Network

nenw

@ SPARCS 20' Summer Wheel Seminar

Caution

이 세미나는 서버 관리를 위한 네트워크의 전반적인 지식을 쉽게 이해하는데 초점을 맞추고 있습니다.
자세한 내용은 CS341에서 배워주시면 감사하겠습니다.

또한, 발표자의 문제로 잘못된 정보가 있을 수 있습니다.
잘못된 게 보이면 빠르게 지적해주시면 감사하겠습니다.

네트워크? 인터넷?

네트워크

컴퓨터와 컴퓨터를 이을 수 있을 거 같은데..?

네트워크

데이터

네트워크

- 그렇게 ARPANET이 생기고 발전되어 전 세계가 연결됨

그래서 내 데이터가 어떻게 그 네트워크를 통해서 지구 반대편의 목적지에 도달할까?

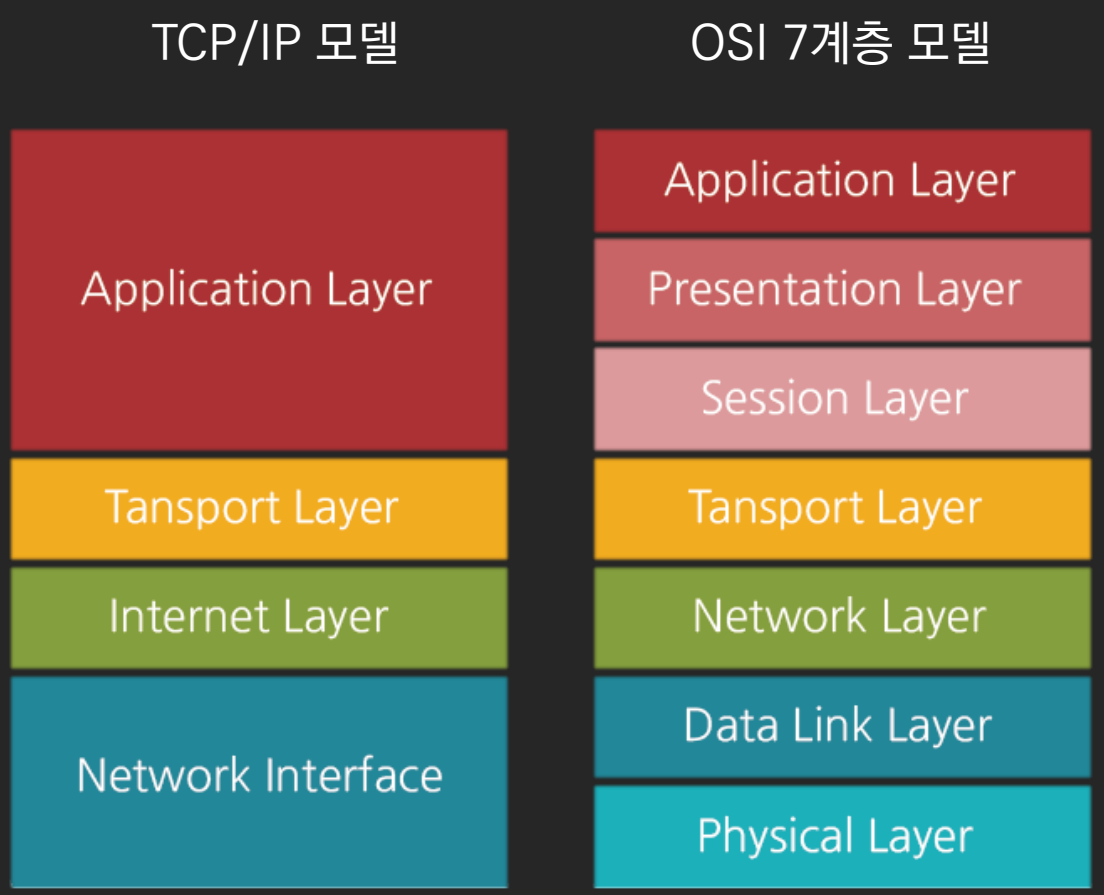
패킷

- 데이터를 MTU 단위로 쪼개서 보냄
- 각 패킷이 독립적으로 목적지를 향해 이동
    - 도착하는 순서가 뒤바뀌거나, 패킷이 누락되거나 할 수 있음

프로토콜

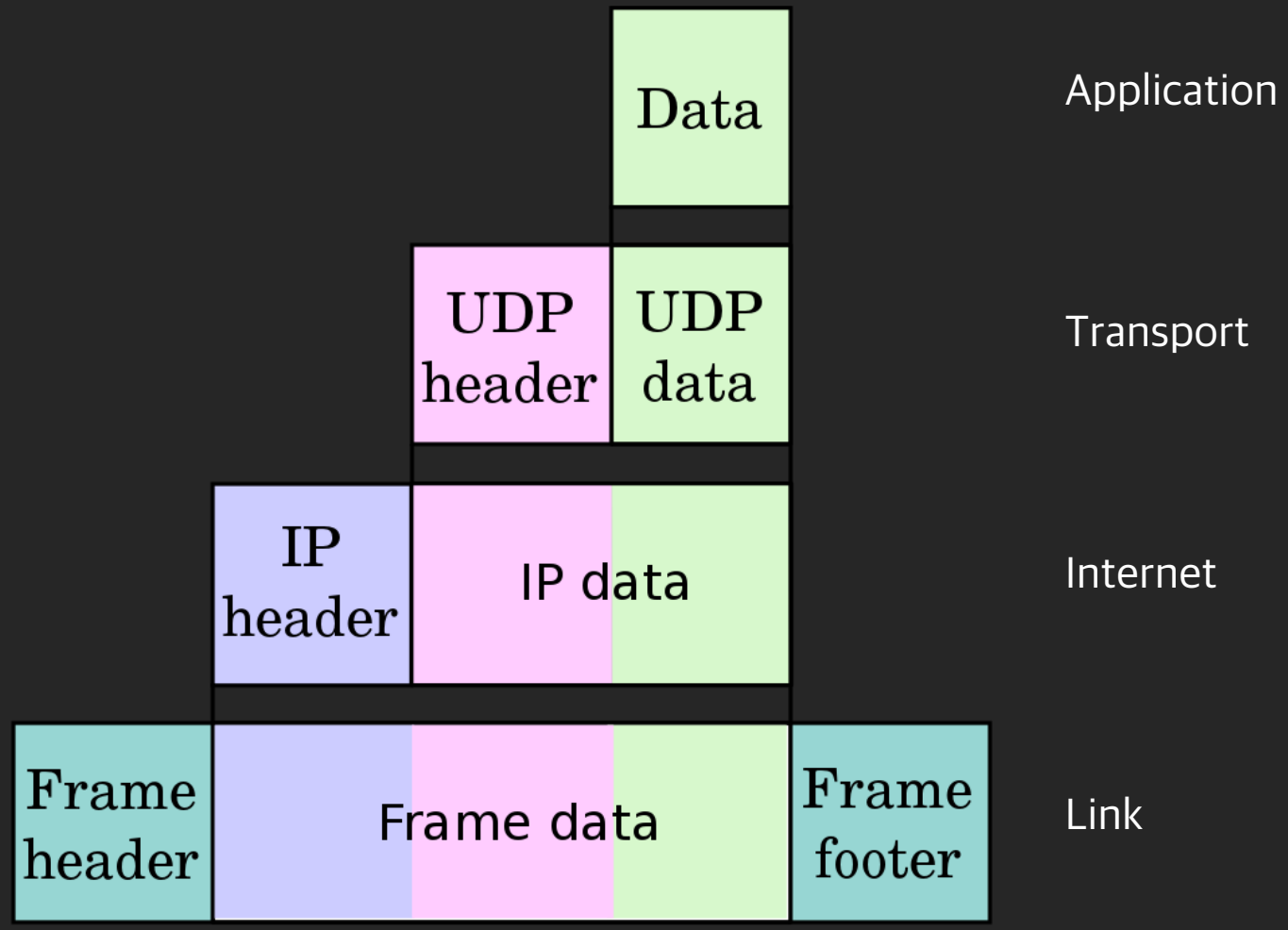
- "통신 규약"
- 정보의 형식, 순서, 에러 검출 방법 등을 결정하는 규칙

OSI 모델


TCP/IP 모델
Application Layer
Tansport Layer
Internet Layer
Network Interface
OSI 7계층 모델
Application Layer
Presentation Layer
Session Layer
Tansport Layer
Network Layer
Data Link Layer
Physical Layer

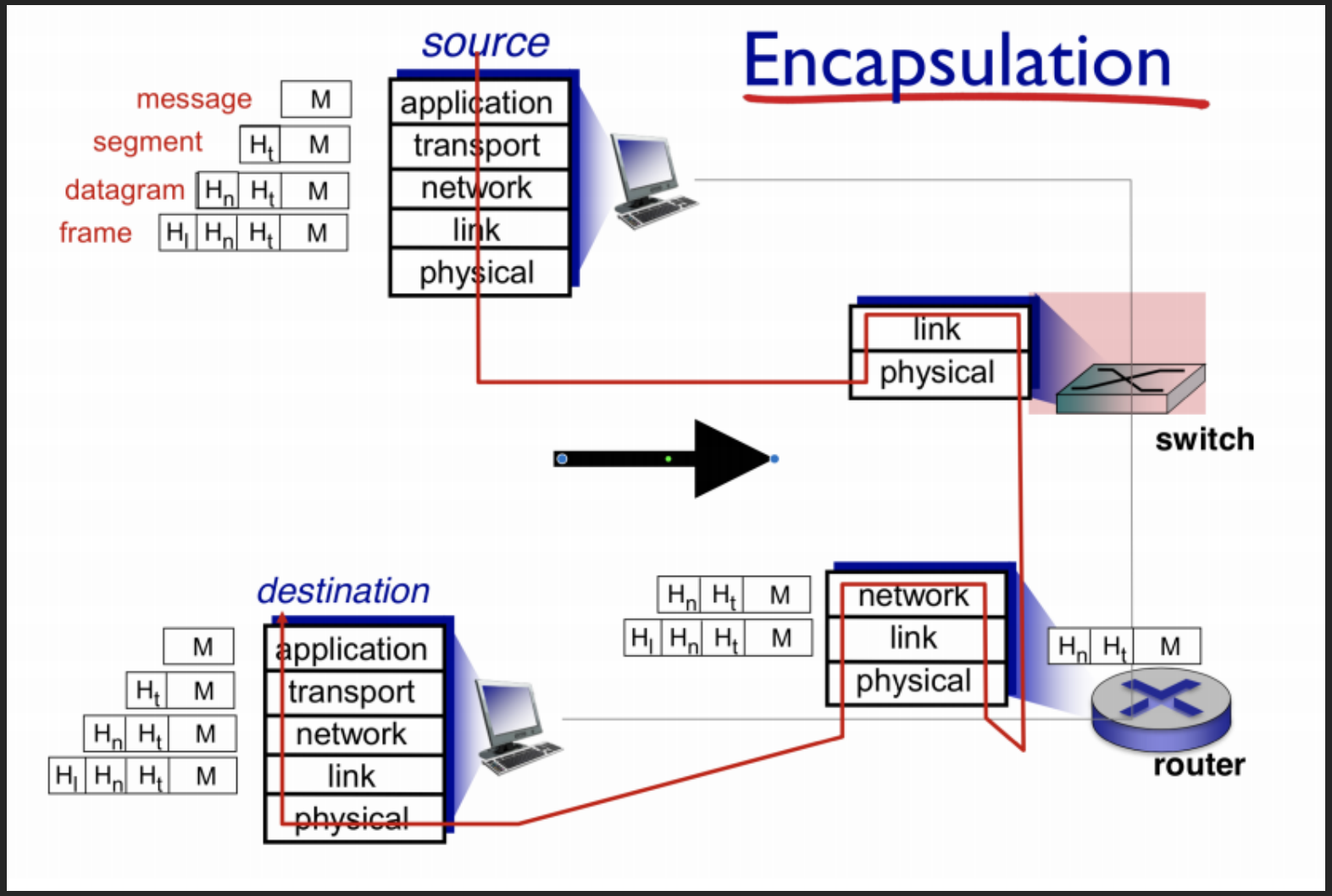
계층

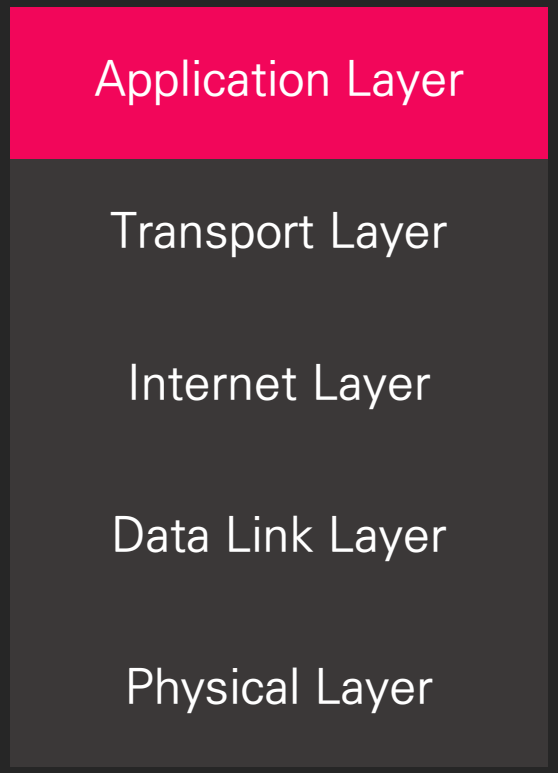
- Encapsulation
  - 각 단계에서 담아야 할 새로운 정보들을 기존 패킷에 추가하고(감싸는 것과 비슷) 하위 계층으로 넘겨준다.
- Decapsulation
  - 각 단계에서 필요한 정보만 추출하고 패킷에서 해당 정보를 삭제한 다음(벗겨내는 것과 비슷) 상위 계층으로 넘겨준다.
- 각 계층은 다른 계층에서 일어나는 일을 신경쓰지 않아도 되므로 개발에 필요한 로드가 줄어든다.


Data
UDP header
UDP data
IP header
IP data
Frame header
Frame data
Frame footer
Application
Transport
Internet
Link

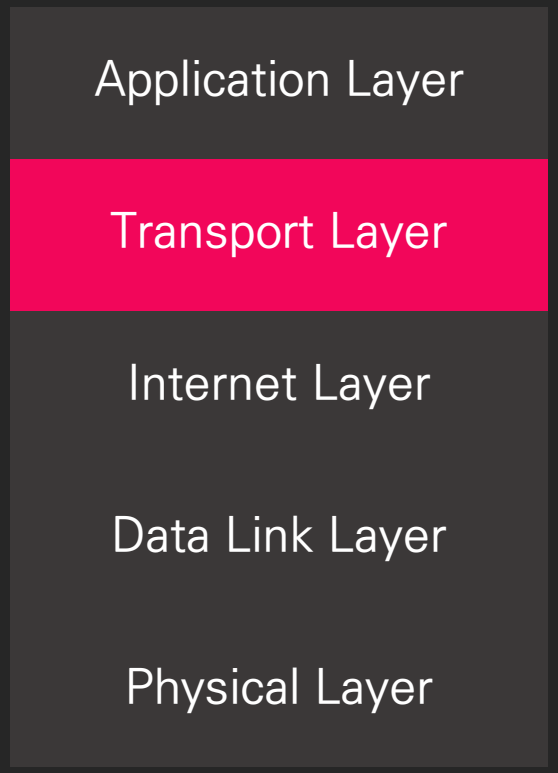
WireShark로 TCP 패킷을 까보자!

유선상어


Encapsulation
source
message
segment
datagram
frame
M
H_t M
H_n H_t M
H_l H_n H_t M
application
transport
network
link
physical
link
physical
switch
destination
application
transport
network
link
physical
network
link
physical
router



- Application Layer
  - 사용자와 가장 가까운 계층
  - 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



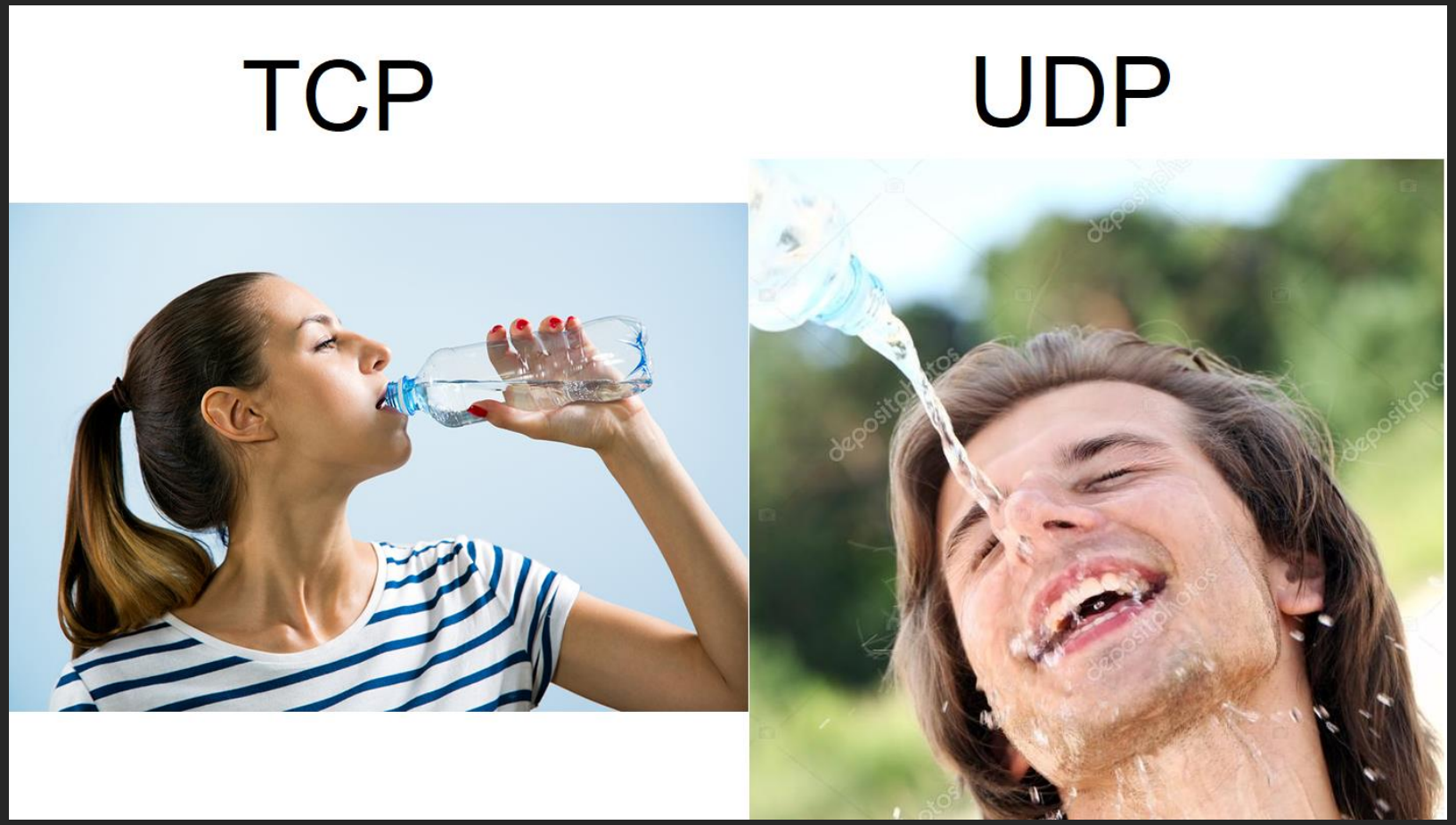
- Transport Layer
  - 두 호스트 간의 통신 방식을 정의
  - TCP / UDP가 여기에 속함
  - TCP와 UDP의 포트는 프로세스에 할당됨
  - 포트의 범위는 0 ~ 65535

포트

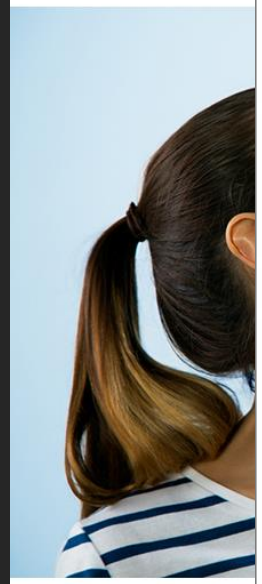
- netstat -tulpn

```
nenv* 〉  sudo netstat -tulpn
[sudo] password for ubuntu:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      11238/nginx: master
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      13124/systemd-resol
tcp        0      0 127.0.0.1:3000          0.0.0.0:*               LISTEN      3689/node
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      11238/nginx: master
tcp        0      0 0.0.0.0:37089           0.0.0.0:*               LISTEN      25001/sshd
tcp        0      0 127.0.0.1:27017         0.0.0.0:*               LISTEN      28162/mongod
tcp6       0      0 :::3021                 :::*                    LISTEN      19773/node /home/ub
tcp6       0      0 :::110                  :::*                    LISTEN      31996/docker-proxy
tcp6       0      0 :::143                  :::*                    LISTEN      31983/docker-proxy
tcp6       0      0 :::80                   :::*                    LISTEN      11238/nginx: master
tcp6       0      0 :::25                   :::*                    LISTEN      32008/docker-proxy
tcp6       0      0 :::3001                 :::*                    LISTEN      3970/docker-proxy
tcp6       0      0 :::3002                 :::*                    LISTEN      30762/node
tcp6       0      0 :::443                  :::*                    LISTEN      11238/nginx: master
tcp6       0      0 :::25565                :::*                    LISTEN      31755/java
```

TCP vs UDP



TCP vs UD



Kirk Bater
@KirkBater

Follow

This image is a TCP/IP Joke. This tweet is a UDP joke. I don't care if you get it.

Thread
iamkirkbater and jkjustjoshing

iamkirkbater Aug 23rd, 2017 at 9:37 AM
in #www

Do you want to hear a joke about TCP/IP?

7

7 replies

jkjustjoshing 5 months ago
Yes, I'd like to hear a joke about TCP/IP

iamkirkbater 5 months ago
Are you ready to hear the joke about TCP/IP?

jkjustjoshing 5 months ago
I am ready to hear the joke about TCP/IP

iamkirkbater 5 months ago
Here is a joke about TCP/IP.

iamkirkbater 5 months ago
Did you receive the joke about TCP/IP?

jkjustjoshing 5 months ago
I have received the joke about TCP/IP.

iamkirkbater 5 months ago
Excellent. You have received the joke about TCP/IP. Goodbye.

TCP

- 가끔씩 생길 수 있는 문제
  - 내 택배가 옥천 TML에서 사라진다. → packet loss
  - 내 택배가 던져지는 과정에서 손상된다. → bit error

- 어떻게 해결하지?
  - bit error: ACK / NACK
    - Acknowledge: 인정, Negative Acknowledge: 노 인정 (using checksum)
  - packet loss: 기다려도 안 오면 다시 보내주세요

TCP

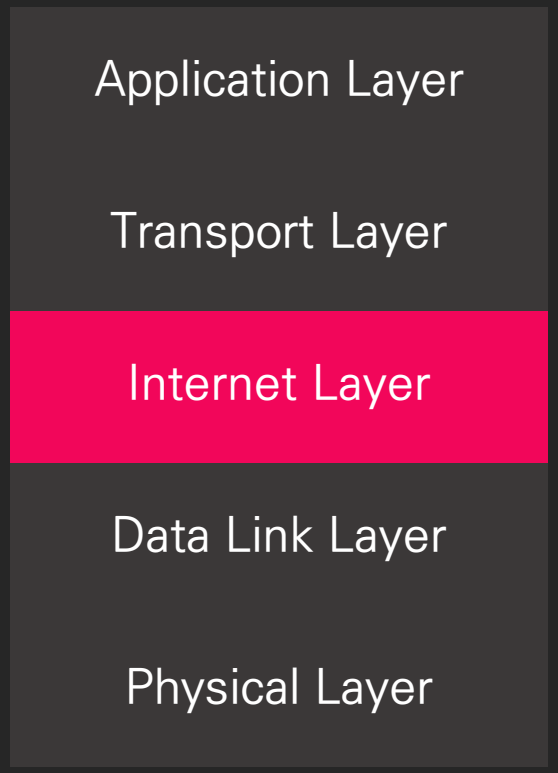
- 그 외
  - 연결 수립: 3-way Handshake
  - 회선이 혼잡해지는 걸 막기 위해: Congestion Control
  - 등등...

그러면 UDP는?

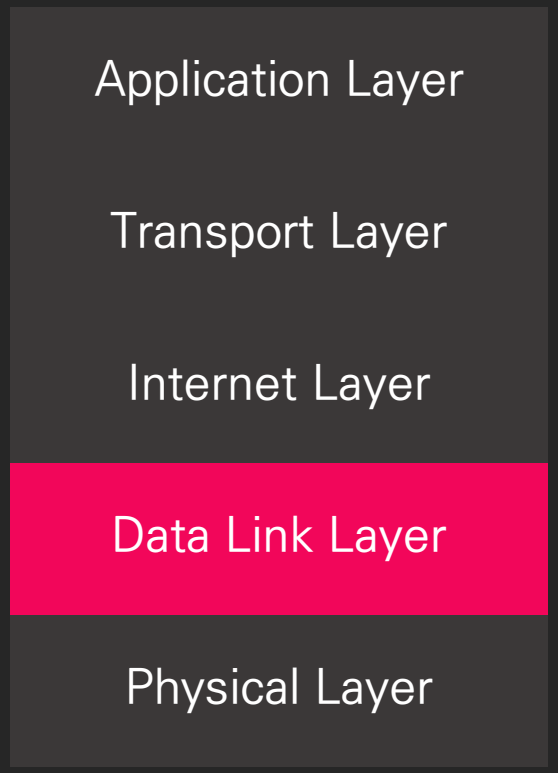
- TCP와 같이 높은 신뢰성이 목표가 아님.
  - 오류 검출은 되지만, 다시 보내는 등의 신뢰장치 없음
  - 패킷 도착 했을 때 순서가 바뀔 수도 있음

UDP

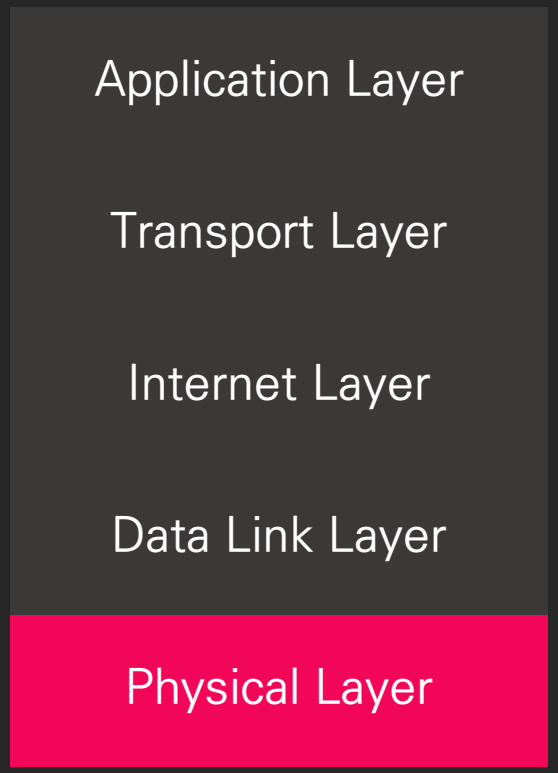
- 그럼 대체 왜 쓰나요?
  - 대신 속도가 빠르고, 오버헤드가 적다.
    - 중간에 잃어버려도 좋으니 빠른 걸 목표로 할 때
  - 게임에서 사용한 스킬이 바로 나가기 위해서
    - (UDP에 ACK, NACK를 따로 구현해서 쓰는 경우도 있음)
  - 음성통화를 하기 위해서
    - 가끔씩 로봇 목소리가 들린다 = packet loss가 있다



- Internet Layer
  - 패킷을 어떻게 보낼 것인지 결정
  - IPv4, IPv6
  - 라우터는 IP (+ 추가적인 정보)를 기반으로 패킷을 라우팅한다



- Data Link Layer
  - 네트워크 상의 기기들을 위한 규약
  - Ethernet이 대표적
  - MAC 주소를 할당
  - 오류 감지 기능을 제공



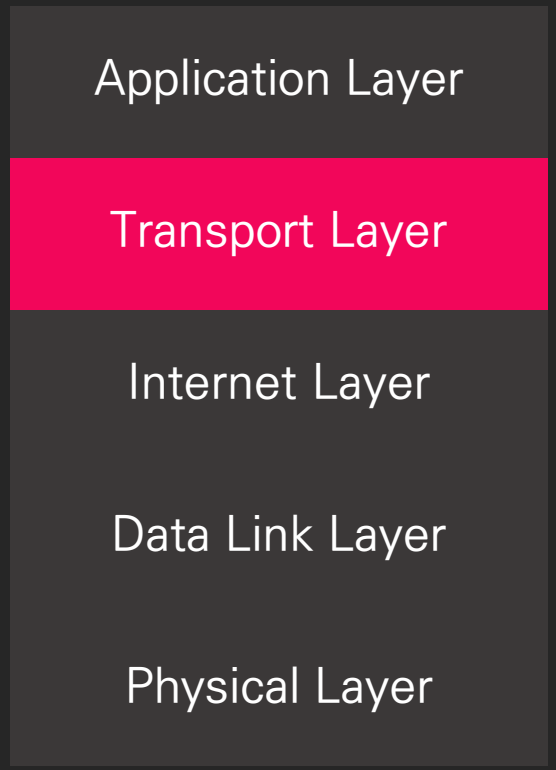
- Physical Layer
  - 실제 물리적인 신호를 정의

구글에 요청을 쏴보자

- **Application Layer**
- Transport Layer
- Internet Layer
- Data Link Layer
- Physical Layer

크롬을 켜고 HTTP로 http://google.com 에 요청을 날린다.
(Hyper Text Transfer **Protocol**)

구글에 요청을 쏴보자



이 요청은 인터넷 상으로 전달되기 위해서 TCP 패킷으로 만들어진다.

구글에 요청을 쏴보자

Application Layer

Transport Layer

Internet Layer

Data Link Layer

Physical Layer

도착지와 출발지 정보를 추가해서 IP 패킷으로 만든다.

구글에 요청을 쏴보자

Application Layer

Transport Layer

Internet Layer

Data Link Layer

Physical Layer

공유기, 이더넷 카드, 랜선 등을 거쳐서 인터넷으로 보내진다.

여러 장치의 도움으로 IP주소를 통해 원하는 곳으로 데이터가 간다.

구글이 요청을 받았다

Application Layer

Transport Layer

Internet Layer

Data Link Layer

Physical Layer

라우터, 스위치, 공유기, 이더넷 카드, 랜선 등을 거쳐서 구글 서버에 데이터가 도착한다

구글이 요청을 받았다

Application Layer

Transport Layer

Internet Layer

Data Link Layer

Physical Layer

메시지에 오류가 있는지 검출해낸다.

구글이 요청을 받았다

Application Layer

Transport Layer

Internet Layer

Data Link Layer

Physical Layer

IP 패킷을 분해해서 누구에게서 온 패킷인지 확인한다.

구글이 요청을 받았다

Application Layer

Transport Layer

Internet Layer

Data Link Layer

Physical Layer

다시 요청하는 등을 해서 문제가 있다면 복구해낸다. 패킷을 재배열한다.

구글이 요청을 받았다

- **Application Layer**
- Transport Layer
- Internet Layer
- Data Link Layer
- Physical Layer

Nginx나 Apache같은 웹서버가 요청을 받아서 응답한다.

(이하 반복)

근데 내 요청이 어떻게 구글로 가?

Application Layer

Transport Layer

Internet Layer

Data Link Layer

Physical Layer

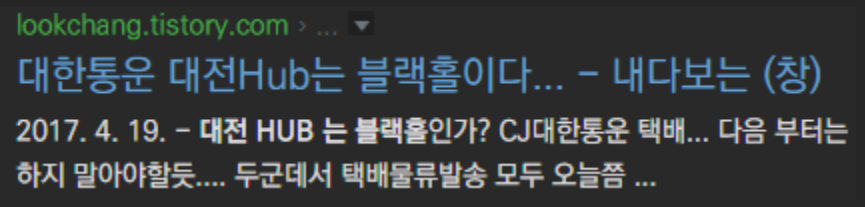
공유기, 이더넷 카드, 랜선 등을 거쳐서 인터넷으로 보내진다.

여러 장치의 도움으로 IP주소를 통해 원하는 곳으로 데이터가 간다.

여러 장치?? IP주소??

택배

- 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 우체국에 도착한다.
- 우체국에 도착한 짐은 대전 우편 집중국으로 옮겨진다.
- 또 다시 택배가 대전 HUB로 옮겨진다.
- ~~3일간 빠져나오지 못한다.~~
- 택배가 군포 HUB로 옮겨진다.
- 택배가 당동 우체국으로 옮겨진다.
- 기사님이 택배를 우리집까지 배송한다. 택배에 적힌 내 이름을 보고 내 택밴지 안다.



인터넷에서의 주소는?


What's your address?
173.168.15.10
VIA 9GAG.COM
No, your local address.
127.0.0.1
I mean your physical address.
29:01:38:62:31:58

주소 체계

- IP주소: 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 구조)
  - 국제 기구인 ICANN에서 할당받아서 사용해야함
  - IPv4: IP주소를 32비트로 나타내므로 약 42억 개의 주소를 할당 가능함
  - IPv6: IP주소를 128비트로 나타내므로 약 340간 개의 주소를 할당 가능함
    - 2^128, 340간 2823구 6692양 938자 4634해 6337경 4607조 4317억 6821만 1456

주소 체계

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

IPv4 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 마스크를 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.

IPv4 클래스

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
  - 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
```

- n는 네트워크 주소를 나타낸다.
- h는 호스트 주소를 나타낸다.
- X는 특별한 목적이 없는 자리를 나타낸다

IPv4 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 급속 도로 소진되었고 다른 해결 방법이 필요해짐.

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

CIDR

- 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 11111111 00000000

  - 앞의 24비트 = 네트워크 영역, 뒤 8비트 = 호스트 주소 영역
  - 나와 같은 서브넷에 존재하는 기기들은 143.248.35.x 의 IP를 공유

특수목적 IPv4

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

인터넷에서의 우체국, 택배허브 등은?

- 허브
  - 1계층에서 동작함 (L1)
  - 패킷이 왔을 때:
    - 모두에게 패킷을 보냄 (많이 연결되면 느려짐)

인터넷에서의 우체국, 택배허브 등은?

- 스위치
  - 우체국과 같은 역할
  - 데이터가 필요한 곳에만 전송
  - 패킷이 왔을 때:
    - 받는 사람이 모르는 MAC 주소 → 모두에게 패킷을 보냄
    - 보내는 사람이 모르는 MAC 주소 → 이 포트에는 이 MAC 주소의 장치가 연결됨을 기억함
    - 아는 MAC 주소 → 해당 포트로 패킷을 보냄

  - L2 스위치는 Data Link Layer까지 접근 가능: 간단하고 빠름
  - L3 스위치는 Network Layer까지 접근 가능: 망 분리 등의 추가적인 기능 제공 가능

주) 여기서 사용하는 포트는 선이 연결된 포트를 뜻한다.

인터넷에서의 우체국, 택배허브 등은?

- 라우터
  - 우편집중국 혹은 택배 허브와 같은 역할 (망 사이를 연결함)
  - Network Layer에서 작동하는 기기로 IP 헤더까지 뜯어볼 수 있음.
  - 분리되어 있는 여러 망 사이에서 적절한 경로를 찾아서 패킷을 전송해줌
    - IP를 보고 패킷이 어느 포트로 나갈지 결정

읽어보면 좋은 만화

  - 아까 L3 스위치랑 뭔 차이에요?
    - 보통 스위치로 망을 만들고 라우터로 망 사이를 연결하는 걸로 알고는 있는데...

주요 특징	Classical Router	Layer 3 Switch
주요 수행 OSI Layer	Layer 3	Layer 3
Routing 수행 방법	Software (CPU + Software)	Hardware (ASIC chip)
지원하는 Layer 2 MAC	Ethernet, TokerRing, FDDI ATM, WAN	FastEthernet Gigabit Ethernet
forwarding performance	Slow (CPU성능과 가격에 따라 다름)	Fast (near wire speed)
Latency	약 200 ms	< 10 ms (100 Mbps)
관리 및 program 가능성	매우 높음	적음
지원 Protocol	All	IP (일부 IPX)
Routing Protocol	All	RIP1,2 OSPF (일부 DVMRP)
WAN 지원	지원함	지원하지 않음
비용	높음	낮음

공유기는 또 뭐에요?

- 평소에 집에서 쓰는 공유기는 위의 저것과 뭔 차이일까요?
  - 몇 개 빼고는 차이가 없어요
  - 근데 주로 집에서 쓰는 공유기는 1개의 IP를 여러 개의 장치에서 공유하는 목적이 긴 하죠

  - 집에서 쓰는 공유기
    - 스위치
    - ISP(KT, SKB, U+, ...)의 망과 연결하는 라우터
    - NAT 기능

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 효과 적으로 사용한 셈이다.

  - TCP / UDP의 체크섬을 다시 계산해야하므로 성능에 영향을 준다

포트 포워딩

- 아까 NAT에서 특정 포트는 특정 IP로만 보내도록 설정

- 예시
  - 우리 집 공유기에서 80번 포트는 집 서버로만 연결시키게 설정

Host IP

- 로컬 네트워크에서 IP를 설정하는 방법은 Static IP Address와 Dynamic IP Address가 있다.
- Static IP Address는 호스트가 자신이 사용할 IP 주소를 미리 정해놓는 방식. 서로 다른 호스트의 IP 주소가 겹칠 위험이 있다.
  - 네트워크 변경이 필요하지 않고 다른 디바이스에서 원격으로 접속하는 경우
  - ex) 서버

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

그래서 실제로 추적해보자

```
nenw* \ tracert google.com

최대 30홉 이상의
google.com [216.58.197.206](으)로 가는 경로 추적:

  1    <1 ms    <1 ms    <1 ms  192.168.25.1
  2     6 ms     4 ms     3 ms  39.119.247.1
  3     2 ms     2 ms     1 ms  100.90.26.117
  4     2 ms    <1 ms    <1 ms  100.90.3.145
  5    <1 ms    <1 ms    <1 ms  10.101.1.56
  6     5 ms     6 ms     6 ms  10.222.10.200
  7     *        3 ms     3 ms  10.222.16.187
  8    48 ms    58 ms    37 ms  72.14.215.199
  9    35 ms    35 ms    35 ms  108.170.242.161
 10    38 ms    38 ms    36 ms  72.14.233.221
 11    37 ms    37 ms    37 ms  nrt13s48-in-f14.1e100.net [216.58.197.206]

추적을 완료했습니다.
```

더 알아보면 좋을 것들

- 자율 시스템
- 라우팅 프로토콜
- 기타 IP 네트워크 관련 프로토콜
  - ICMP, SNMP 등

Network - Appendix

nenw

@ SPARCS 20' Summer Wheel Seminar

Table-of-Contents

방화벽

Firewall

방화벽

- 여러 규칙을 통해 데이터를 허용/거부 등
  - iptables
  - ufw

ufw

- 사용하기 쉬운 방화벽
  - 주의! 이거 설정 잘못할 경우 여러분의 서버를 여러분의 집에서 접속하는게 불가능해집니다.
  - 기본 규칙: 아웃바운드 전부 허용, 인바운드 전부 거부

ufw

- 명령어
  - 활성화
    - $ sudo ufw enable
  - 현재 상태 보기
    - $ sudo ufw status verbose
  - 포트 허용
    - $ sudo ufw allow 22/tcp
  - 특정 IP대역에서 포트 허용
    - $ sudo ufw allow from 192.168.0.0/24 to any port 22
  - 규칙 삭제
    - $ sudo ufw status numbered
    - $ sudo ufw delete (숫자)

ufw

- 앱
  - /etc/ufw/applications.d 에 있음
  - 새로 만들 수도 있음
  - 저는 그냥 방화벽 설정을 보기 좋게 관리하려고 사용 중

  - `$ sudo ufw allow "OpenSSH"`
  - `$ sudo ufw allow from 121.152.53.34 to any app OpenSSH`

HTTP

Hyper Text Transfer Protocol

HTTP

- TCP 기반 프로토콜
- 인간이 읽을 수 있는 프로토콜
- Stateless

Request

```
POST /aaaa HTTP/1.1
Host: nenw.dev
User-Agent: MyAgent/1.0
...

asdfasdfasdfasdfasdf
```

Method, Path, Protocol Version

Header

Body

Response

HTTP/1.1 200 OK — Protocol Version, Status

Content-Length: 12345 — Header

...

asdfasdfasdfasdfasdf — Body

주요 Request 헤더

- Accept-*
- Authorization
- Cookie
- Connection
- Host
- Range
- Referer
- User-Agent
- X-Forwarded-*

주요 Response 헤더

- Access-Control-*
- Content-Security-Policy
- Set-Cookie

주요 메서드

- GET
- POST
- PATCH
- PUT
- DELETE
- OPTIONS

nginx

an HTTP and reverse proxy server, ...

Apache

- HTTP 웹 서버용 소프트웨어
- 확장성이 좋음 (= 설정이 복잡함)
- 가장 많이 사용되는 웹 서버, 전통 강호, 하락세

Apache

- MPM 방식 (Multi Processing Module) 방식으로 요청을 처리
  - 새로운 요청이 들어올 때마다 복제하는 방식

- 프로세스 복제할 시 메모리까지 복사
  - 요청이 많아지면 메모리 사용량도 많아짐

Nginx

- Apache와 비교했을 때 높은 트래픽, 정적 파일 서빙에 강함
  - 약 4배의 요청을 처리 가능, 메모리도 적게 먹음
- 간편한 설정
  - .htaccess 파일 설정 필요 없음
- AWS 내 점유율 50% 이상, 신흥 강자, 상승세

Nginx

- Event-driven 비동기 방식(Asynchronous)으로 동작
- Single-threaded(worker 프로세스)

Nginx 설정

- /etc/nginx/ 폴더에 위치

- nginx.conf: 메인 파일
- conf.d/: 추가 설정 파일들을 넣는 폴더
- sites-available/: 사이트의 설정 파일들이 위치함
- sites-enabled/: 사이트의 설정 중에 현재 사용하는, 활성화된 설정 파일들이 위치함
  - sites-available/에 설정을 만들고 심볼릭링크로 sites-enabled/에 추가하는 식으로 사용
- snippets/: 주로 사용하는 내용을 저장해두고, include 디렉티브로 가져옴

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

블록

디렉티브

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 여러 개를 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.

- event 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

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

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

- upstream 블록
  - Fast CGI나 Reverse Proxy 등에서 로드 밸런싱을 하기 위해 사용한다. 예를 들어, 80%의 요청은 A서버로 처리하고 나머지 20%의 요청은 B서버로 처리하며, 3번 이상 응답이 없으면 죽은 걸로 취급하고 쓰지 않는다 같은 설정을 이로 한다.

Server 블록

- 디렉티브
  - server_name: 이 서버의 주소, 여러개 명시나 정규식 명시 등 가능
    - server_name: nenw.dev *.nenw.dev;
  - listen: 어떤 포트에서 들을지
    - listen 80; # HTTP
    - listen 3000 http2; # Custom Port(3000), HTTP/2
    - listen [::]:443 ssl; # HTTPS, IPv6
    - listen [::]:80 default_server; # IPv6, 기본 서버
      - 다른 server_name 이 전부 충족하지 않을 경우 이 서버 사용, 1개 서버만 default_server 명시 가능

Location 블록

- 웹 상의 특정 위치에 적용하는 내용
  - /api의 내용에 접근하면 api 서버로 보내고, 아닐 경우 정적 파일 서빙하는 식의 설정
  - 우선순위가 있다!
- 일반 선언 (~~로 시작하는 주소)
  - location /asdf { }
- 주소와 정확하게 일치
  - location = /asdf { }
- 정규표현식과 일치 (매치된 그룹은 $1와 같이 사용 가능)
  - location ~ (.*) { }

기타 디렉티브

- include
- return
- rewrite
- client_max_body_size
- log_not_found

- 등등...

Nginx의 기능들

- Static Serving
- FastCGI
- Reverse Proxy

- 그 외에도 Load Balancing 등 알아두면 좋은 기능이 많음

Static Serving

- 말 그대로 정적 파일을 서빙함
- 이미지, html, js, css 등 요청에 맞게 동적으로 생성해내는 것이 아닌 데이터를 서빙

- 주로 사용하는 디렉티브
  - root
    - 이 디렉토리 안에서 파일을 제공
  - try_files [파일1] [파일2] ...
    - 파일1을 시도, 실패하면 파일2, 실패하면 ...
  - error_page
    - 오류가 났을 때 제공할 파일

Static Serving 예제

```
$ cat /etc/nginx/sites-available/example.com
server {
        listen 80;
        listen [::]:80;

        server_name example.com;

        root /usr/share/nginx/html;

        location / {
                try_files $uri $uri/ =404;
                error_page 404 = /404.html;
        }
}
```

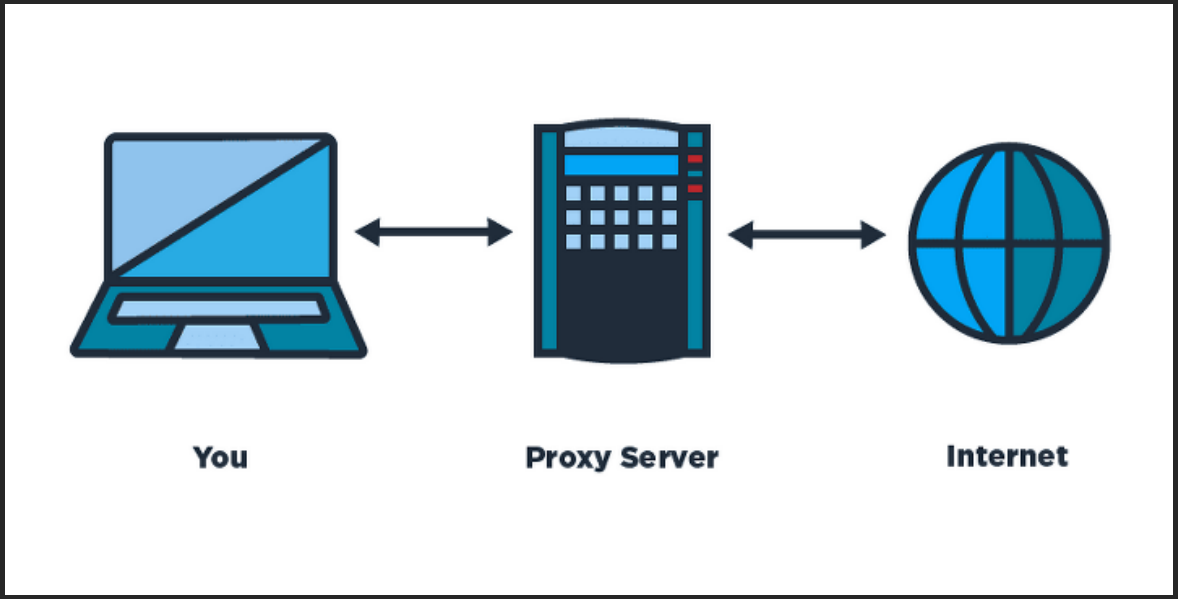
Fast CGI

- 다른 프로그램과 웹 서버와 통신하기 위한 프로토콜
- 요청이 들어왔을 때 프로세스에 전해주고, 결과를 다시 보내주는 등

- 주로 php 등에서 쓰는데 저희가 그걸 안 써서 패스할게요

Reverse Proxy

- Proxy: 대행자
  - Proxy Server를 통해서 요청하면 나 대신 그 서버가 요청을 해줌



Reverse Proxy

- Reverse Proxy: 그 반대
  - 요청을 보낼 때 Proxy를 사용한다면, Reverse Proxy는 요청을 받을 때 사용함
  - 그러니까...
    - 사용자가 nginx로 요청을 보내면 nginx는 그걸 받아서 다른 서버에 전달해줌

  - 예제)
    - aaa.nenw.dev로 오는 요청은 nginx가 80번에서 받은 후,
      localhost:3000 에서 도는 서버가 처리
    - bbb.nenw.dev로 오는 요청도 nginx가 80번에서 받은 후,
      localhost:3001 에서 도는 서버가 처리

Reverse Proxy

- 활용)
  - static file은 nginx가 직접 처리해주고, 나머지 요청은 서버에 보내기
  - 80번 포트로 오는 요청을 nginx를 통해 다른 포트로 분배해주기
  - 등등...

Reverse Proxy

- 디렉티브:
  - proxy_pass
    - 다른 서버로 넘겨줌
  - proxy_redirect
    - 서버가 Location 헤더 등을 사용해서 리다이렉트 할 때, 넘겨준 서버의 주소로 리다이렉션시킬 경우 그걸 다시 다른 주소로 바꾸는 기능
    - 보통 proxy_redirect off; 하면 된다.
  - proxy_set_header
    - 전달해줄 때 헤더를 변경하거나 새로 추가함
    - 보통 X-Forwarded-For 같은 여러 헤더를 추가해 원래 정보를 같이 전달해준다.

Reverse Proxy 예제

```
$ cat /etc/nginx/snippets/proxy.conf
# Reverse Proxy 사용할 때 include 후 proxy_pass
proxy_set_header Host                   $http_host;
proxy_set_header X-Real-IP              $remote_addr;
proxy_set_header X-Forwarded-For        $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto      $scheme;
proxy_set_header X-NginX-Proxy          true;

proxy_redirect off;
```

Reverse Proxy 예제

```
$ cat /etc/snippets/proxy_ws.conf
# WebSocket을 사용할 때 proxy.conf와 같이 include
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "Upgrade";
proxy_set_header Origin "";
```

Reverse Proxy 예제

```
$ cat /etc/nginx/sites-available/example.com

server {
        listen 80;
        listen [::]:80;
        server_name myserver.example.com;

        location / {
                include snippets/proxy.conf;
                include snippets/proxy_ws.conf;
                proxy_pass http://127.0.0.1:3000/;
        }
}
```

그 외

- gzip을 이용해서 압축해서 전송할 수 있음

그 외: https

- 디렉티브
  - ssl_certificate
  - ssl_certificate_key

실습

지금까지 휠 세미나에서 배운 개념들 중 여러 개를 같이 이용해서 실습하겠습니다

실습

- [nickname].1e-9.space
  - / → /usr/share/nginx/static/
  - /dist → /usr/share/nginx/dist/
  - on 404 → /usr/share/nginx/static/error.html
- blog.[nickname].1e-9.space
  - reverse-proxy to docker
- Automatically redirect http to https
- TLS using letsencrypt

실습 세팅

- nginx 설치
- static 파일들 만들기
- 미리 docker 블로그 띄워두기
- letsencrypt 인증서 발급받기
  - crontab으로 자동갱신

실습 세팅: nginx

- `$ sudo apt install nginx`
- `$ sudo systemctl start nginx`

- 서버 설정이 바뀌었을 때는
  - `$ sudo nginx -s reload`

실습 세팅: static 파일

- 슬랙에서 zip 파일 받은 후 압축 풀기
- scp로 홈 디렉토리와 같은 위치에 넣은 후 /usr/share/nginx/에 파일 복사

실습 세팅: static 파일

```
/usr/share/nginx/
 ├ static
 │  ├ index.html
 │  └ error.html
 └ dist
    ├ files
    │  └ ...
    └ wheelseminar.bundle.css
    └ ...
```

실습 세팅: 도커

- 사용할 이미지: ghost:3-alpine
  - https://hub.docker.com/_/ghost 를 참고해서 docker-compose 구성
  - url 은 https://blog.[nickname].1e-9.space 로

실습 세팅: letsencrypt

- 크론탭에 1주일 간격으로
  - certbot renew --renew-hook="sudo nginx -s reload" 걸어두기

실습

1. nginx 설치하기
2. static 파일 옮기기
3. docker 띄우기
    - docker-compose up -d

실습

4. nginx 설정 만들기
   - /etc/nginx/sites-available/blog.[nickname].1e-9.space
   - /etc/nginx/sites-available/[nickname].1e-9.space
   - ln -s 이용해 심볼릭 링크 걸기
     - /etc/nginx/sites-available/[nickname].1e-9.space → /etc/nginx/sites-enabled/[nickname].1e-9.space
     - /etc/nginx/sites-available/blog.[nickname].1e-9.space → /etc/nginx/sites-enabled/blog.[nickname].1e-9.space

실습

- 할 수 있는 만큼 최대한 해보고 힘들 경우 헬프데스크에 오기

감사합니다.