네트워크 프로그래밍

Ch01_네트워크 기초

강 민 우

강의 개요

지도목표	◦ TCP/IP 프로토콜의 동작 원리를 이해할 수 있다. ◦ 소켓을 활용하여 TCP, UDP 서버/클라이언트 프로그램을 작성할 수 있다.

차시	일자	시간	학습 내용
1	9.13.	3시간	네트워크 기초 용어
2	9.20.	3시간	소켓 시작하기
3	10.4.	3시간	소켓 주소 구조체 다루기
4	10.10. (화)	3시간	TCP의 동작 원리
5	10.25.	3시간	데이터 전송하기
6	11.1.	3시간	멀티쓰레드(리눅스)
7	11.15.	3시간	UDP의 동작 원리
8	11.22.	3시간	TCP, UDP 서버/클라이언트
9	12.6.	3시간	소켓 입출력 모델(리눅스)
10	12.11. (월)	3시간	IPv4 TCP, UDP 지원 데이터 전송 프로그램

Part 1

우리가 지금은 알고 있어야 하는 것

- 1 [byte] = 8 [bit]
- 비트 논리 연산의 방법
- 2진수를 16진수로 변환하기
- 단위 변환 → 1[GB] = 1024[MB]
- 프로세스와 프로그램
- OSI 7 Layer
- User mode와 Kernel mode
- Buffer

Part 1

실습 환경

- 개인용 노트북
- 참고자료(PPT, 소스코드)
- Ubuntu Linux → WSL 활용
- Git 저장소 → git clone https://github.com/goodgeni/network
- g++ 컴파일러
- make 패키지
- 열정과 의지

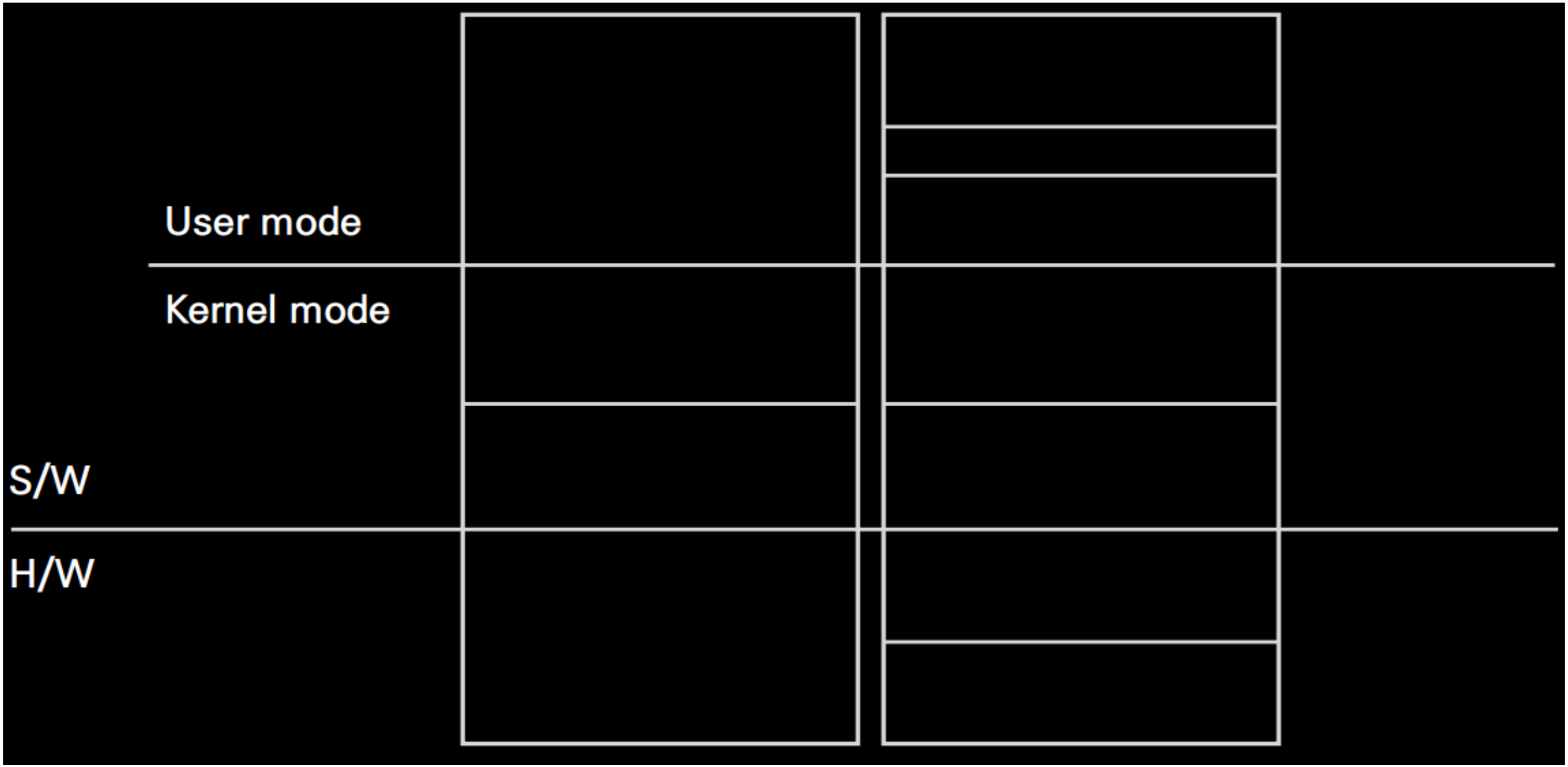
Part 1

네트워크 필수 지식

- OSI 7 Layer와 각 계층의 프로토콜, 식별자
- Host, End-Point, Switch, Router
- IP 주소와 MAC 주소
- 네트워크의 데이터 단위 (MSS, MTU)
- PORT 번호

Part 1

User Mode와 Kernel Mode / OSI 7 Layer와 식별자



Part 1

OSI 7 Layer와 식별자(암기)

- **L2 Frame에서는 MAC 주소**

  : 48bit 이며 보통 16진수로 표기됨

- **L3 Packet에서는 IP 주소**

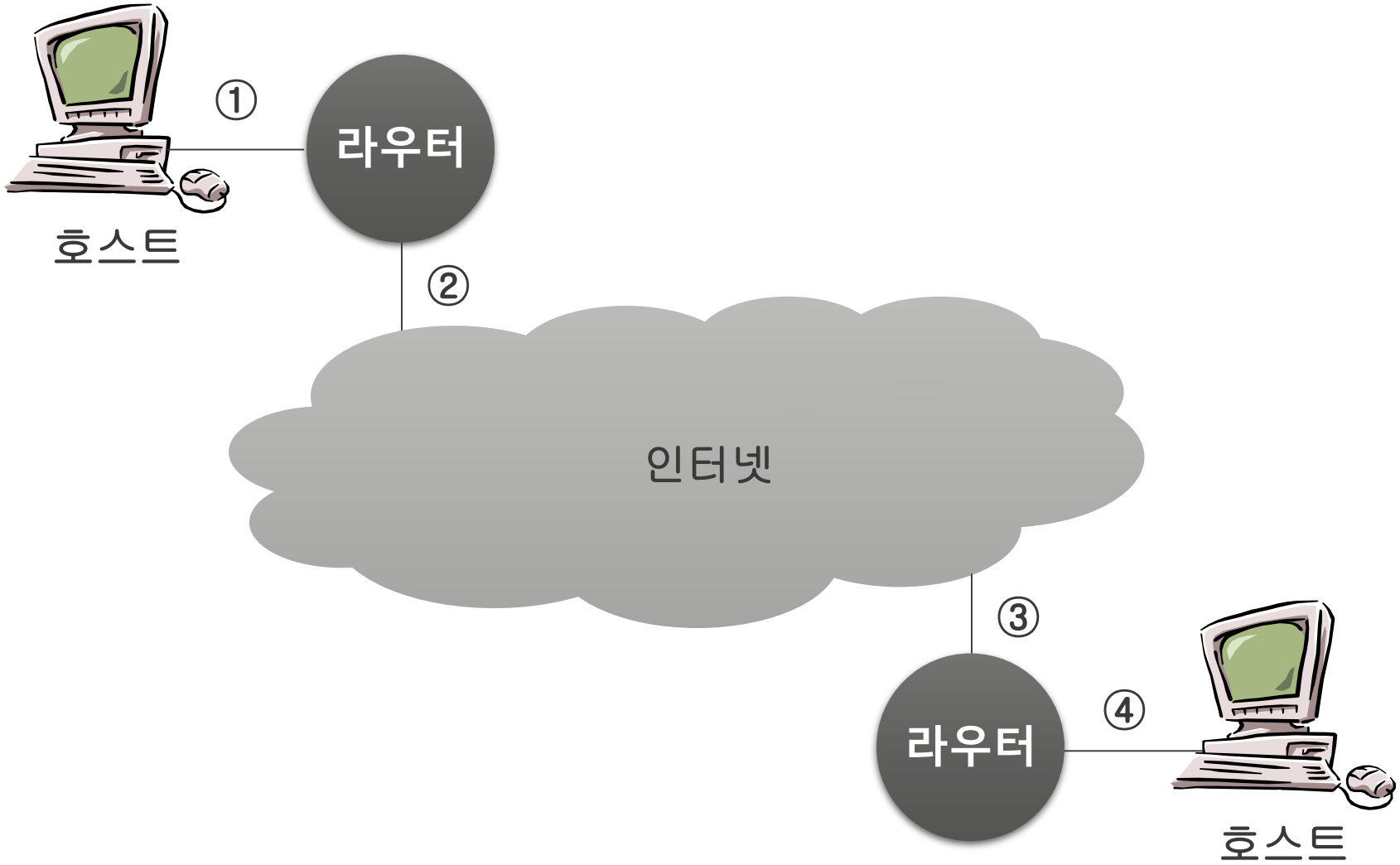
  : IPv4에서는 32bit이며 10진수(8bit)씩 끊어 점으로 구분

- **L4 수준에서는 PORT 번호**

  : 16bit 양의 정수

Part 1

인터넷의 구성 요소



Part 1

인터넷의 구성 요소

호스트

- 최종 사용자의 응용 프로그램을 수행하는 주체

라우터

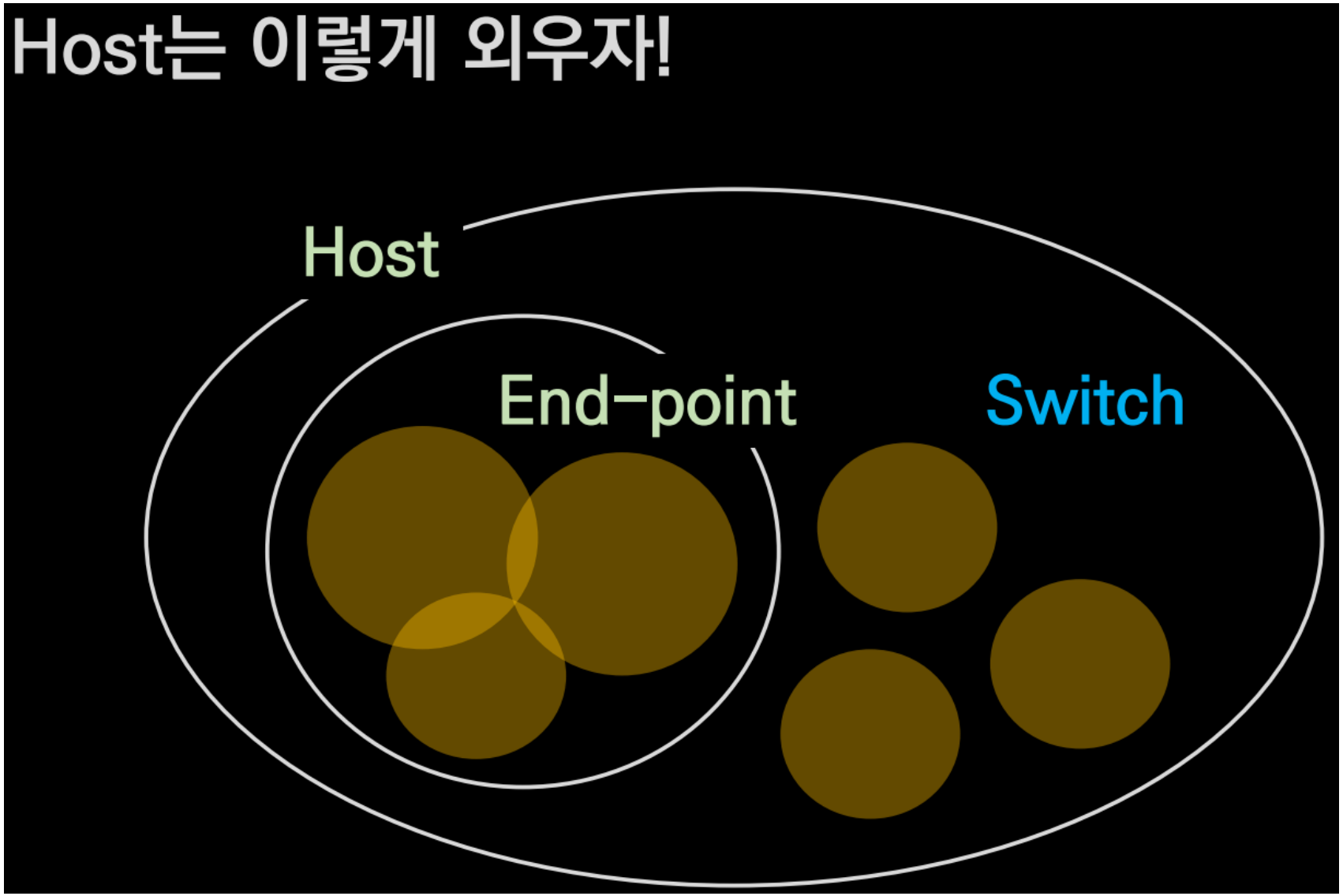
- 호스트에서 생성된 데이터를 여러 네트워크를 거쳐 전송함으로써 서로 다른 네트워크에 속한 호스트 간에 데이터를 교환할 수 있게 하는 장비

통신 프로토콜

- 호스트와 라우터, 라우터와 라우터, 호스트와 호스트가 통신하기 위한 정해진 절차와 방법

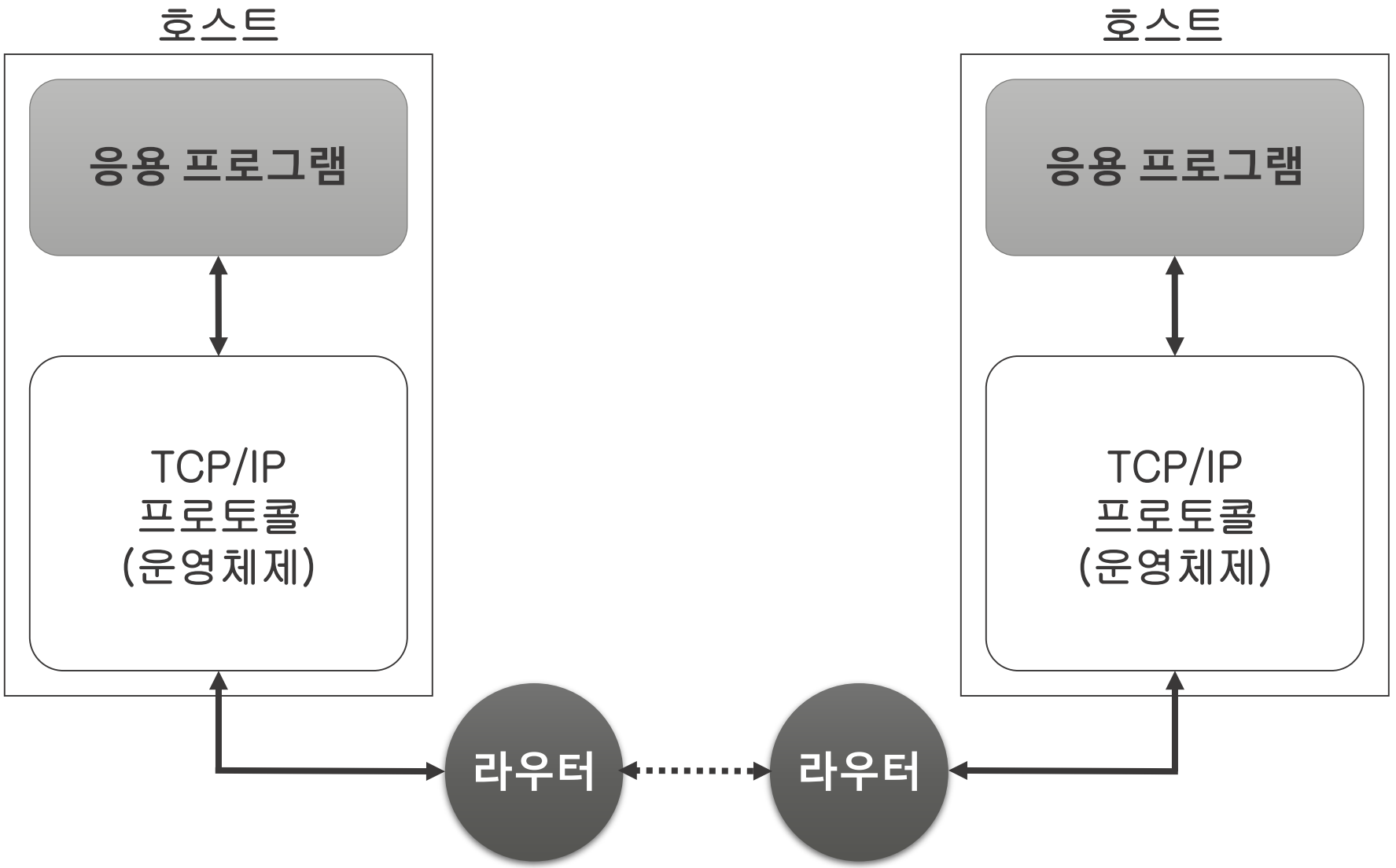
Part 1

인터넷의 구성 요소



Part 1

TCP/IP 프로토콜



Part 1

TCP/IP 프로토콜의 구조

Telnet, FTP, HTTP,
SMTP, MIME, SNMP, …

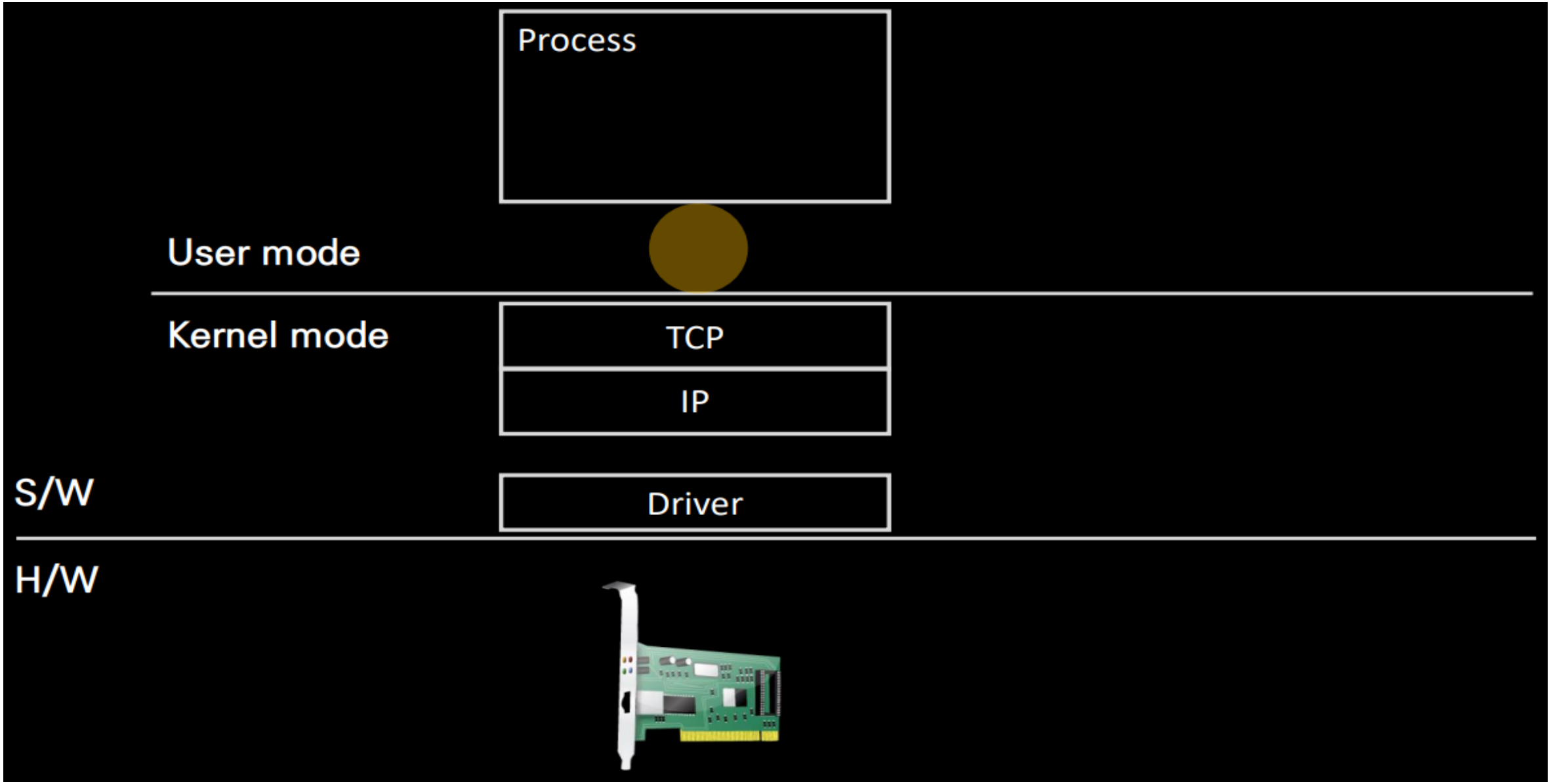
TCP, UDP

IP

디바이스 드라이버
네트워크 하드웨어(NIC)

Part 1

계층별 데이터 단위



Part 1

L1, L2 계층

네트워크 접근 계층

- 역할 : 물리적 네트워크를 통한 데이터 송수신
- 구성 요소 : 네트워크 하드웨어(NIC) + 장치 드라이버
- 주소 지정 방식 : 물리 주소

  예) 이더넷: 48비트 물리 주소

```
goodgeni@강민우:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.220.144.102  netmask 255.255.255.0  broadcast 10.220.144.255
        inet6 fe80::6b94:512d:f200:f387  prefixlen 64  scopeid 0xfd<compat,link,site,host>
        ether 00:01:2e:6d:6c:6b  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 1500
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0xfe<compat,link,site,host>
        loop  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Part 1

L3 계층

인터넷 계층

- 역할
  - 네트워크 접근 계층의 도움을 받아 데이터를 목적지 호스트까지 전달
- 구성 요소
  - IP 주소 + 라우팅(*or* 라우터)
- 주소 지정 방식
  - IP 주소
    - 소프트웨어적으로 정의된 논리 주소
    - 전 세계적인 유일성과 하드웨어 독립성을 가짐
- 라우팅
  - 데이터를 목적지까지 전달하는 일련의 작업
    - 라우팅에 필요한 정보 수집
    - 라우팅 정보를 기초로 데이터 전달

Part 1

L4 계층

전송 계층

- 역할
  - 최종 통신 목적지(응용 프로그램)를 지정하고, 오류 없이 데이터를 전송
    - 데이터 손실 또는 손상을 검출해 잘못된 데이터가 목적지에 전달되는 일을 방지
- 주소 지정 방식
  - 포트 번호
- 대표 프로토콜
  - TCP
  - UDP

Part 1

L4 계층 → TCP와 UDP

TCP	UDP
연결형(connection-oriented) 프로토콜 - 연결 설정 후 통신 가능	비연결형(connectionless) 프로토콜 - 연결 설정 없이 통신 가능
신뢰성 있는 데이터 전송 - 데이터를 재전송함	신뢰성 없는 데이터 전송 - 데이터를 재전송하지 않음
일대일 통신(unicast)	일대일 통신(unicast), 일대다 통신(broadcast, multicast)
데이터 경계 구분 안 함 - 바이트 스트림(byte-stream) 서비스	데이터 경계 구분함 - 데이터그램(datagram) 서비스

응용 계층

- 역할
  - 전송 계층을 기반으로 한 다수의 프로토콜과 이 프로토콜을 사용하는 응용 프로그램을 포괄
- 대표 프로토콜
  - Telnet, FTP, HTTP, SMTP, ...

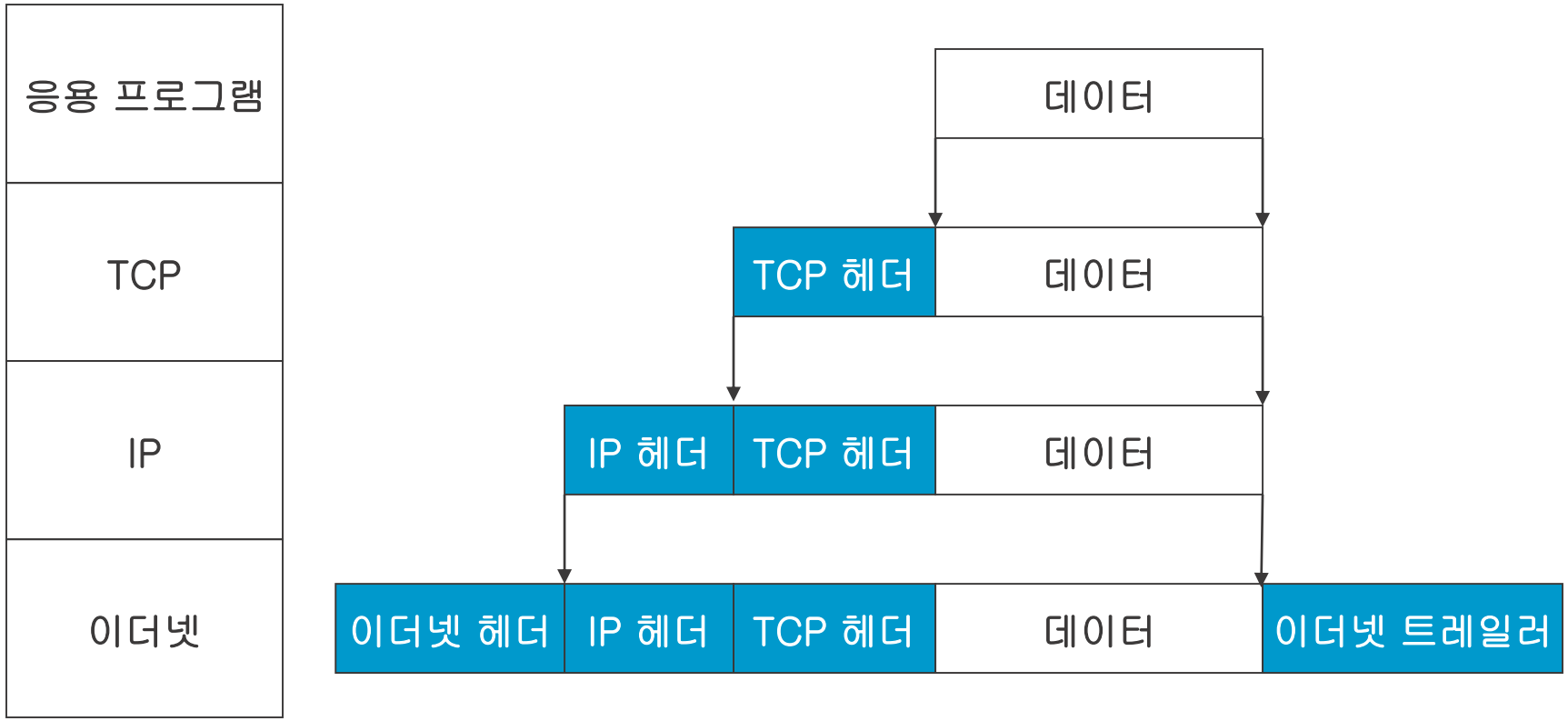
데이터 전송 원리 (1)

- **패킷이란?**
  - 각 프로토콜에서 정의한 제어 정보(IP 주소, 포트 번호, 오류 체크 코드 등) + 데이터
  - 제어 정보의 위치에 따라 앞쪽에 붙는 헤더(*header*)와 뒤쪽에 붙는 트레일러(*trailer*)로 구분

데이터 전송 원리 (2)

패킷 전송 형태

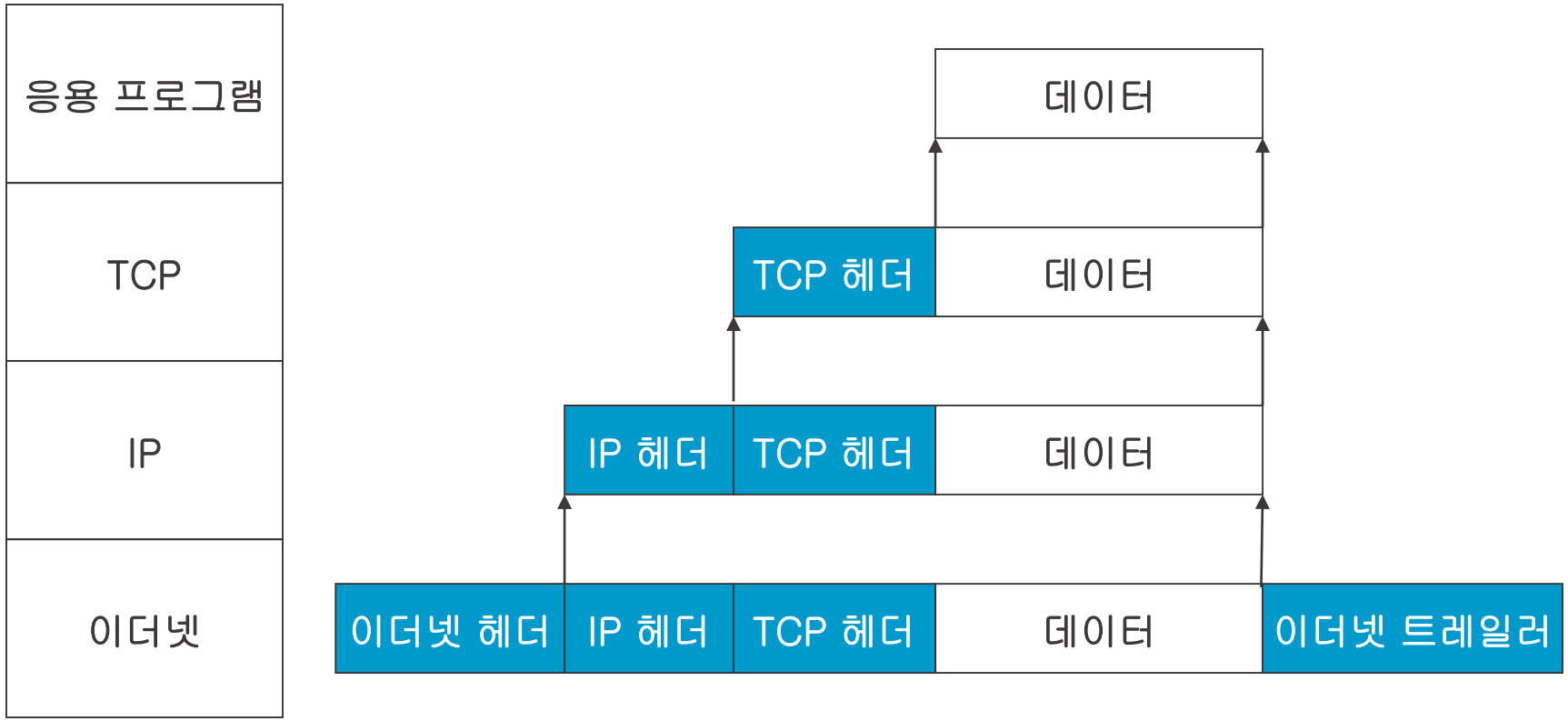
- 송신측



데이터 전송 원리 (3)

패킷 전송 형태

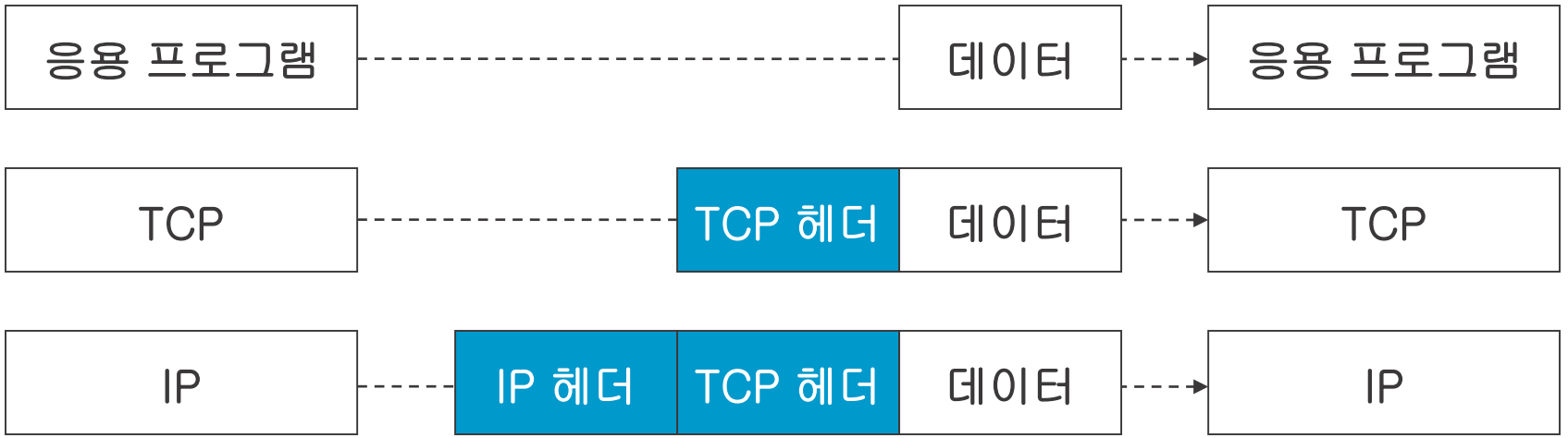
- 수신측



데이터 전송 원리 (4)

패킷 전송 형태

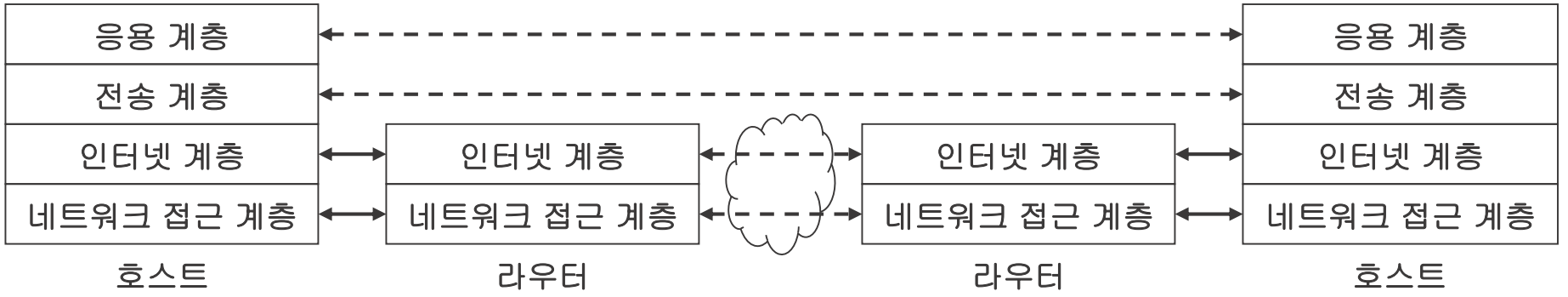
- 계층별
  - 각 계층은 동일 위치의 상대 계층과 통신하는 것으로 간주



데이터 전송 원리 (5)

패킷 전송 형태

- 인터넷
  - 응용 계층, 전송 계층
    - 하부 계층이 제공하는 가상적인 연결을 사용해 동작
  - 인터넷 계층
    - IP 주소와 라우팅 기능을 이용해 패킷 전송 경로 결정
  - 네트워크 접근 계층
    - 물리 주소를 사용해 실제 패킷 전송



IP 주소, 포트 번호 (1)

IP 주소

- 인터넷에 있는 호스트와 라우터의 식별자
  - 폐쇄된 네트워크이거나 IP를 공유하는 경우가 아니면 전 세계적으로 값이 유일
- IPv4는 32비트, IPv6는 128비트 사용
- IPv4는 8비트 단위로 .(dot)로 구분하여 10진수 4개로 표기 ☞ *dotted-decimal notation*
  - 예) 147.46.114.70
- IPv6는 16비트 단위로 :(colon)으로 구분하여 16진수 8개로 표기 ☞ *colon-hexadecimal notation*
  - 예) 2001:0230:abcd:ffab:0023:eb00:ffff:1111

IP 주소, 포트 번호 (2)

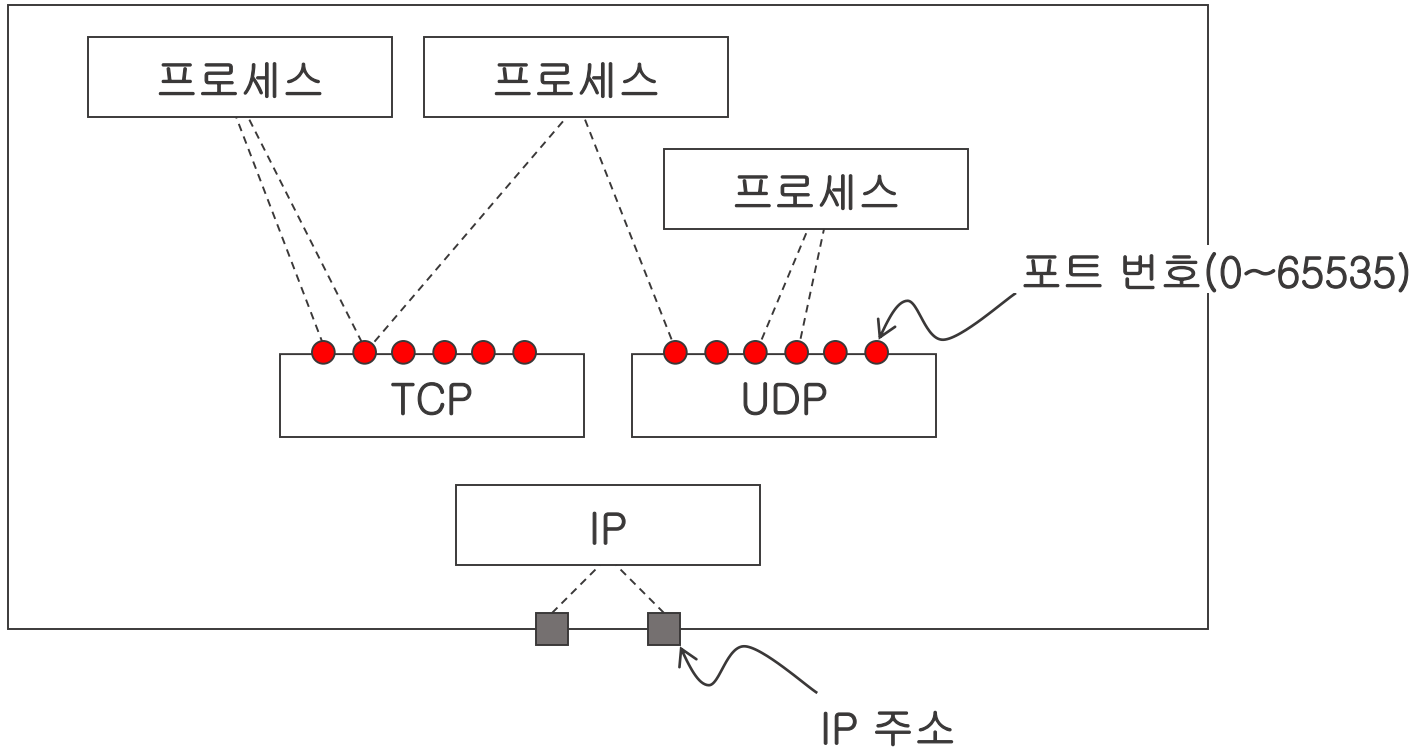
포트 번호

- 인터넷 통신의 종착점(하나 혹은 여러 프로세스)을 나타내는 식별자
- TCP와 UDP는 포트 번호로 부호 없는 16비트 정수를 사용하므로 0~65535 범위가 가능
- 영역별 포트 번호

포트 번호	분류
0 ~ 1023	알려진 포트(well-known ports)
1024 ~ 49151	등록된 포트(registered ports)
49152 ~ 65535	동적/사설 포트(dynamic and/or private ports)

IP 주소, 포트 번호 (3)

IP 주소와 포트 번호



IP 주소, 포트 번호 (4)

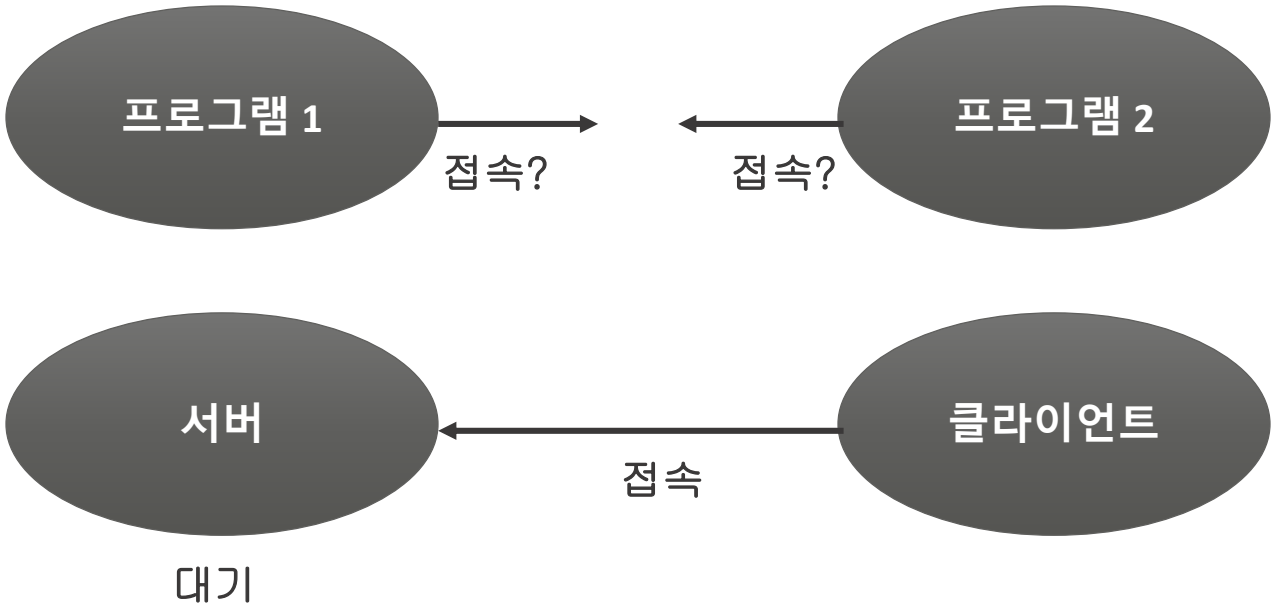
도메인 이름

- IP 주소에 대한 (기억하기 쉬운) 별명
- 실제 통신할 때는 IP 주소로 변환해야 함

클라이언트-서버 모델

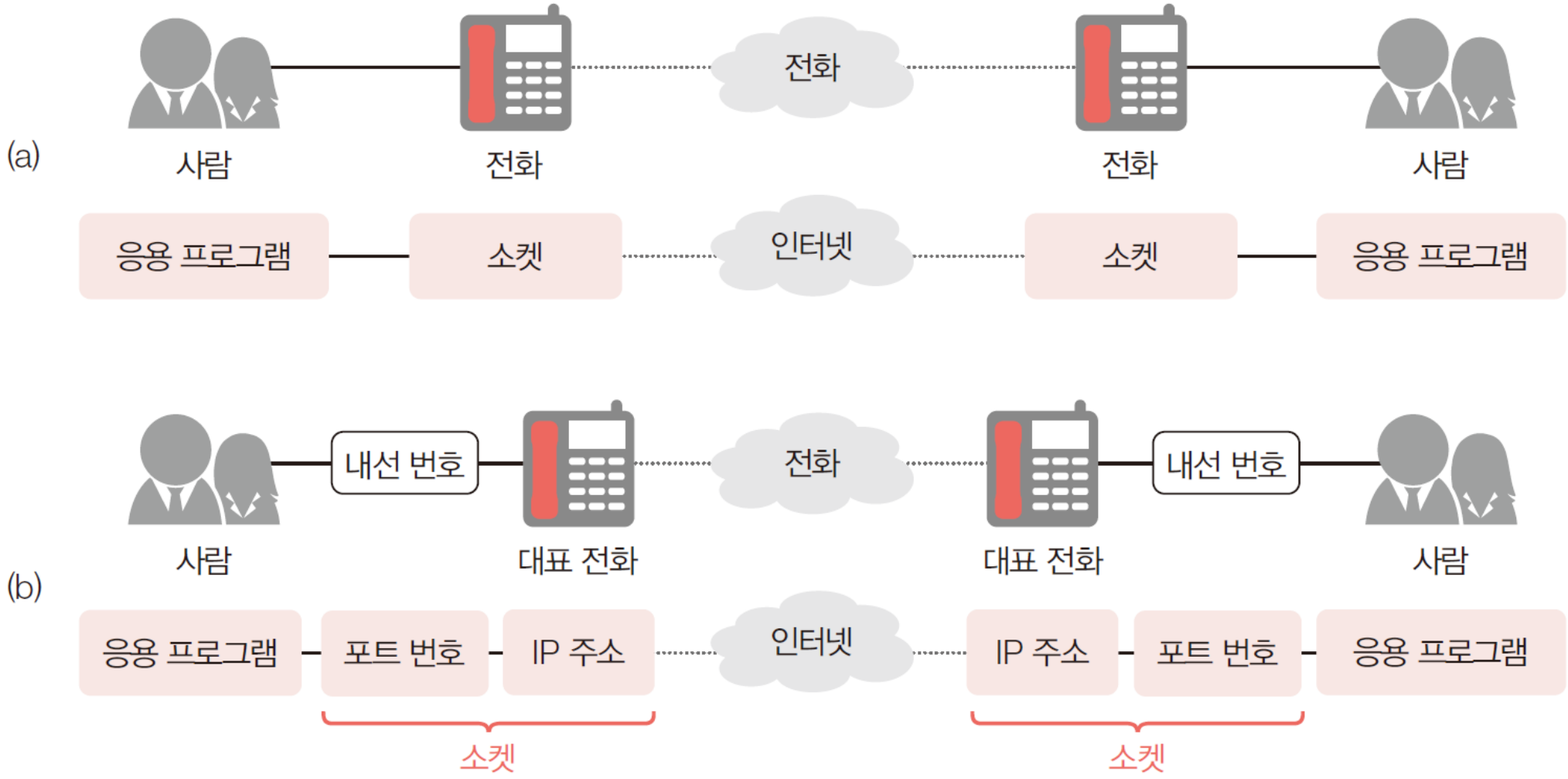
클라이언트-서버(Client/Server) 모델

- 두 프로그램이 상호 작용하는 방식을 나타내는 용어
- 서비스를 요청하는 쪽은 클라이언트(Client), 클라이언트가 요청하는 서비스를 처리하는 쪽은 서버(Server)



소켓의 개념 (1)

전화 통신과 소켓 통신 비교



소켓의 개념 (2)

- **세 가지 관점**
  - ① 데이터 타입
  - ② 통신 종단점
  - ③ 네트워크 프로그래밍 인터페이스

소켓의 개념 (3)

데이터 타입

- 파일 디스크립터 혹은 핸들과 유사한 개념
- 생성과 설정 과정이 끝나면 운영체제의 통신 관련 정보를 참조해 다양한 작업을 편리하게 할 수 있는 데이터 타입

```
int fd = open("myfile", ...); // 파일 생성
...
read(fd, ...);  // 데이터 읽기
write(fd, ...); // 데이터 쓰기
```

(a) 리눅스 파일 입출력

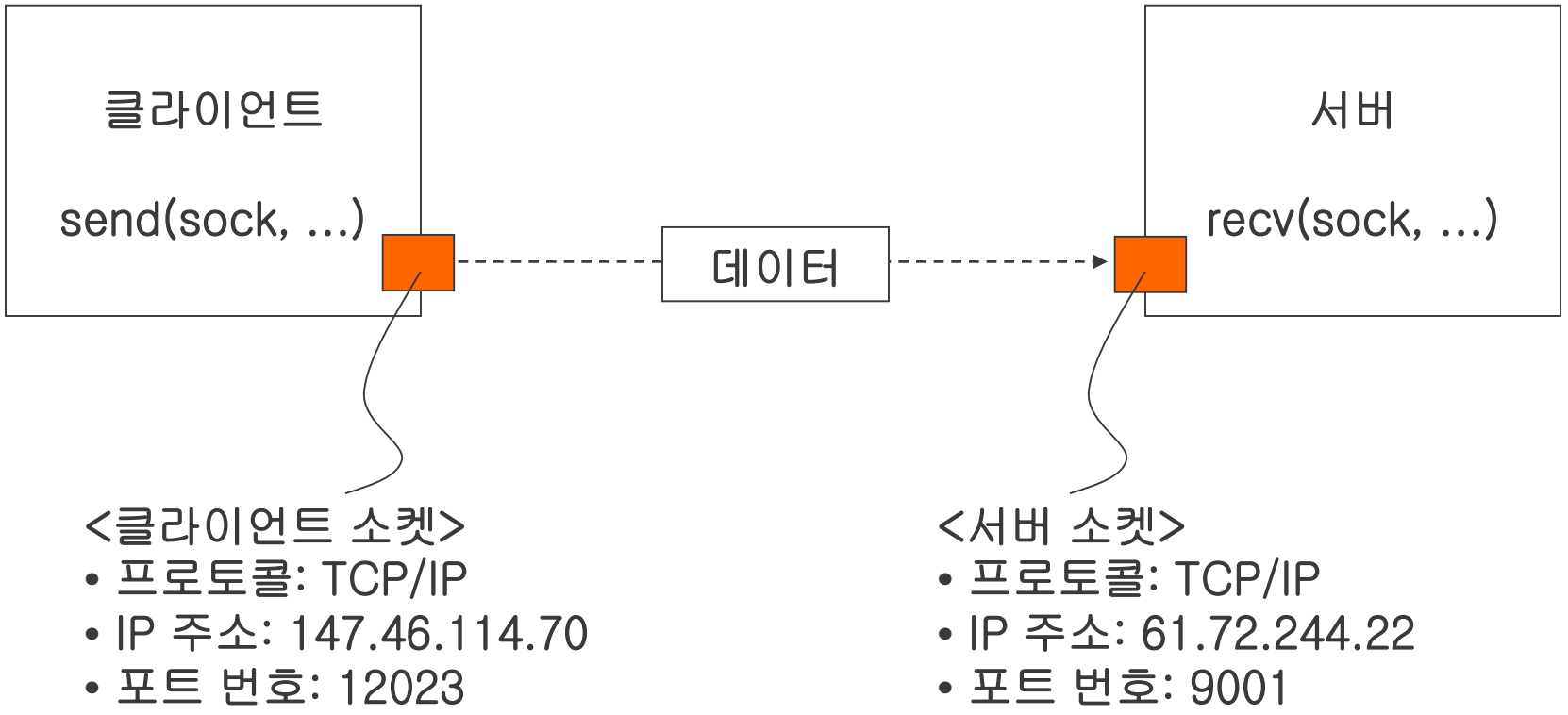
```
SOCKET sock = socket(...); // 소켓 생성
...
recv(sock, ...); // 데이터 받기
send(sock, ...); // 데이터 보내기
```

(b) 윈도우 소켓 통신

소켓의 개념 (4)

통신 종단점

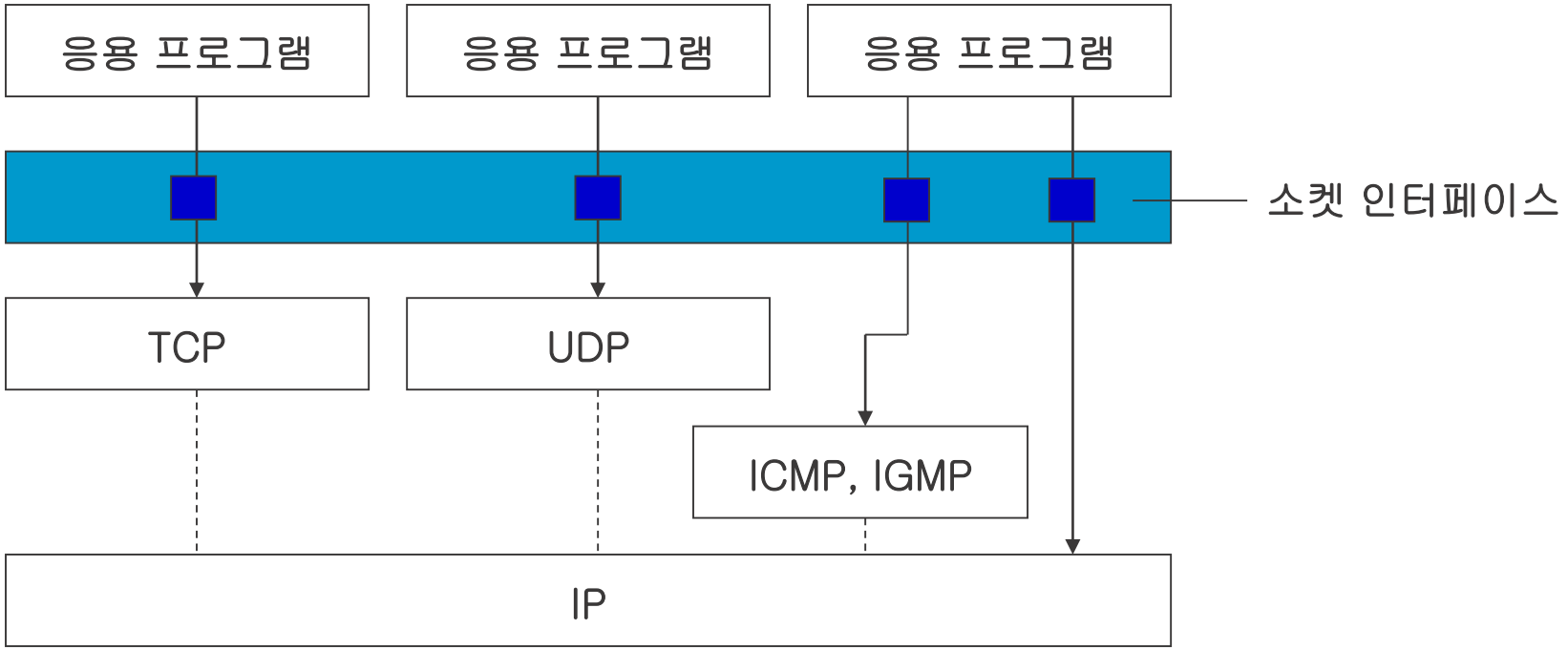
- 응용 프로그램은 자신의 소켓이 상대편의 소켓과 연결된 것으로 생각하고 데이터를 주고받음



소켓의 개념 (5)

네트워크 프로그래밍 인터페이스

- 통신 양단이 모두 소켓을 사용할 필요는 없음
- TCP/IP 프로토콜에서 (일반적으로) 응용 계층과 전송 계층 사이에 위치하는 것으로 간주



윈도우 소켓 (1)

윈도우 소켓(윈속)

- 버클리 유닉스에서 개발한 네트워크 프로그래밍 인터페이스를 윈도우 환경에서 사용할 수 있게 만든 것
- 윈도우 95 버전부터 API에 정식으로 포함하여 제공

윈도우 소켓 (2)

윈속의 장점

- 유닉스 소켓과 소스 코드 수준에서 호환성이 높으므로 기존 코드를 이식하여 활용하기 쉬움
- 가장 널리 사용되는 네트워크 프로그래밍 인터페이스이므로 한번 배우면 여러 운영체제(윈도우, 리눅스 등)에서 사용 가능
- TCP/IP 외의 프로토콜도 지원하므로 최소 코드 수정으로 응용 프로그램이 사용할 프로토콜 변경 가능
- 비교적 저수준 프로그래밍 인터페이스이므로 세부 제어가 가능하며 고성능 네트워크 프로그램 개발 가능

윈도우 소켓 (3)

윈속의 단점

- 응용 프로그램 수준의 프로토콜을 프로그래머가 직접 설계해야 함
  - 주고받는 데이터 형식이나 전송 절차 등을 고려해 프로그래밍해야 하며, 설계 변경 시에는 코드 수정이 불가피함
- 서로 다른 바이트 정렬 방식을 사용하거나 데이터 처리 단위가 서로 다른 호스트끼리 통신할 경우, 응용 프로그램 수준에서 데이터 변환을 처리해야 함

리눅스 소켓 프로그램 맛보기 (1)

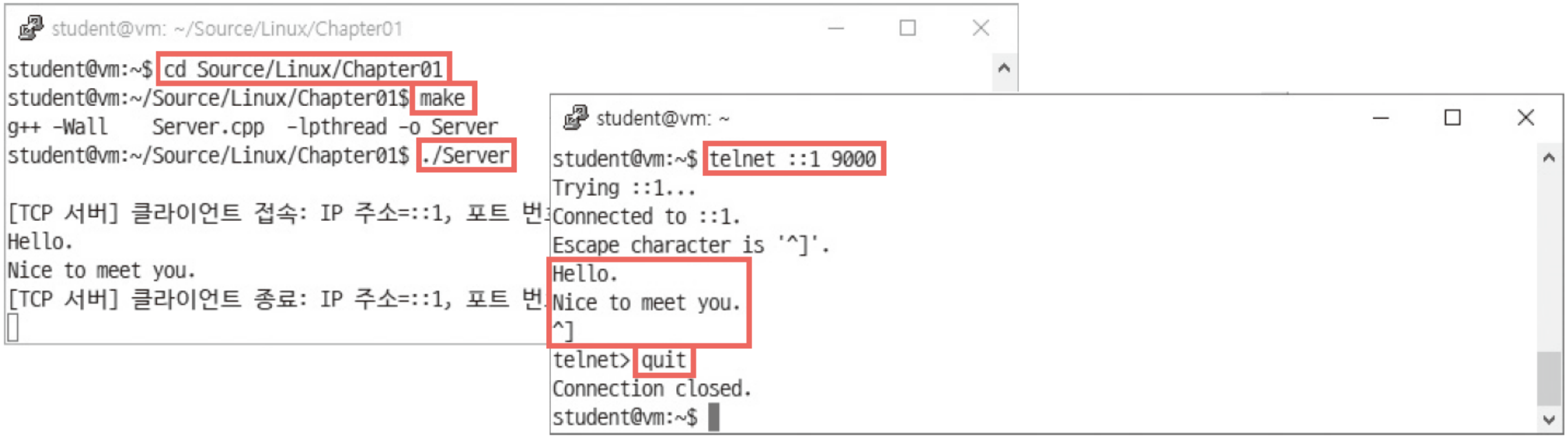
g++ 컴파일러와 make 명령 확인

```
student@vm:~$ g++ --version
g++ (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

student@vm:~$ make --version
GNU Make 4.1
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

리눅스 소켓 프로그램 맛보기 (2)

실행 화면



감사합니다