

# Linux System Administration

Network Administration : 네트워크 관리

# 리눅스 네트워크 관리

: 네트워크의 이해

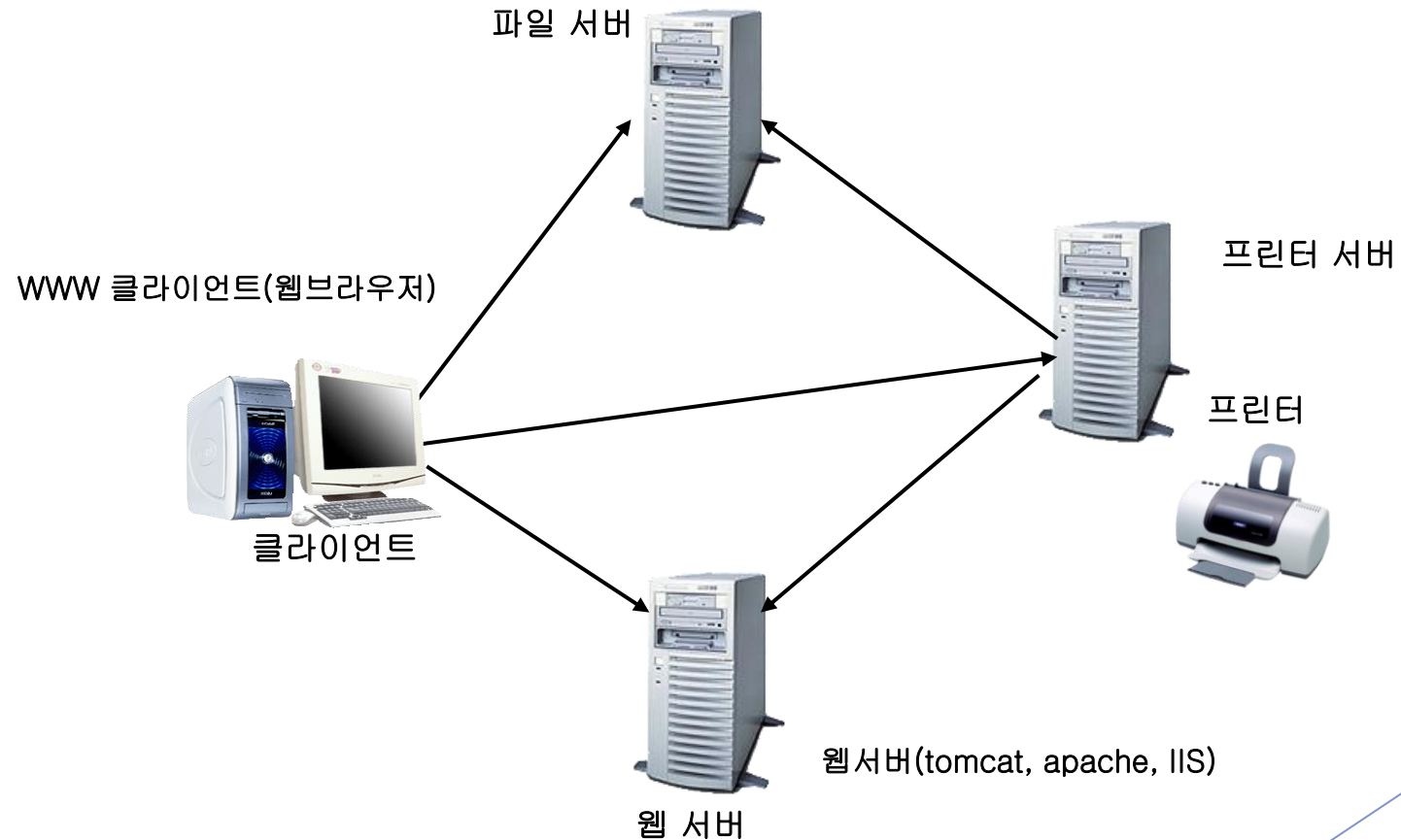
“유/무선으로 연결되어 있는 **Device**들의 집합 ”

- ▶ 유/무선으로 연결
  - ▶ Bluetooth, Wi-Fi, RFID, 적외선 통신(IrDA) 등 근거리 무선 통신
  - ▶ WCDMA, LTE 등 이동통신 기술
  - ▶ Ethernet, Serial 통신 등 유선통신
  - ▶ GPS
- ▶ Device
  - ▶ 네트워크에 연결된 컴퓨터
  - ▶ 컴퓨터가 아닌 다른 장치들  
예) 프린터, 모바일 장치, 가전제품, 웨어러블 등 다양한 임베디드 제품들
  - ▶ 포괄적 개념으로 네트워크에 연결되어 있는 것들을 총칭하는 용어

# 리눅스 네트워크 관리

: 네트워킹의 이해

“네트워크에 연결된 디바이스들 간에  
미리 약속된 프로토콜을 사용하여  
데이터를 교환하는 것 ”



# 리눅스 네트워크 관리

## : 네트워킹의 이해

### ▶ Protocol (프로토콜)

- ▶ 약속, 규약
- ▶ 디바이스 상호간 데이터 통신을 위해 필요한 규약
- ▶ 통신 규약

### ▶ 데이터 통신 방법

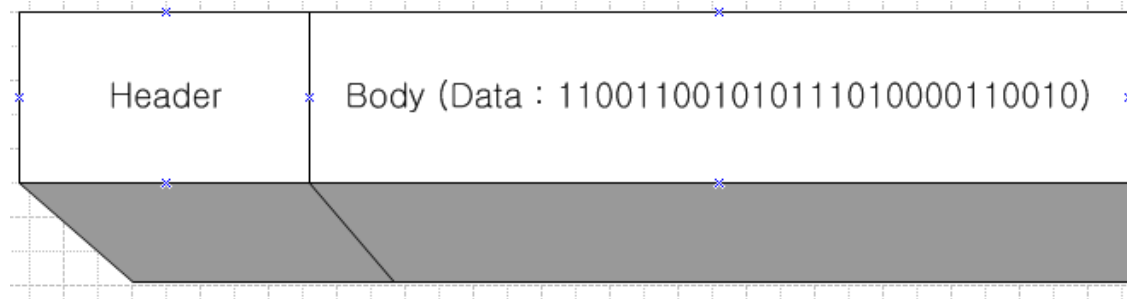
- ▶ 두 개의 상이한 디바이스가 물리적으로 떨어져 있는 경우, 유/무선을 통해 네트워크에 연결되어야 한다
- ▶ 연결된 두 디바이스는 전류, 전파, 빛 등의 방식으로 데이터 통신을 한다
- ▶ 이 데이터는 0/1 또는 on/off의 bit의 조합으로 표현된다
- ▶ 주소(address)
  - ▶ 두 디바이스가 데이터 통신을 하기 위해서는 서로의 위치를 알아야 한다
  - ▶ 이 위치를 네트워크에서는 node라 부른다
  - ▶ 각 node마다 고유의 주소를 가지고 있어야 한다
- ▶ 데이터 통신을 할 때는 데이터 외에 어디로 보내야 하는가 또는 누가 보내는가 등의 정보를 담고 있어야 한다

# 리눅스 네트워크 관리

## : 데이터 통신 방법

### ▶ Packet(패킷)

- ▶ 실제 네트워크를 통한 데이터 통신을 할 때에는 패킷을 사용한다



**Header** : 송신자/수신자의 주소, 체크섬(Checksum), 기타 제어 정보

**Body** : 전송할 데이터 (주로) **byte** 단위로 포함

# 리눅스 네트워크 관리

## : 인터넷의 이해

- ▶ 인터넷 != WWW(World Wide Web)
- ▶ 인터넷 기반 서비스

| 이름    | 프로토콜           | 포트         | 기능         |
|-------|----------------|------------|------------|
| WWW   | HTTP           | 80         | 웹서비스       |
| Email | SMTP/POP3/IMAP | 25/110/114 | 이메일 서비스    |
| FTP   | FTP            | 21         | 파일 전송 서비스  |
| DNS   | DNS            | 23         | 네임서비스      |
| NEWS  | NNTP           | 119        | 인터넷 뉴스 서비스 |

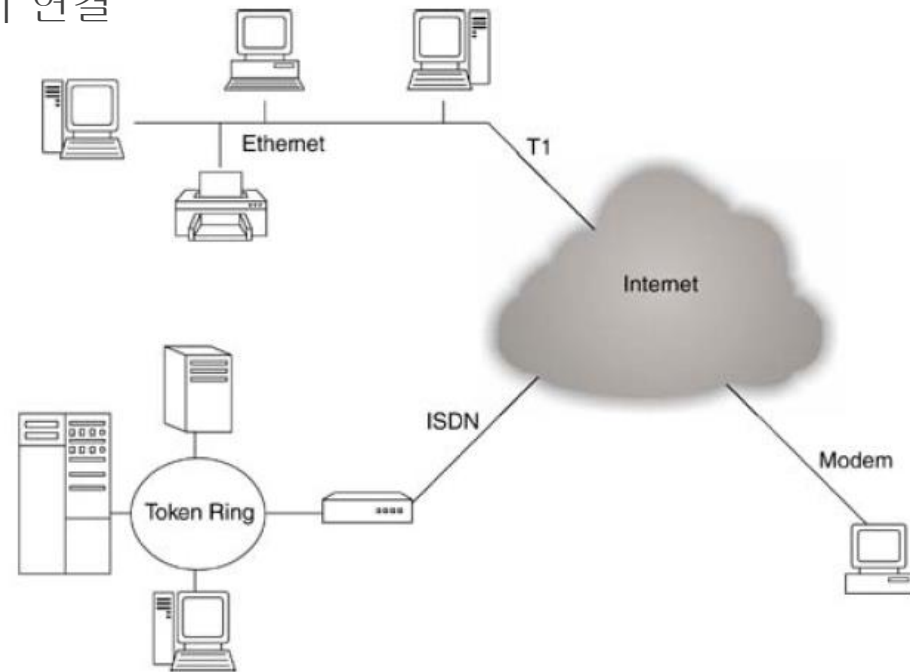
- ▶ 인터넷 (Internet)
  - ▶ TCP/IP 기반의 네트워크가 전 세계적으로 확대되어 하나로 연결된 네트워크들의 네트워크 (네트워크의 결합체)

# 리눅스 네트워크 관리

## : 인터넷의 이해

### ▶ 인터넷의 역사

- ▶ 미국방성의 연구 목적 네트워크 ARPANET(1969년)이 시초
- ▶ 미 과학재단 네트워크인 NSFNET이 연결(1986년)
- ▶ 1990년 이후 일반 산업 목적의 네트워크들이 연결



# 리눅스 네트워크 관리

## : 인터넷의 이해

### ▶ OSI 7계층과 TCP/IP 4계층

- ▶ 하드웨어, 운영체제, 접속 매체와 관계 없이 동작할 수 있는 개방형 구조
- ▶ OSI 7 계층을 4계층으로 단순화

| OSI 7계층   | TCP/IP 4계층 |  |
|-----------|------------|--|
| 응용 계층     | 응용 계층      | 네트워크를 사용하는 WWW, FTP, 텔넷, SMTP 등의 응용 프로그램으로 구성.               |
| 표현 계층     |            |  |
| 세션 계층     |            |  |
| 전송 계층     | 전송 계층      | 도착지까지 데이터를 전송<br>각각의 시스템을 연결<br>TCP 프로토콜을 이용하여 데이터를 전송       |
| 네트워크 계층   | 인터넷 계층     | 데이터를 정의 및 경로 지정<br>정확한 라우팅을 위해 IP 프로토콜을 사용<br>IP 주소가 위치하는 계층 |
| 데이터 링크 계층 | 링크 계층      | 물리적 계층 즉 이더넷 카드와 같은 하드웨어                                     |
| 물리 계층     |            |  |



# 리눅스 네트워크 관리

## : 인터넷의 이해 - 계층과 프로토콜

### ▶ Link 계층 (Link Layer)

- ▶ Host(호스트)간의 네트워크 데이터 통신을 위한 물리적 연결 (유/무선)에 대한 표준
- ▶ LAN, WAN, MAN과 같은 네트워크 구성을 정의
- ▶ 상위 계층인 **Internet** 계층에서 형성된 패킷을 전기신호 또는 광신호로 바꾸어 전달하는 역할 담당
- ▶ 응용프로그램 개발자가 직접 접근할 수 있는 계층이 아니고 보통 네트워크 장비나 드라이버 개발자들이 관심을 갖는 계층



# 리눅스 네트워크 관리

## : 인터넷의 이해 - 계층과 프로토콜

### ▶ Internet 계층

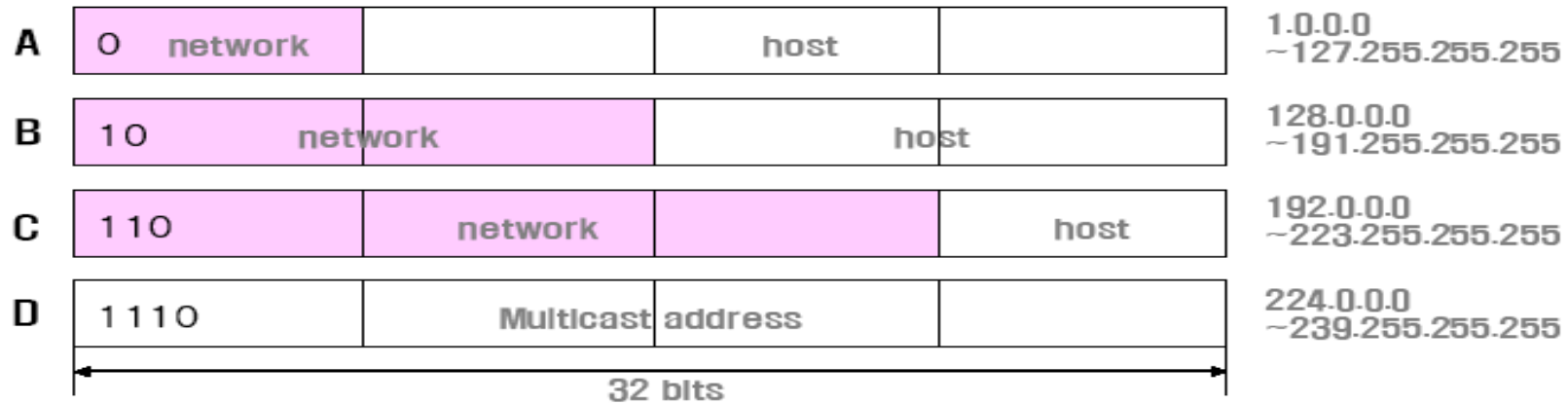
- ▶ Link 계층을 통해 물리적으로 연결된 각 Host간의 Packet 전달 경로를 결정
- ▶ IP (Internet Protocol)이 이 계층에 위치
- ▶ IP 프로토콜은 IP 주소(Address)를 부여하는 방법과 체계를 정의
- ▶ IP 프로토콜은 IP 주소를 기반으로 경로를 라우팅(Routing) 하는 방법을 정의
- ▶ 전송 계층(Transport Layer)과 함께 인터넷에서 중요한 계층
- ▶ 데이터가 상대방에게 안전하게 전송되는 것을 보장하지는 않음
- ▶ 전송 계층이 데이터 전달에 대한 신뢰성을 책임진다는 가정 하에 목적지로 패킷을 어떤 경로로 전송할 것인가에 대한 문제만 해결하는 계층



# 리눅스 네트워크 관리

## : 인터넷의 이해 - 계층과 프로토콜

- ▶ IP 주소 (IP Address)의 종류
  - ▶ IPv4 (Internet Protocol Version 4) : 4바이트 주소 체계
  - ▶ IPv6 (Internet Protocol Version 6) : 16바이트 주소 체계
- ▶ IPv4 주소는 Network 주소와 Host 주소로 구분됨

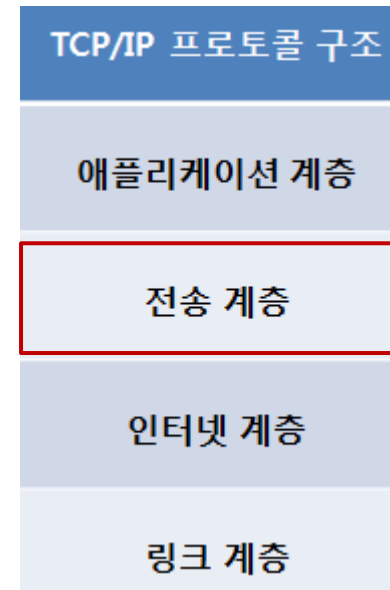


# 리눅스 네트워크 관리

## : 인터넷의 이해 - 계층과 프로토콜

### ▶ 전송 계층(Transport Layer)

- ▶ 하위 인터넷 계층의 IP가 해결한 목적지까지의 네트워크 경로에서 실제 데이터를 전송하는 역할을 담당
- ▶ TCP와 UDP라는 데이터의 전달을 책임지는 프로토콜이 존재
- ▶ IP는 하나의 패킷이 전달되는 과정에만 중심을 두고 설계되었기 때문에 여러 패킷으로 나뉘어 전달되는 데이터의 순서와 전송 자체는 신뢰할 수 없다
- ▶ 데이터의 순서와 신뢰할 수 있는 데이터 전송을 보장하는 역할을 전송 계층의 프로토콜이 담당



# 리눅스 네트워크 관리

## : 인터넷의 이해 - 계층과 프로토콜

### ▶ TCP(Transmission Control Protocol)

- ▶ 연결지향 프로토콜: 데이터 전송/수신 이전에 소켓을 통해 양쪽 연결이 성립
- ▶ 연결이 성립되면 **TCP**는 데이터의 손실이나 중복 없이 목적지에 확실하게 전달
- ▶ 전송 도중 데이터의 일부가 유실되었다면 수신자는 발신자에게 해당 데이터의 재 전송을 요청
- ▶ **UDP**에 비해 프로토콜이 더 복잡하고 속도도 느리다
- ▶ 흔히, 전화 통화에 비유
- ▶ **UDP**에 비해 신뢰성 있는 데이터 전송이 가능하다는 장점 때문에 **HTTP, FTP, TELNET** 등 대부분 응용 계층 프로토콜은 전송 계층으로 **TCP**를 이용

# 리눅스 네트워크 관리

## : 인터넷의 이해 - 계층과 프로토콜

### ▶ UDP (User Datagram Protocol)

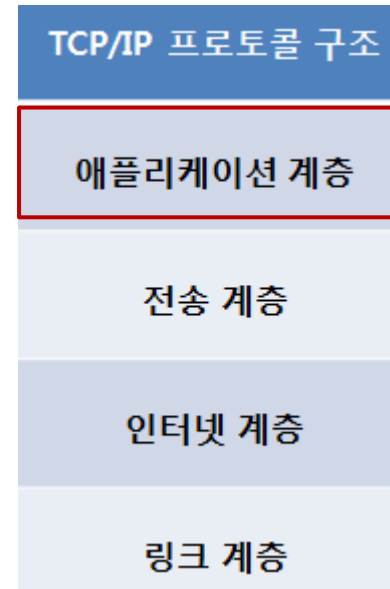
- ▶ UDP는 비연결지향 프로토콜
- ▶ 전송한 데이터가 잘 전달되었는지 확인하지 않고 단지 데이터를 보내는 것으로 자신의 임무를 다한 것으로 간주
- ▶ TCP에 비해 신뢰성이 떨어지는 프로토콜
- ▶ 흔히, 편지를 보내는 것에 비유
- ▶ 음악이나 동영상의 **Streaming** 서비스 등에 적당한 프로토콜

# 리눅스 네트워크 관리

## : 인터넷의 이해 - 계층과 프로토콜

### ▶ 응용 계층(Application Layer)

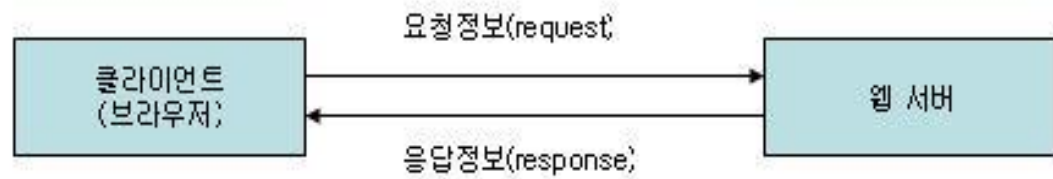
- ▶ 하위 계층이 목적지가 되는 호스트에 데이터를 안전하게 전달한다는 신뢰를 바탕으로 응용프로그램(프로세스)들이 통신을 하게 된다
- ▶ 응용 계층의 응용 프로그램(프로세스)들의 데이터 통신은 매우 다양
  - ▶ 예) 메일 보내기(SMTP), 파일 전송(FTP), 웹사이트 접속(HTTP)
  - ▶ 목적에 맞는 데이터 통신을 위한 응용 프로토콜이 미리 정의되어 있기도 하고 개발자가 직접 설계하여 인터넷 기반의 서비스를 개발할 수도 있다
- ▶ 소켓(Socket)은 응용 계층에서 개발되는 응용 프로그램에서 하위 계층의 TCP/IP의 역할을 감추어 준다 (투명성)
- ▶ 소켓(Socket)이라는 도구를 사용하면 응용 프로그램간의 성격에 따라 미리 설계된 응용 프로토콜을 이용한 프로그램을 개발하거나 직접 프로토콜을 설계하고 구현만 하면 됨
- ▶ 대부분의 네트워크 프로그래밍은 소켓(Socket)을 사용하여 위와 같은 작업을 하는 것이라 할 수 있다



# 리눅스 네트워크 관리

## : HTTP

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

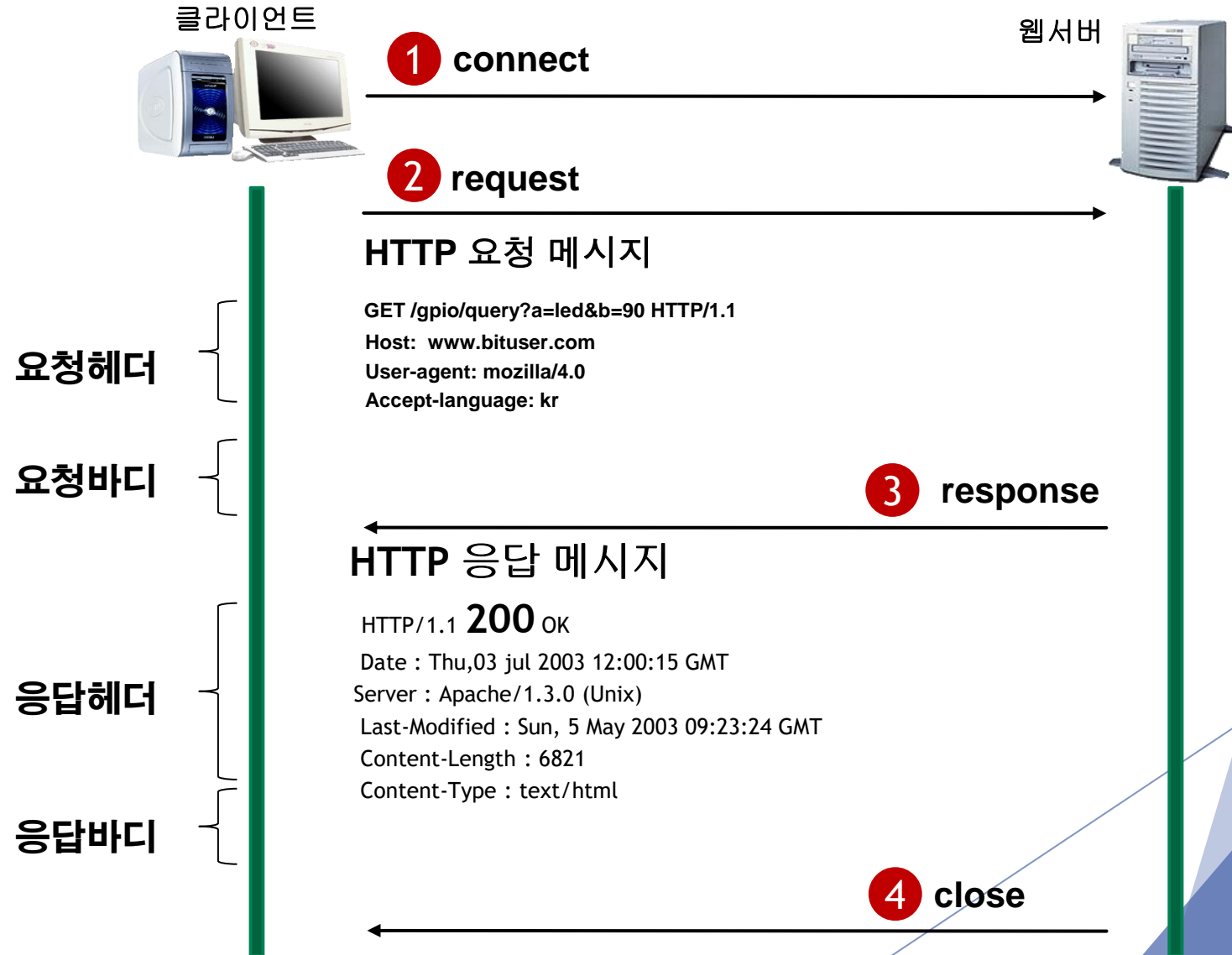


- ▶ HTTP 프로토콜의 작동 방식
  - ▶ 브라우저는 웹 서버에게 요청 정보를 보낸다
  - ▶ 웹 서버는 요청 정보를 분석하여 응답 정보를 브라우저에게 보낸다
  - ▶ 비연결 지향의 프로토콜



# 리눅스 네트워크 관리

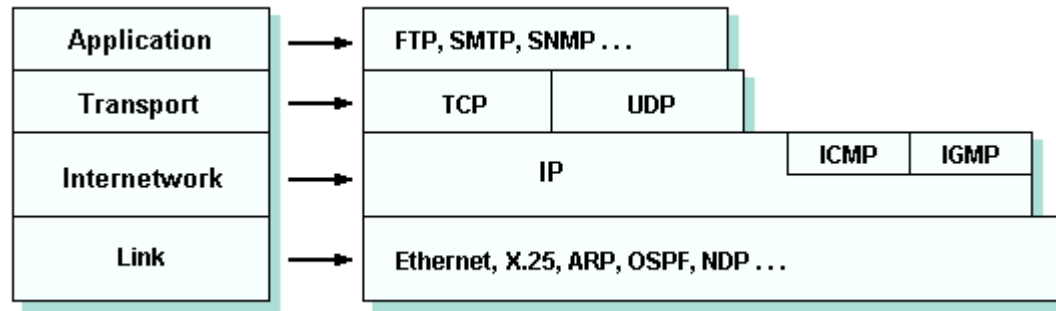
: HTTP



# 리눅스 네트워크 관리

## : TCP/IP 프로토콜 스택(Stack)

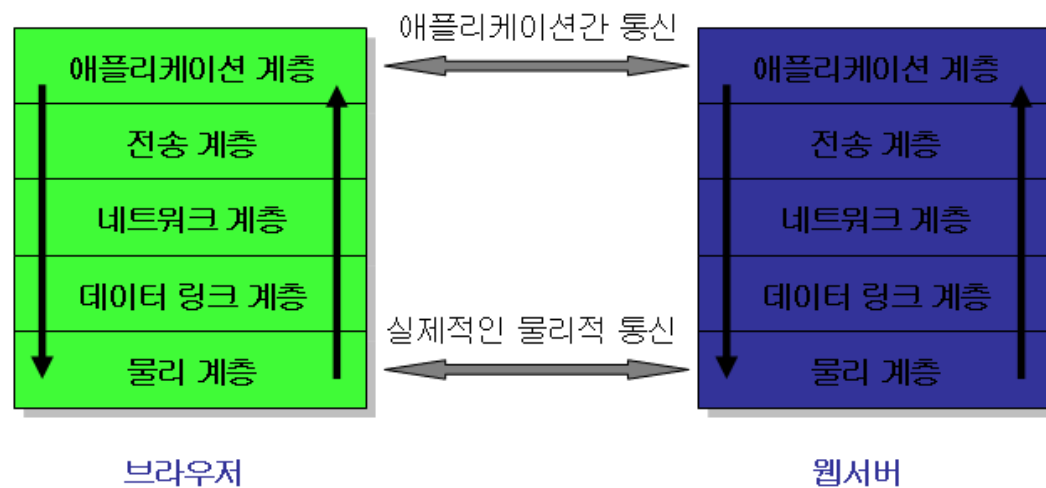
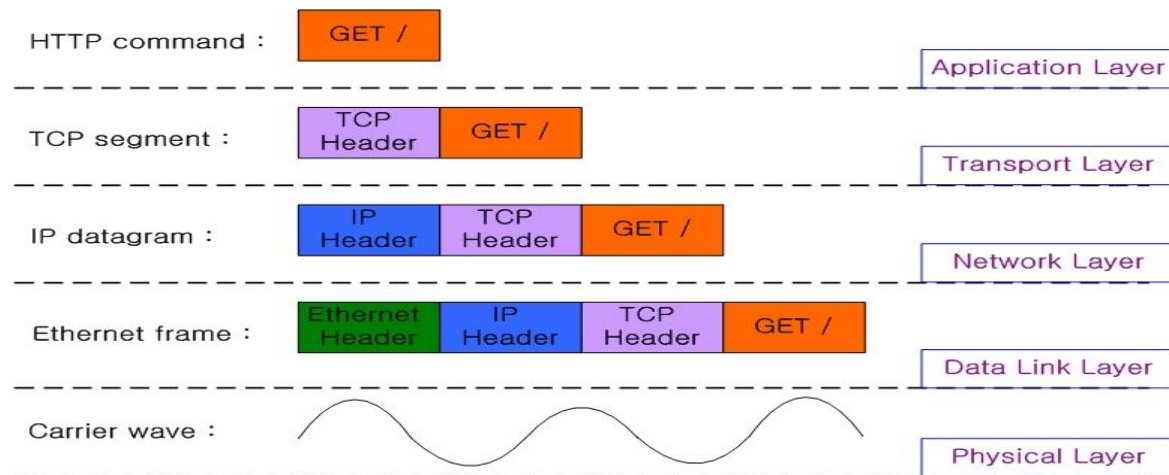
- ▶ TCP/IP 프로토콜 스택은 총 4개의 부분으로 구분
- ▶ 네트워크 데이터 통신 과정을 4개의 영역으로 계층화한 것
- ▶ 데이터 통신 과정을 하나의 덩치 큰 프로토콜로 해결하지 않고 작게 나눠서 계층화 하려는 노력의 결과



- ▶ 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_DGRAM             |
|-------------------------|------------------------|
| 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 );
```

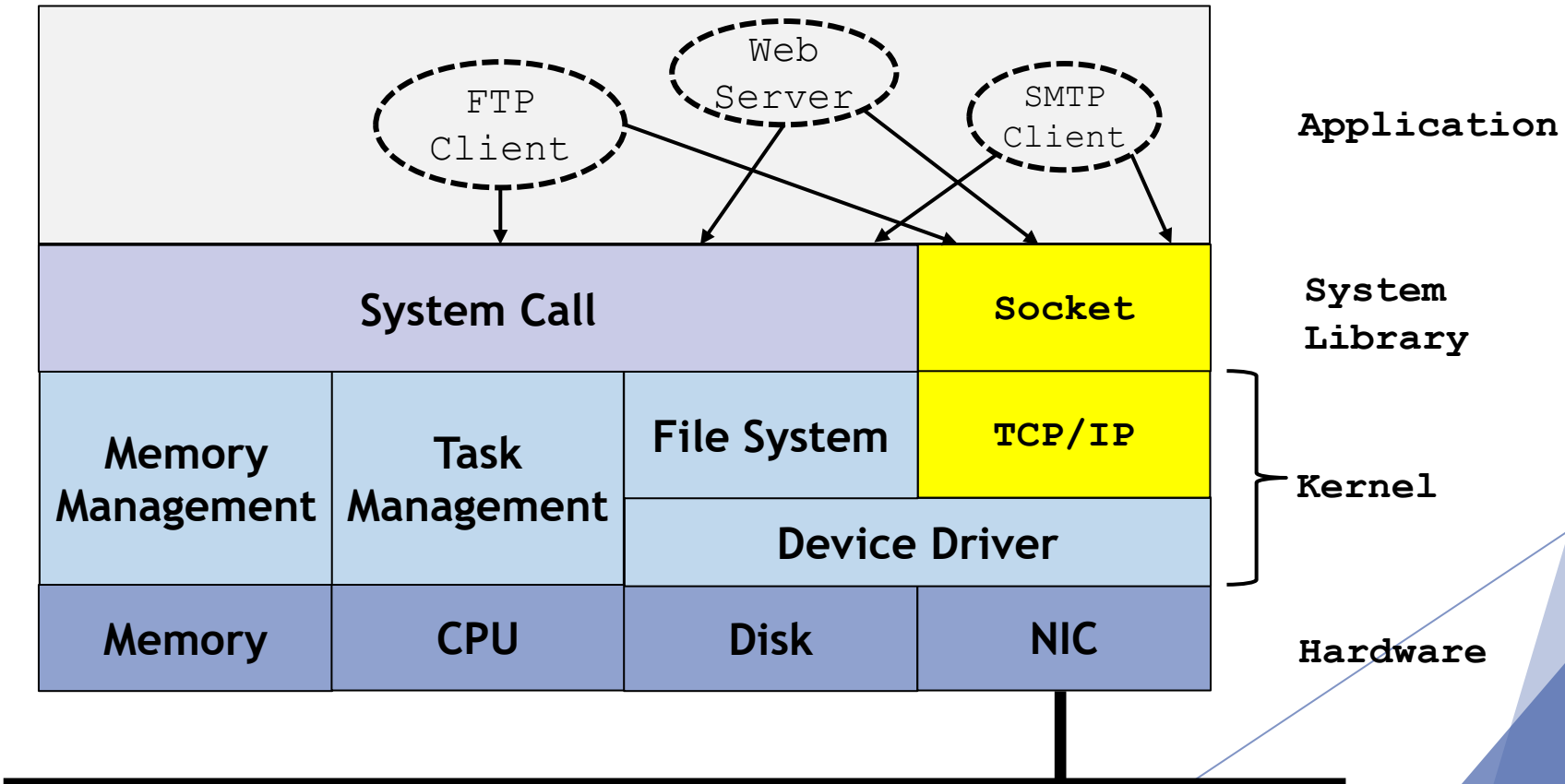
▶ **protocol : 프로토콜 선택**

- ▶ 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

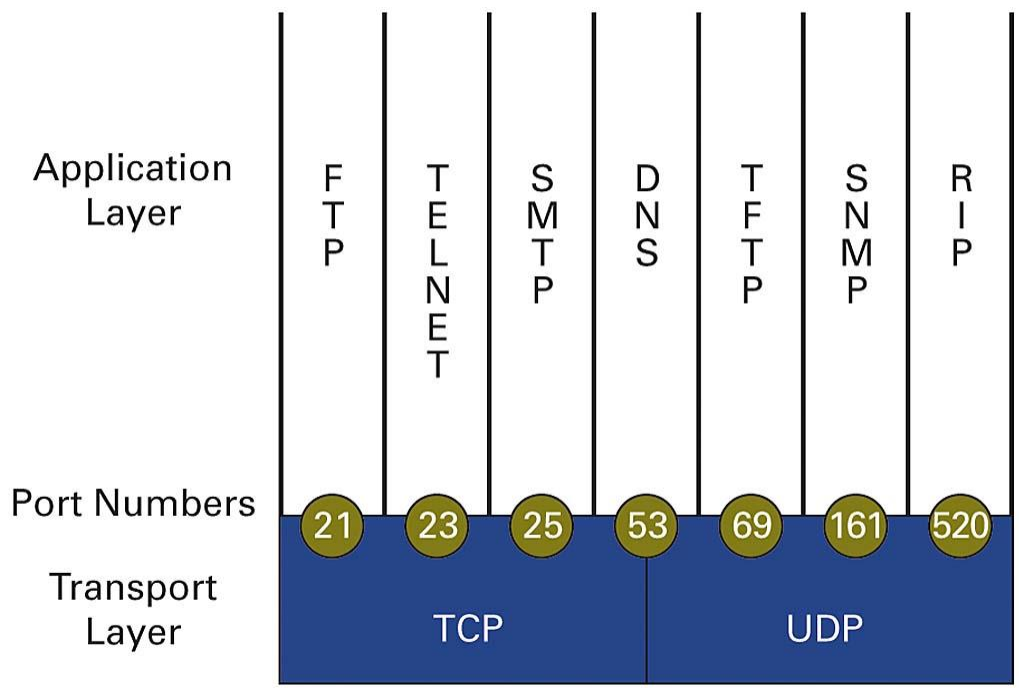
### ▶ 포트의 필요성

- ▶ 실제적인 데이터 통신은 연결된 두 **Host**(컴퓨터)의 **Process**(프로그램) 사이에서 이루어짐
- ▶ 여러 계층을 통해 애플리케이션 계층으로 들어온 데이터를 해당 **Process**에만 정확히 전달해야 할 필요가 있음
- ▶ 하나의 **Host**(컴퓨터)에는 여러 개의 **Process**(프로그램)가 각각의 소켓을 사용하여 데이터 통신을 하고 있기 때문에 **TCP**에서는 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

## : 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

# 리눅스 네트워크 관리

## : 원격 시스템 네트워크 동작 확인 - ping

- ▶ [실습] [www.google.com](http://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
```

- ▶ 기본적으로 옵션 없이 사용하면 지속적으로 패킷을 보냄: **Ctrl + C**로 중지

# 리눅스 네트워크 관리

## : 원격 시스템 네트워크 동작 확인 - ping

- ▶ [실습] [www.google.com](http://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 확인 하는데 유용하지만, 과도한 사용은 서버에 부담을 줄 수 있음
- ▶ 외부의 과도한 ping을 막기 위해 ICMP(Internet Control Message Protocol) echo 를 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
```

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

- ▶ 설정을 완료하고 외부에서 ping을 시도해서 테스트 해 봅니다

# 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 대화형으로 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.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

[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 이름을 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

## : 네트워크 상태 확인 - netstat

- ▶ [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

## : 네트워크 상태 확인 - netstat

- ▶ [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

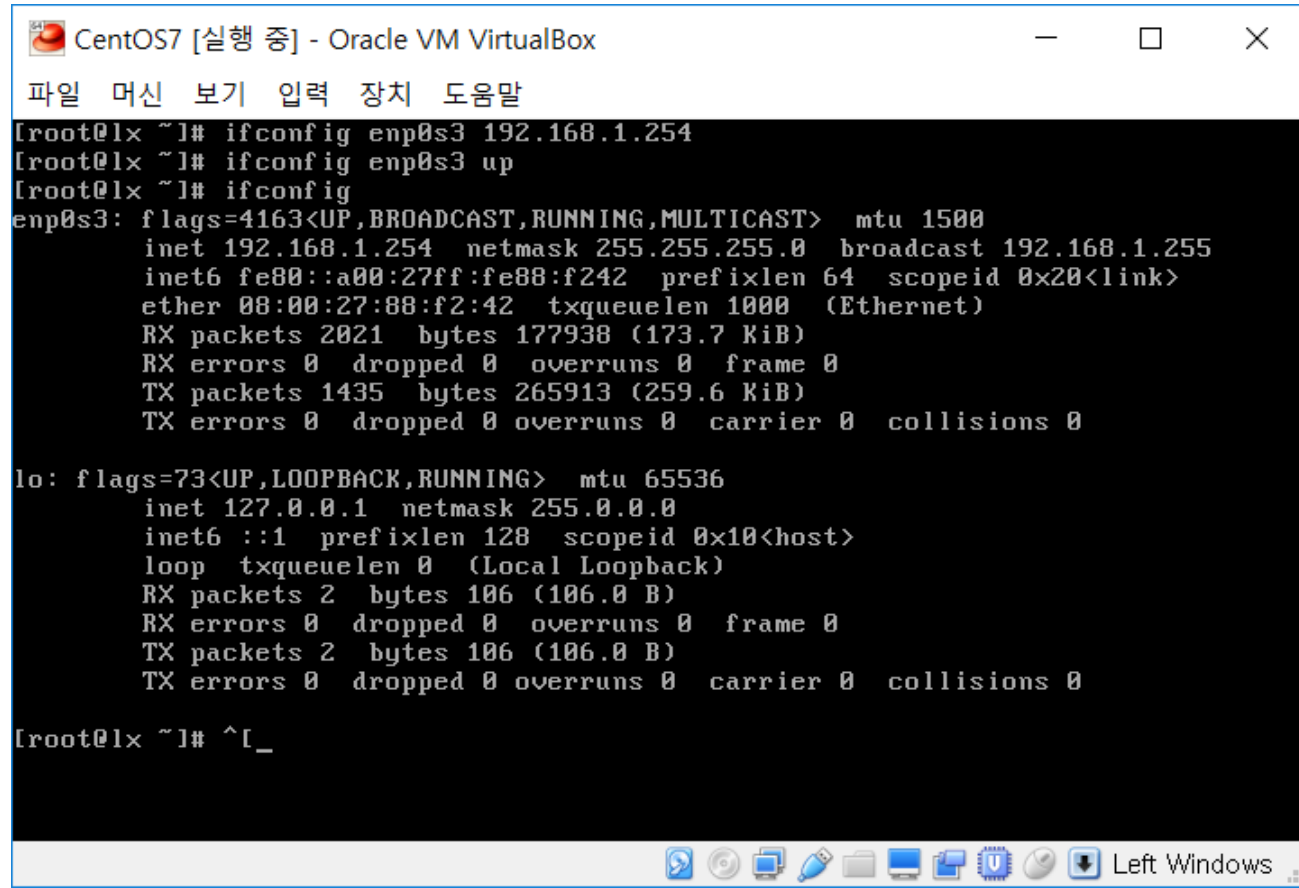
```
[root@lx ~]# ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.0.3  netmask 255.255.255.0  broadcast 192.168.0.255
    inet6 fe80::a00:27ff:fe88:f242  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:88:f2:42  txqueuelen 1000  (Ethernet)
    RX packets 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



```
CentOS7 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
[root@lx ~]# ifconfig enp0s3 192.168.1.254
[root@lx ~]# ifconfig enp0s3 up
[root@lx ~]# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.1.254  netmask 255.255.255.0  broadcast 192.168.1.255
    inet6 fe80::a00:27ff:fe88:f242  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:88:f2:42  txqueuelen 1000  (Ethernet)
    RX packets 2021  bytes 177938 (173.7 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 1435  bytes 265913 (259.6 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

[root@lx ~]# ^[_
```

# 리눅스 네트워크 관리

## : 고정 IP 설정 연습

- ▶ `ifconfig`에서 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
```