

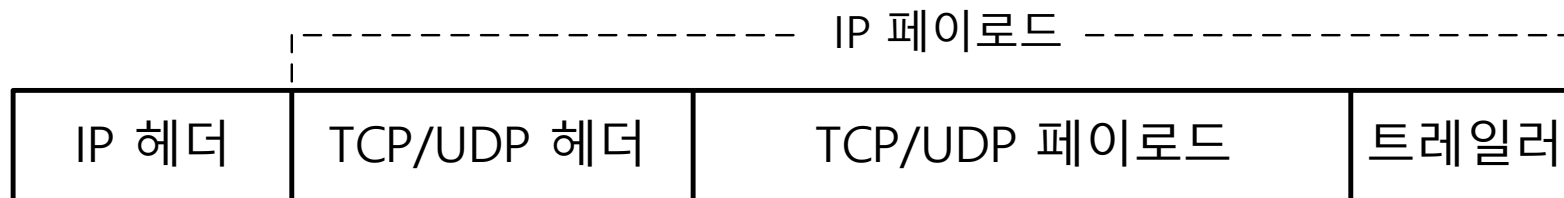
# TCP/IP 모델

## ① 네트워크 인터페이스 계층

- 물리 계층에서 발생할 수 있는 오류를 찾아내고 수정하기 위한 기능적 수단을 제공
  - CRC를 기반으로 인접 노드 간의 비트 오류가 있는지의 여부를 판단
- 네트워크 adapter(예, LAN 카드)를 위한 장치 드라이버 및 소프트웨어
  - MAC 주소 지정: LAN 카드의 일련번호

## ② 네트워크 계층

- 전송 데이터를 데이터그램으로 구성하여 전송
- 경로 배정 기능
  - IP 헤더에 들어있는 목적지 주소를 기반으로 경로를 찾아 데이터그램을 전송
- IP 주소 지정: 인터넷 상의 노드(호스트) 구분
- IP 패킷의 구조



# TCP/IP 모델

## ③ 전송 계층

- ❑ 오류 제어와 흐름 제어를 이용하여 프로세스 사이의 전송을 책임진다
- ❑ 전송 계층 프로토콜: TCP와 UDP가 있음
- ❑ 포트 번호(Port number) 지정 : 최종 통신 목적지(응용 프로그램)

<TCP 헤더>

송신 포트번호	수신 포트번호
패킷순서번호	
패킷확인응답번호	
헤더길이 및 제어 플레그	윈도우 크기
체크섬	긴급 데이터 포인터
옵션과 패딩	

<UDP 헤더>

발신포트	수신포트	UDP 길이	UDP 체크섬
------	------	--------	---------

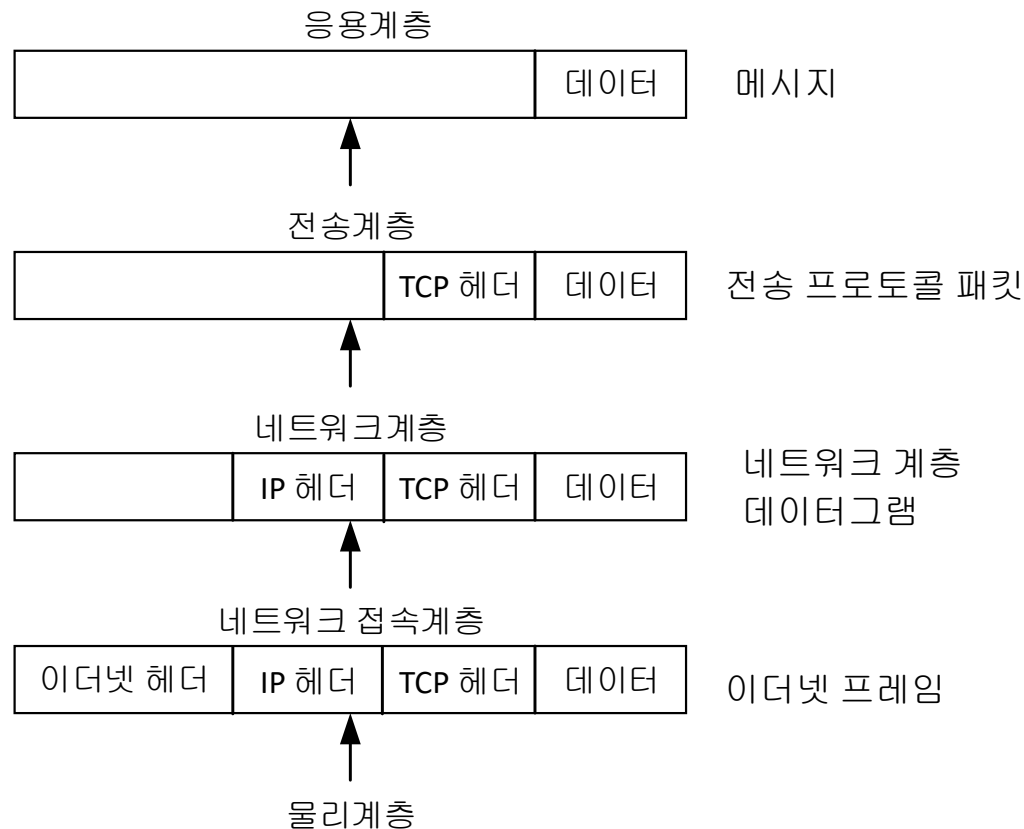
## ④ 응용 계층

- ❑ 최종 사용자에게 서비스를 제공하는 역할
- ❑ 프로토콜: WWW, e-mail, FTP, Telnet 등

# TCP/IP 프로토콜

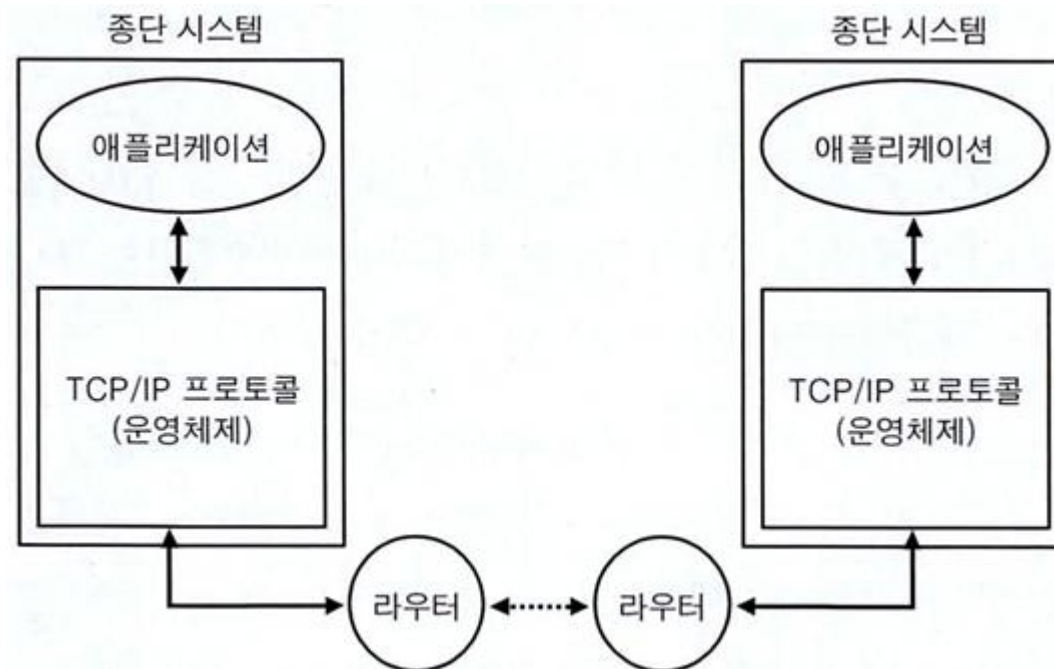
## □ TCP/IP 데이터 송수신 과정

- 상위 계층에서 하위계층으로 전달되면서 각 계층의 헤더가 추가됨
- 하위 계층 → 상위계층으로 전달될 때는 헤더 제거



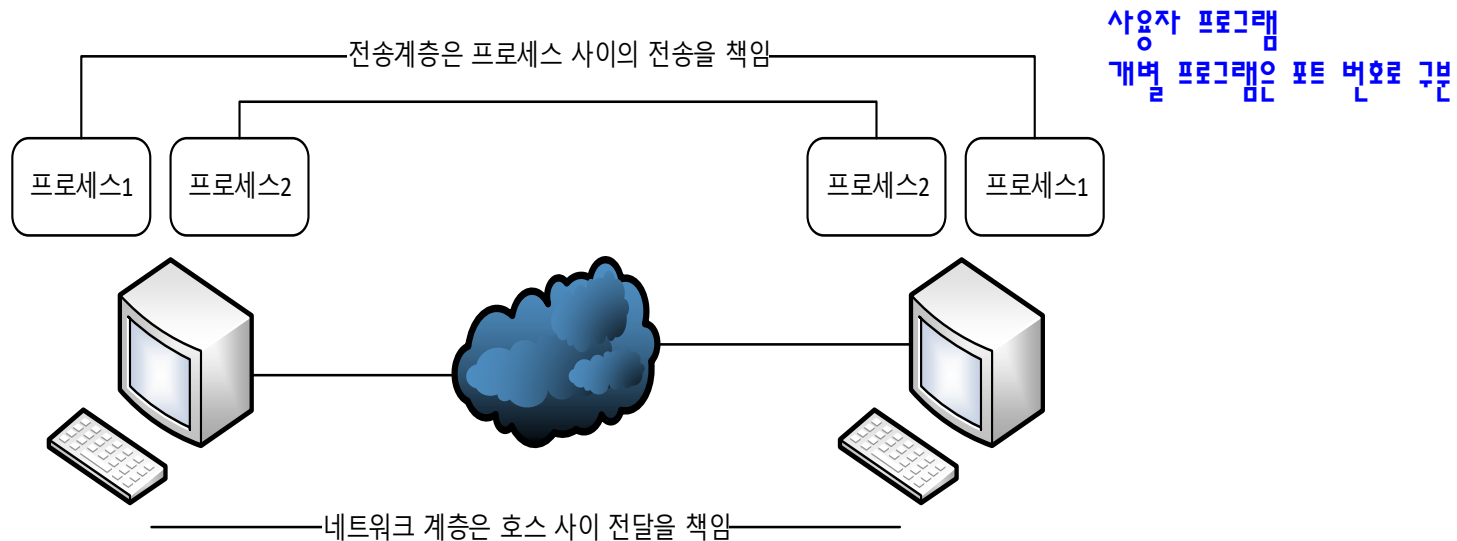
# TCP/IP 프로토콜

- 인터넷 프로그램에서 사용자 프로그램의 위치
  - ▣ 사용자 프로그램은 전송 계층의 서비스를 받아 동작하는 응용 계층 프로그램
  - ▣ 전송 계층까지는 사용자가 프로그램할 필요 없음
  - ▣ 전송 계층이 제공하는 서비스(함수)를 이용하여 사용자 프로그램이 동작

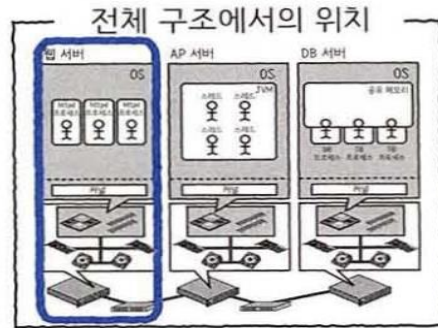


# 전송 종단점(end point) – 소켓(socket)

- 실제 데이터 전송은 프로세스와 프로세스 사이에서 이루어진다.
- 컴퓨터 내부에서 동시에 실행되는 프로세스를 구분하기 위해 포트번호 사용
- 전체적으로 프로세스의 위치를 정확히 나타내기 위해 사용함. 송신자 주소와 포트 번호, 수신자 주소와 포트 번호가 필요 → 전송 종단점은 주소와 포트 번호로 표시



# 소켓: TCP/IP 4계층 모델



## 애플리케이션 계층

이미지 데이터를 보내고 싶어.  
송신은 다른 계층에 맡기자!

## 전송 계층

애플리케이션이 의뢰한 데이터는 책임지고 상대방에게 전달한다!

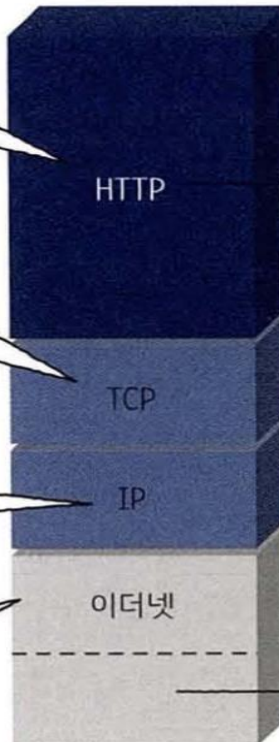
## IP 계층

데이터를 최종 위치까지 운반할게!

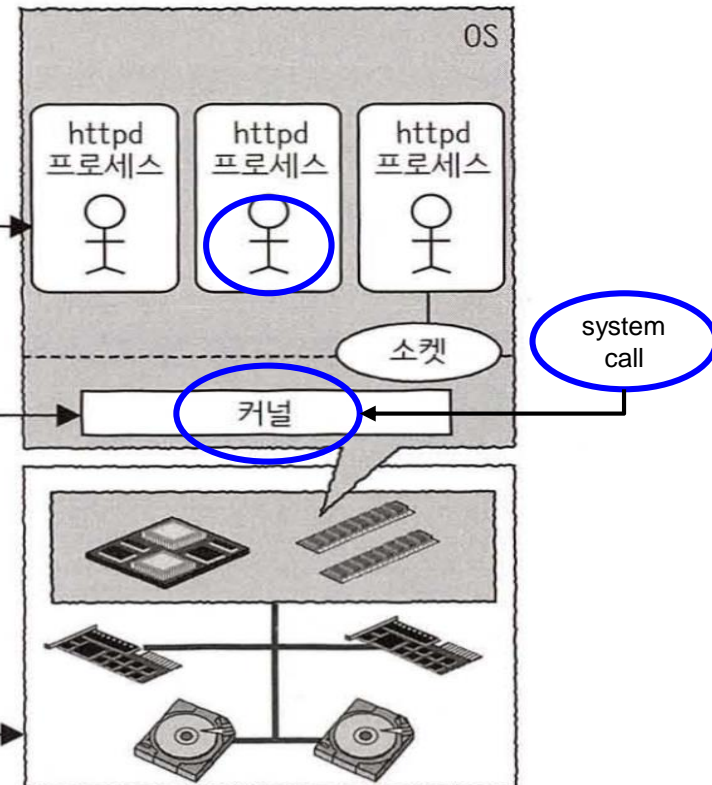
## 링크 계층

직접 연결돼 있는 주변 장비에게도 보내자!

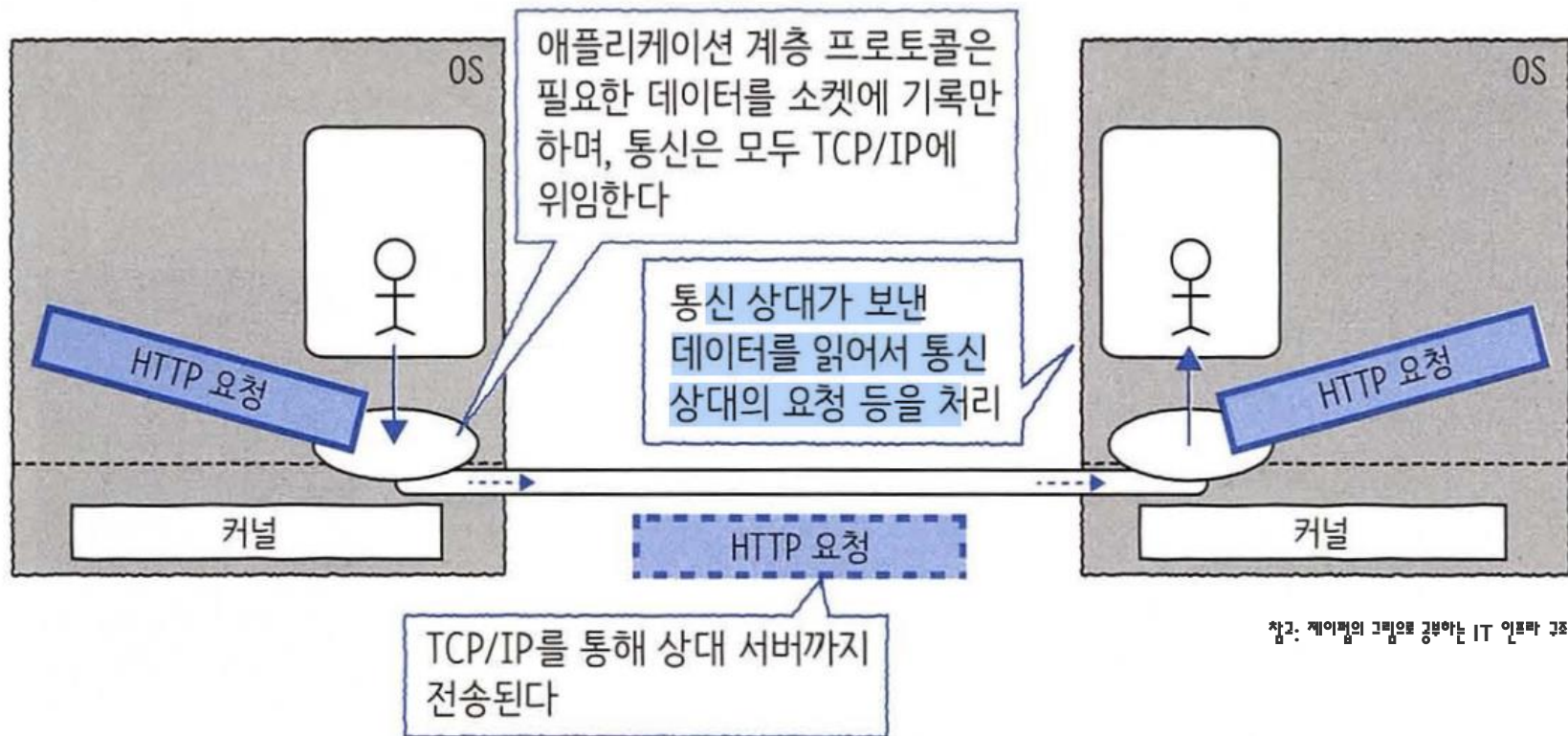
## TCP/IP 4계층 모델 (HTTP 통신 예)



## 각 계층이 실제로 담당하는 위치



# 소켓: L7 (애플리케이션 계층)과 HTTP 프로토콜



참고: 제이펍의 그림으로 공부하는 IT 인프라 구조

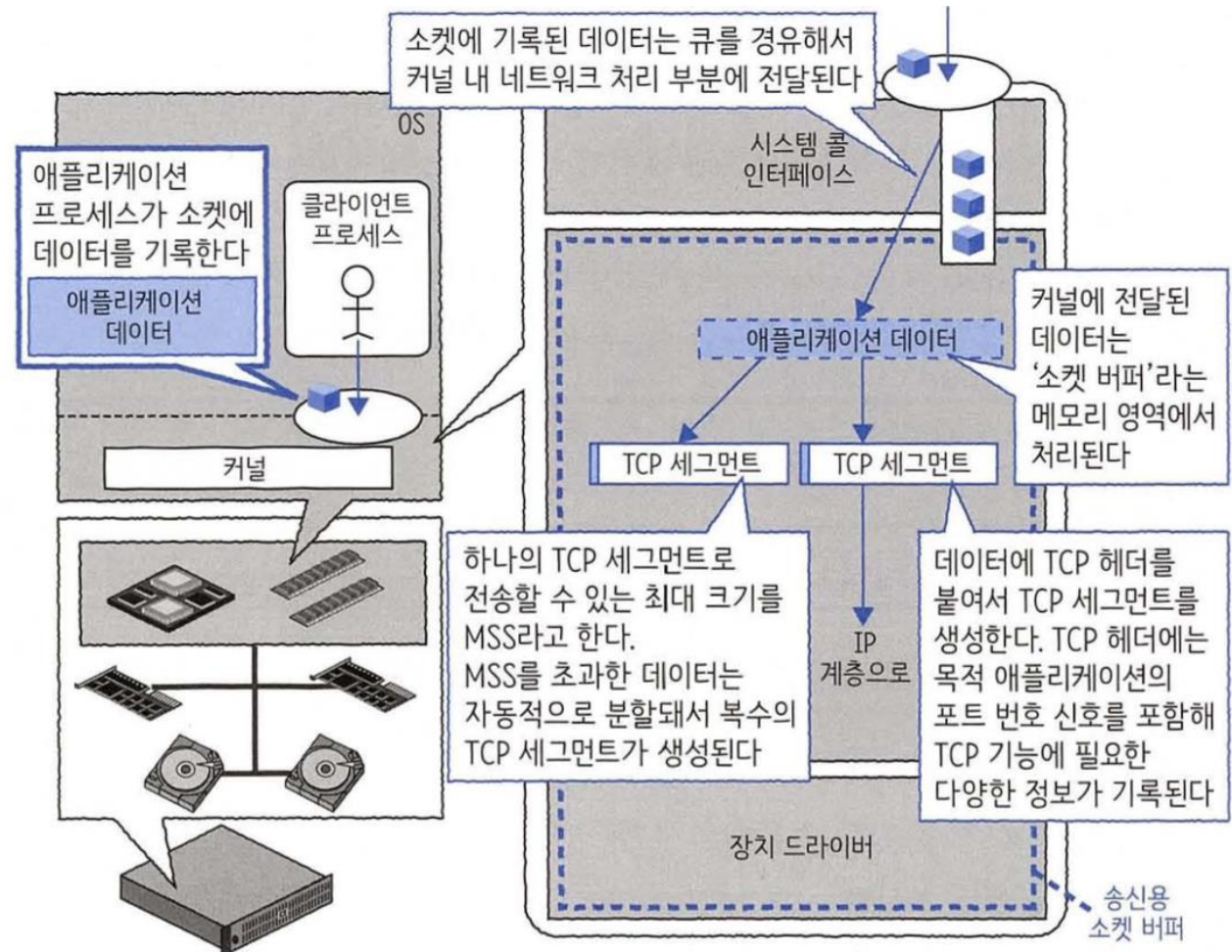
- 운영체제의 커널(Kernel)은 하드웨어와 응용 프로그램 사이에서 인터페이스를 제공하는 역할을 수행한다.
- 운영체제에서 하드웨어의 자원을 프로세스에 나눠주고, 프로세스 제어(작업 관리), 메모리 제어, 프로그램이 운영 체제에 요구하는 시스템 콜 등을 수행한다.
- 하드웨어에 직접적으로 명령하는 작업을 커널이 한다. 현재 많이 사용되고 있는 운영 체제는 커널 위에 여러 가지 소프트웨어 계층을 올린 것이다.

# 소켓: L7 (애플리케이션 계층)과 HTTP 프로토콜

- 애플리케이션 계층에서 사용하는 프로토콜은 HTTP/HTTPS가 있다.
- 애플리케이션 계층은 **소켓**을 사용해 통신처리를 진행한다.
- 애플리케이션 계층에서는 커널에 **상대방 애플리케이션과 통신할 수 있는 회선인 소켓을 열어달라** 요청한다. 이후 열려진 소켓에다 보낼 데이터만 기록하고 나머지 통신 처리는 모두 TCP/IP에게 위임한다.
- 소켓을 열어달라고 의뢰할 때 커널에다가 IP주소와 TCP가 사용할 포트번호를 전달해주면 상대방 서버에도 소켓이 열려 통신을 할수있는 가상경로가 생성된다.(실제는 물리적인 통신케이블을 통해 데이터가 전달된다.)

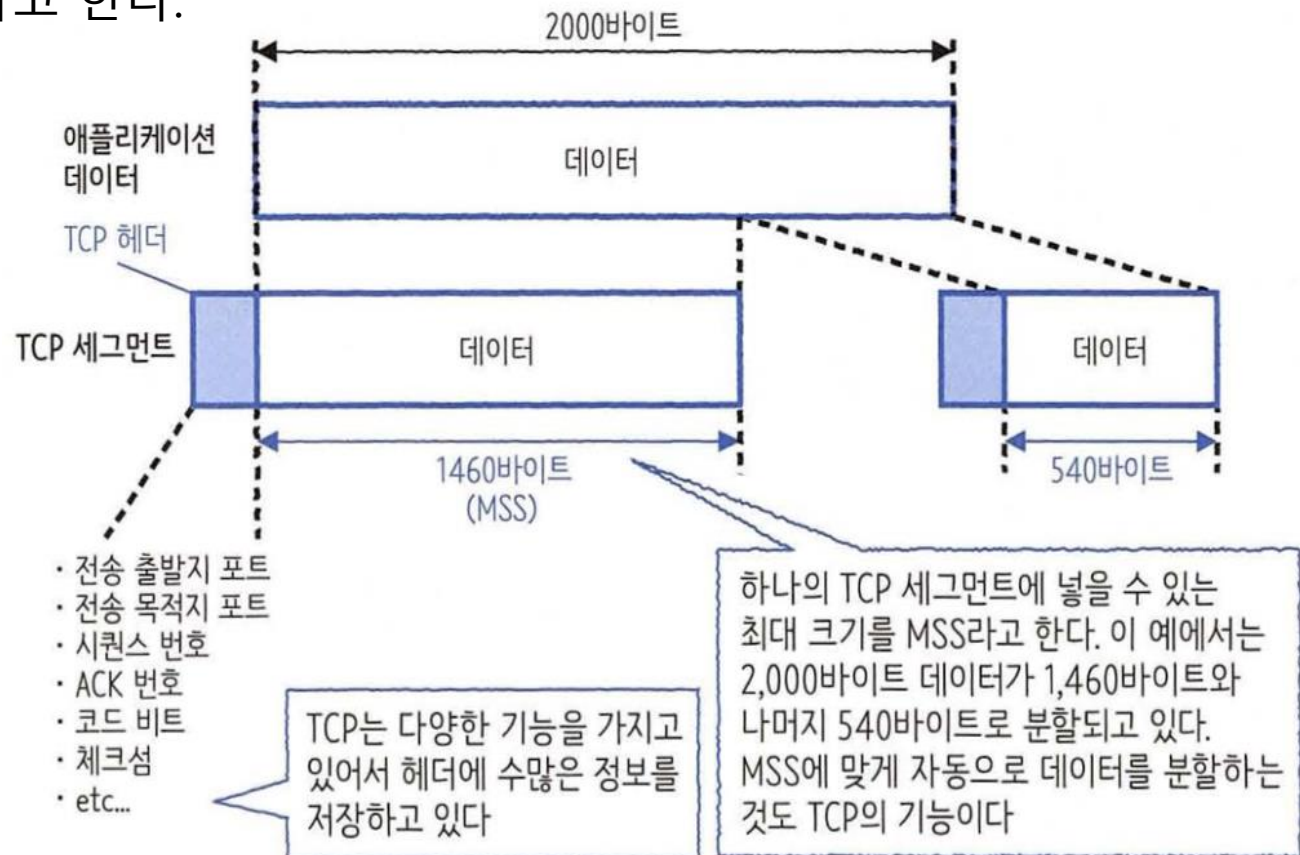
# 소켓: L4 (전송계층)과 TCP 프로토콜

- 소켓에 기록된 애플리케이션 데이터는 소켓의 큐를 경유해서 소켓 버퍼라 불리는 메모리 영역에서 처리된다.
- 소켓 버퍼는 소켓별로 준비된 전용 메모리 영역으로, 이후 계속되는 IP나 이더넷까지의 일련의 처리도 소켓 버퍼 내에서 이루어진다.



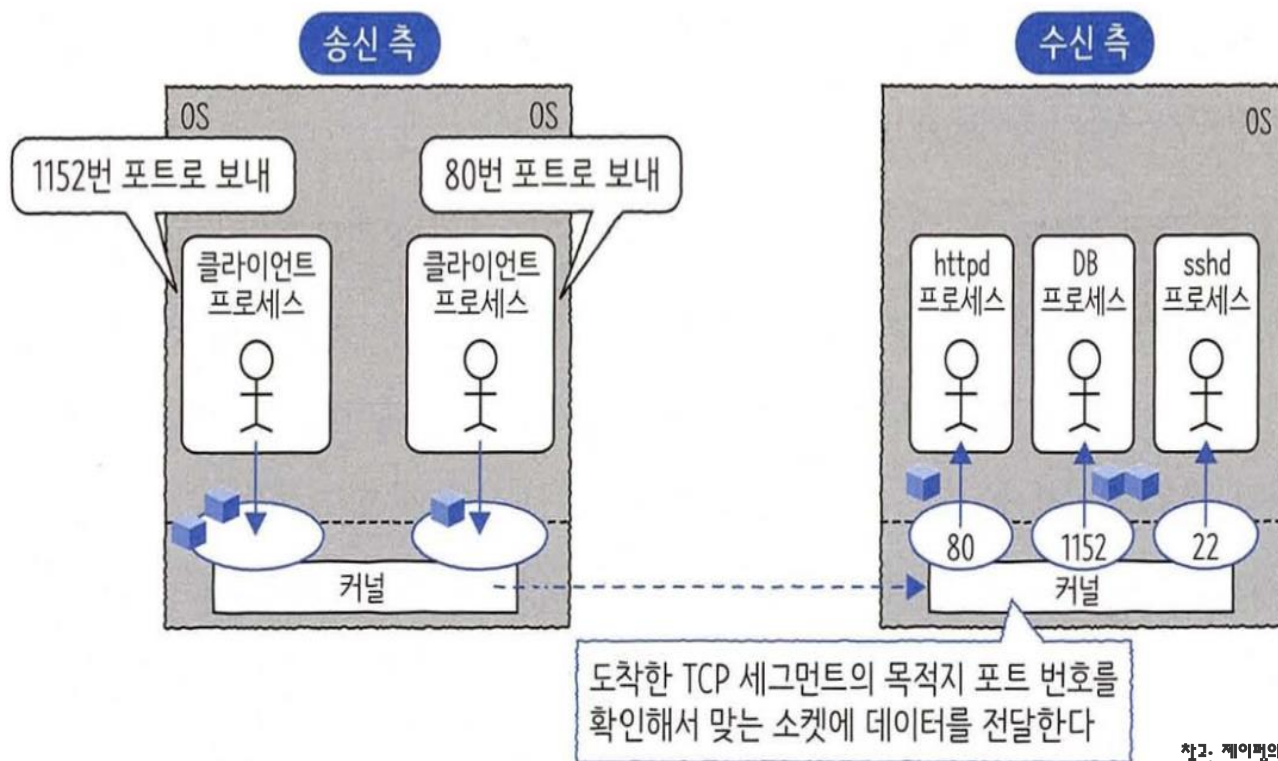
# 소켓: L4 (전송계층)과 TCP 프로토콜

- TCP는 애플리케이션 데이터에 TCP 헤더를 붙여서 TCP 세그먼트를 작성한다. 이 TCP 헤더에는 도착 지점 포트 번호를 포함해서 TCP 기능을 표현하기 위한 수 많은 정보가 기록된다.
- 하나의 TCP 세그먼트로 전송할 수 있는 최대 데이터 크기를 MSS (Maximum Segment Size) 라고 한다.

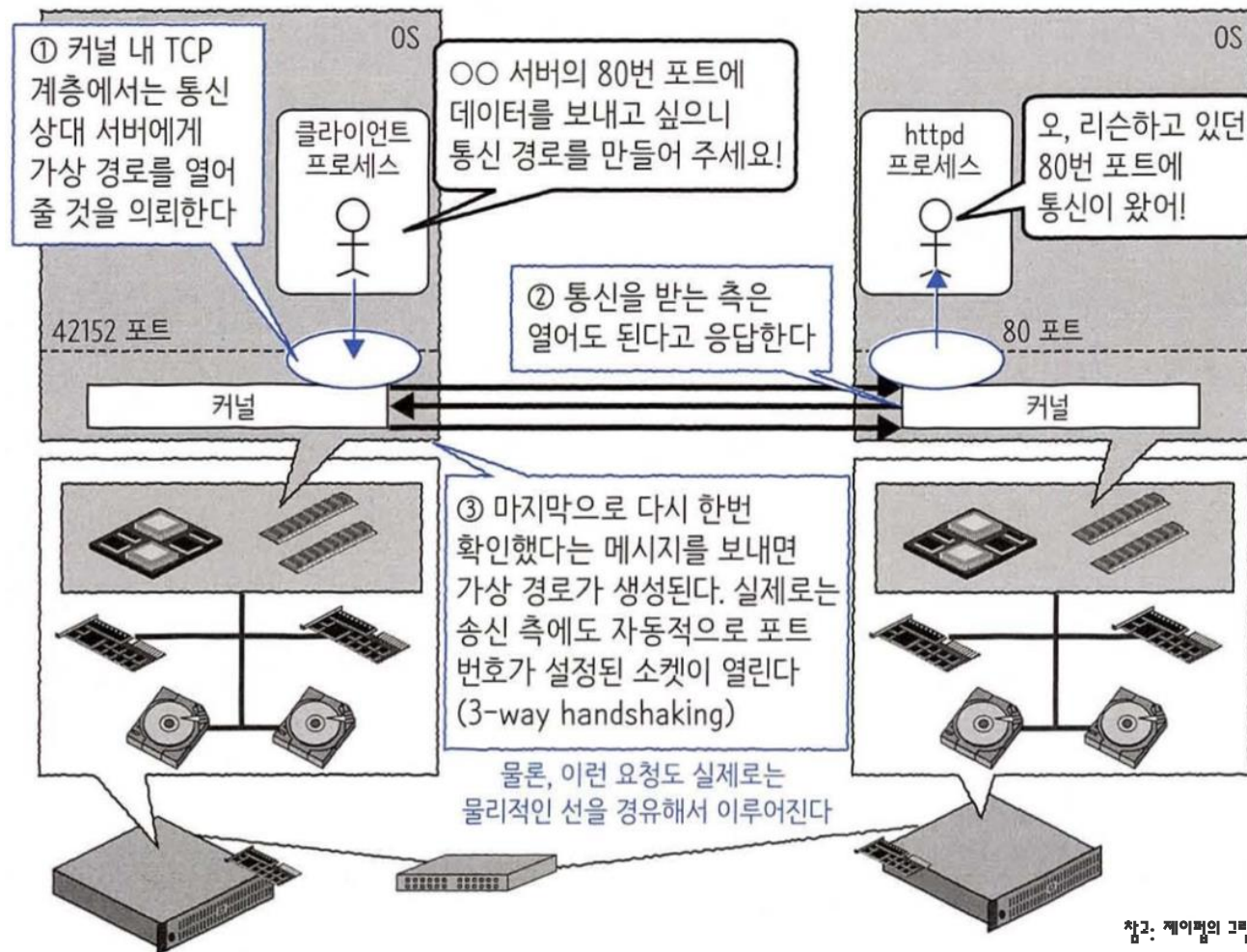


# 소켓: 포트 번호를 이용한 데이터 전송

- 상대 서버에 데이터가 도착했다고 해도 어떤 애플리케이션용 데이터인지 알 수 없다. TCP에서는 포트 번호를 사용해서 어떤 애플리케이션에 데이터를 전달할지 판단한다.
- TCP 포트 번호는 0~65535까지의 숫자를 이용한다.



# 소켓: TCP의 연결(Connection) 생성



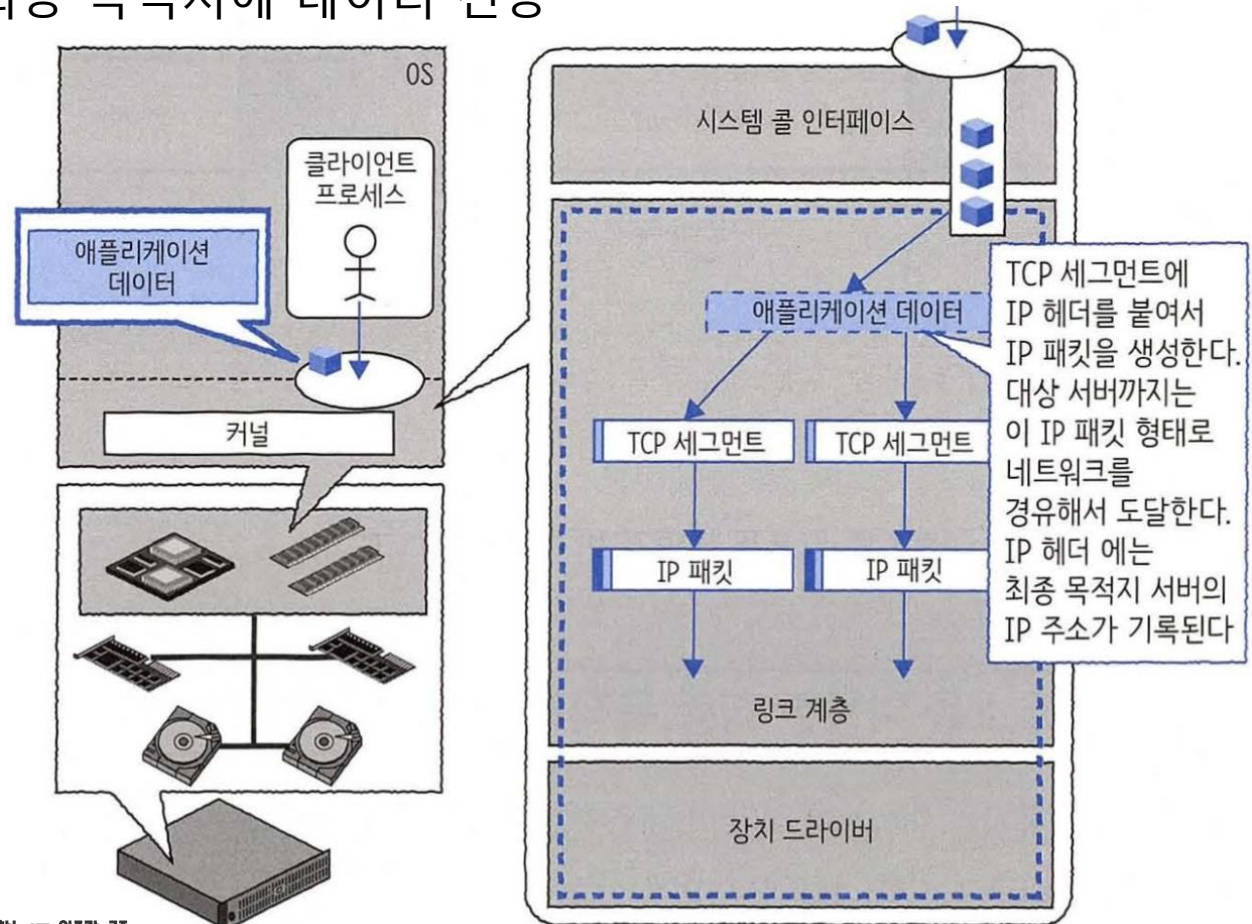
# 소켓: TCP의 연결(Connection) 생성

- ① 서버 프로세스는 통신을 위한 특정 포트를 열어둔다.(Listen)
- ② 클라이언트 프로세스는 연결을 생성하기위해 서버 프로세스가 listen하고있는 포트에 통신할 경로를 열어도 되는지 의뢰한다.
- ③ 서버 프로세스는 클라이언트 프로세스에게 통신 경로를 열어도 된다는 응답을 준다.
- ④ 클라이언트 프로세스는 서버 프로세스가 응답한 내용을 확인했다는 메시지를 다시한번 서버 프로세스에게 보낸다.
- ⑤ 2~4 의 3-way handshaking 이 끝나면 통신할 수 있는 가상경로가 생성된다.



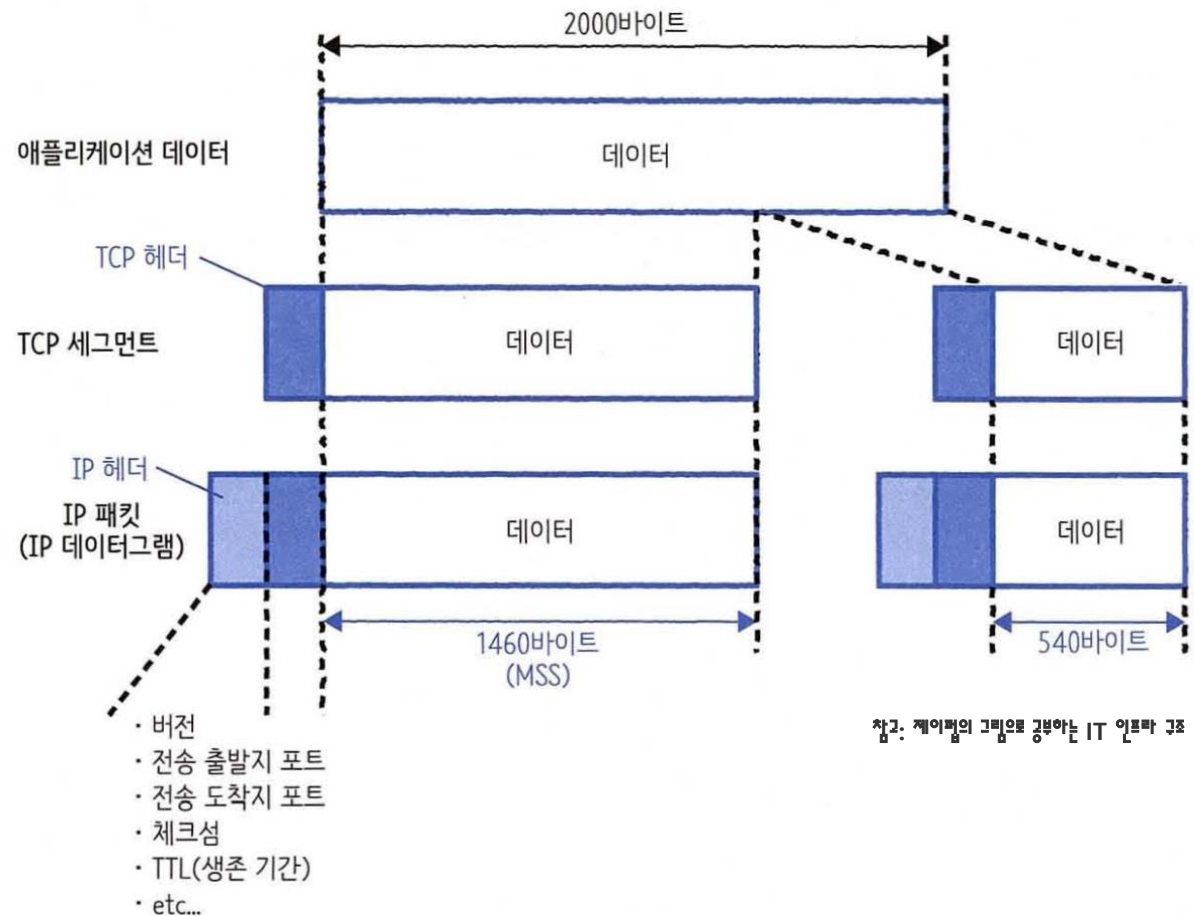
# 소켓: L3(네트워크 계층)과 IP 프로토콜

- IP의 역할은 지정한 대상 서버까지 전달받은 데이터를 전달하는 것
- IP에서는 반드시 전달된다고 보장하지 않음
- IP가 담당하는 기능
  - IP 주소를 이용해서 최종 목적지에 데이터 전송
  - 라우팅(Routing)



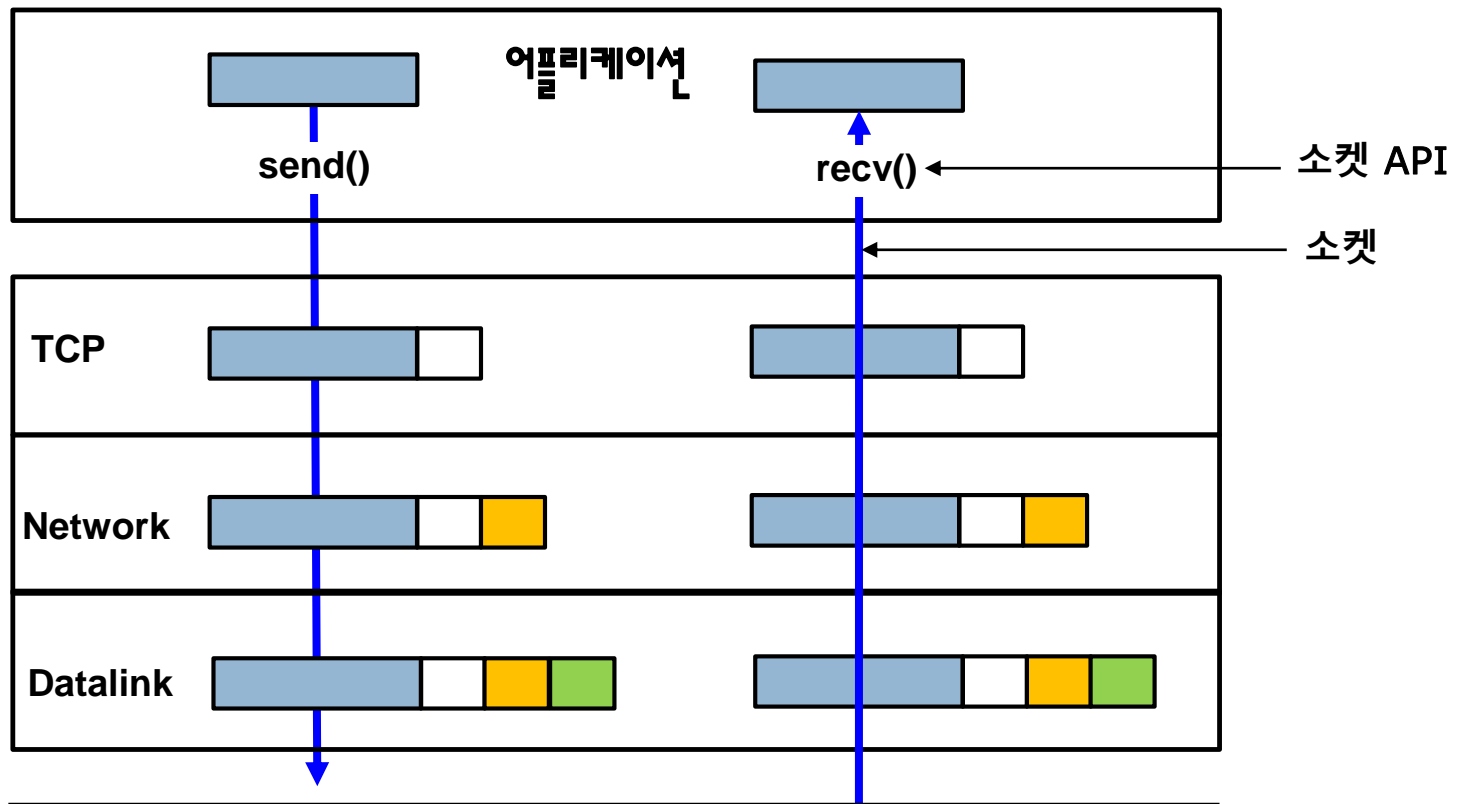
# 소켓: L3(네트워크 계층)과 IP 프로토콜

- IP 계층에서는 최종 목적지가 적힌 IP 헤더를 TCP 세그먼트먼트에 추가해서 IP 패킷을 생성한다.
- 최종 목적지의 IP를 가지는 대상 서버까지 복수의 네트워크를 라우팅을 통해 경유해서 데이터를 전송한다.



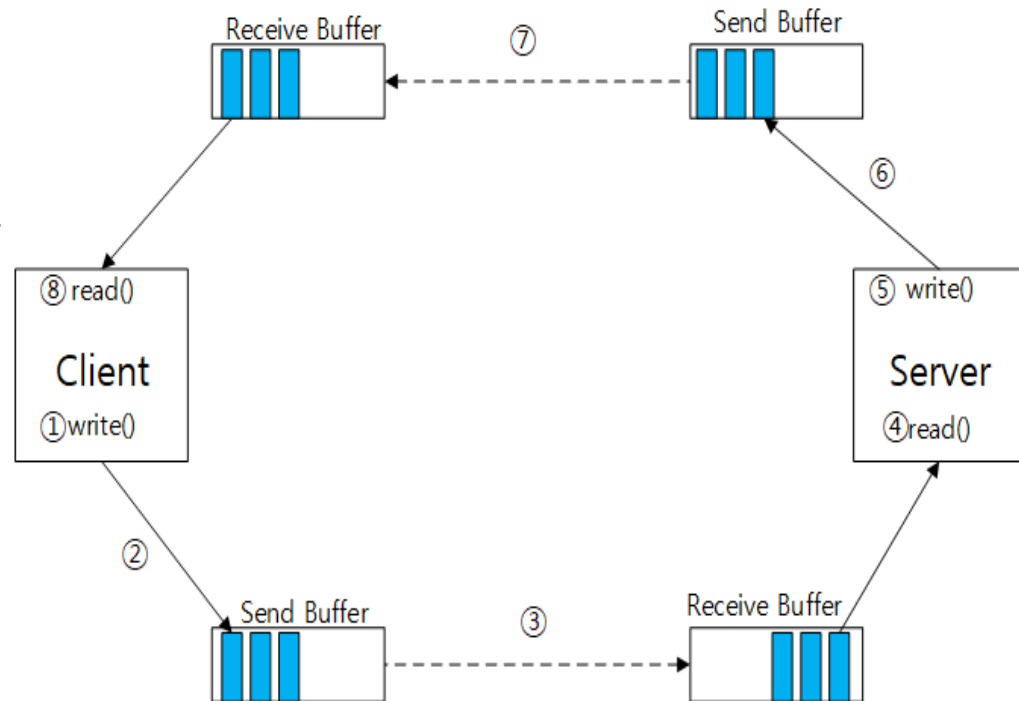
# 소켓: 기능

- 통신 상대방의 IP, Port 번호 설정
- 데이터의 송수신 요청



# 소켓: 송수신 버퍼

- TCP는 데이터를 효율적으로 전송하기 위해 버퍼라고 하는 임시 데이터 공간을 사용한다.
- 응용프로그램으로부터 생성된 소켓은 운영체제로부터 일정 크기의 송신버퍼와 수신버퍼를 할당받는다.
  - 클라이언트, 서버 각각 소켓 버퍼를 가지고 있다.
- 응용 프로그램에서는 데이터를 read/write (recv/send) 하게 되고 해당 데이터는 버퍼를 통해 전송된다.
- 수신자가 소켓을 닫으면
  - 출력 버퍼에 남아 있는 데이터는 계속 해서 전송이 이루어진다.
  - 입력 버퍼에 남아있는 데이터는 소멸된다.



## 소켓: 송수신 버퍼

- ① 서버에 보낼 데이터를 write 한다.
  - ② 데이터가 클라이언트의 송신 버퍼(Send Buffer)에 적재된다.
  - ③ 적재된 데이터가 서버의 수신 버퍼(Receive Buffer)에 전송된다.
  - ④ 서버가 수신버퍼에 적재된 데이터를 read한다.
  - ⑤ 서버에서 클라이언트에게 응답할 데이터를 write 한다.
  - ⑥ 데이터가 서버의 송신버퍼에 적재된다.
  - ⑦ 적재된 데이터가 클라이언트의 수신 버퍼에 전송된다.
  - ⑧ 클라이언트가 수신버퍼에 적재된 데이터를 read한다.
- 사용자가 직접 버퍼 사이즈를 조정할 수도 있다. 단, **설정 값이 그대로 적용되는게 아니라 커널상에서 사용자가 설정한 값을 기반으로 임의로 적용된다는 것이다.**

## 소켓: 송수신 버퍼

- 수신자의 입력 버퍼의 크기가 50byte이고 송신자가 100byte를 보냈을 경우 버퍼가 오버플로우되어 문제가 발생한다?
  - TCP에서는 슬라이딩 윈도우가 있어서 이런 문제가 발생하지 않는다.
  - 슬라이딩 윈도우란 수신자가 자신의 수신 가능한 사이즈(Window Size) 값을 송신자에게 알려주소, 송신자는 수신자가 수신 가능한 만큼만 데이터를 보내는 것이다. 이러한 TCP의 기능을 **흐름 제어(Flow Control)**라 한다.
    - 수신자: 한번에 16658 Bytes까지 수신가능하다(윈도우 사이즈가 16658 Bytes 임)
    - 송신자: 수신자의 요청을 확인
  - 이와 같이 수신자가 수신가능한 만큼만 데이터를 전송하므로 버퍼가 오버플로우되는 경우는 없다.