server

Apache & Nginx

Virtual Host

-> 한 서버에서 여러 개의 사이트를 운영할 수 있음

-> 여러 개의 도메인 이름, 하나의 ip 주소, 여러 개의 서비스
ex) 한 서버에서 www.example.com과 example.com 을 모두 서비스하려고 할 때

-> 다른 ip 주소인데 같은 서비스
ex) 서버 컴퓨터에 IP 주소가 두개 (192.168.1.1과 172.20.30.40) 있다. 컴퓨터는 내부 (인트라넷) 네트워크와 외부 네트워크 사이에 위치한다. 네트워크 밖에서 server.example.com은 외부 주소를 (172.20.30.40) 의미하고, 네트웍 내부에서 같은 이름을 내부 주소로 (192.168.1.1) 사용한다.

-> 여러 포트에서 서비스
ex) 80번 포트와 443포트로 들어오는 요청을 같은 서버로 요청한다.
ex) 10000번 포트는 otl 서버로, 10001번 포트는 ara 서버로 연결한다.

-> 등등 IP, DNS, port로 짬뽕할 수 있는 모든 것

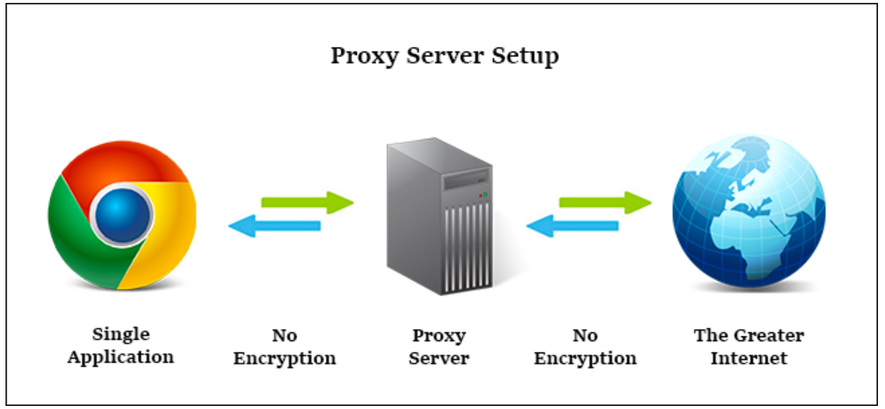
Reverse Proxy

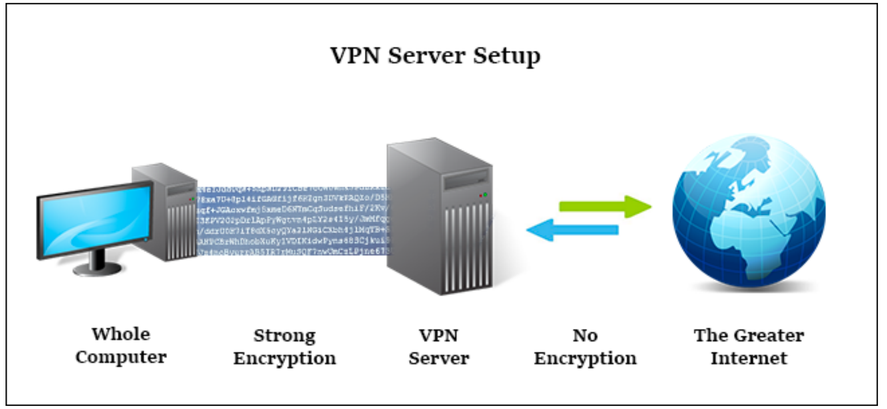
Proxy : 대행자

Proxy Server

어떤 요청을 proxy server를 통해서 요청을 보내면 요청자의 정보를 숨길 수 있다.

VPN 또한 proxy server로 볼 수 있다. VPN은 OS 레벨에서 암호화를 진행한다는 차이가 있다.





Reverse Proxy

Proxy가 요청을 보낼 때 proxy server를 사용한다면 Reverse Proxy는 요청을 받기 전에 요청을 처리해주는 서버를 둔다.

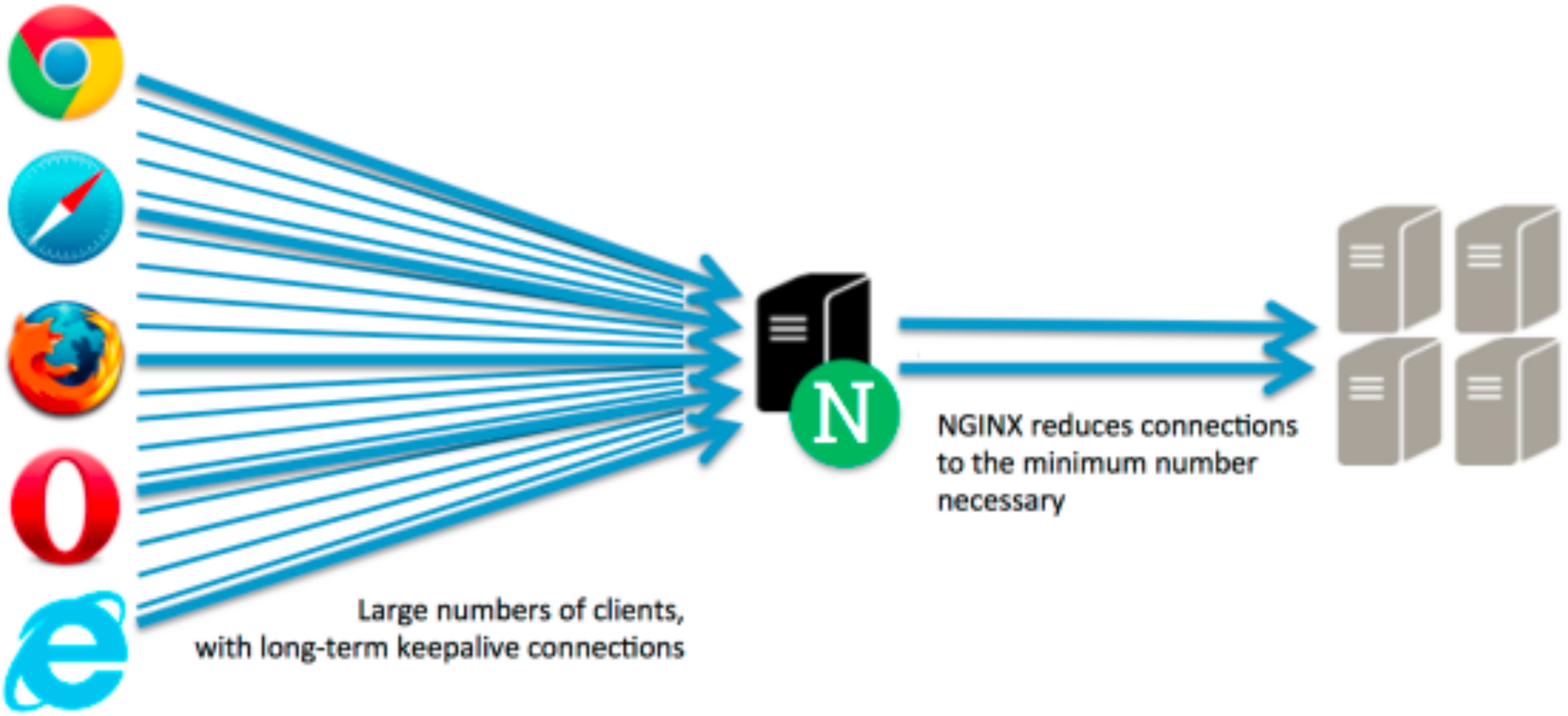
Nginx와 Apache가 그러한 역할을 하는데, 들어온 요청들을 여러 프로세스 또는 서버에 적절히 분배하는 역할을 한다.

활용

-> 서버에 요청을 넣기 전에 IP black list 처리를 할 수 있다.

-> 이미지 파일이나 동영상 파일처럼 static file은 서버로 처리를 넘기지 않고 reverse proxy에서 직접 처리해줄 수 있다.

Reverse Proxy



Apache vs Nginx

-> 앞에서 말한 기능을 처리해줄 수 있는 두 종류의 대표주자가 있다. 하나는 Apache, 하나는 Nginx ~~(Microsoft IIS 도 있는데 linux에서 도는게 아니라 우리가 알아야하나 싶다.)~~

-> 둘 중 뭘 골라야 할까?

Apache

-> HTTP Web 서버용 소프트웨어

-> Apache Software 재단에서 제작

-> Linux에서 구동되는 것을 목표로 만들어짐

-> 확장성이 좋음 ~~(= 설정할게 많음)~~

-> 가장 많이 사용되는 웹 서버, 전통 강호, 하락세

Apache - MPM

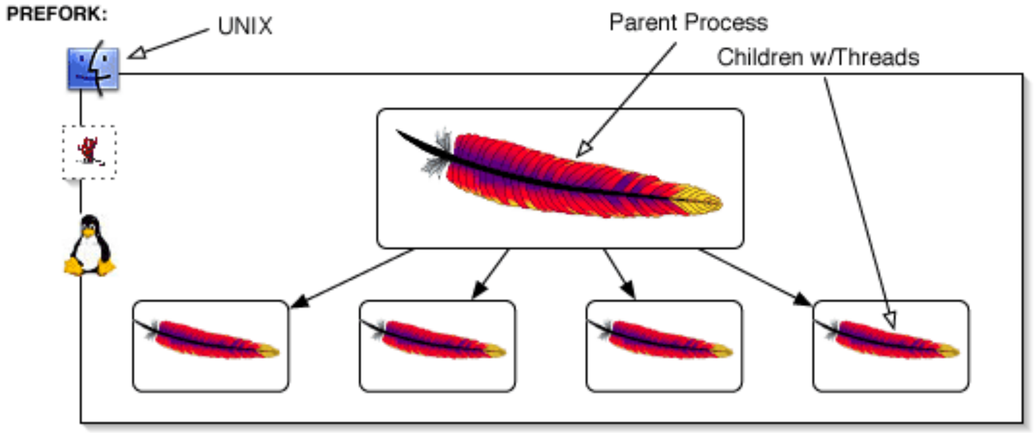
-> MPM (Multi Processing Module) 방식으로 요청을 처리.

-> 새로운 요청이 들어올 때 마다 프로세스를 복제하는 방식.

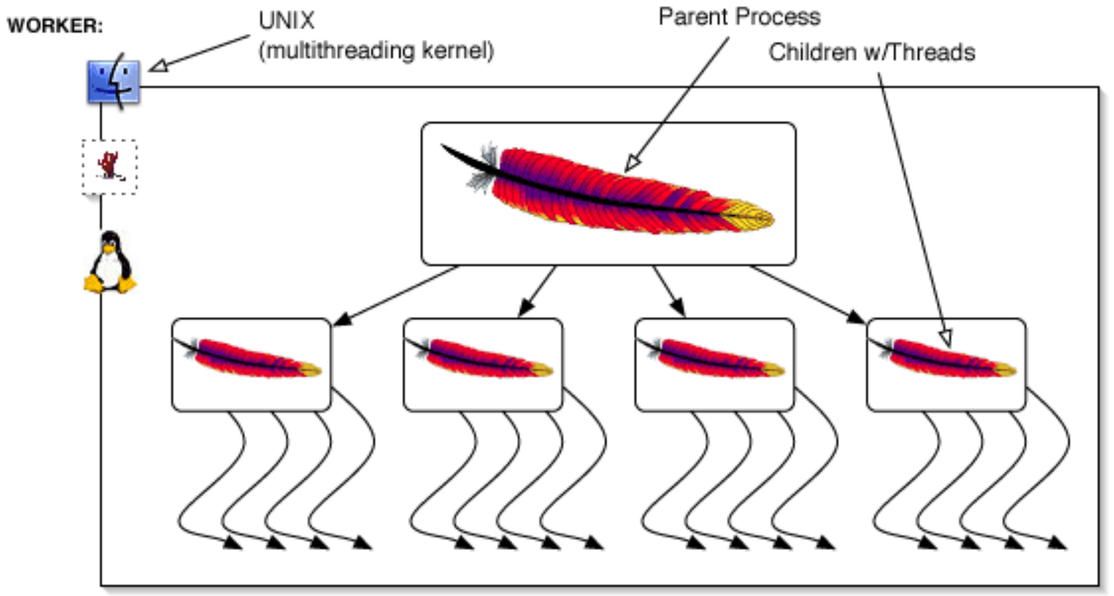
-> 프로세스 복제 시 메모리 영역까지 복사. 요청량이 많아지면 메모리 사용량이 크게 증가한다.

-> 멀티 쓰레드 OS에서는 메모리 사용량을 줄이기 위해서 여러 쓰레드가 메모리를 공유할 수 있다. 하지만 요청이 많아지면 프로세스를 복제해야해서 여전히 메모리 사용량이 크다.

Apache - MPM



Prefork MPM (single thread)



Worker MPM (multi thread)

Nginx

-> Apache와 비교해서 높은 트래픽, 정적 파일 서빙에 강함

  -> 약 4배의 요청을 처리 가능. 메모리도 적게 먹는다.

-> 간편한 설정

  -> .htaccess파일을 설정해줄 필요 없음

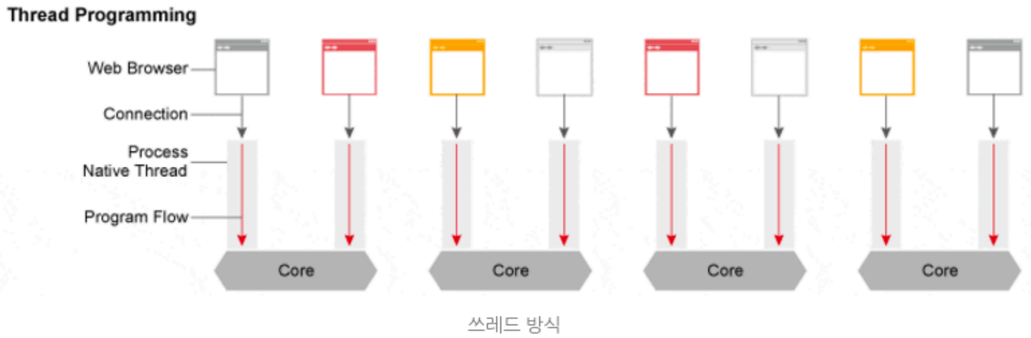
  -> 접근 제어의 자유도가 떨어짐

-> AWS내 점유율 50%+ 신흥 강자, 상승세

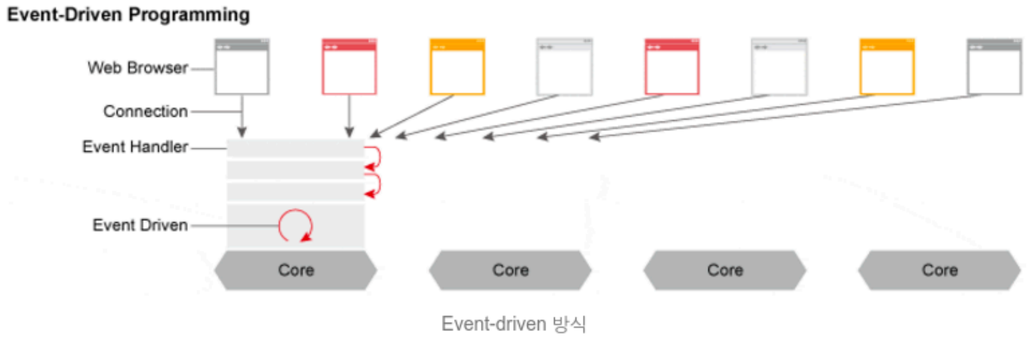
Nginx

-> Event-driven 비동기 방식(Asynchronous)으로 동작

-> Single-threaded(worker 프로세스)



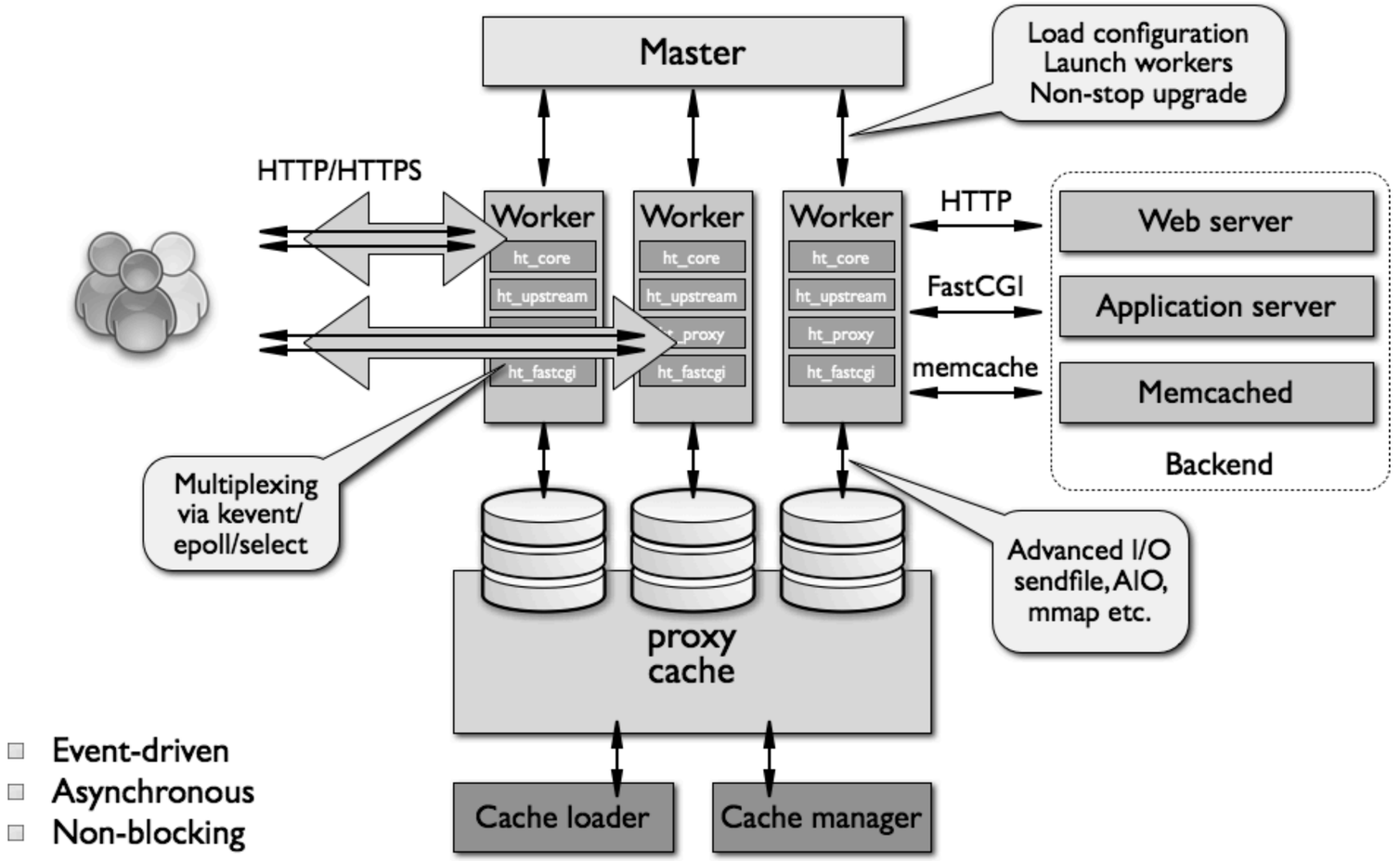
쓰레드 방식



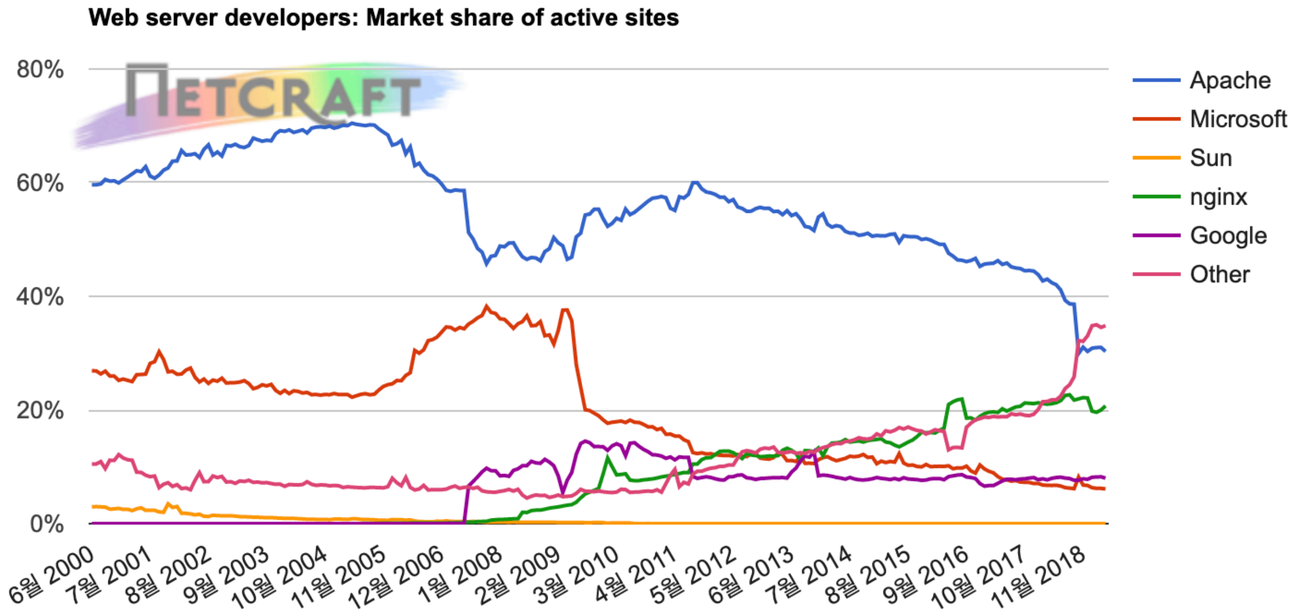
Event-driven 방식

* 참고자료

Nginx


Master
Load configuration
Launch workers
Non-stop upgrade
HTTP/HTTPS
Worker
ht_core
ht_upstream
ht_proxy
ht_fastcgi
HTTP
FastCGI
memcache
Web server
Application server
Memcached
Backend
Multiplexing via kevent/ epoll/select
Advanced I/O sendfile, AIO, mmap etc.
proxy cache
Cache loader
Cache manager
Event-driven
Asynchronous
Non-blocking

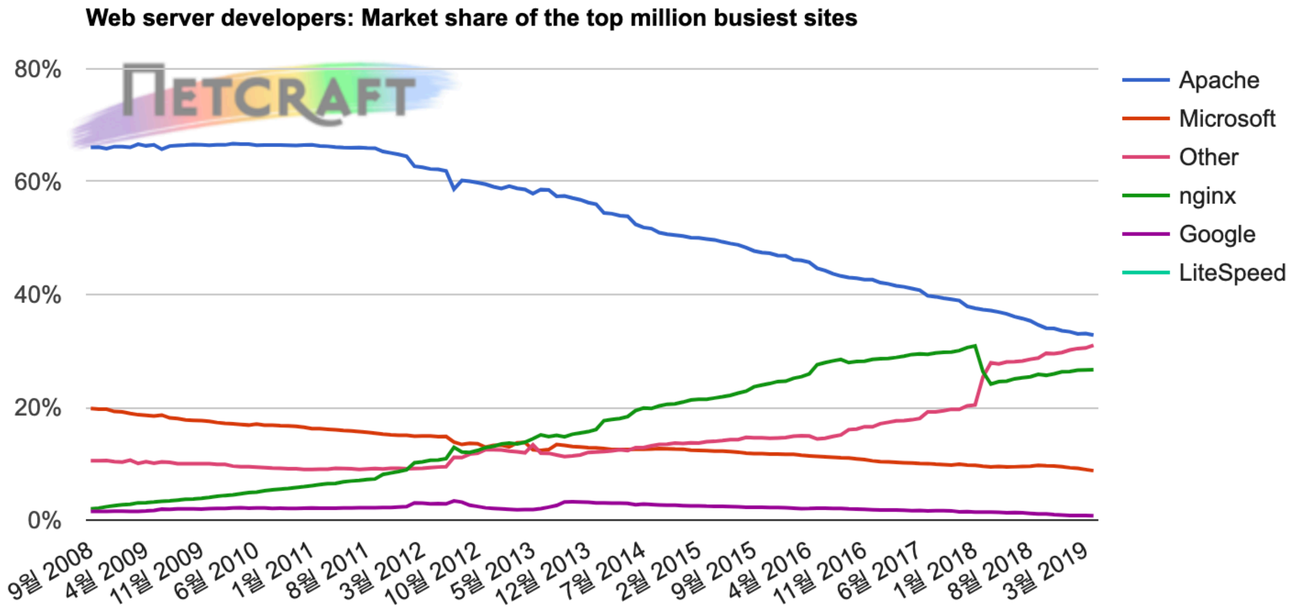
Apache vs Nginx



Developer	March 2019	Percent	April 2019	Percent	Change
Apache	56,236,608	31.01%	55,982,911	30.30%	-0.71
nginx	36,268,792	20.00%	38,293,931	20.73%	0.73
Google	14,953,818	8.25%	14,801,433	8.01%	-0.23
Microsoft	11,254,841	6.21%	11,298,492	6.12%	-0.09

April 2019 survey we received responses from **1,445,266,139** sites across **233,886,577** unique domain names and **8,613,630** web-facing computers.

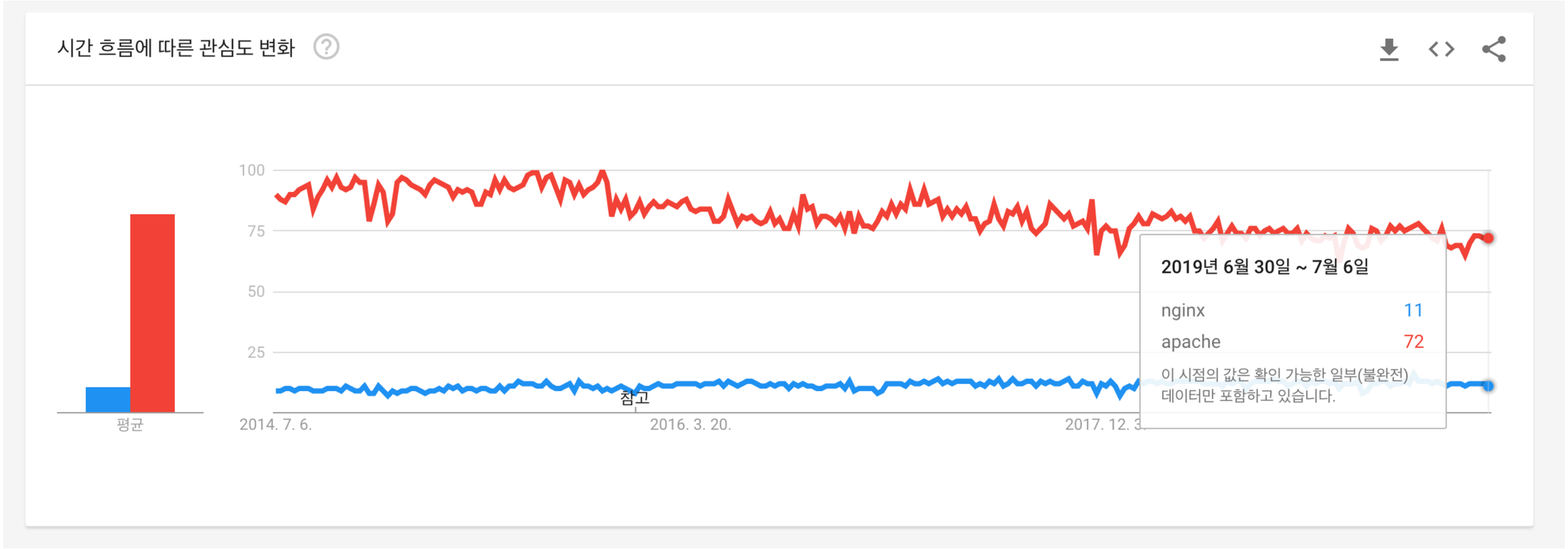
Apache vs Nginx



Developer	March 2019	Percent	April 2019	Percent	Change
Apache	325,237	32.52%	322,376	32.24%	-0.29
nginx	262,032	26.20%	262,220	26.22%	0.02
Microsoft	88,181	8.82%	86,118	8.61%	-0.21
LiteSpeed	16,536	1.65%	17,068	1.71%	0.05

April 2019 survey we received responses from **1,445,266,139** sites across **233,886,577** unique domain names and **8,613,630** web-facing computers.

Apache vs Nginx



Apache vs Nginx

그래서 뭘 써야하는가??

1. 많이 쓰이는건 아직 Apache. 커뮤니티 파워를 무시할 순 없다.

2. 하지만 Nginx도 사용자가 매우매우 많기 때문에 커뮤니티 파워는 충분하다.

3. 유저가 많아질 수록 Nginx가 좋다.

4. 그러니 그냥 nginx를 쓰자!

Nginx

Ubuntu에서 sudo apt-get install nginx 명령어로 설치할 수 있다.

/etc/nginx 디렉토리 아래에 설정 파일들이 위치

-> nginx.conf : 메인 설정 파일

-> conf.d/ : 추가 설정 파일들을 넣는 폴더 (첨엔 비어있다.)

-> ~~fcgi.conf : FastCGI 환경설정 파일 (몰라도 된다)~~

-> sites-available/ : 비활성화된 사이트들의 설정 파일들이 위치. 사이트 설정을 변경하고 싶으면 이 폴더 안에서 설정 파일을 추가하거나 수정하면 된다. (첨엔 default 라는 설정 파일이 하나 있다.)

-> sites-enabled/ : 활성화된 사이트들의 설정 파일들이 위치. sties-available/ 폴더 안에서 활성화 하고 싶은 서버 설정들의 링크 파일을 해당 폴더에 생성하는 방식으로 활성화 시킨다.

Nginx 설정

```
sparcs@211158846f12:/etc/nginx$ ls
conf.d        fastcgi_params  koi-win     nginx.conf    scgi_params      sites-enabled  uwsgi_params
fastcgi.conf  koi-utf         mime.types  proxy_params  sites-available  snippets       win-utf
sparcs@211158846f12:/etc/nginx$ ls -al sites-enabled/
total 8
drwxr-xr-x 2 root root 4096 May 24 19:55 .
drwxr-xr-x 6 root root 4096 Jul  5 09:42 ..
lrwxrwxrwx 1 root root   34 May 24 19:55 default -> /etc/nginx/sites-available/default
sparcs@211158846f12:/etc/nginx$ ls -al sites-available/
total 12
drwxr-xr-x 2 root root 4096 Jul  5 09:37 .
drwxr-xr-x 6 root root 4096 Jul  5 09:42 ..
-rw-r--r-- 1 root root 1003 May 27 06:58 default
sparcs@211158846f12:/etc/nginx$
```

Nginx 설정

```
worker_processes  1;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    server {
        listen       80;
        location / {
            root   html;
            index  index.html index.htm;
        }
    }
}
```

Nginx 설정

Core 모듈 설정

-> 위의 예의 work_processes와 같은 지시자 설정 파일 최상단에 위치하면서 nginx의 기본적인 동작 방식을 정의한다.

http 블록

-> http 블록은 이후에 소개할 server, location의 루트 블록이라고 할 수 있고, 여기서 설정된 값을 하위 블록들은 상속한다. http 블록은 여러개를 사용할 수 있지만 관리상의 이슈로 한번만 사용하는 것을 권장한다.

-> http, server, location 블록은 계층구조를 가지고 있다. 많은 지시어가 각각의 블록에서 동시에 사용할 수 있는데, http의 내용은 server의 기본값이 되고, server의 지시어는 location의 기본값이 된다. 그리고 하위의 블록에서 선언된 지시어는 상위의 선언을 무시하고 적용된다.

server 블록

-> server 블록은 하나의 웹사이트를 선언하는데 사용된다. 가상 호스팅(Virtual Hosting)의 개념이다. 예를들어 하나의 서버로 http://zabo.sparcs.org 과 http://zabo.kaist.ac.kr 을 동시에 운영하고 싶은 경우 사용할 수 있는 방법이다.

location 블록

-> location 블록은 server 블록 안에 등장하면서 특정 URL을 처리하는 방법을 정의한다. 이를테면 http://zabo.sparcs.org/ 과 http://opentutorials.org/api 로 접근하는 요청을 다르게 처리하고 싶을 때 사용한다.

events 블록

-> 이벤트 블록은 주로 네트워크의 동작방법과 관련된 설정값을 가진다. 이벤트 블록의 지시어들은 이벤트 블록에서만 사용할 수 있고, http, server, location와는 상속관계를 갖지 않는다.

Nginx 설정

설정 파일을 열어보면 자동으로 sites-enabled/* 를 가져오고 있는걸 알 수 있다.
즉 sites-enabled/example 이란 설정파일을 만들고 안에 server 블록을 선언할 수 있다.

```
 9  http {
10
11  ...
28
29      ##
30      # Virtual Host Configs
31      ##
32
33      include /etc/nginx/conf.d/*.conf;
34      include /etc/nginx/sites-enabled/*;
35  }
```

Nginx 실습

노트

- EC2 IP 주소 : 54.180.150.186

- AWS Console IAM 유저

=> 계정 :

=> username : route53admin

=> passwd:

AWS EC2 접속하기

-> ~/.ssh/id_rsa 파일이 있는지 확인

-> 없으면 ssh-keygen 명령어로 생성

-> ~/.ssh/id_rsa.pub 파일의 내용을 연사에게 공유!

-> 설정이 완료되면 ssh ubuntu@[ec2-ip-address] 로 접속

DNS 설정 추가하기

1. console.aws.amazon.com 접속

2. 로그인

3. Route53 서비스 => 호스팅 영역 => sparcs.org => 레코드 세트 생성 => 이름 : [nickname].sparcs.org, 값: [ec2 ip 주소]

* 다른 설정들을 건드리지 않도록 주의하자!

HTML 파일 추가

cd /var/www/html

sudo mkdir [nickname]

sudo cp cookie/index.html [nickname]/index.html

sudo vi [nickname]/index.html => 안의 내용 마음대로 수정

Nginx 설정

1. cd /etc/nginx
2. sudo cp sites-available/cookie.sparcs.org sites-available/[nickname].sparcs.org
3. sudo vi sites-available/[nickname].sparcs.org
4. server 블록의 root 폴더 디렉토리를 /var/www/html/[nickname]으로 변경
5. server 블록의 server_name을 [nickname].sparcs.org로 수정
6. 파일 저장
7. sudo ln -s /etc/nginx/sites-available/[nickname].sparcs.org /etc/nginx/sites-enabled/[nickname].sparcs.org
8. sudo systemctl reload nginx
9. [nickname].sparcs.org 접속해서 잘 나오는지 확인!