목차

- 네트워크?
- 프로토콜
- OSI 7계층 모델과 TCP/IP 4계층 모델
- TCP/IP의 구조도
- 계층에서의 식별자

네트워크가 뭐야?

개념적

- 응용 프로그램 사이의 원활한 통신을 제공하기 위한 것

응용 프로그램? : 크롬, 서버 등 프로세스
통신? : 통신 주체 사이의 데이터 교환

물리적

- 여러 링크, 스위치, 라우터의 집합

프로토콜

- 통신을 하려면 서로 같은 방법으로 통신을 해야 한다.

- 이렇게 **통신에 사용되는 약속**을 **프로토콜**이라고 한다.

- 그런데 기술 발달로 시스템이 매번 변하는 상황에서 어떻게 프로토콜을 설정 하는지가 중요

그래서 아주 기가 막힌 방법을 생각하게 되는데…

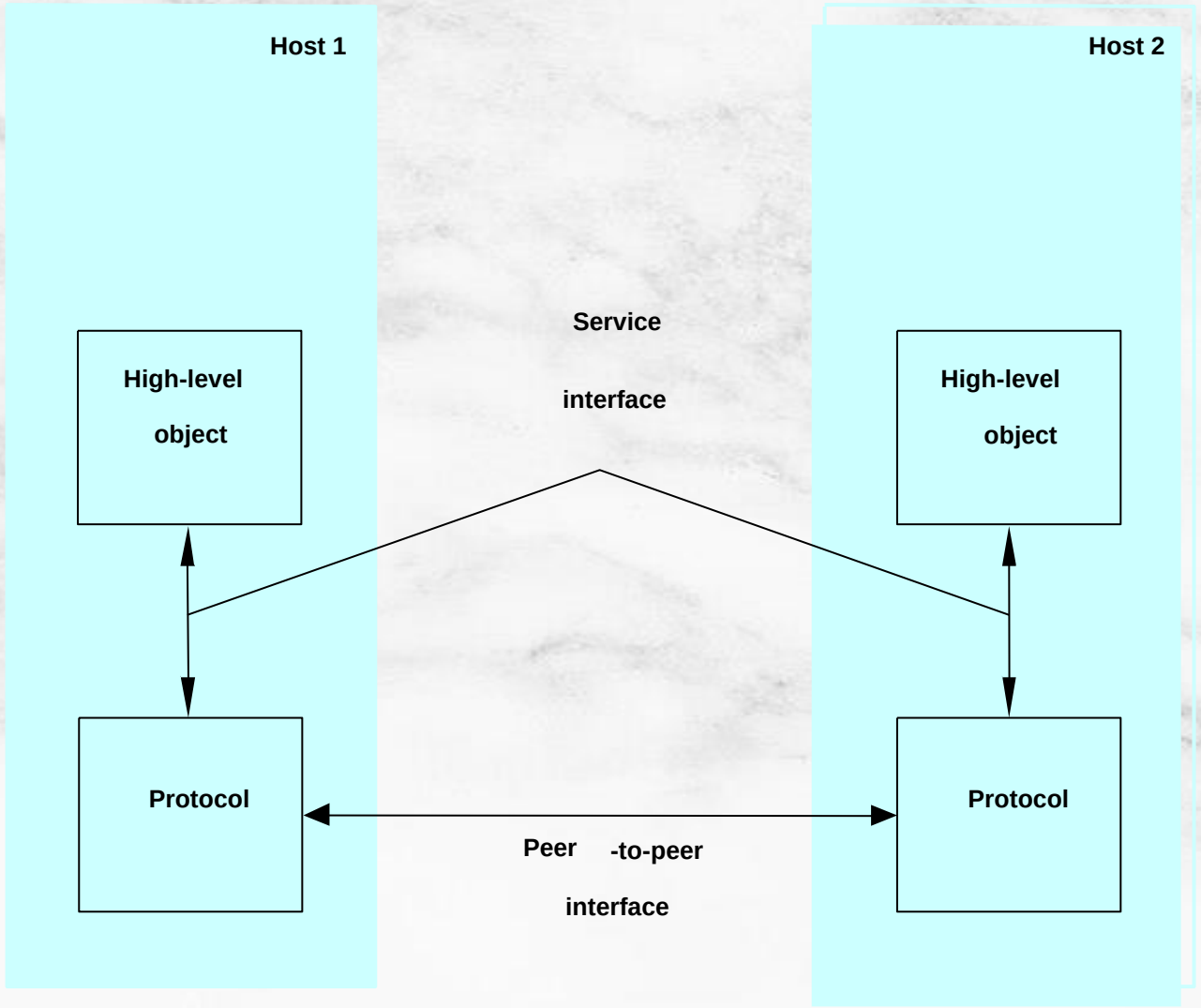
기가 막힌 방법

추상화 (짬 때리기)
- 복잡한 내용은 숨기자. ex) 일단 이런 기능이 있다~
- 추상화된 내용-> 또 추상화

이렇게 되면 자연스럽게 **계층화**가 된다.

따라서 **통신 프로토콜 : 여러 계층으로 정의**

프로토콜의 특징



2개의 인터페이스를 가진다.

1. 서비스 인터페이스
 - 해당 프로토콜의 작업 정의

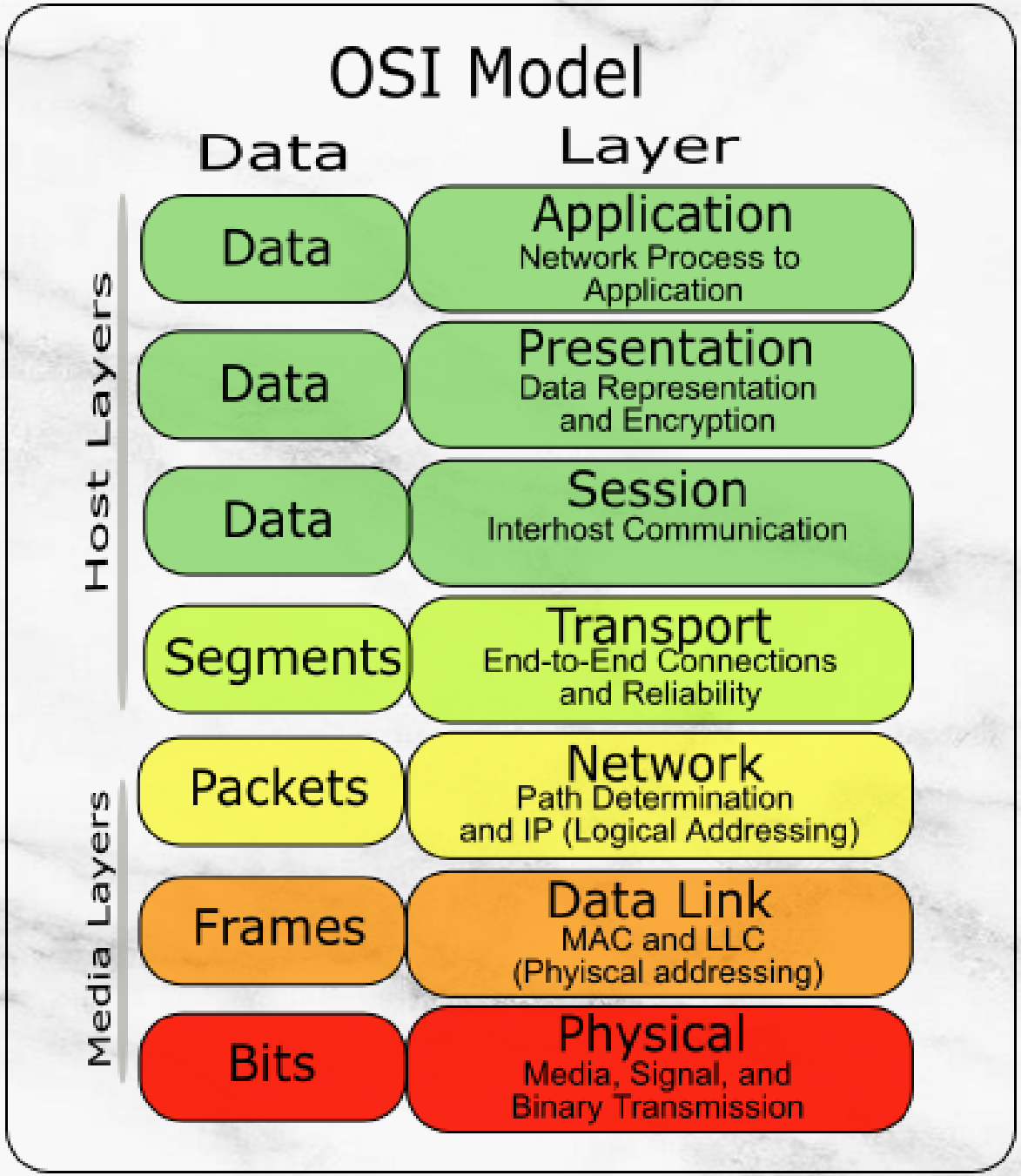
2. 동료 인터페이스
 - 통신 대상의 같은 계층 프로토콜과 교환하는 메시지 정의

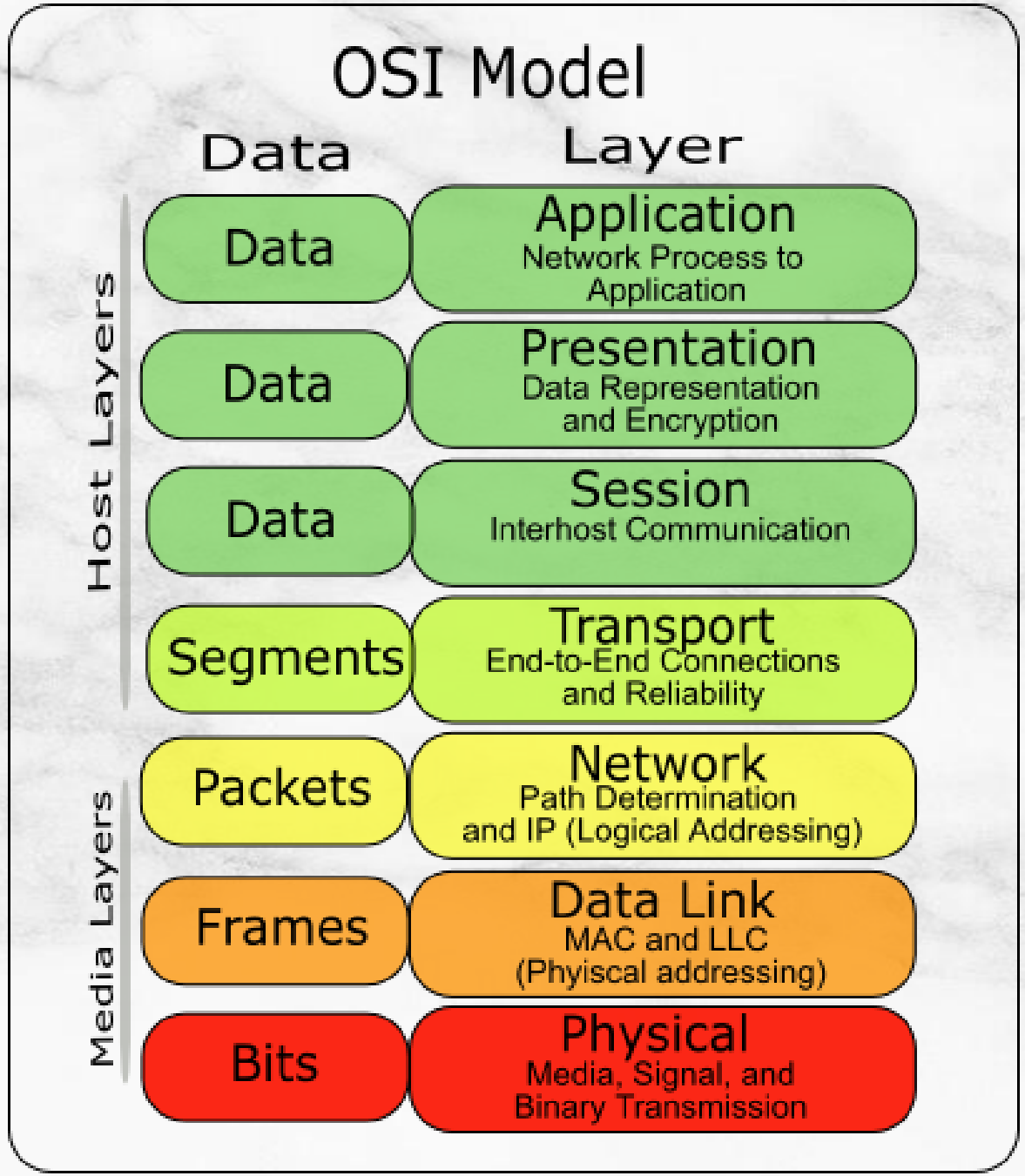
OSI 7계층 모델

Open
Systems
Interconnect (OSI)

Architecture

통신문제를 **나누어** 생각하는 틀





응용 계층 : 응용 자체와 관련된 사항만.

프리젠테이션 계층 : 데이터 표현 방법과 관련된 사항

세션 계층 : 대화 패턴과 관련된 사항 담당 (RPC)

트랜스포트 계층 : **통신 응용** 사이의 **신뢰성 있는** 메시지 교환 (process간 channel 제공)

네트워크 계층 : 네트워크를 통해 연결된 **호스트**(단말) 사이의 데이터 (**패킷**) 교환

링크 계층 : 하나의 링크로 연결된 **노드** 사이의 비트묶음 (**프레임**) 교환

물리 계층 : 물리적인 **신호** 교환과 관련된 사항 담당

TCP/IP 4계층

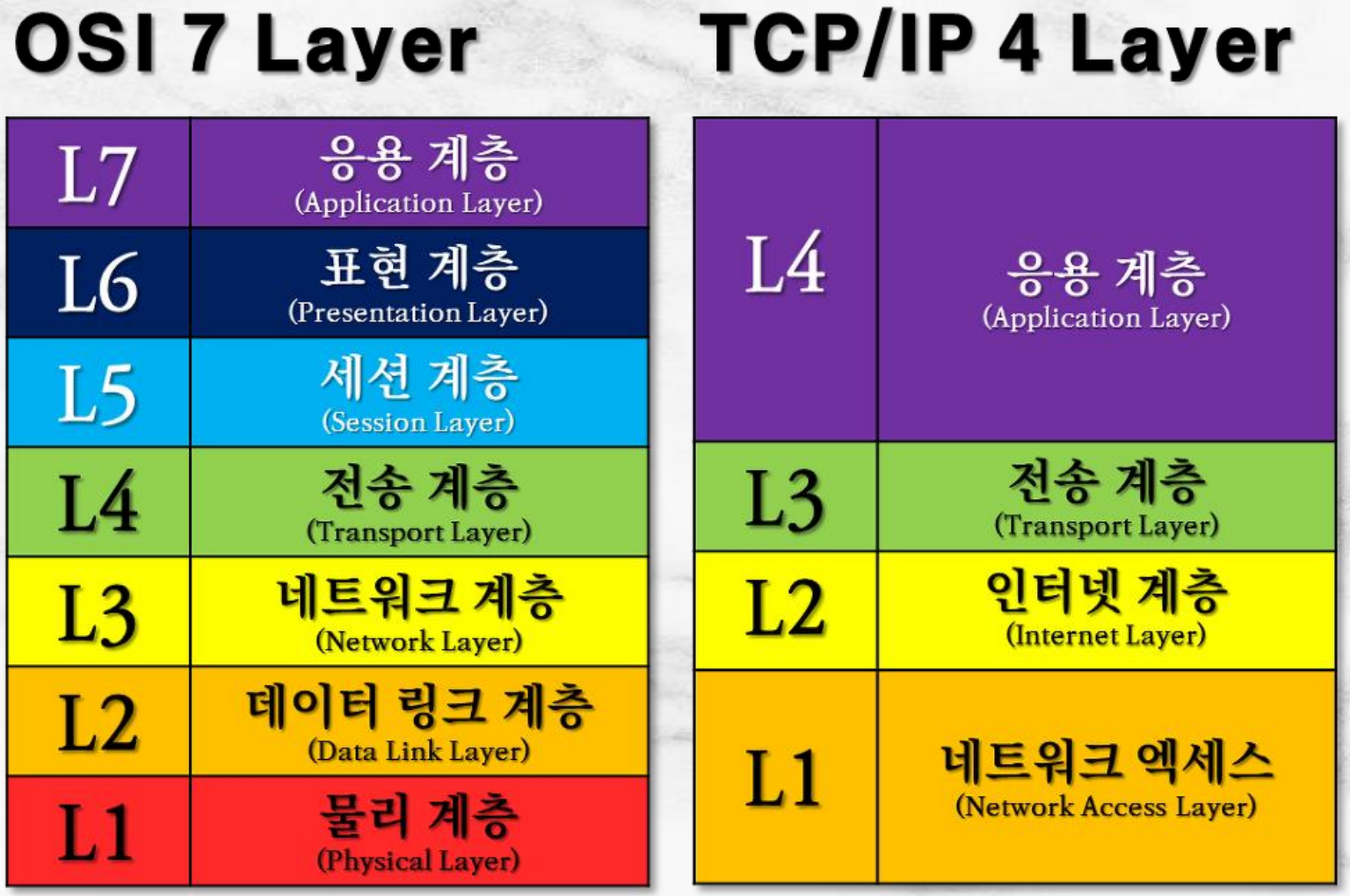
- 인터넷에서 사용하는 통신 프로토콜
- osi -> interface
- tcp/ip -> implementation

- TMI : OSI를 설계 한 곳과 TCP/IP를 설계 한 곳은 다르다
- TCP, UDP, IP는 TCP/IP스펙 표준에 명시되어 있다.
(OSI X)

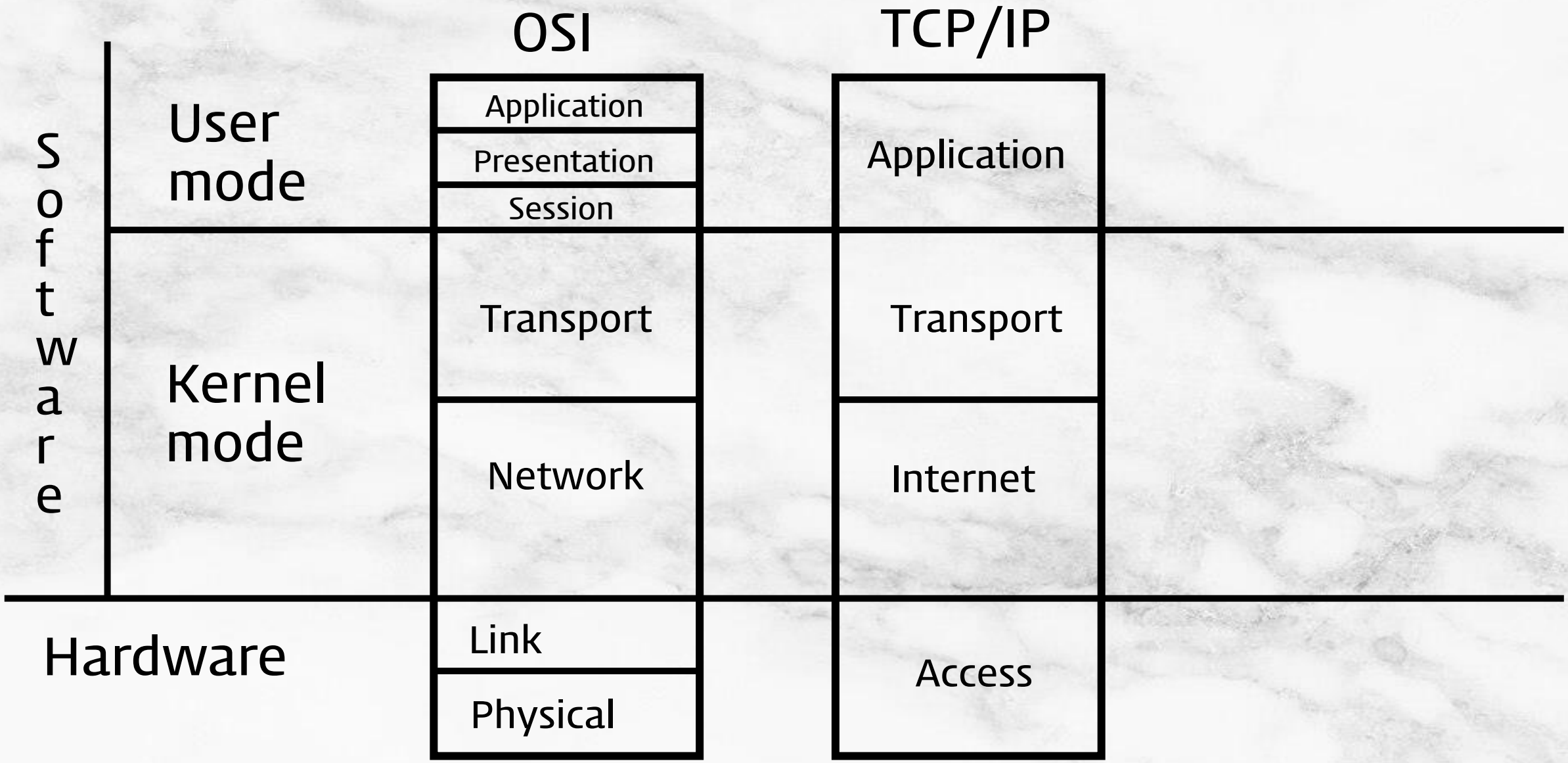
TCP/IP 4계층

- 인터넷에서 사용하는 통신 프로토콜
- osi가 interface라면 tcp/ip는 implementation

TCP/IP 4계층


OSI 7 Layer
L7
응용 계층
(Application Layer)
L6
표현 계층
(Presentation Layer)
L5
세션 계층
(Session Layer)
L4
전송 계층
(Transport Layer)
L3
네트워크 계층
(Network Layer)
L2
데이터 링크 계층
(Data Link Layer)
L1
물리 계층
(Physical Layer)
TCP/IP 4 Layer
L4
응용 계층
(Application Layer)
L3
전송 계층
(Transport Layer)
L2
인터넷 계층
(Internet Layer)
L1
네트워크 엑세스
(Network Access Layer)

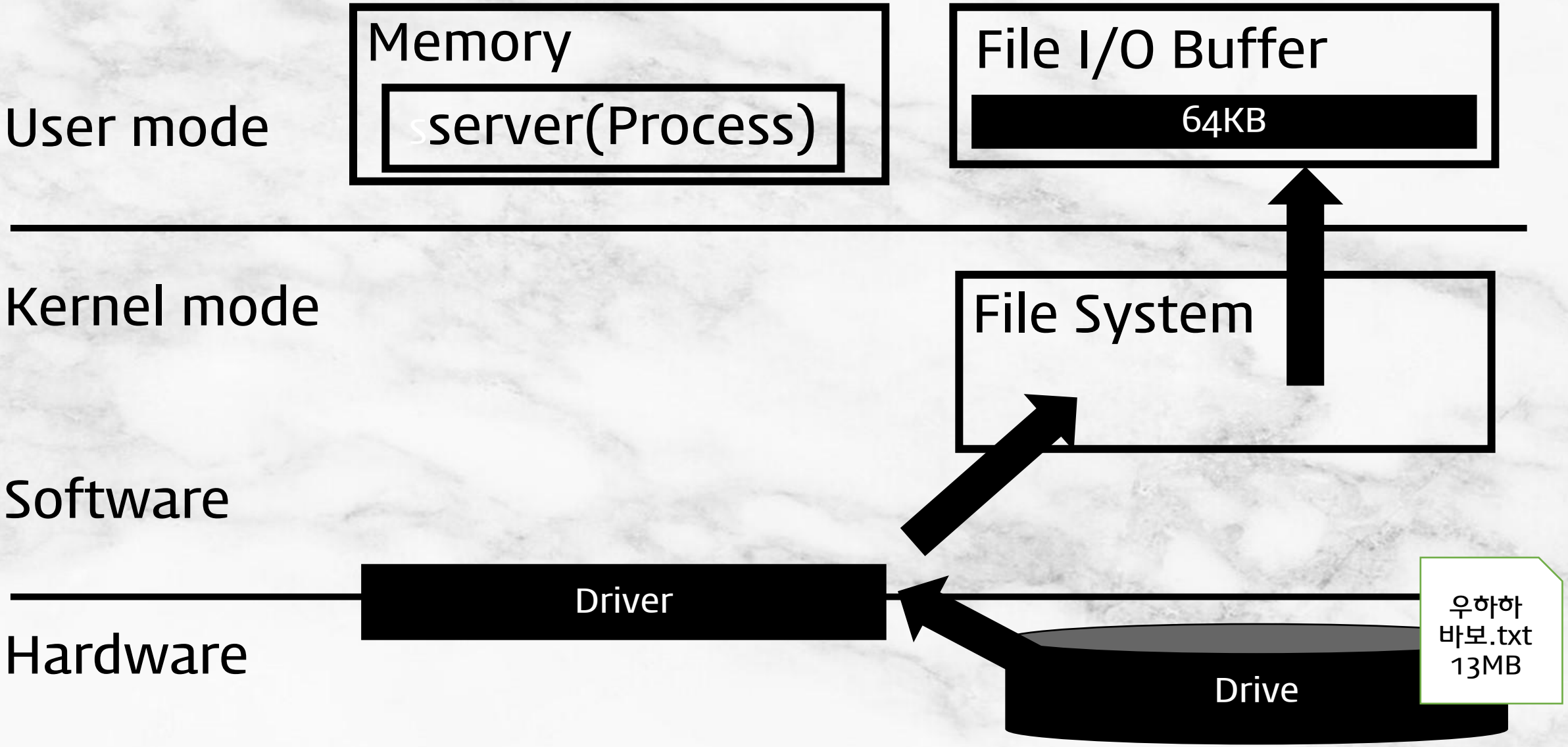
TCP/IP 4계층 : 실제는?



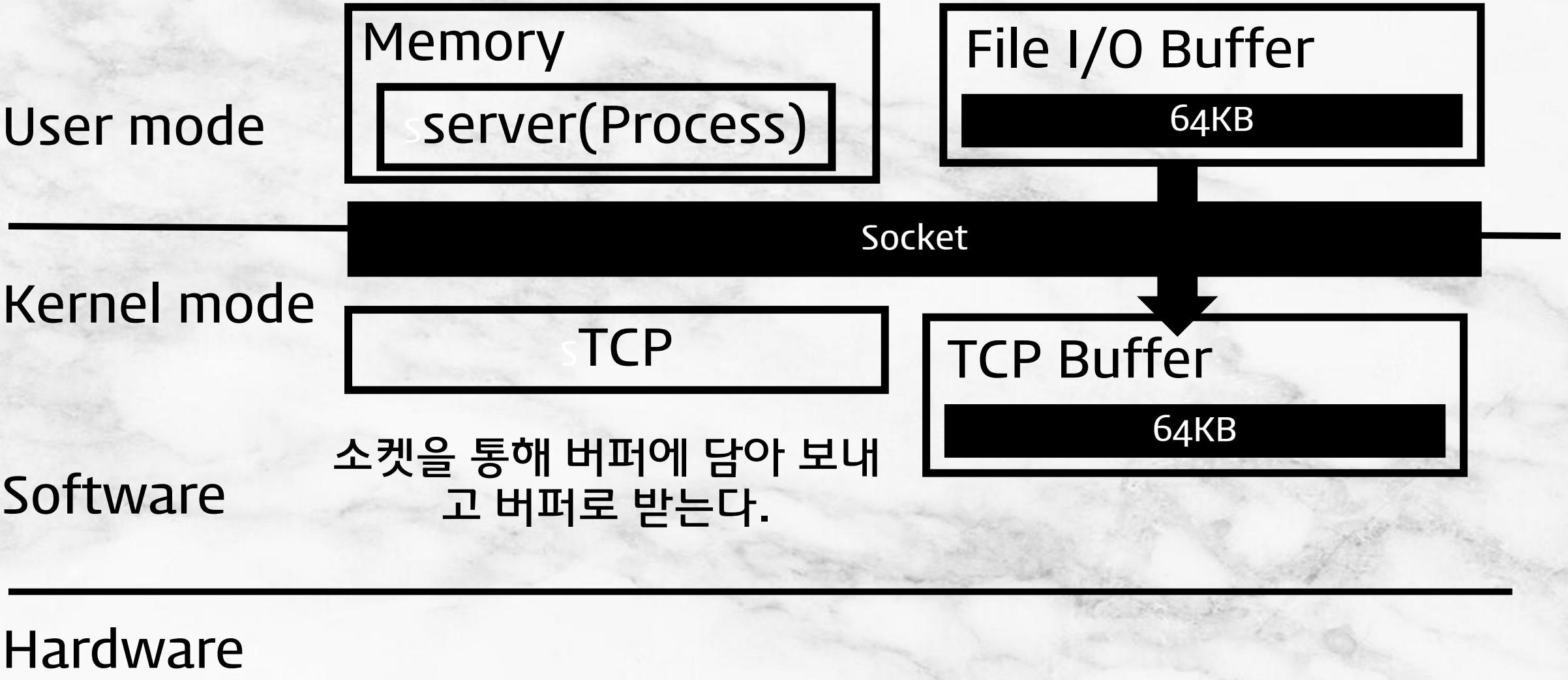
통신, 해보자!

한 컴퓨터(서버)에 있는 파일을 다른 컴퓨터로
보낼 때 어떤 일이 일어나는가?

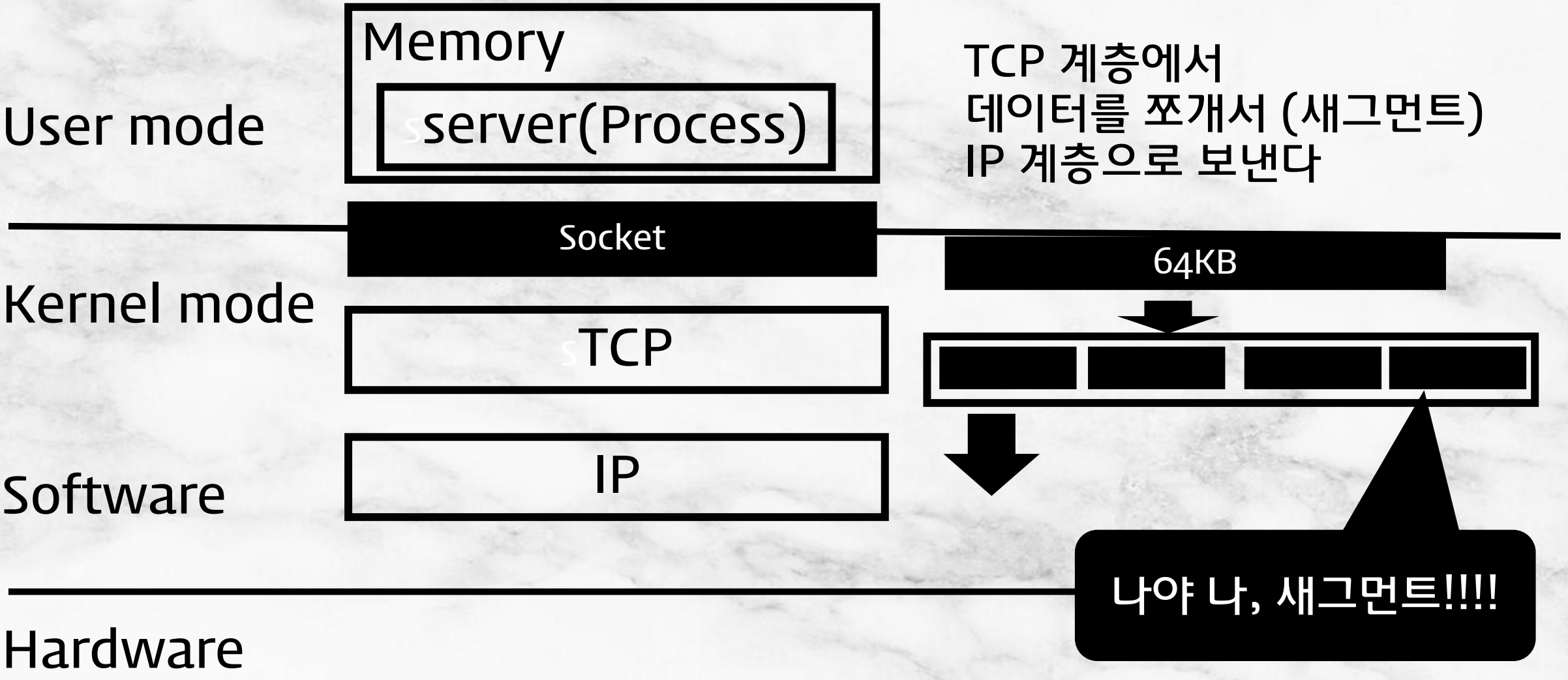
보내는 쪽 (send) : hdd -> Application



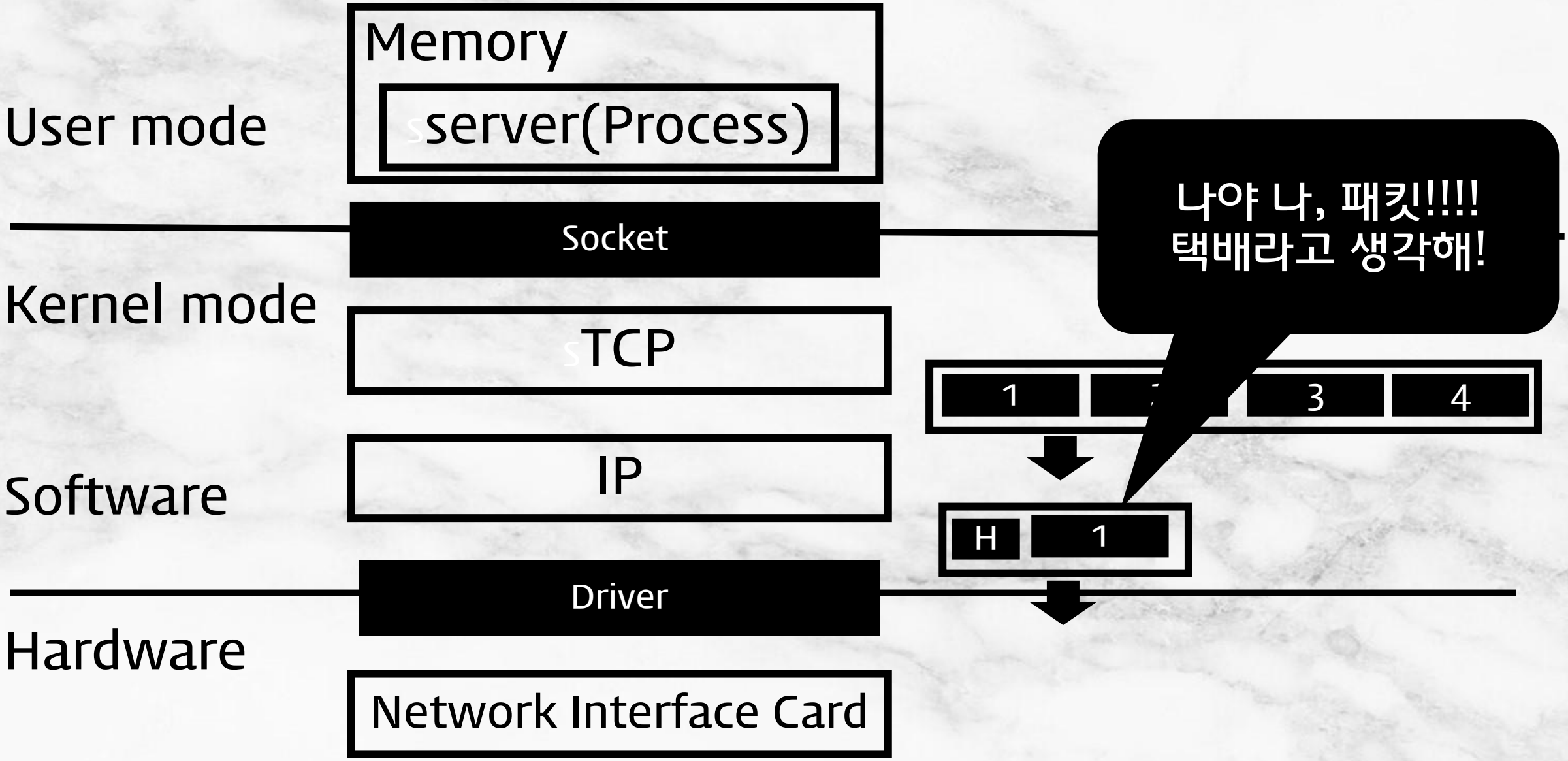
보내는 쪽 (send) : Application -> TCP



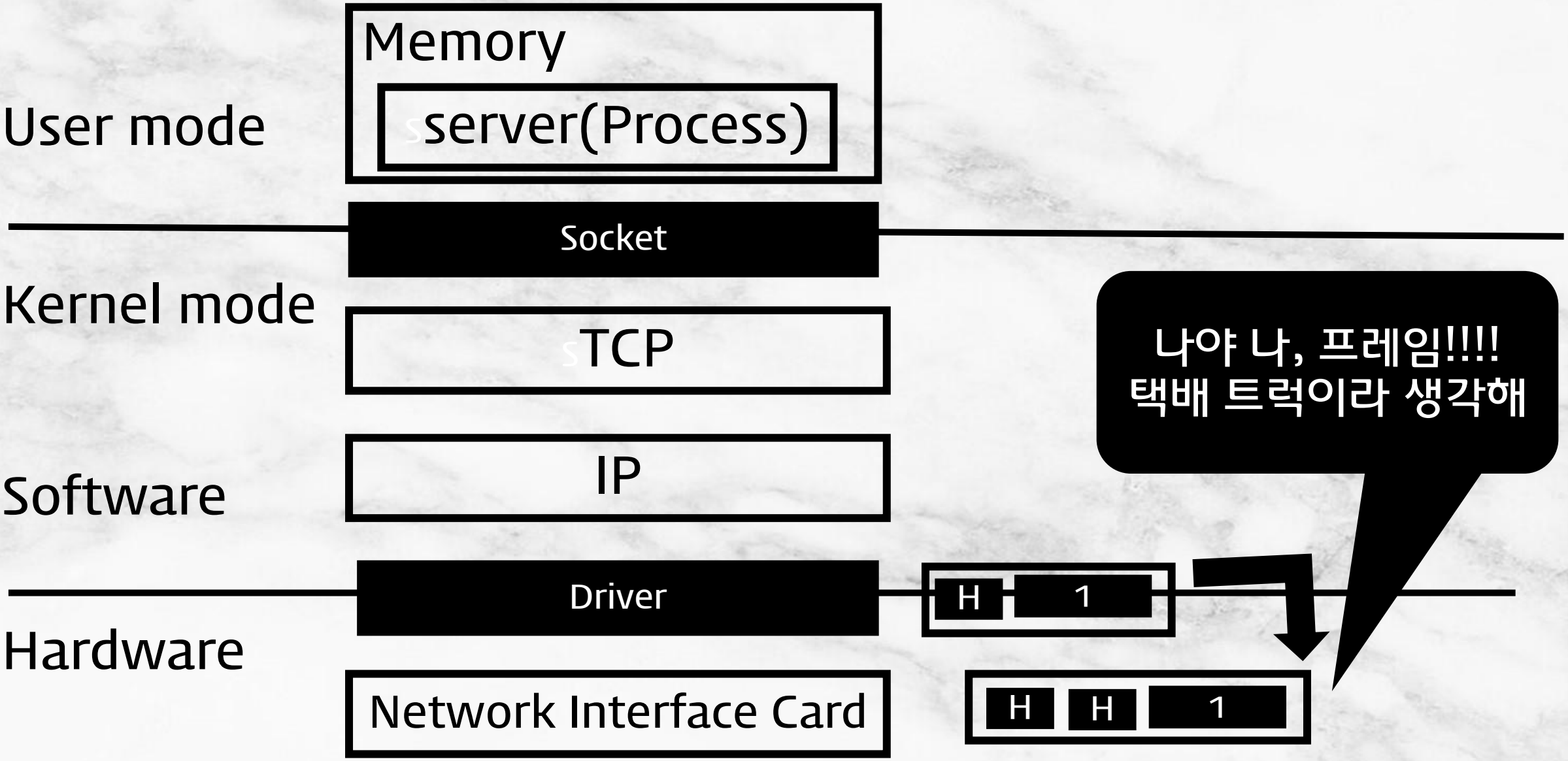
보내는 쪽 (send) : TCP -> IP



보내는 쪽 (send) : IP -> Access

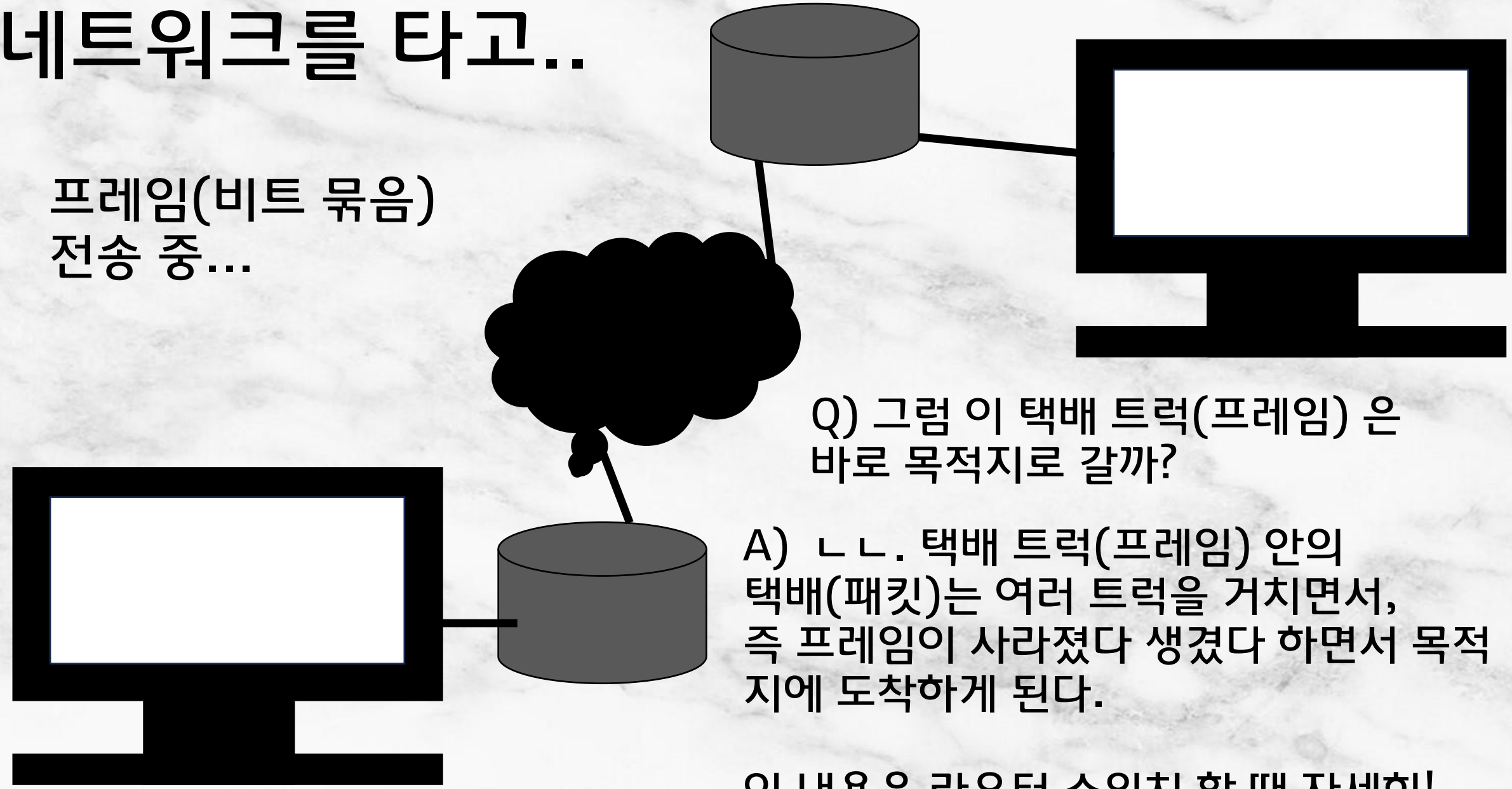


보내는 쪽 (send) : Access Layer



네트워크를 타고..

프레임(비트 묶음)
전송 중...

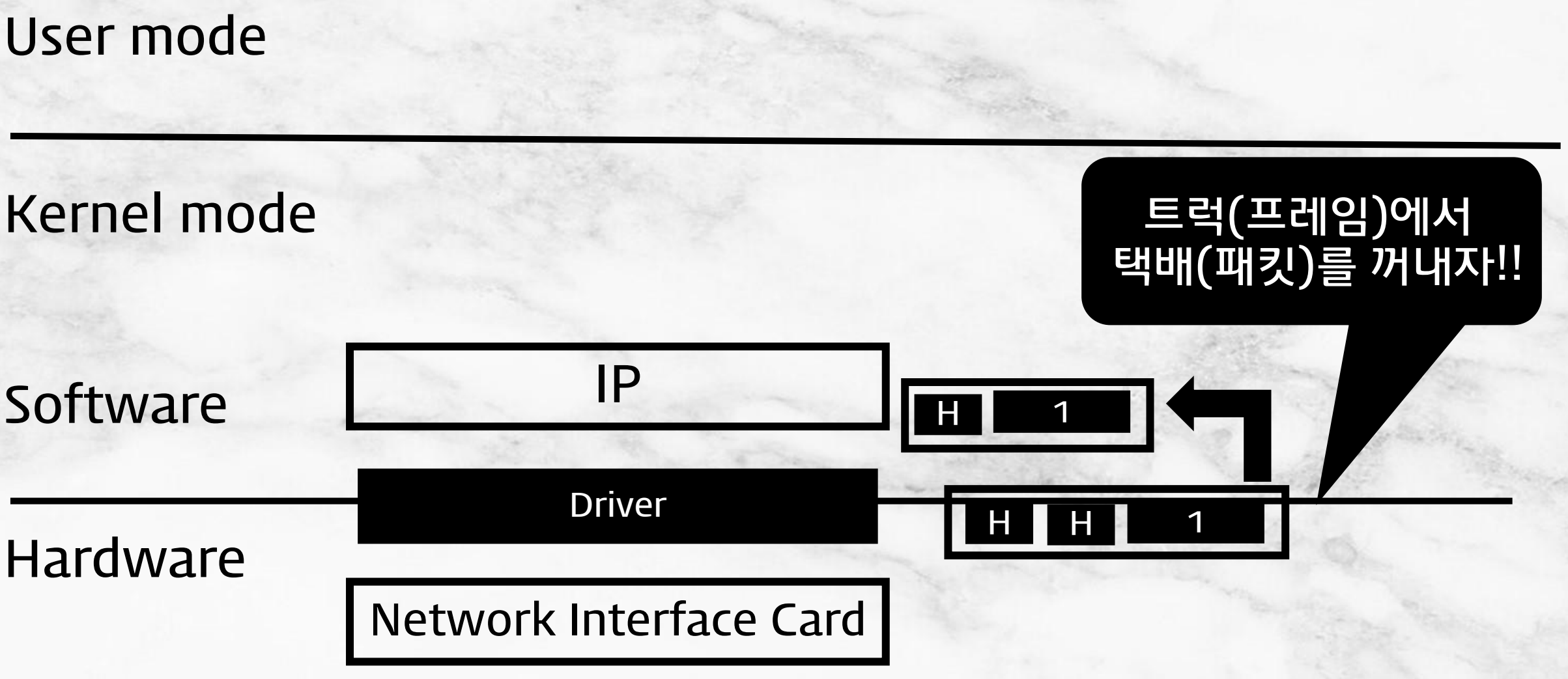


Q) 그럼 이 택배 트럭(프레임) 은 바로 목적지로 갈까?

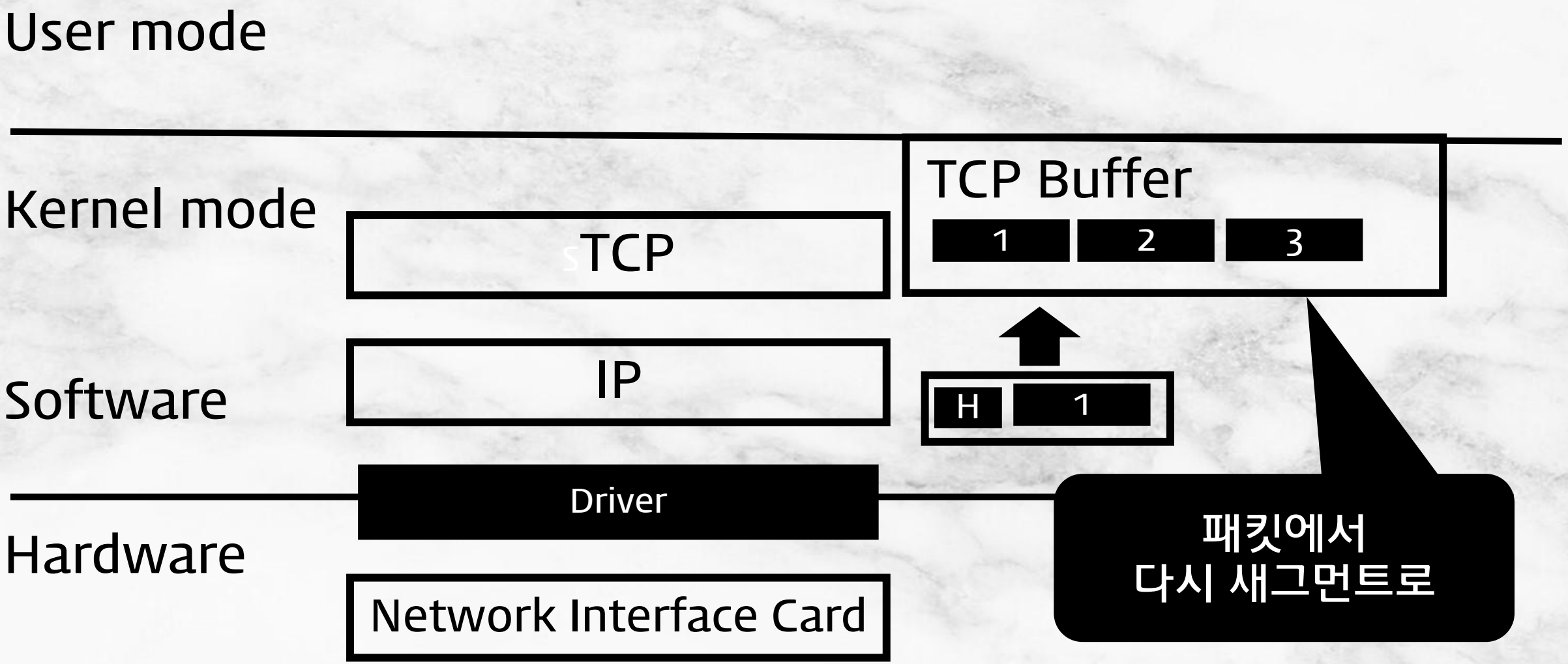
A) ㄴㄴ. 택배 트럭(프레임) 안의 택배(패킷)는 여러 트럭을 거치면서, 즉 프레임이 사라졌다 생겼다 하면서 목적지에 도착하게 된다.

이 내용은 라우터 스위치 할 때 자세히!

받는 쪽 (receive) : Access -> IP

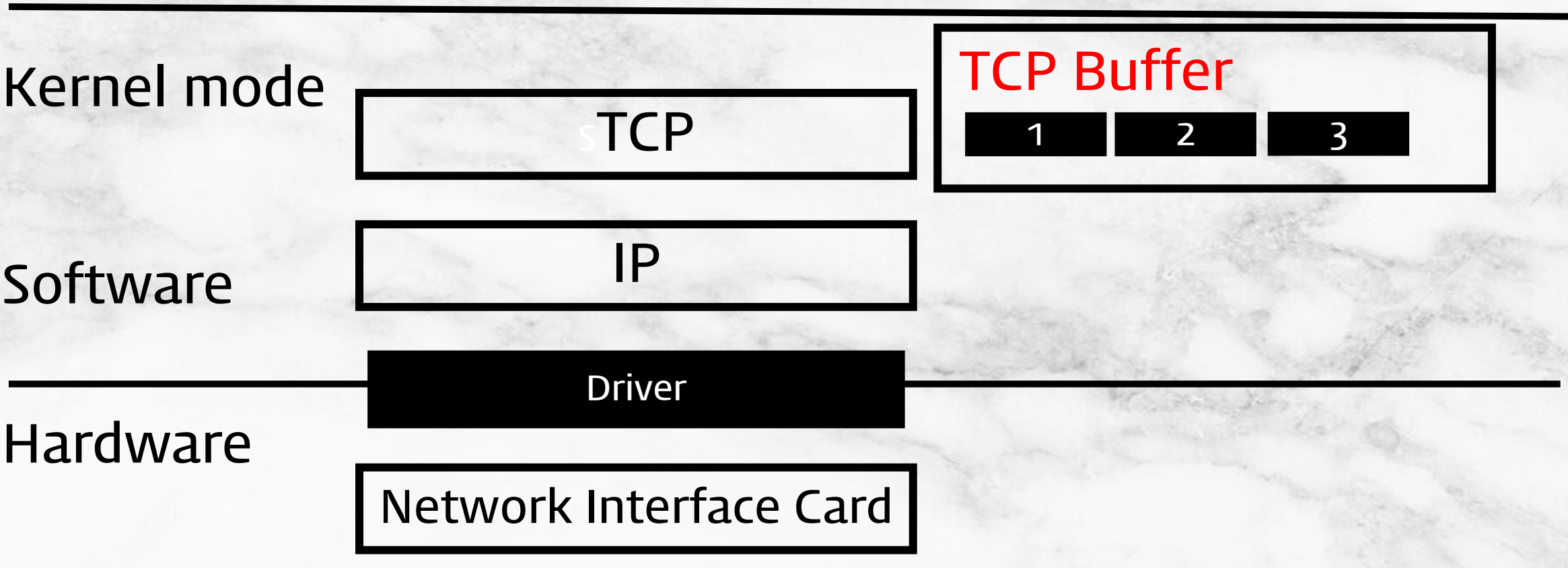


받는 쪽 (receive) : IP -> TCP

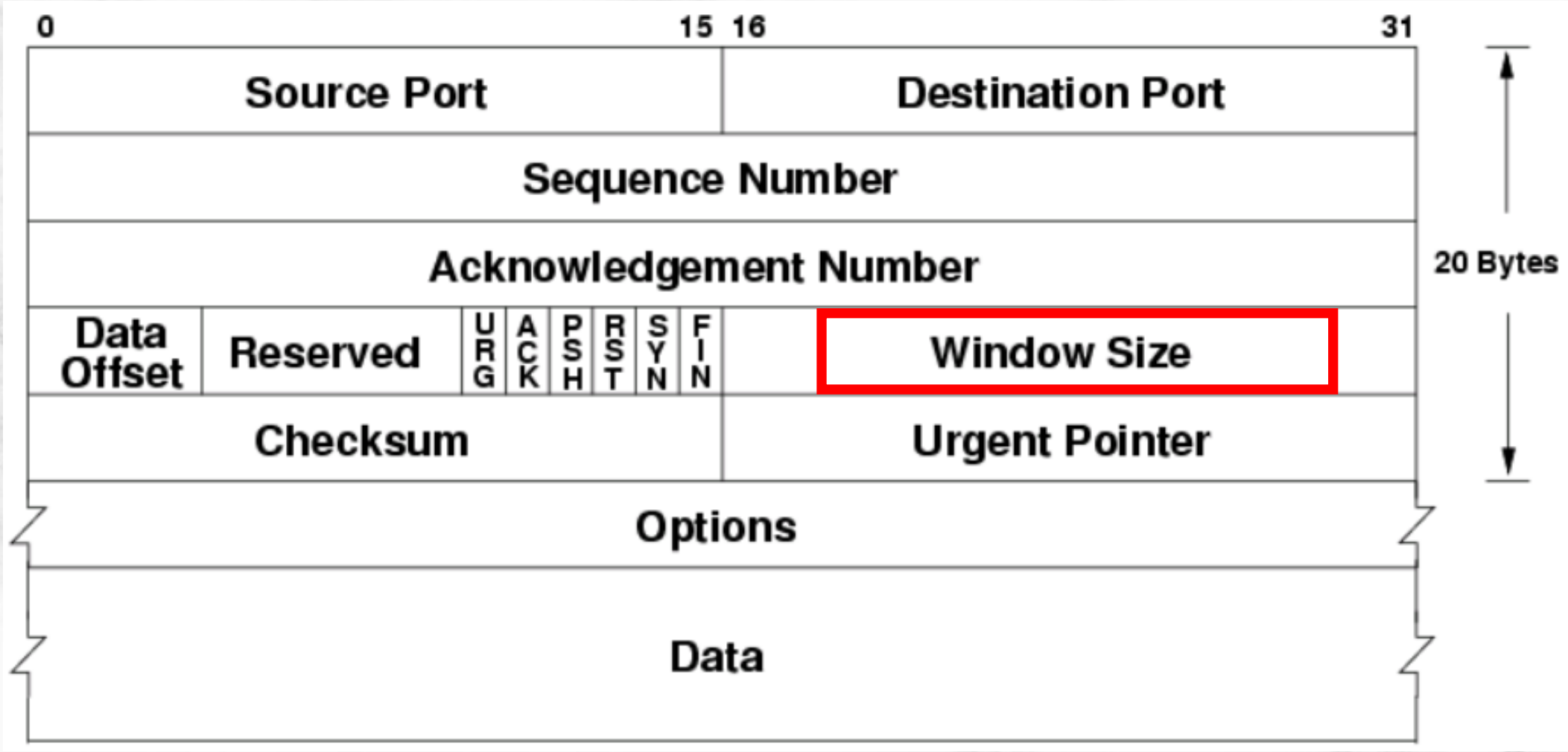


깜짝 퀴즈 : 이 TCP Buffer의 이름은?

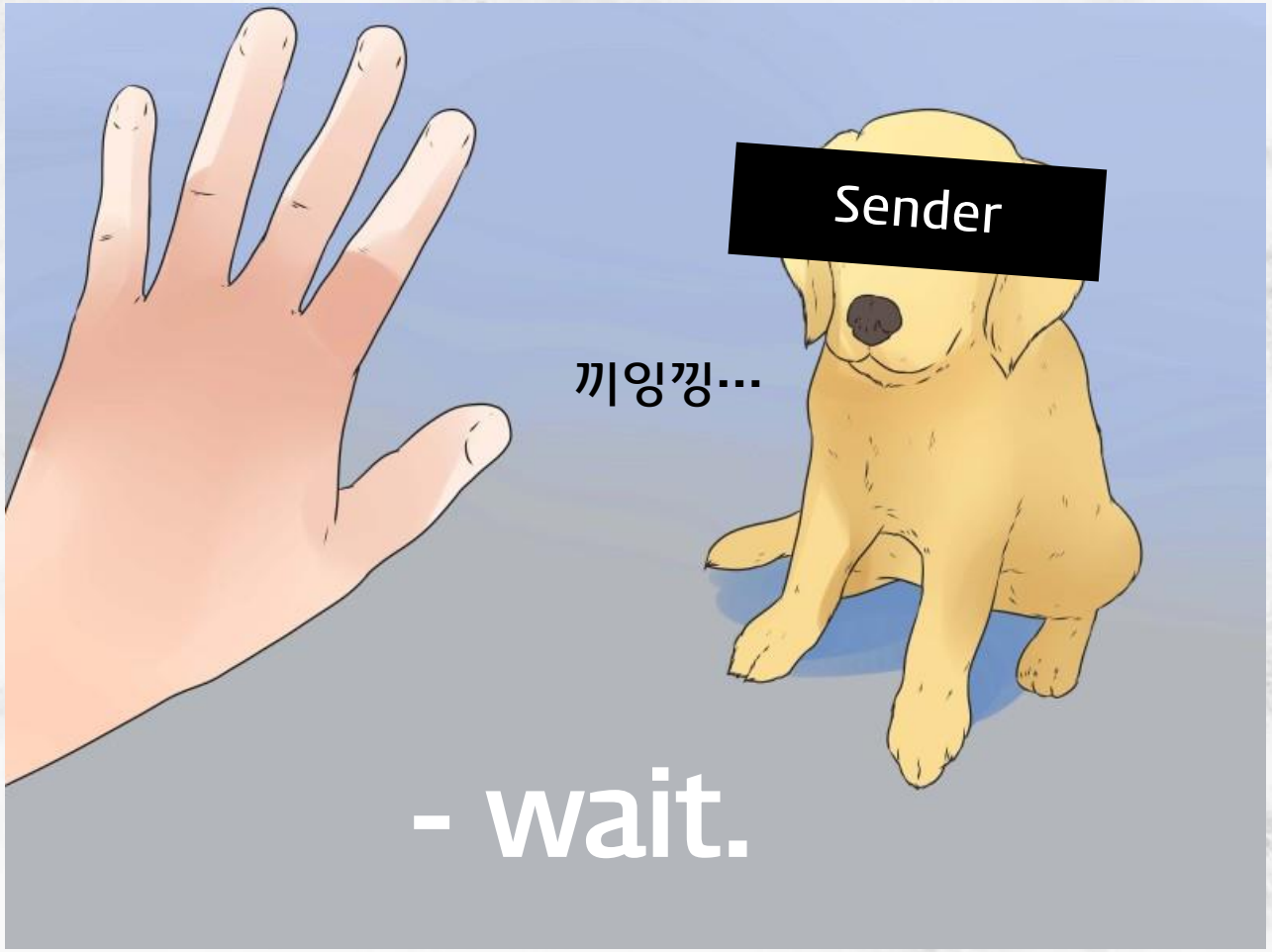
A) Window



그래서 TCP header에 window size가?!



그래서 Tcp가 udp보다 느려..



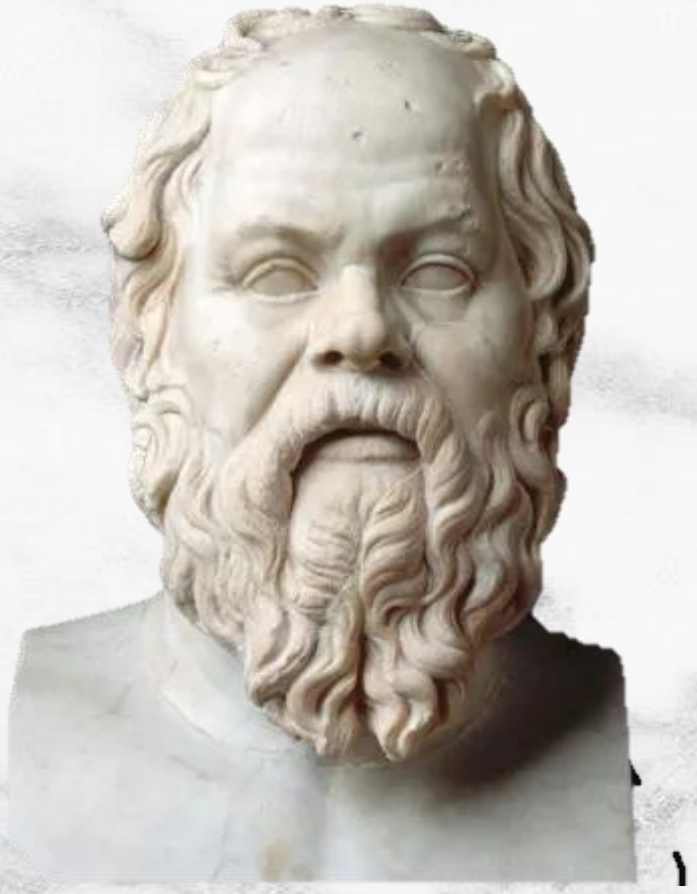
지금까지

- 네트워크에 대한 간략한 정의
- 통신을 위한 약속, 프로토콜
- 프로토콜의 규격 OSI 7 layer, 구현체 TCP/IP stack
- 실제로 TCP/IP가 어떻게 작동하는지
- 단편화 된 기억들을 약간이나마 모아보았다!

아직 남았습니다 끝 아님

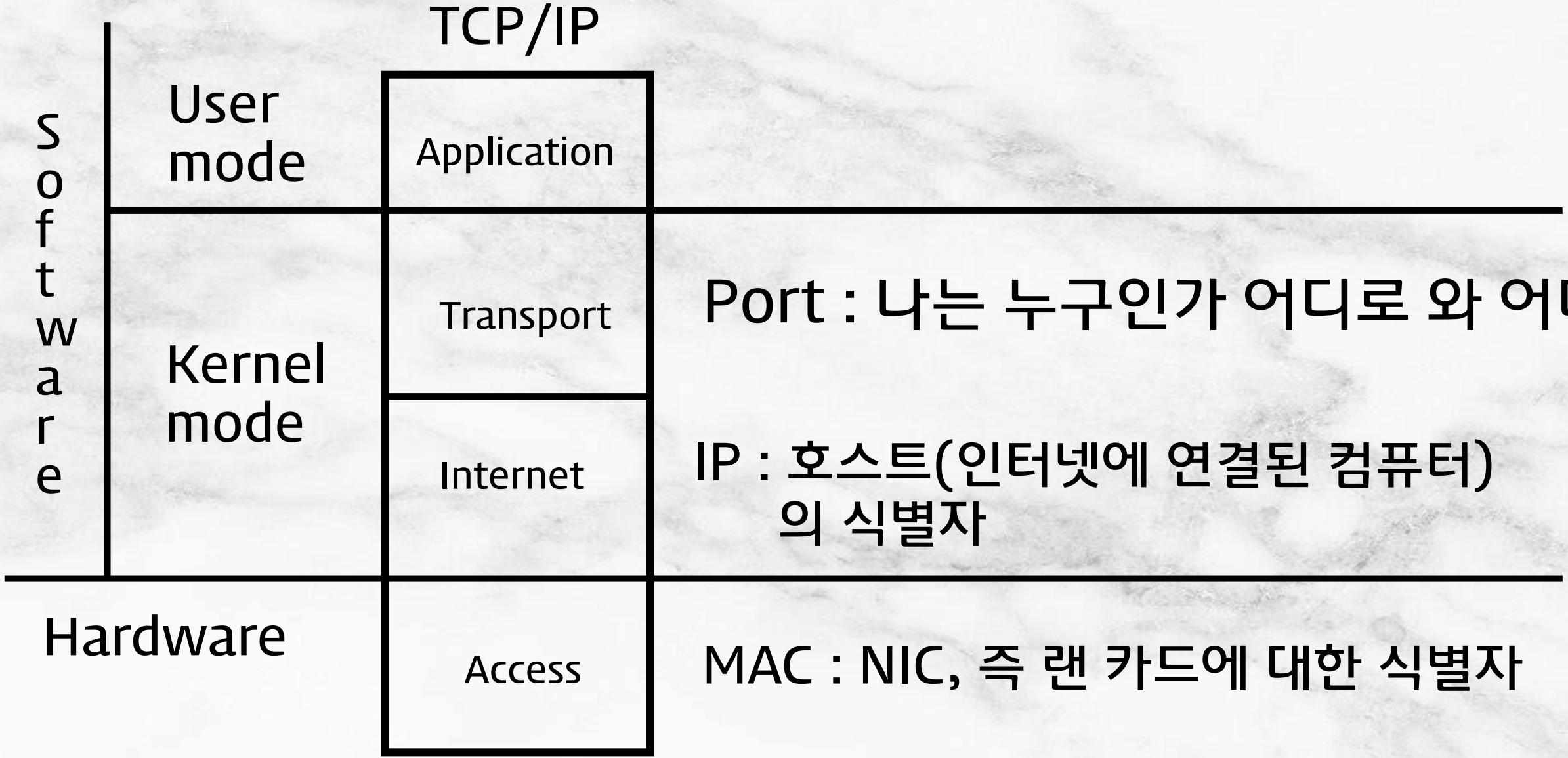
식별자

통신에서 알아야 할 것
- 나
- 너
- 그리고 우리

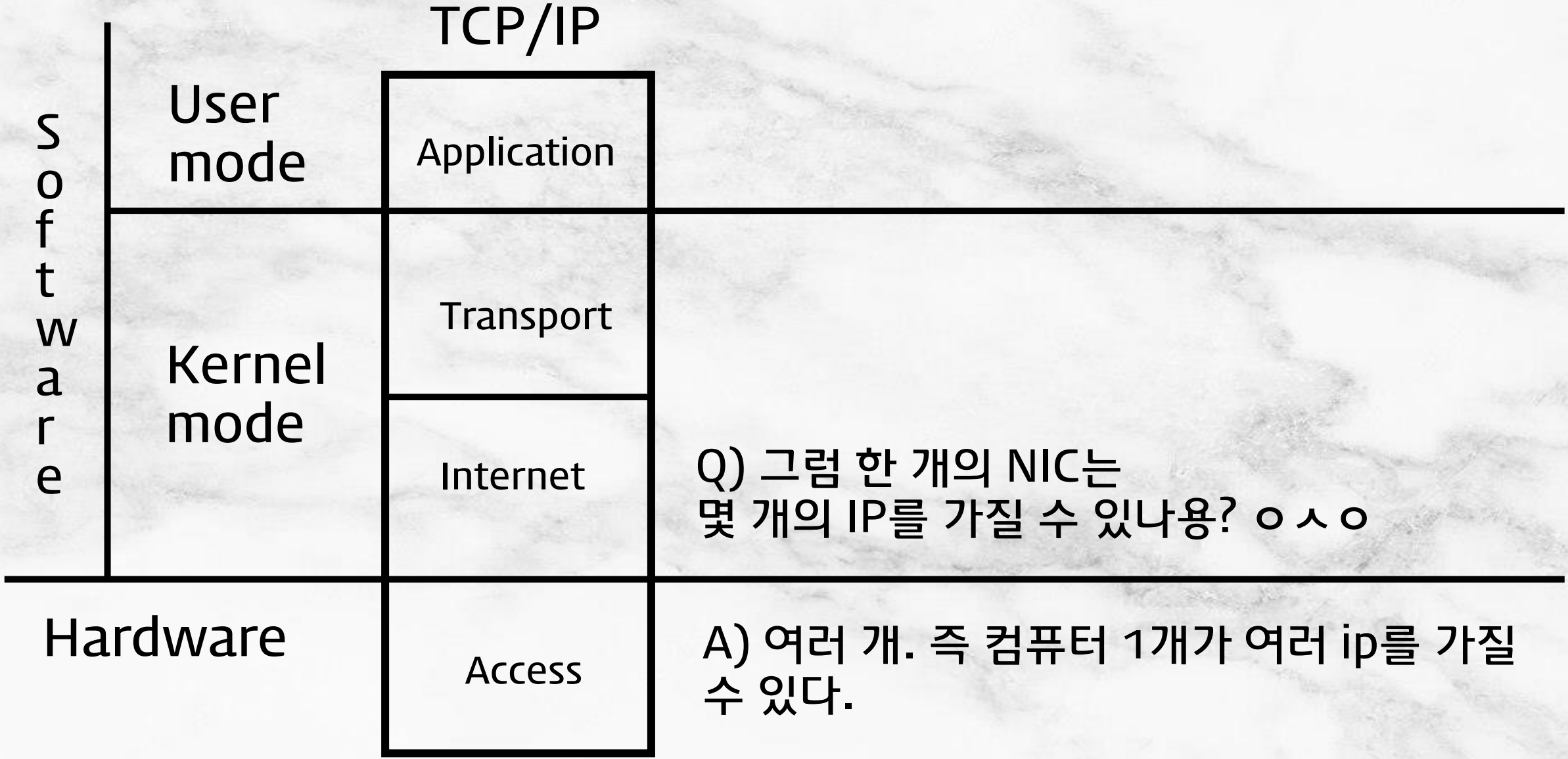


각 계층에서 대상을 식별하기 위해 사용

TCP/IP 프로토콜의 식별자



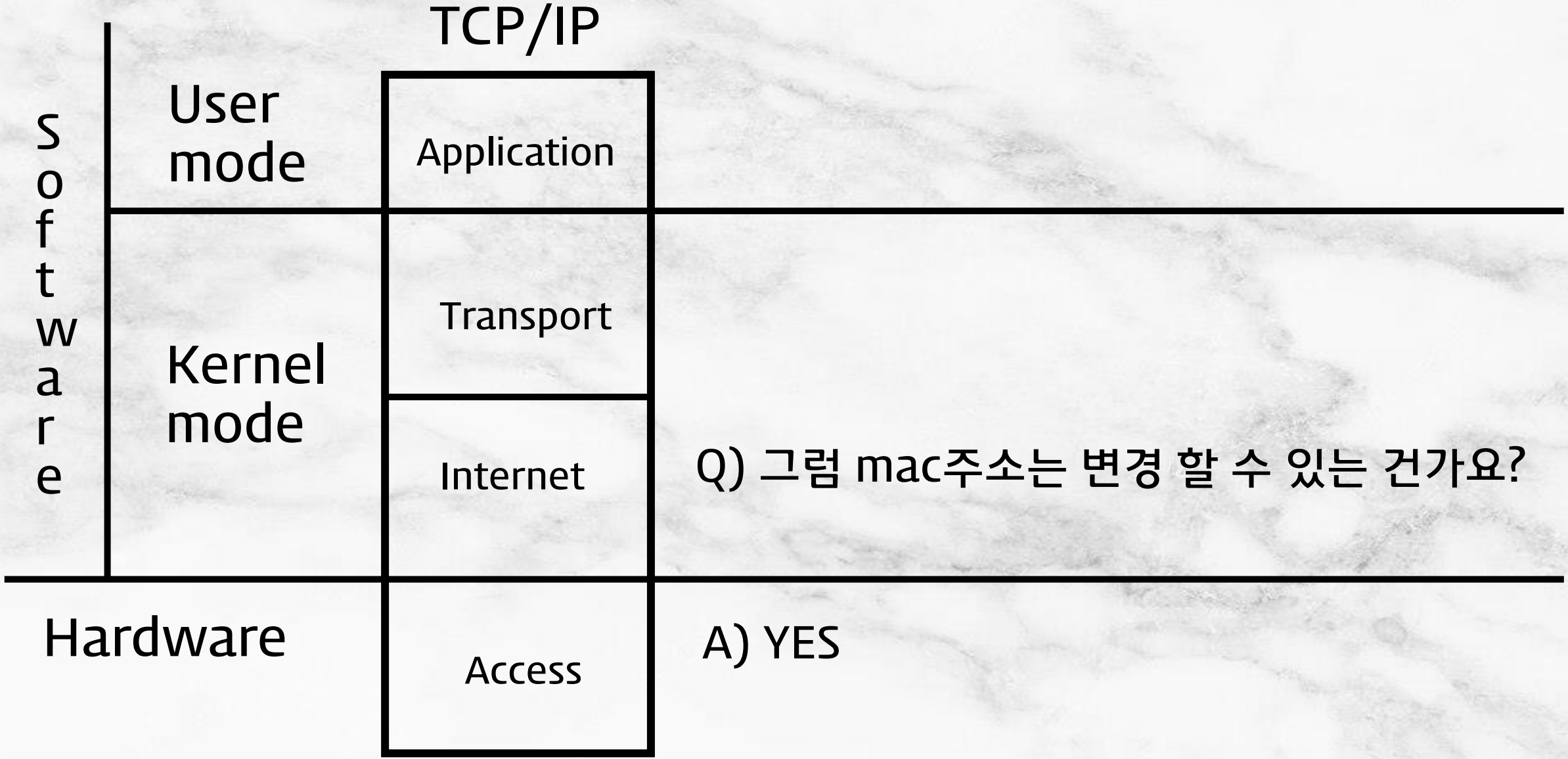
이유를 알 수 없는 퀴즈1



Q) 그럼 한 개의 NIC는
몇 개의 IP를 가질 수 있나용? ㅇㅅㅇ

A) 여러 개. 즉 컴퓨터 1개가 여러 ip를 가질 수 있다.

이유를 알 수 없는 퀴즈 2


TCP/IP
Software
User mode
Kernel mode
Application
Transport
Internet
Access
Hardware
Q) 그럼 mac주소는 변경 할 수 있는 건가요?
A) YES

그럼 Port 넌 뭐니?

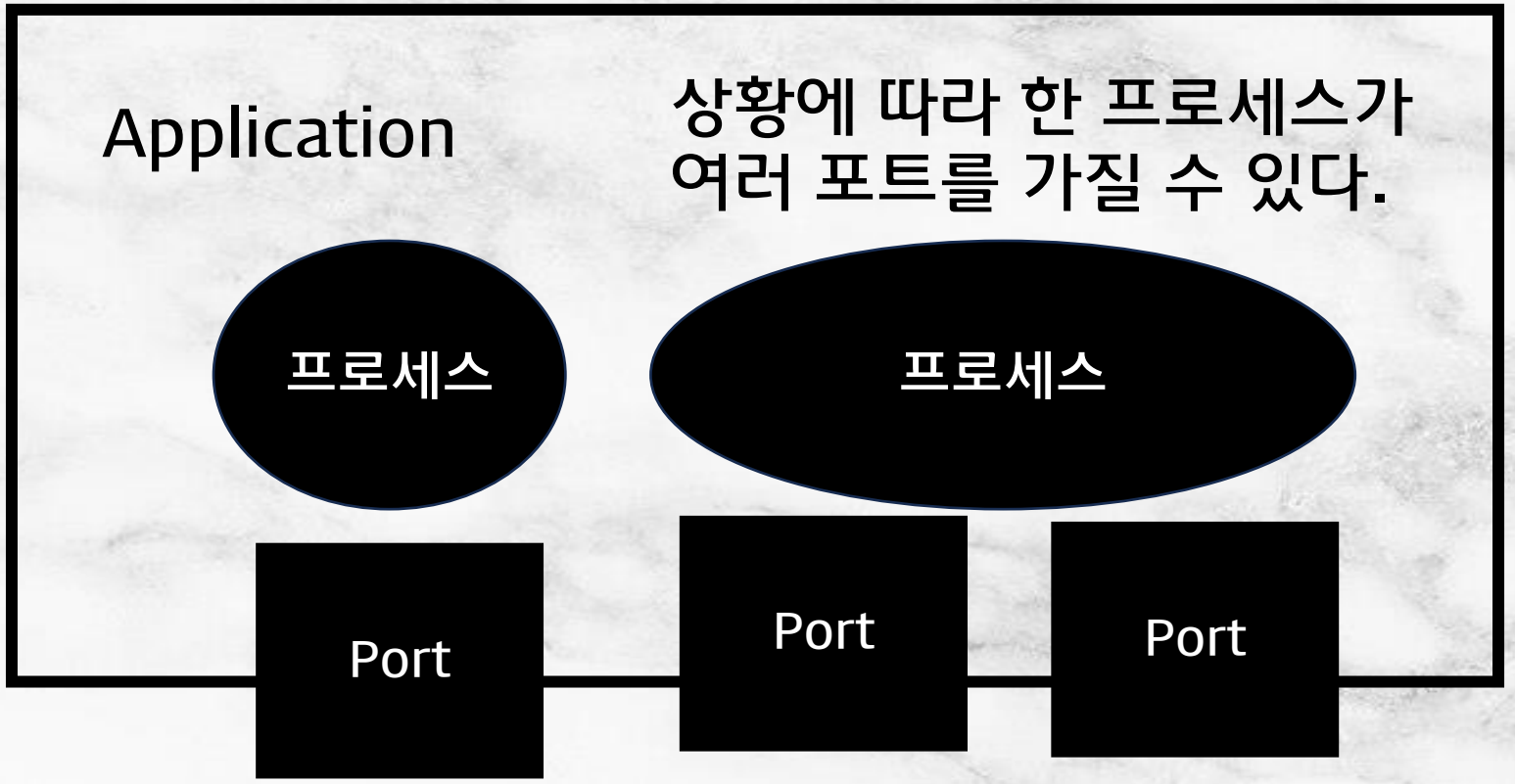
소프트웨어 개발자 기준

프로세스와 연결된 data path 혹은 data channel

아니 이거 그냥 숫자 8080 이런거 아님?

Port의 개념

어플리케이션 영역에 있는
각각의 프로세스들이 포트를
이용하여 통신을 한다

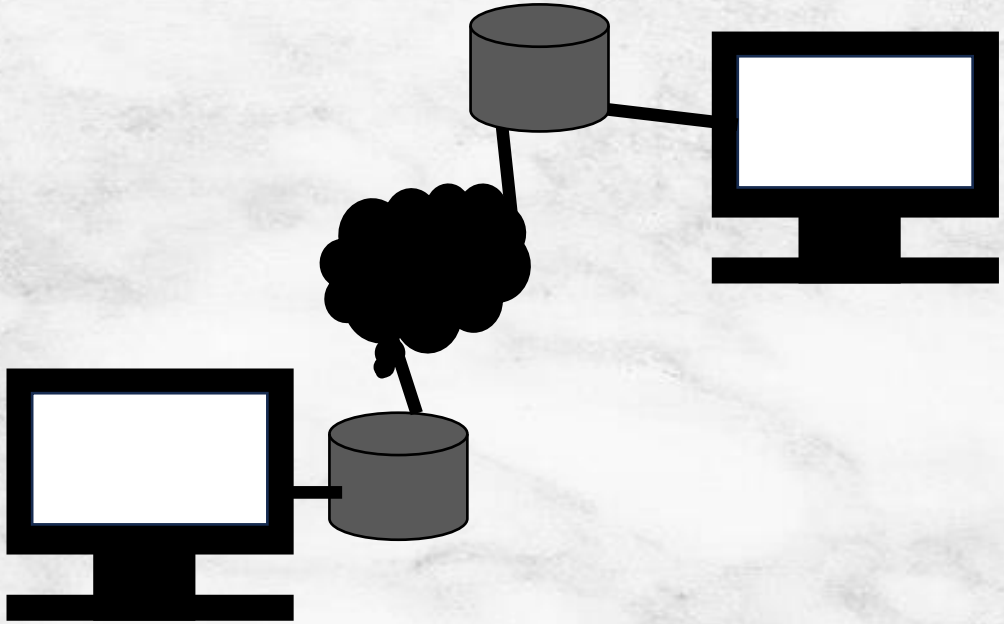


User
mode

Kernel
mode

Port name으로 각 포트를 식별

Port를 이용하여 통신을 하는데..



그럼 왜 이렇게 설계해ㅑ ㅆ냐?
그만큼 단순해지므로

TCP/IP의

Internet Protocol은
데이터를 목적지까지의 최적의 경로를 라우팅 하여 전송하는 기능이다.

근데 이 녀석의 특징 : **Unreliable**
데이터가 유실 될 수도 있으며,
데이터가 순서대로 안 올 수도 있다.

즉 모자라지만 그래도 최선을 다 하는 녀석

그런데 프로세스와 프로세스 사이의 데이터 전달은 유실이나 순서가 반드시 맞아야 한다.

즉 프로세스 간의 통신에서는
데이터를 안정적으로 주고 받을 수 있는 프로토콜이 필요!!!!

그게 Transmission Control Protocol,
프로세스 간 데이터를 안정적으로 주고 받을 수 있는 프로토콜
TCP다.

그럼 Port 넌 뭐니?

즉 미리 **Connection**을 만들고
데이터를 보내면 reliable하다!!!!
다 보낸 후에는 연결 종료

이 Connection을 위해 서로를 식별해야 하는데,
그 식별자가 바로 Port다.

Port 부연 설명

Port (number) : 16 bits로 이루어진 숫자 (0~65535)

근데 겨우 6만으로는 인터넷 상의 모든 프로세스를 식별하기 어려워. 즉 unique하게 식별 불가!!!

생각해보니 Internet Address(IP)로 각 host를 식별 할 수 있다고 했잖아. 그렇다면?

Port 부연 설명

그러면 둘이 합치면? 인터넷의 모든 Port들을 식별 가능

Internet Address + Port number

Q) 그럼 이걸 뭐라 불러?

소켓

Socket의 여러 정의 in TCP/IP

인터넷 상에 존재하는 각 port를 유니크하게 식별하기 위한 주소

이런 이유로 확장해서 생각하면 소켓 = 포트 가 되는거다.
즉 각각의 소켓은 인터넷 상에서 고유(unique)해야 한다.

그럼 Connection을 Socket의 관점으로 본다면

각 Connection을 고유하게 식별 할 수 있어야 한다.
즉, 한 쌍의 Socket은 connection을 고유하게 식별한다.

즉
< src internet addr, src port, dst internet addr, dst port>
를 통해 Connection을 식별 할 수 있다.

그럼 Socket을 Connection의 관점으로 본다면

하나의 소켓은 동시에 여러 Connection에 사용 될 수 있다.
(1:N 프로세스의 연결을 생각. 그럼 1의 소켓은 여러 conn에 사용된다.)

발을 살짝 담가보자.

아니 나는 데이터가 유실되든 상관 없어.
Internet Protocol처럼
유실 순서 여부에 상관없이 받는 방법이 있을까..?

그게 바로 User Datagram Protocol
UDP다.

UDP

Connectionless : 연결? 안함 바로 진행시켜

그럼 연결이 없으니깐 여기서 소켓은 어떻게 되는거지?
>> 원래는 UDP표준(RFC 768)을 보면 socket이라는 단어가 아예 등장 안 한다.

사실 소켓은 Port를 인터넷에서 식별하기 위해 사용 된 것이므로 Port자체를 Udp에서도 쓸 수 있다. 물론 식별을 위해 추가 작업을 해야겠지만.

그래서 그냥 udp를 쓰면서 자연스럽게 socket을 쓰게 되었다.

결국 Socket in Tcp/Ip stack은

즉 udp에서도 소켓이란 개념을 쓰게 되니깐 ip+port만으로는 고유하게 식별이 안된다. 따라서

<Protocol, IP address, port number>

를 이용하여 고유하게 인터넷 상의 프로세스를 식별한다.

끝!

Tcp/udp원리, handshake, ip주소체계,
http, ssl 등은 다음에!

출처

https://www.youtube.com/watch?v=X73Jl2nsqiE&t=1214s

https://www.youtube.com/watch?v=K9L9YZhEjCo&t=1662s

https://www.youtube.com/watch?v=JDh_lzHO_CA