네트워크-보충

학습 목표

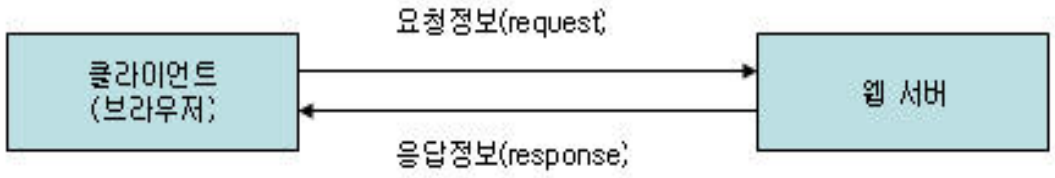
- 네트워크 유틸을 사용 할 수 있다.

네트워크 툴

1. **송수신 호스트 사이의 패킷 전달 경로를 선택**
2. **네트워크 계층 주요기능: 라우팅, 패킷의 분할과 병합**
3. **라우팅**
   A. 송수신 호스트 사이의 패킷 전달 경로를 선택하는 과정
4. **패킷의 분할과 병합**
   A. 상위 계층에서 내려온 데이터는 하위 계층인
   링크 계층의 프레임 구조에 정의된 형식으로 캡슐화 되어야 함
   B. 송신 호스트는 전송 전에 적절한 크기로 데이터를 분할
   C. 수신 호스트는 분할되어 수신한 데이터를 다시 병합

리눅스 네트워크 관리
: HTTP

- HTTP (Hyper Text Transfer Protocol)
  - WWW(World Wide Web)의 기반이 되는 프로토콜



  - HTTP 프토토콜의 작동 방식
    - 브라우저는 웹 서버에게 요청 정보를 보낸다
    - 웹 서버는 요청 정보를 분석하여 응답 정보를 브라우저에게 보낸다
    - 비연결 지향의 프로토콜

리눅스 네트워크 관리
: HTTP

1 connect

2 request

HTTP 요청 메시지

GET /gpio/query?a=led&b=90 HTTP/1.1
Host: www.bituser.com
User-agent: mozilla/4.0
Accept-language: kr

요청헤더

요청바디

3 response

HTTP 응답 메시지

HTTP/1.1 **200** OK
Date : Thu,03 jul 2003 12:00:15 GMT
Server : Apache/1.3.0 (Unix)
Last-Modified : Sun, 5 May 2003 09:23:24 GMT
Content-Length : 6821
Content-Type : text/html

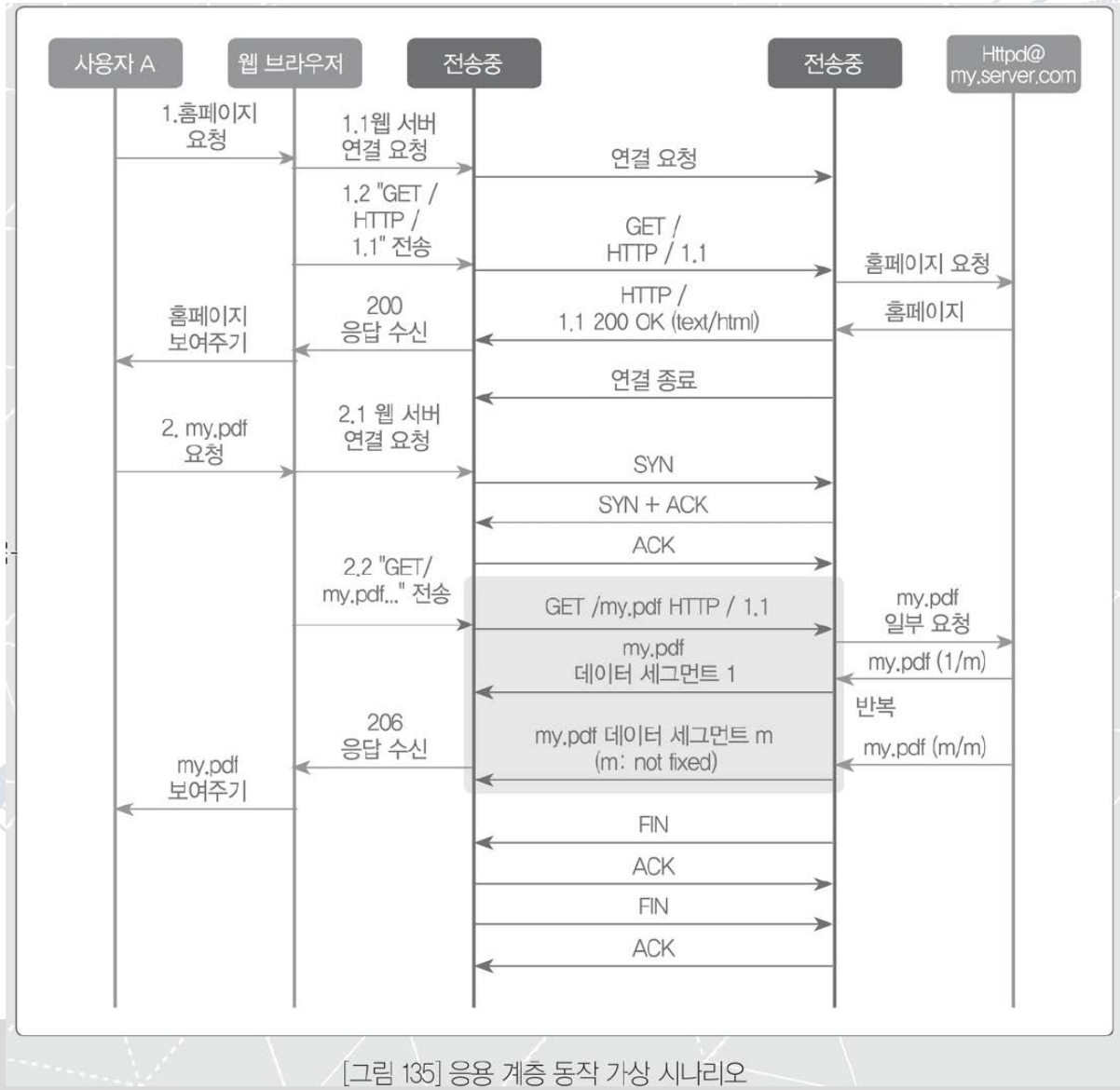
응답헤더

응답바디

4 close

리눅스 네트워크 관리
: HTTP

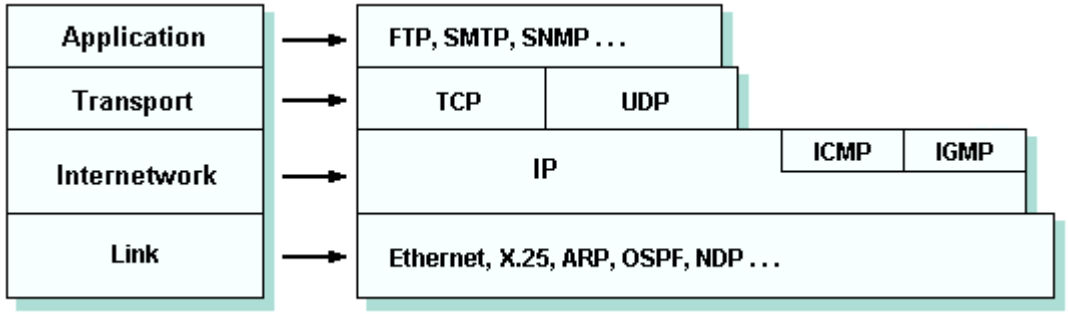
- 단계 1 의 연결 요청은 실제로는 단계 @"-'(!)을 간단하게 그린 것이며. 단계 @의 연결 종료 역시 단계 @rv@를 간단하게 그린 것이다. 단계 0'-'@. ®"'®)온 TCP 헤더 바로 뒤에 붙는 응용 계층 테이터이다 이 데이터는 KTTP 프로토콜에 따라 서 사용자 A의 컴퓨터에 있는 웹 브라우저와 my.server.com 웹 서버 프로그램이 생성하게 된다. 이제 TCP 프로토콜을 좀 더 상세하고 알아보고, 해당되는 딘계를 살펴보자특히 단계 @-v@과 같이 PDF 파일과 같이 많은 앙의 데이터가 한번에 전송되는 경우에는 여러 네트워크 경로로 IP 패킷이 전달될 수 있어 TCP 흐름 제어가 중요하며 네트워크 경로상의 호스트나 인터넷 선로에 의해 발생하는 전송 오류에 대처하기 위한 오류 제어가 중요하다. 또한 네트워크에서 단위 시간에 처리 가능한 용량(capacrty)보다 많은 패킷이 전송될 때 이를 제어하기 위한 혼잡 제어 역시 TCP 성능에 많은 영향을 미치므로 이런 상황을 이해하기 위한 노력이 필요



[그림 135] 응용 계층 동작 가상 시나리오

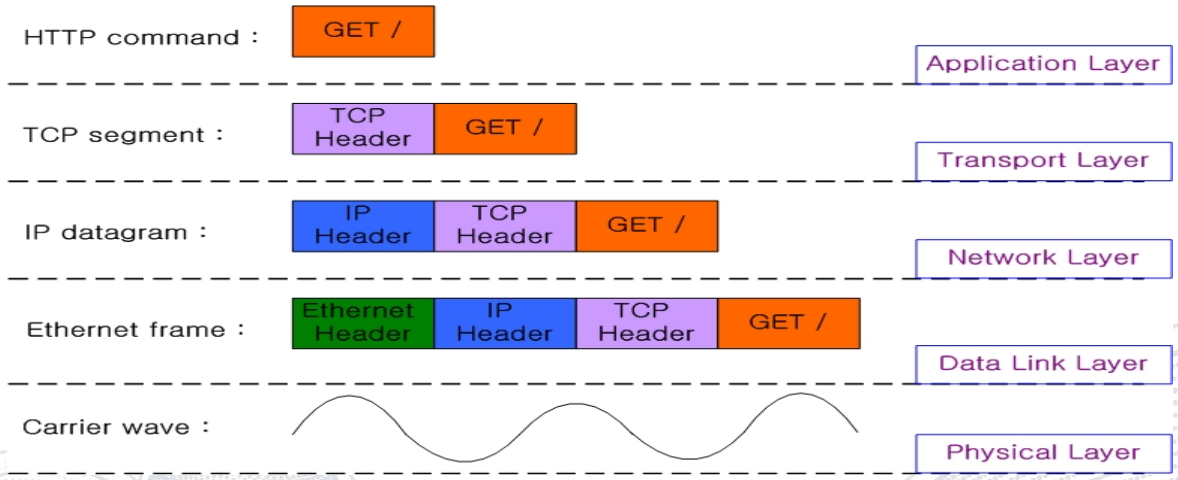
리눅스 네트워크 관리
: TCP/IP 프로토콜 스택(Stack)

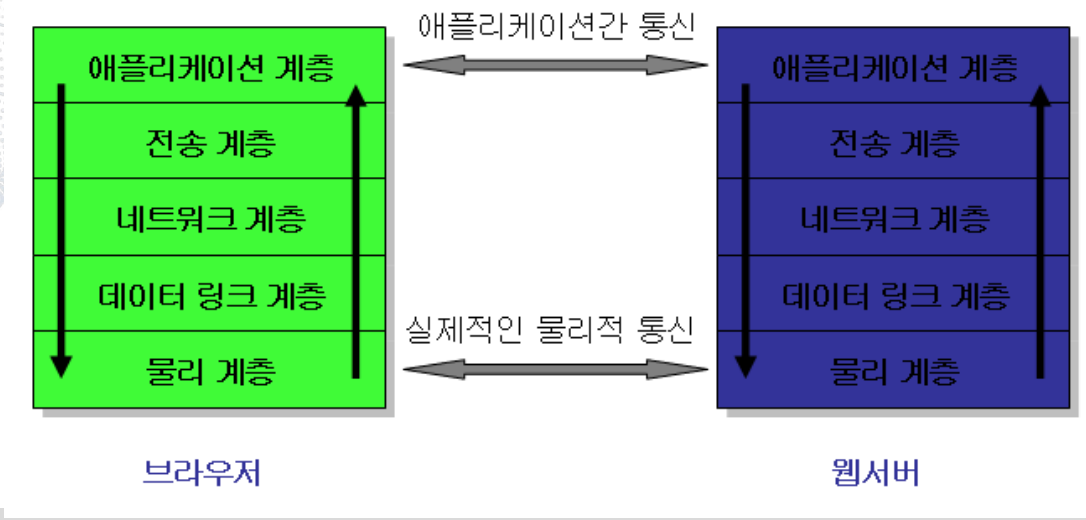
- TCP/IP 프로토콜 스택은 총 4개의 부분으로 구분
- 네트워크 데이터 통신 과정을 4개의 영역으로 계층화한 것
- 데이터 통신 과정을 하나의 덩치 큰 프로토콜로 해결하지 않고 작게 나눠서 계층화 하려는 노력의 결과



- 왜 OSI 7 계층을 모두 이용하지 않는가?
  - OSI 7 계층은 이론적인 면과 하드웨어 장비 표준을 위해 제정한 것
  - 실무에서 네트워크 프로그래밍은 90% 이상 위 프로토콜 스택을 기반으로 작업이 진행

리눅스 네트워크 관리
: 프로토콜 계층간의 데이터 캡슐화





리눅스 네트워크 관리
: 소켓의 이해

- TCP/IP 프로토콜의 프로그래머 인터페이스
- 네트워크 프로그래밍에서 개발자에게 네트워크에 접근할 수 있는 인터페이스 제공
- 1986년 BSD Unix 4.3 개정된 소켓 사용
- 프로세스 간의 통신 방식(클라이언트-서버 모델)
- 3가지 과정으로 사용
  - 소켓 생성(소켓 열기)
  - 소켓을 통한 송/수신
  - 소켓 소멸(소켓 닫기)

리눅스 네트워크 관리
: 소켓의 이해 – 소켓 생성

- **`int socket(int domain, int type, int protocol)`**

```
#include <sys/socket.h>

int sockfd = socket( PF_INET, SOCK_STREAM, IPPROTO_TCP );
```

- domain : 소켓이 사용할 프로토콜 체계(Protocol Family)
  - PF_INET : IPv4 인터넷 프로토콜 체계
  - PF_INET6 : IPv6 인터넷 프로토콜 체계
  - PF_LOCAL : 로컬 통신을 위한 UNIX 프로토콜 체계
  - PF_PACKET : Low Level 소켓을 위한 프로토콜 체계
  - PF_IPX : IPX 노벨 프로토콜 체계

리눅스 네트워크 관리
: 소켓의 이해 – 소켓 생성

- **`int socket(int domain, int type, int protocol)`**

```
#include <sys/socket.h>

int sockfd = socket( PF_INET, SOCK_STREAM, IPPROTO_TCP );
```

- type : 소켓의 유형

SOCK_STREAM	SOCK_DATAGRAM
TCP 통신 소켓	UDP 통신 소켓
Stream 방식의 연결지향 소켓 생성	Datagram 방식 비연결성 소켓 생성
양방향 통신	일방적 송신
가변길이 Byte Stream	고정 길이 메시지 사용
신뢰도 높음(3 Way Handshake)	신뢰도 낮음
전달된 순서대로 수신	전달된 순서대로 수신되지 않음

리눅스 네트워크 관리
: 소켓의 이해 – 소켓 생성

- **int socket(int domain, int type, int protocol)**

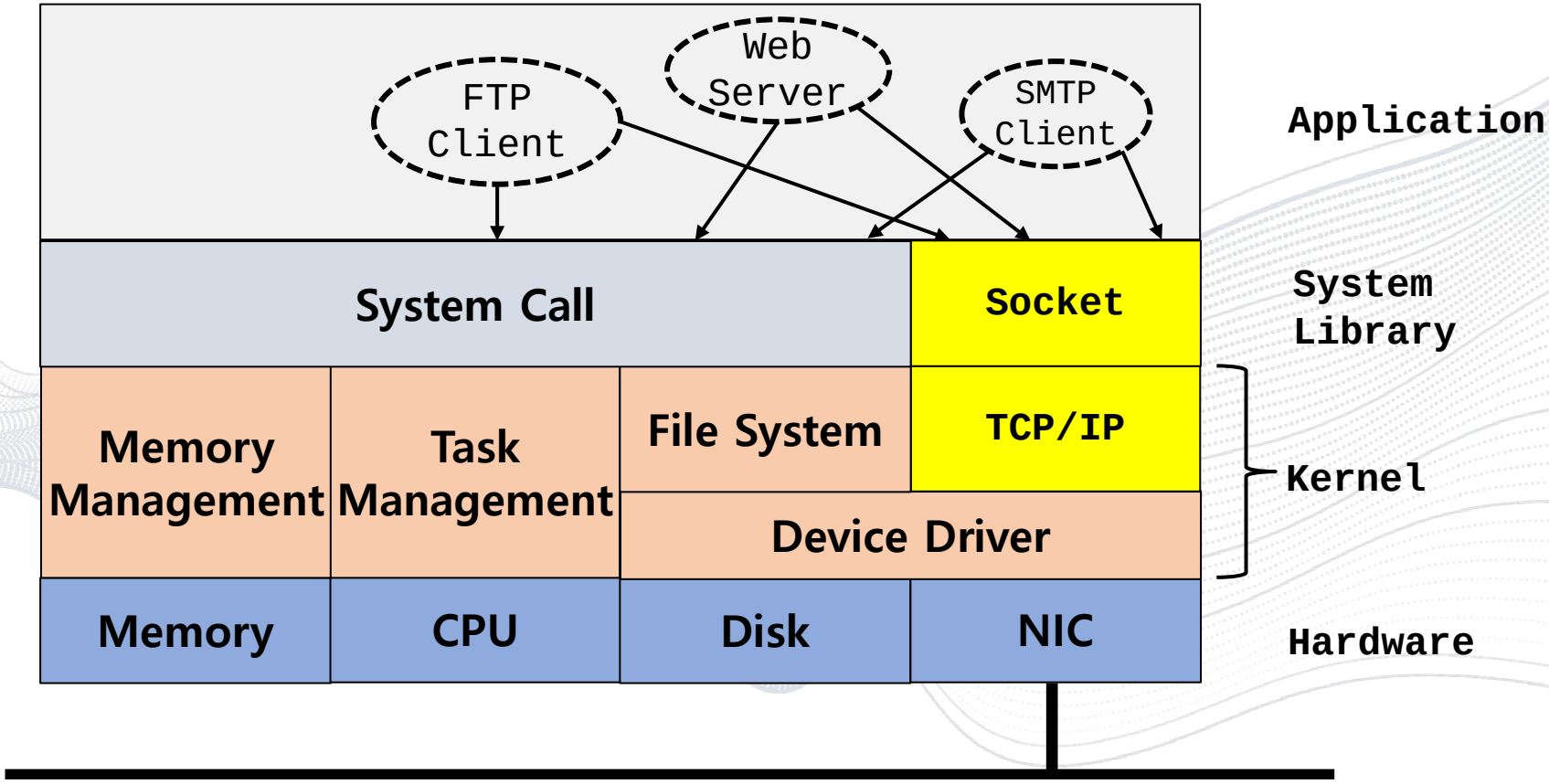
```
#include <sys/socket.h>

int sockfd = socket( PF_INET, SOCK_STREAM, IPPROTO_TCP );
```

- protocol : 프로토콜 선택
  - 지정하지 않아도 충분히 원하는 소켓을 선택할 수 있음
  - 하나의 프로토콜 체계에서 데이터 전송방식이 동일한 프로토콜이 2개 이상 존재하기 때문에 마지막 파라미터를 통해 원하는 프로토콜 정보를 조금 더 구체화
  - IPv4 인터넷 프로토콜 체계에서 연결지향형 데이터 송수신 소켓 선택
    - IPPROTO_TCP
  - IPv6 인터넷 프로토콜 체계에서 비 연결 지향 데이터 송수신 소켓 선택
    - IPPROTO_UDP

리눅스 네트워크 관리
: 소켓의 이해 - 포트번호

- OS 구조와 TCP/IP 그리고 socket의 위치



리눅스 네트워크 관리
: 소켓의 이해 - 포트번호

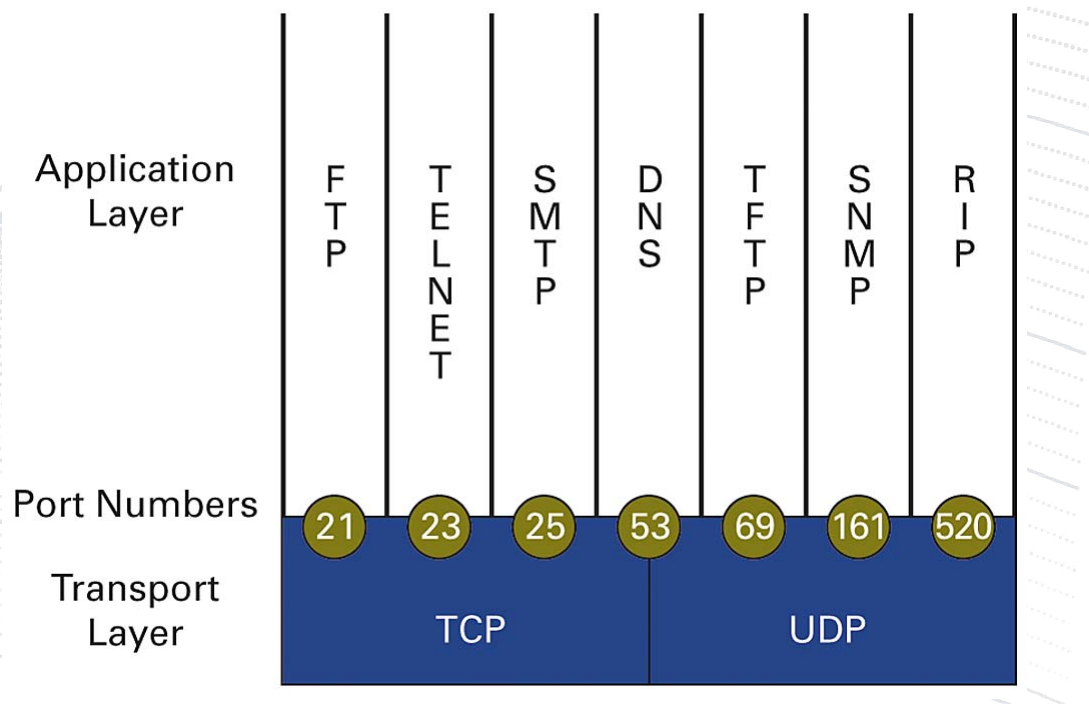
- 포트의 필요성
  - 실제적인 데이터 통신은 연결된 두 Host(컴퓨터)의 Process(프로그램) 사이에서 이루어짐
  - 여러 계층을 통해 애플리케이션 계층으로 들어온 데이터를 해당 Process에만 정확히 전달해야 할 필요가 있음
  - 하나의 Host(컴퓨터)에는 여러 개의 Process(프로그램)가 각각의 소켓을 사용하여 데이터 통신을 하고 있기 때문에 TCP에서는 각 소켓을 구분해야 할 필요가 있음
  - 이때 각각의 소켓을 구분할 때 사용하는 것이 포트
  - 쉬운 예시:

    "아파트(Host)에 사는 사람(Process)에게 편지(Data)를 보낼 때,

    동(IP Address)과 호(Port)를 봉투(Packet)에 기입해야 한다."

리눅스 네트워크 관리
: 포트 번호

- 포트 번호는 16비트 정수를 사용 (0 ~ 65535)
- 포트의 종류
  - 1 ~ 255 : 잘 알려진 인터넷 서비스 포트 (Well-Known Port)
  - 256 ~ 1023 : 그 밖의 인터넷 서비스
  - 1024 ~4999 : 시스템 예약
  - 5000 ~ 65535 : 사용자 사용
- 포트의 중복은 불가능하다
- 하지만, 같은 UDP 포트와 TCP 포트는 중복하여 사용할 수 있다



리눅스 네트워크 관리
: 포트 번호

- Well-known 포트번호

서비스	TCP 포트	서비스	TCP 포트
FTP	2(제어),20(데이터)	DNS	53
SSH	22	HTTP	TCP 80
telnet	23	POP3	110
SMTP	25	IMAP4	143

리눅스 네트워크 관리
: CIDR

- CIDR(Classless Inter-Domain Routing)
  - 개념
    - CIDR은 여러 개의 C 클래스 주소 범위를 하나의 네트워크에 결합시기기 위한 방법으로, 이 방법을 통해 라우팅 테이블의 크기를 줄이고 다양한 서브넷 형태의 IP주소를 사용할 수 있다.
  - 역할
    - A,B,C 클래스 별로 IP 주소를 구분하지 않고, 네트워크 식별자 범위를 자유롭게 지정할 수 있도록 하여 IP 주소 운영의 융통성 제공
    - 네트워크 주소를 자유롭게 설정할 수 있기 때문에 IP 주소 낭비 방지 및 효과적인 네트워크 구성 가능
    - 도메인간의 라우팅에 사용되는 IP주소를 효과적으로 관리 가능
  - 표현형태
    - IP 주소의 서브넷 부분은 임의의 길이를 가지며, 주소형식은 a.b.c.d/x이다.
    - 주소 형식에서 x는 주소의서브넷 부분이다.

리눅스 네트워크 관리
: CIDR

- CIDR(Classless Inter-Domain Routing)

[그림 124] CIDR 표현형태

리눅스 네트워크 관리
서브네트워킹

IPv4 Subnetting

Ping My Network

Subnets size

Private IPv4

Prefix	Ranges
10.0.0.0/8	10.0.0.0 – 10.255.255.255
172.16.0.0/12	172.16.0.0 – 172.31.255.255
192.168.0.0/16	192.168.0.0 – 192.168.255.255

Binary calculation

Subnet Mask		Wildcard	
255	1111 1111	0	0000 0000
254	1111 1110	1	0000 0001
252	1111 1100	3	0000 0011
248	1111 1000	7	0000 0111
240	1111 0000	15	0000 1111
224	1110 0000	31	0001 1111
192	1100 0000	63	0011 1111
128	1000 0000	127	0111 1111
0	0000 0000	255	1111 1111

IPv4 Classes

Classes	Ranges
A	0.0.0.0 – 127.255.255.255
B	128.0.0.0 - 191.255.255.255
C	192.0.0.0 - 223.255.255.255

Subnetting

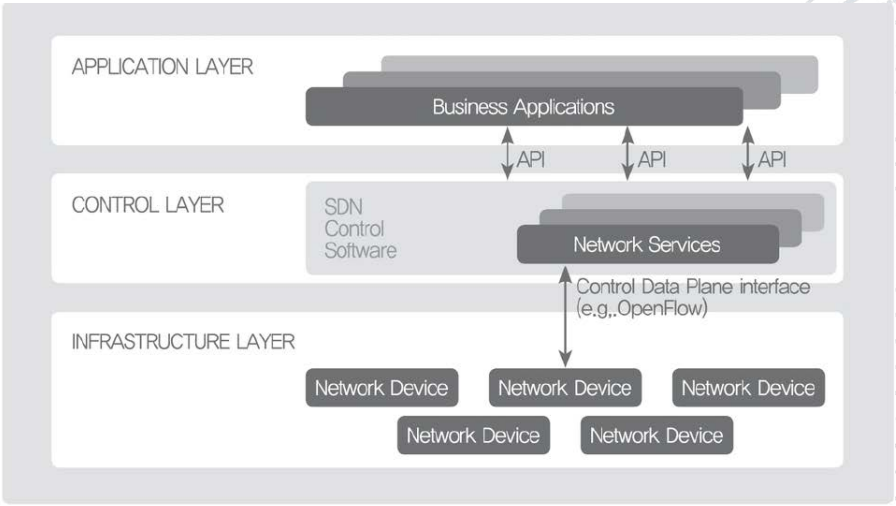
CIDR	Subnet mask	Wildcard	Adresses
/32	255.255.255.255	0.0.0.0	1
/31	255.255.255.254	0.0.0.1	2
/30	255.255.255.252	0.0.0.3	4
/29	255.255.255.248	0.0.0.7	8
/28	255.255.255.240	0.0.0.15	16
/27	255.255.255.224	0.0.0.31	32
/26	255.255.255.192	0.0.0.63	64
/25	255.255.255.128	0.0.0.127	128
/24	255.255.255.0	0.0.0.255	256
/23	255.255.254.0	0.0.1.255	512
/22	255.255.252.0	0.0.3.255	1,024
/21	255.255.248.0	0.0.7.255	2,048
/20	255.255.240.0	0.0.15.255	4,096
/19	255.255.224.0	0.0.31.255	8,192
/18	255.255.192.0	0.0.63.255	16,384
/17	255.255.128.0	0.0.127.255	32,768
/16	255.255.0.0	0.0.255.255	65,536
/15	255.254.0.0	0.1.255.255	131,072
/14	255.252.0.0	0.3.255.255	262,144
/13	255.248.0.0	0.7.255.255	524,288
/12	255.240.0.0	0.15.255.255	1,048,576
/11	255.224.0.0	0.31.255.255	2,097,152
/10	255.192.0.0	0.63.255.255	4,194,304
/9	255.128.0.0	0.127.255.255	8,388,608
/8	255.0.0.0	0.255.255.255	16,777,216
/7	254.0.0.0	1.255.255.255	33,554,432
/6	252.0.0.0	3.255.255.255	67,108,864
/5	248.0.0.0	7.255.255.255	134,217,728
/4	240.0.0.0	15.255.255.255	268,435,456
/3	224.0.0.0	31.255.255.255	536,870,912
/2	192.0.0.0	63.255.255.255	1,073,741,824
/1	128.0.0.0	127.255.255.255	2,147,483,648
/0	0.0.0.0	255.255.255.255	4,294,967,296

리눅스 네트워크 관리
: 최신 네트워크

- 소프트웨어 기반 네트워크 (Software Defined Network)
  - 기존 통신환경의 한계와 패러다임의 변화
    - 트래픽 패턴 변화, 가상화 기술 활성화, 네트워크 구조 복잡, 네트워크 관리,설계 어려움,장비 제조사 의존성 확대
  - SDN(software Defined Network)
    - 소프트웨어 프로그래밍을 통해네트워크 경로 설정과 제어 및 복잡한 운영관리 처리
    - IETF, ITU-T 등에서 표준 개발 시도, 산업체 간의 기술 표준 채택 경쟁
    - 2010년 스탠포드 대학교에서 오픈네트워크 서밋에서 제안하여 ONF(Open Networking Foundation) 결성

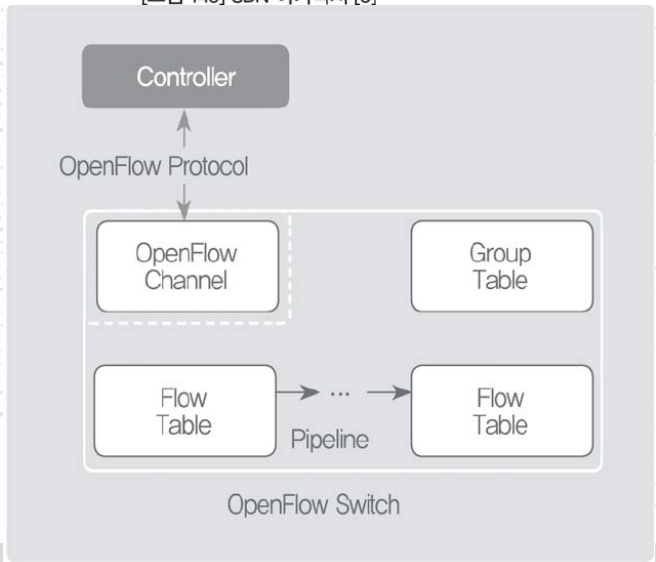
리눅스 네트워크 관리
: 최신 네트워크

- 소프트웨어 기반 네트워크 (Software Defined Network)
  - SDN(software Defined Network)
    - 개념 :제어플레인과 데이터 플레인으로 구성 Openflow같은 산업 표준 프로토콜을 사용



[그림 146] SDN 아키텍처 [5]

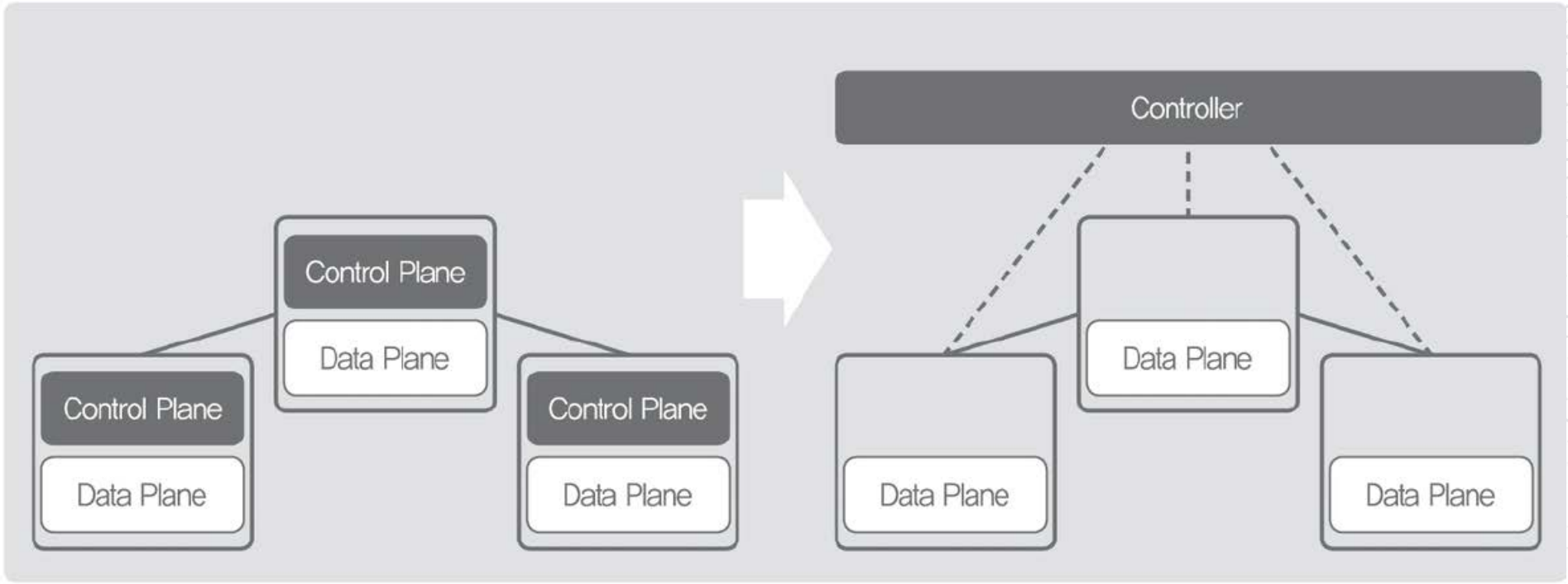
- 오플 플로우(Openflow)기술
  - 개념 : 컨트롤로와 스위치로 구성하여 패킷의 흐름을 수행



[그림 147] 오픈플로우 스위치 주요 컴포넌트 [6]

리눅스 네트워크 관리
: 최신 네트워크

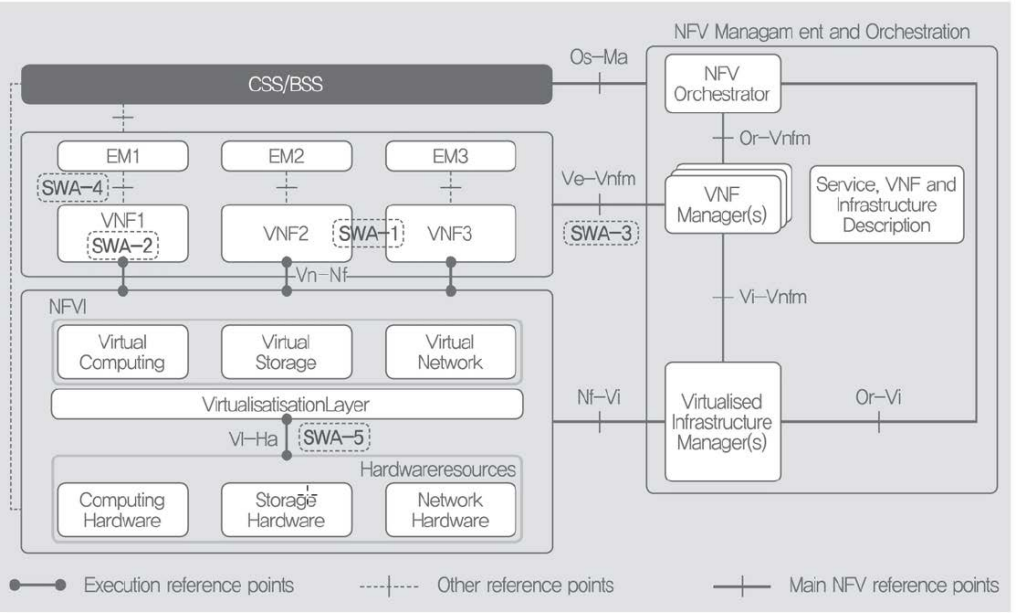
- SDN 적용 사례
  - 네트워크 장비의 구성이 콘트롤, 데이터 플레인으로 구성되어 있는데, SDN도입으로 콘트롤 플레인이 하나로 통합됨



[그림 148] SDN 적용 전/후의 네트워크 구조

리눅스 네트워크 관리
: 최신 네트워크

- 소프트웨어 기반 네트워크 (Software Defined Network)
  - NFV(Network Function Virtualization)
    - 도입배경 : 네트워크 속도 증가 및 다양한 서비스, 장비의 수명과 교체시 수익 확보에 대한 대책
    - 개념 : 고성능 x86 플랫폼에서 실행되면 사용자가 필요한 네트워크 기능을 활성화. VM 또는 서비스 프로파일 작성하여 구현



[그림 149] NFV 아키텍처 프레임워크(5개 VNF 인터페이스 타입) [7]

리눅스 네트워크 관리
: 최신 네트워크

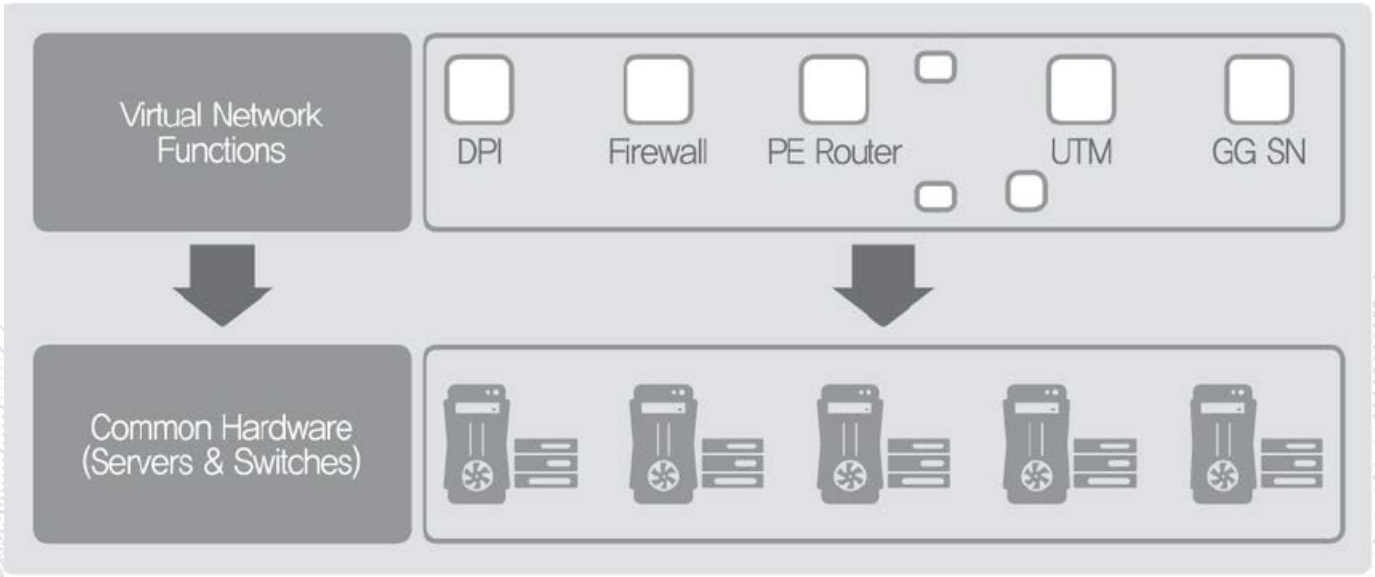
- 소프트웨어 기반 네트워크 (Software Defined Network)

〈표 81〉 NFV 아키텍처 프레임워크 구성

구분	내용
VNFs	Virtual Network Functions 여러 응용 프로그램을 지원하기 위한 네트워크 기능 집합 소프트웨어
NFVI	NFV Infrastructure 컴퓨팅, 저장소, 네트워크 기능을 지원하는 물리적 하드웨어 자원, 가상화 지원 기능 및 VNF 실행을 지원하는 기능을 제공
Management & Orchestration	하드웨어적, 소프트웨어적 자원관리, 전달, VFN 관리기능 제공

리눅스 네트워크 관리
: 최신 네트워크

- NFV 적용 사례
  - 가상화 기능을 이용한 네트워크 장비 구현 방식으로 x86서버에서 장비의 기능 구현

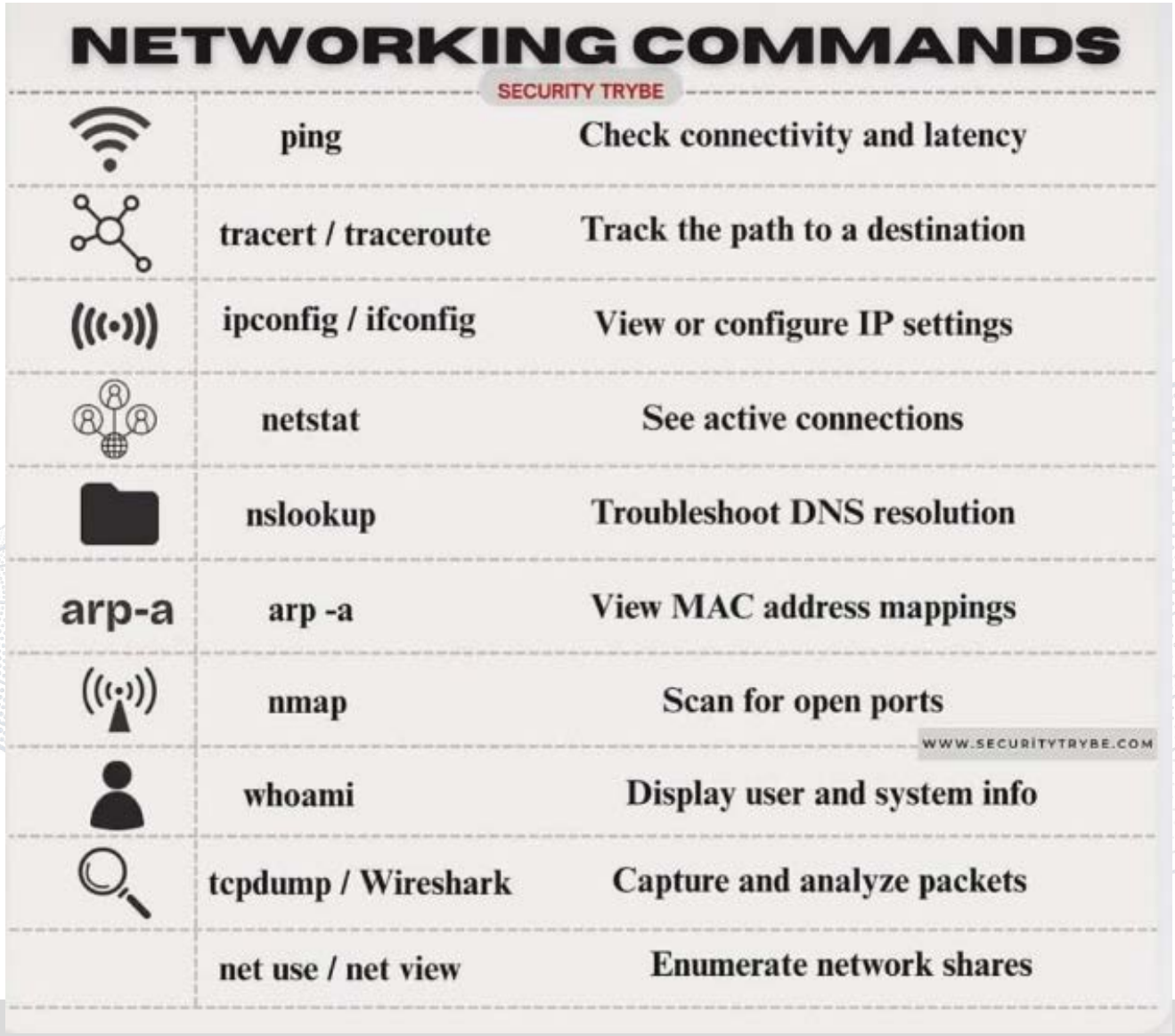


[그림 150] NFV 구현방식

리눅스 네트워크 관리
: 최신 네트워크

- 소프트웨어 기반 네트워크 (Software Defined Network)
  - 기존 통신환경의 한계와 패러다임의 변화
  - SDN(software Defined Network)
  - NFV(Network Function Virtualization)
    - 도입배경 : 네트워크 속도 증가 및 다양한 서비스, 장비의 수명과 교체시 수익 확보에 대한 대책
    - 개념 : 고성능 x86 플랫폼에서 실행되면 사용자가 필요한 네트워크 기능을 활성화. VM 또는
    - 서비스 프로파일 작성하여 구현

리눅스 네트워크 관리


NETWORKING COMMANDS
SECURITY TRYBE
ping
Check connectivity and latency
tracert / traceroute
Track the path to a destination
ipconfig / ifconfig
View or configure IP settings
netstat
See active connections
nslookup
Troubleshoot DNS resolution
arp-a
arp -a
View MAC address mappings
nmap
Scan for open ports
WWW.SECURITYTRYBE.COM
whoami
Display user and system info
tcpdump / Wireshark
Capture and analyze packets
net use / net view
Enumerate network shares

리눅스 네트워크 관리
: 원격 시스템 네트워크 동작 확인 - ping

- ping : 원격 시스템의 네트워크가 현재 동작 중인지 확인하는 명령
  - 사용법
    - ping [옵션] 호스트주소
  - 옵션
    - -s : 패킷 사이즈를 지정
    - -q : 종합 결과만 보여줌
    - -i : 지연 시간을 설정
    - -c : 보낼 패킷 수를 지정

apt install iputils-ping
apt install net-tools
apt install iproute2

리눅스 네트워크 관리
: 원격 시스템 네트워크 동작 확인 - ping

- [실습] www.google.com 에 ping으로 네트워크를 확인해 봅니다

```
[root@localhost ~]# ping www.google.com
PING www.google.com (59.18.45.34) 56(84) bytes of data.
64 bytes from cache.google.com (59.18.45.34): icmp_seq=1 ttl=57 time=74.7 ms
64 bytes from cache.google.com (59.18.45.34): icmp_seq=2 ttl=57 time=151 ms
64 bytes from cache.google.com (59.18.45.34): icmp_seq=3 ttl=57 time=187 ms
64 bytes from cache.google.com (59.18.45.34): icmp_seq=4 ttl=57 time=200 ms
64 bytes from cache.google.com (59.18.45.34): icmp_seq=5 ttl=57 time=121 ms
. . .
. . .

--- www.google.com ping statistics ---
89 packets transmitted, 85 received, 4% packet loss, time 105734ms
rtt min/avg/max/mdev = 2.892/173.967/340.652/79.367 ms
```

리눅스 네트워크 관리
: 원격 시스템 네트워크 동작 확인 - ping

- [실습] www.google.com 에 패킷 횟수를 5로 지정하여 ping을 전송해 봅니다

```
[root@localhost ~]# ping -c 5 www.google.com
PING www.google.com (59.18.45.54) 56(84) bytes of data.
64 bytes from cache.google.com (59.18.45.54): icmp_seq=1 ttl=57 time=228 ms
64 bytes from cache.google.com (59.18.45.54): icmp_seq=2 ttl=57 time=186 ms
64 bytes from cache.google.com (59.18.45.54): icmp_seq=3 ttl=57 time=222 ms

--- www.google.com ping statistics ---
5 packets transmitted, 3 received, 40% packet loss, time 4001ms
rtt min/avg/max/mdev = 186.741/212.701/228.387/18.497 ms
```

- [실습] 패킷의 크기를 지정하여 ping을 전송해 봅니다

```
[root@localhost ~]# ping -s 1000 -c 5 www.google.com
PING www.google.com (59.18.45.35) 1000(1028) bytes of data.
1008 bytes from cache.google.com (59.18.45.35): icmp_seq=1 ttl=57 time=165 ms
1008 bytes from cache.google.com (59.18.45.35): icmp_seq=2 ttl=57 time=157 ms
1008 bytes from cache.google.com (59.18.45.35): icmp_seq=4 ttl=57 time=230 ms
1008 bytes from 59.18.45.35: icmp_seq=5 ttl=57 time=187 ms

--- www.google.com ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 8396ms
rtt min/avg/max/mdev = 157.476/185.342/230.987/28.545 ms
```

리눅스 네트워크 관리
: 원격 시스템 네트워크 동작 확인 - ping

- ping 명령은 상대 호스트 또는 자신이 정상적으로 네트워크 작동을 하는지 확인하는데 유용하지만, 과도한 사용은 서버에 부담을 줄 수 있음
- 외부의 과도한 ping을 막기 위해 ICMP(Internet Control Message Protocol) echo를 ignore 시킬 수 있음
- [실습] ping 응답 설정 여부를 확인해 봅니다

```
[root@localhost ~]# cat /proc/sys/net/ipv4/icmp_echo_ignore_all
0
[root@localhost ~]#
```

- [실습] ping 응답을 막기 위해 /etc/sysctl.conf를 열고 다음을 추가해 봅니다

```
# For more information, see sysctl.conf(5) and sysctl.d(5).

net.ipv4.icmp_echo_ignore_all=1
```

- 설정을 완료하고 외부에서 ping을 시도해서 테스트 해 봅시다

```
[root@localhost ~]# sysctl -p
net.ipv4.icmp_echo_ignore_all = 1
[root@localhost ~]#
```

리눅스 네트워크 관리
: 도메인 네임서버 질의 - nslookup

- nslookup : 도메인 네임서버에 질의를 할 수 있는 명령. 도메인 이름의 호스트 IP를 검색할 수 있고 네임서버가 올바르게 작동하는지 확인 가능
  - 사용법
    - nslookup [도메인]
  - 도메인을 입력하지 않으면 대화영으로 프로그램이 작동
- [실습] 한 개의 도메인을 검색

```
[root@localhost ~]# nslookup www.naver.com
Server:                 168.126.63.1
Address:        168.126.63.1#53

Non-authoritative answer:
www.naver.com   canonical name = www.naver.com.nheos.com.
Name:           www.naver.com.nheos.com
Address: 125.209.222.142
Name:           www.naver.com.nheos.com
Address: 202.179.177.21

[root@localhost ~]#
```

리눅스 네트워크 관리
: 도메인 네임서버 질의 - nslookup

- [실습] 대화형 방식으로 여러 도메인을 질의해 봅니다

```
[root@localhost ~]# nslookup
> www.naver.com
Server:                 168.126.63.1
Address:        168.126.63.1#53

Non-authoritative answer:
www.naver.comcanonical name = www.naver.com.nheos.com.
Name:           www.naver.com.nheos.com
Address: 125.209.222.142
Name:           www.naver.com.nheos.com
Address: 202.179.177.21
> www.bitacademy.co.kr
Server:                 168.126.63.1
Address:        168.126.63.1#53

Non-authoritative answer:
Name:           www.bitacademy.co.kr
Address: 218.145.65.233
> www.facebook.com
Server:                 168.126.63.1
Address:        168.126.63.1#53
> exit

[root@localhost ~]#
```

리눅스 네트워크 관리
: 호스트 네임 확인 - hostname

- hostname : 호스트네임을 화면에 출력하는 명령
  - 사용법
    - hostname
- [실습] 현재 호스트 명을 확인해 봅시다

```
[root@localhost etc]# hostname
localhost.localdomain
[root@localhost etc]#
```

- [실습] 호스트명을 변경해 봅시다
  - /etc/hostname 파일을 vi 에디터로 열고 다음과 같이 수정

```
lx.mydomain.com
```

  - 저장하고 반영하기 위해 재부팅

```
[bituser@lx ~]$ clear
[bituser@lx ~]$ hostname
lx.mydomain.com
[bituser@lx ~]$
```

리눅스 네트워크 관리
: 네트워크 상태 확인 - netstat

- netstat : 네트워크 연결, 라우팅 테이블, 네트워크 장치의 통계 정보 등 네트워크 관련 여러 정보들을 확인하는 명령
  - 사용법
    - netstat [옵션]
  - 옵션
    - -a : 연결된 모든 소켓을 출력
    - -n : 호스트, 포트 등의 정보를 이름 대신 숫자로 표시
    - -p : 소켓을 열고 있는 프로세스의 아이디(PID)를 출력
    - -r : 라우팅 테이블을 출력
    - -t : TCP 연결에 대한 소켓을 출력
    - -u : UDP 연결에 대한 소켓을 출력

리눅스 네트워크 관리
: 도메인 네임서버 질의 - nslookup

- [실습] 현재 시스템의 라우팅 테이블을 확인해 봅니다

```
[root@lx ~]# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         192.168.0.1     0.0.0.0         UG        0 0          0 enp0s3
192.168.0.0     0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
[root@lx ~]#
```

- [실습] 현재 열려 있는 TCP 포트 정보를 출력해 봅니다

```
[root@lx ~]# netstat -ant | grep LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:25                  :::*                    LISTEN
[root@lx ~]#
```

리눅스 네트워크 관리
: 도메인 네임서버 질의 - nslookup

- [실습] 현재 열려 있는 TCP 포트의 프로세스까지 함께 출력해 봅시다

```
[root@lx ~]# netstat -anpt | grep LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      701/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      780/master
tcp6       0      0 :::22                   :::*                    LISTEN      701/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      780/master
[root@lx ~]#
```

  - 출력된 PID를 이용, 어떤 프로세스가 이 포트를 사용하고 있는지 확인해 봅시다

리눅스 네트워크 관리
: 네트워크 인터페이스 설정 - ifconfig

- ifconfig : 네트워크 인터페이스를 설정하고, 현재 네트워크 인터페이스의 정보를 알아보는 명령. 대부분 네트워크 설정을 확인하는 명령어로 많이 이용
  - 사용법
    - ifconfig [옵션]
- [실습] 전체 네트워크 인터페이스의 설정을 확인해 봅니다 (-a 옵션)

```
[root@lx ~]# ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.3  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe88:f242  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:88:f2:42  txqueuelen 1000  (Ethernet)
        RX packets 1841  bytes 162376 (158.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1297  bytes 250857 (244.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 2  bytes 106 (106.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 106 (106.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

리눅스 네트워크 관리
: 네트워크 인터페이스 설정 - ifconfig

- [실습] 특정 네트워크 인터페이스에 대한 정보만 출력해 봅니다

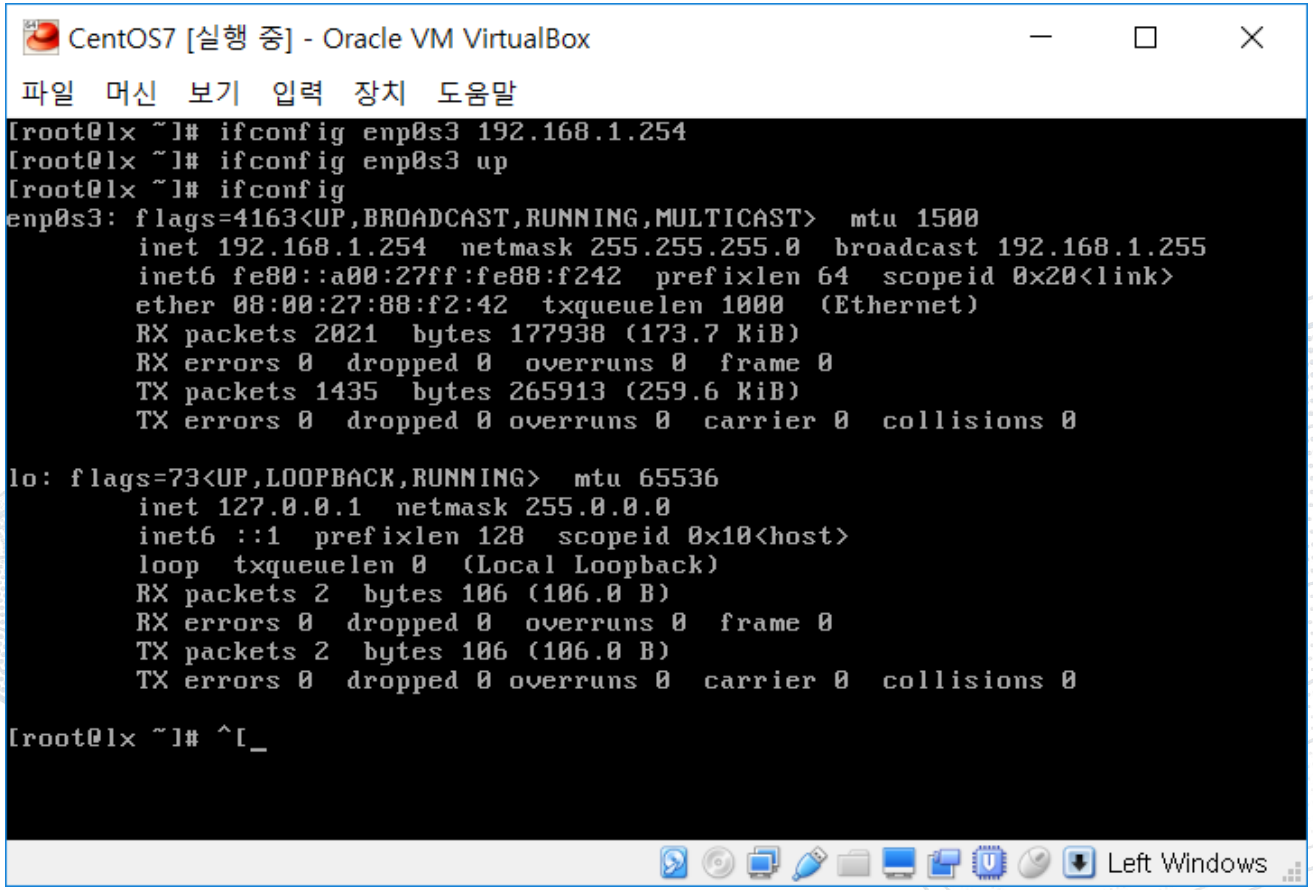
```
[root@lx ~]# ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.3  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe88:f242  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:88:f2:42  txqueuelen 1000  (Ethernet)
        RX packets 1952  bytes 171942 (167.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1365  bytes 259601 (253.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

  - 인터페이스(interface)는 NIC(Network Interface Card)를 의미하며 보통 랜카드(이더넷 카드)라고 부릅니다

- [실습] 네트워크를 중지해 봅니다
  - ifconfig {인터페이스명} down

리눅스 네트워크 관리
: 네트워크 인터페이스 설정 - ifconfig

- [실습] 네트워크 인터페이스에 IP address 를 변경하고 다시 시작해 봅니다



리눅스 네트워크 관리
: 고정 IP 설정 연습

- ifconfig에서 설정된 IP 주소는 시스템이 재시작되면 반영이 되지 않음
- 시스템의 네트워크를 설정하여 영구적으로 반영되도록 해야 한다

- 설정 전 확인 사항
  - IP Address
  - Subnet Mask
  - Gateway IP Address
  - DNS Server IP Address

리눅스 네트워크 관리
: 고정 IP 설정 연습

- [실습] 네트워크 설정은 /etc/sysconfig/network-scripts/ifcfg-인터페이스명 파일에서 합니다. 현재 설정 내용을 확인해 봅시다

```
[bituser@lx ~]$ cat /etc/sysconfig/network-scripts/ifcfg-enp0s3
TYPE="Ethernet"
BOOTPROTO="dhcp"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
NAME="enp0s3"
UUID="c6755cb5-a27b-4260-a305-aa260b00c275"
DEVICE="enp0s3"
ONBOOT="yes"
PEERDNS="yes"
PEERROUTES="yes"
IPV6_PEERDNS="yes"
IPV6_PEERROUTES="yes"
IPV6_PRIVACY="no"
[bituser@lx ~]$
```

리눅스 네트워크 관리
: 고정 IP 설정 연습

- 현재는 BOOTPROTO가 dhcp, 즉 동적 할당으로 되어 있음
- 고정 IP로 설정하기 위해 다음과 같이 설정

```
TYPE="Ethernet"
BOOTPROTO="static"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
...


IPADDR=192.168.1.101
NETMASK=255.555.255.0
GATEWAY=192.168.1.1
DNS1=168.126.63.1
```

- 수정 후 네트워크 재시작

```
[root@lx ~]# systemctl restart network.service
```

리눅스 네트워크 관리
: 고정 IP 설정 연습

```
network:
  ethernets:
    # 네트워크 인터페이스 이름(이더넷명 : enp0s3 ) 사용자 마다 다름
    enp0s3:
      addresses:
        # IP주소 / subnet mask CIDR 표기법(/24 = 255.255.255.0  의미는 8byte * 3 = 24byte)
        - 192.168.123.61/24
      nameservers:
        # DNS 서버 주소 []안에 하나만 입력하거나 한칸 아래 "- 주소" 입력 가능하다
        # ex)  addresses: [1.1.1.1,8.8.8.8]
        addresses:
          - 8.8.8.8
      routes:
        - to: default
          #gateway 주소 입력
          via: 192.168.123.1
  version: 2
```

Ubuntu 18.04 이후 Yaml 파일로 변경, netplan으로 적용

```
#변경사항 저장하기
sudo netplan apply
```

네트워크 트래픽 모니터링

도구	bps	pps	NIC 구분/출력	한 화면 여러 NIC	Process 구분/출력	Address 구분/출력	Protocol 구분/출력	그래프 시각화	History	기타
1. iftop	O	X	/	/	X	O	X	O	X	
2. bmon	O	O	O	O	X	X	X	O	O	tc도 구분
3. slurm	O	△	/	/	X	X	X	O	O	
4. tcptrack	O	X	/	/	X	O	/	X	X	
5. nethogs	O	X	O	O	O	X	O	X	X	
6. ifstat	O	X	O	O	X	X	X	X	X	
7. nettop	O	X	X	O	O	O	O	X	X	
8. nload	O	X	O	X	X	X	X	O	O	
9. netstat	X	O	O	O	X	X	X	X	X	
10. dstat	O	X	X	X	X	X	X	X	O	
11. iptraf	O	O	O	O	X	O	O	X	X	
12. speedometer	O	X	O	O	X	X	X	O	O	
13. vnstat	O	O	O	O	X	X	X	X	O	daemon
14. cbm	O	X	O	O	X	X	X	X	X	
15. bandwhich	O	X	O	O	O	O	O	X	X	
16. darkstat	O	X	O	O	X	O	X	O	O	daemon, GUI

pps: 밴드위스 정보를 제공 유무

pps: 패킷 개수 제공 유무

NIC : 여러 개의 Nic을 구별해 주며 보여주는 유무

PROCESS, Address, Protocol 구분/출력

각각의 요소를 화면내에 통계를 수집하여 출력하는 경우

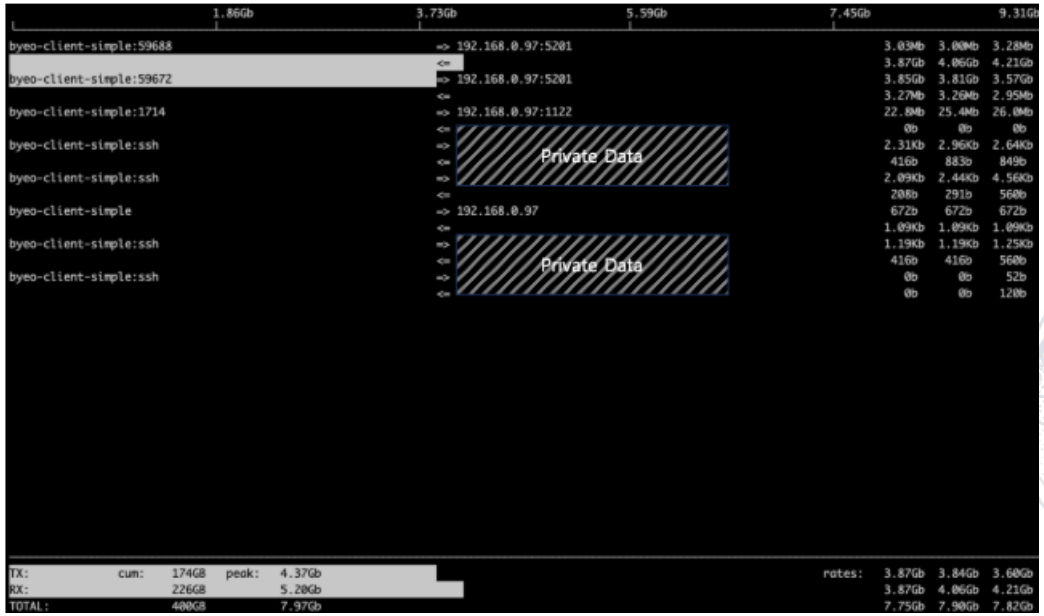
그래프 시각화 : 시각화 유무

History: 현재 및 과거 상황도 그래프 또는 텍스트로 제공 유무

네트워크 트래픽 모니터링

- iftop
  - sudo apt install iftop
  - Iftop -p

화면



장점

- Simple하다!
- NIC이 사용하는 트래픽 양을 주소별로 알 수 있다. (Tx / Rx 별도)
- 글자에 하얀색 배경은 화면 최상단의 bar를 통해서 어느정도 쓰는지 알 수 있다. 즉, 정보의 시각화를 제공한다.
- 우측에 2초, 10초, 40초 평균을 제공한다.

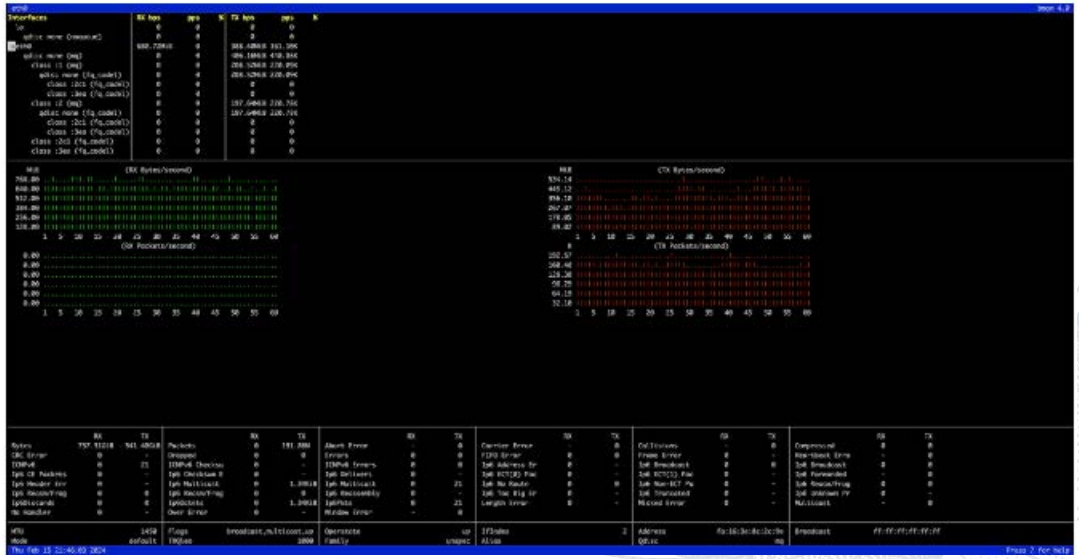
단점

- 프로토콜과 관련된 정보가 보이지 않는다.
- 프로세스와 관련된 정보도 없다. 만약 5-tuple에서 protocol만 다른 경우, 프로세스가 다를 수도 있다.
- 한 화면에 인터페이스 1개다. -i 옵션으로 지정하지 않으면 interface list에서 가장 첫 interface를 보여준다.

네트워크 트래픽 모니터링

- bmon
  - sudo apt install bmon
  - bmon

화면



장점

- NIC별로 bps와 pps를 한 눈에 보여준다.
- 선택한 1개의 NIC에 대해서 시각화하여 그래프도 그려준다.
- tc (traffic control qdisc)와 관련된 정보까지 세세히 제공한다.

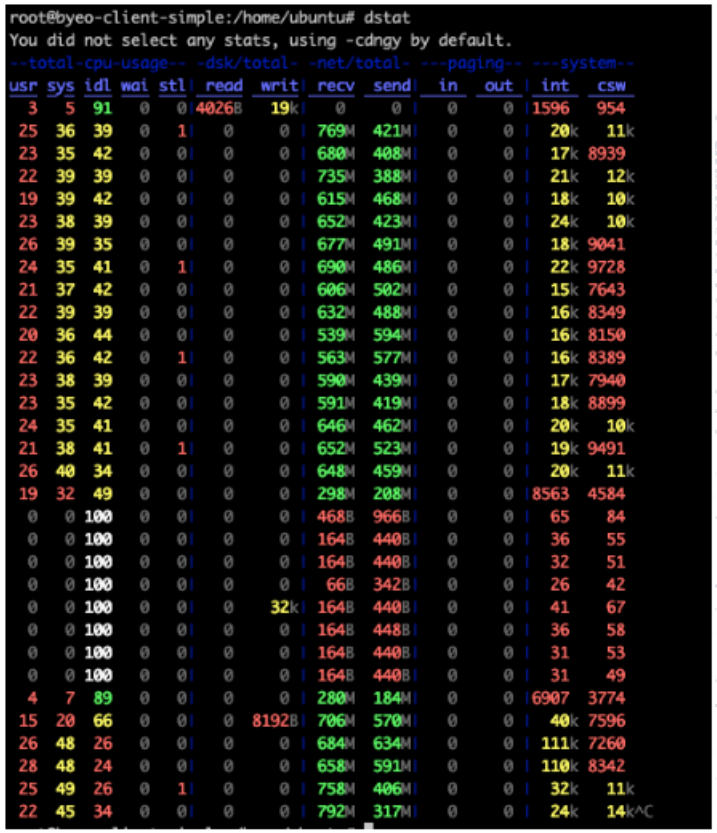
단점

- NIC 단위에 대한 정보에 집중해있어서 process 별 정보, five-tuple 별 정보 등은 얻기 힘들다.
- 8을 곱하면 되긴 하나.. bps가 Byte단위로 나온다.

네트워크 트래픽 모니터링

- dstat
  - sudo apt install dstat
  - nload

화면



장점

- 네트워크 상태 뿐만 아니라 disk, cpu 사용량 등 다양한 정보를 제공한다.

단점

- NIC별로, 프로세스 별로 등 확인하기 어렵다.

네트워크 트래픽 모니터링

- 기타 항목 조사 실습 해보세요 ^^