쿠키와 세션
&
웹소켓과 소켓 통신

이수현

목 차

쿠키와 세션

웹소켓과 소켓 통신

쿠키와 세션

쿠키?

쿠키 (Cookie)

클라이언트에 저장되는
key-value 형태의
작은 데이터 파일

세션?

세션 (Session)

쿠키를 기반으로 동작하지만,
사용자 정보를
클라이언트 측이 아닌
서버측에서 관리

세션 (Session)

서버는 각 클라이언트에
고유 session ID를 할당하고,
이 ID를 통해 사용자의 상태를
관리할 수 있습니다.

쿠키와 세션의 차이

쿠키와 세션의 차이

	쿠키	세션
저장 위치	브라우저	서버
만료 시점	쿠키 저장 시 설정 (브라우저 종료해도 쿠키 남아있음)	브라우저 종료시 삭제 (기간 지정 가능)
요청 속도	세션보다 빠름	쿠키보다 느림
보안	좋지 않음	쿠키보다 좋음

쿠키와 세션은 왜 필요할까?

쿠키와 세션을 사용하는 이유

HTTP 의 특성이자 약점을 보완하기 위해 사용

HTTP의 특성

connectionless(비연결성)

- 서버에 요청 후 응답을 받으면 TCP/IP 연결을 끊음
  - 서버 자원 효율적 관리
  - 새로운 연결 시 3-way handshake에 따른 시간 소요

stateless(무상태성)

- 서버가 클라이언트의 이전 상태를 보존하지 않음
  - 어느 서버가 응답해도 무관, 서버 스케일 아웃에 유연
  - 클라이언트가 많은 양의 데이터를 전송해야 하는 단점

HTTP의 특성

서버는 클라이언트가 누구인지
계속해서 인증 필요

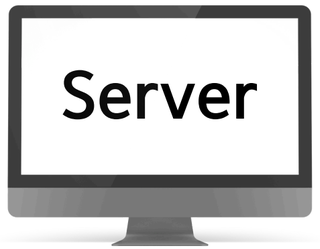
HTTP의 특성

서버는 클라이언트가 누구인지
계속해서 인증 필요

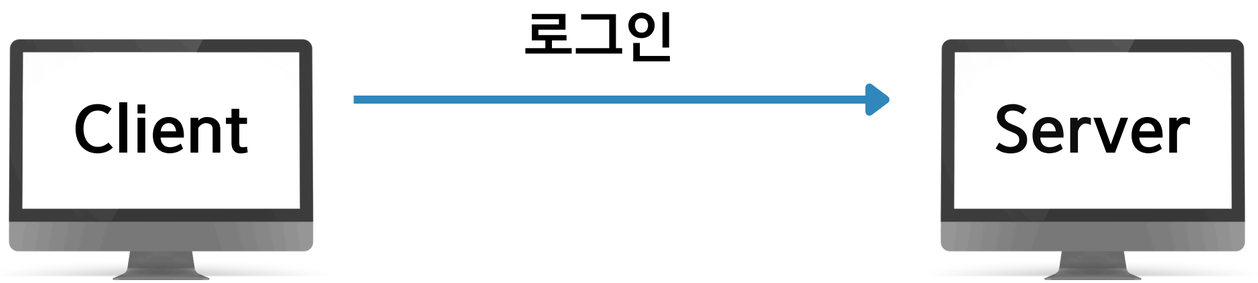
쿠키와 세션을 이용하면
상태 유지 가능!

동작 방식

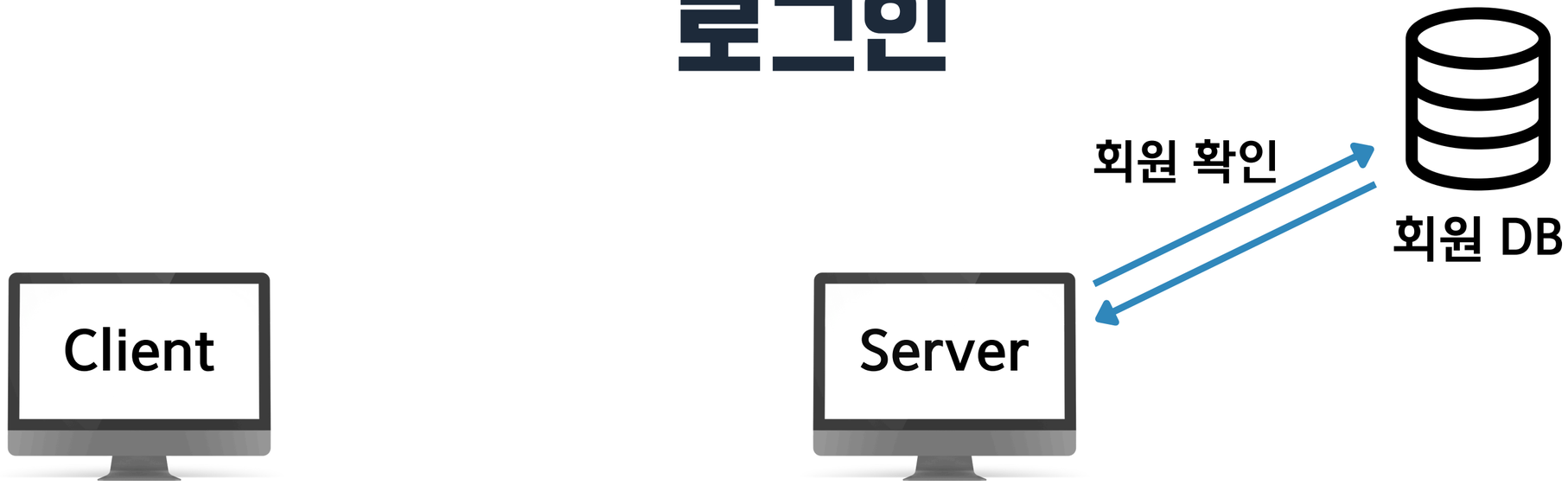
로그인



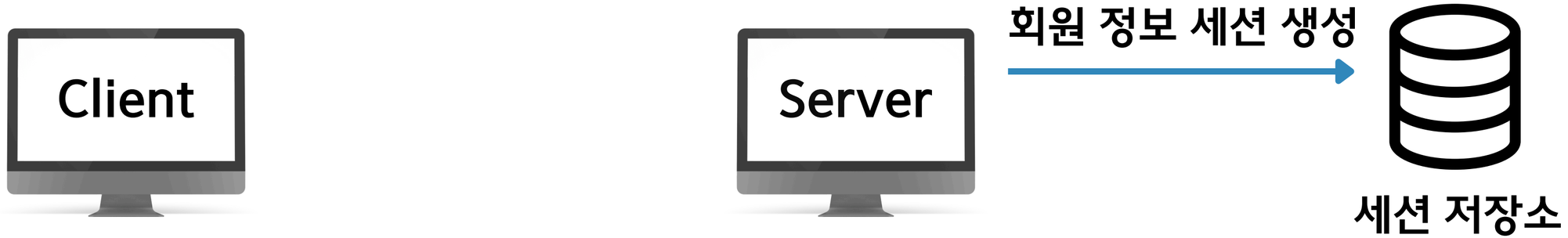
로그인



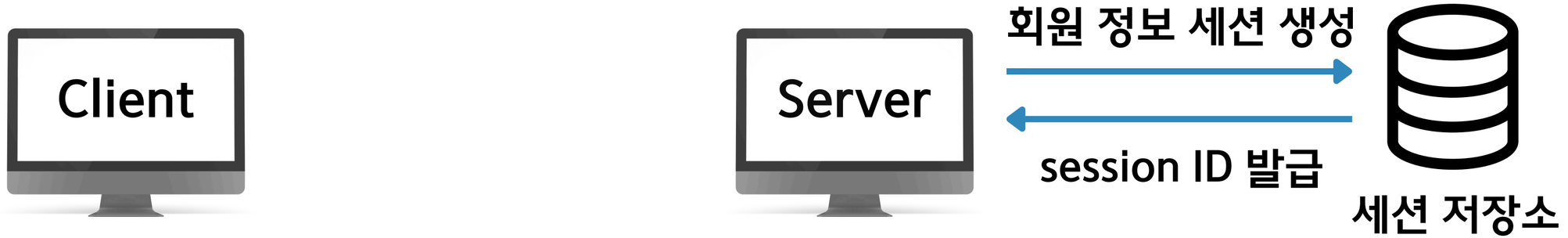
로그인



로그인



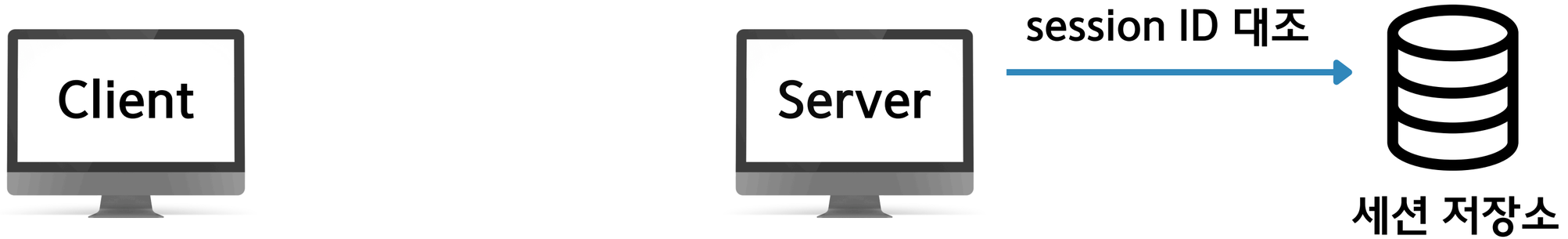
로그인



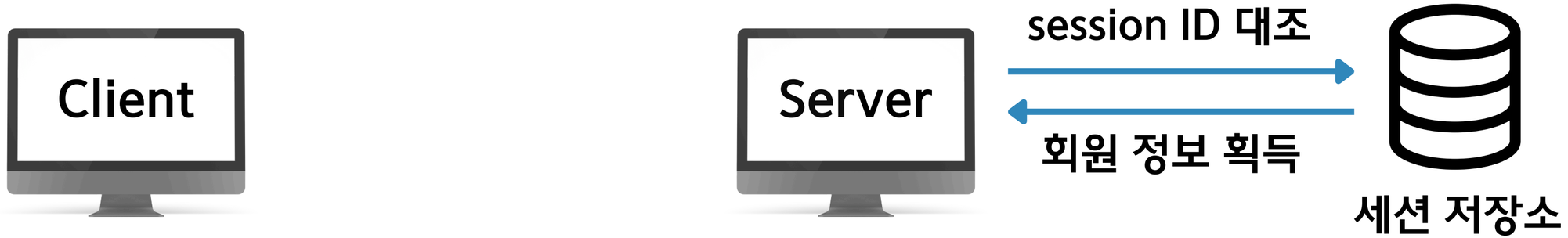
로그인

로그인 유지

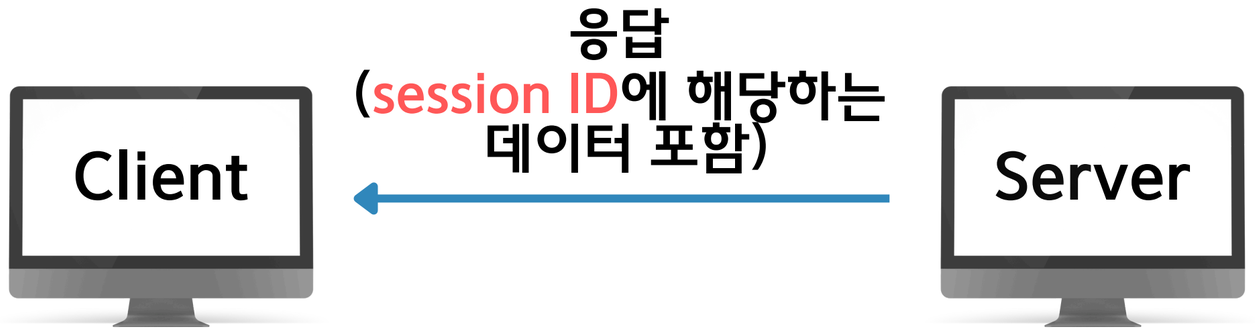
로그인 유지



로그인 유지



로그인 유지



규모가 커져 서버가 여러 개가 된다면,
세션을 어떻게 관리할 수 있을까요?

규모가 커져 서버가 여러 개가 된다면,
세션을 어떻게 관리할 수 있을까요?

1. 한 클라이언트가 반드시 동일 서버로 응답하게 하여 세션을 관리한다

   해당 서버에 문제가 생기게 된다면 연속성을 유지할 수 없다

2. 공용 세션 저장소를 만들어 모든 서버가 해당 저장소를 참조할 수 있도록 한다

   어떤 서버로 연결되든 연속성을 유지할 수 있다

   네트워크 비용이 증가한다

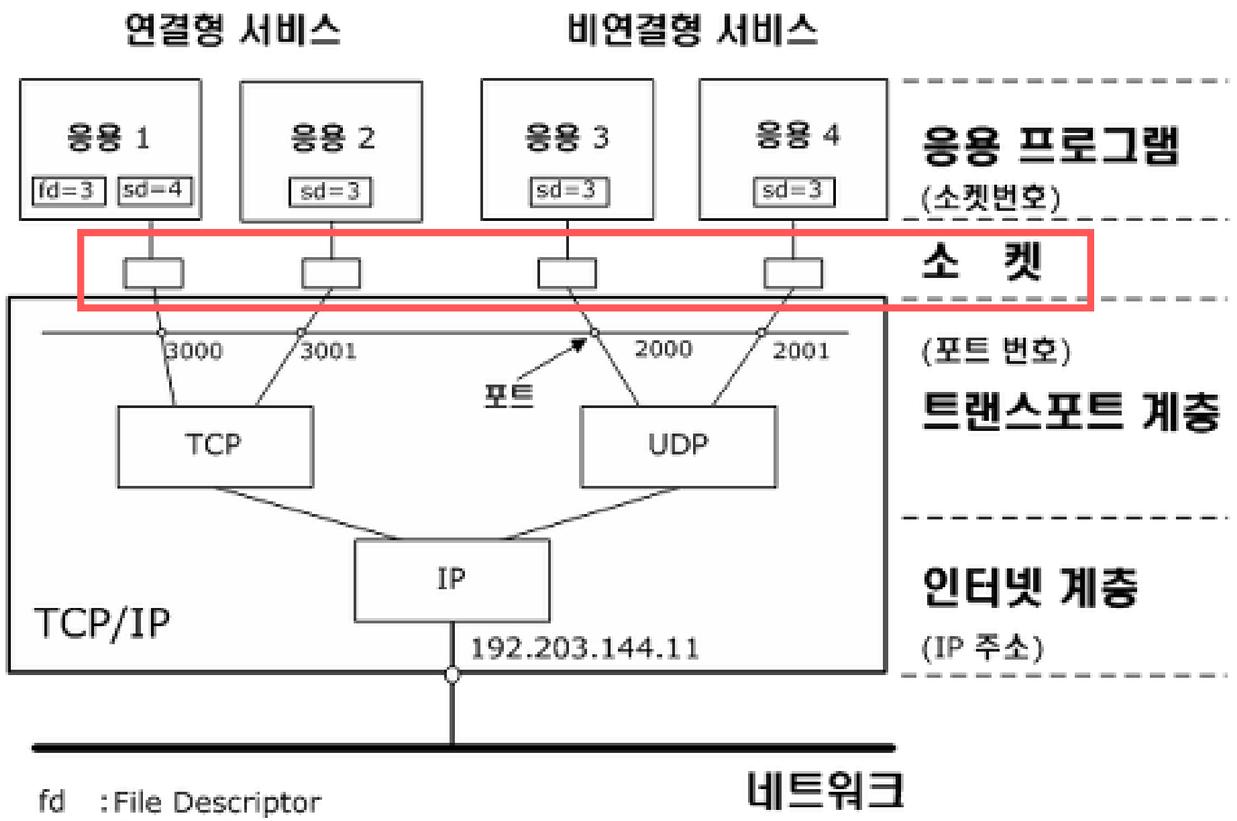
웹소켓과 소켓 통신

쏘켓?

소켓

프로그램이 네트워크 상에서
데이터를 송수신 하기 위한 연결부

일반적으로 TCP/IP 프로토콜을 이용

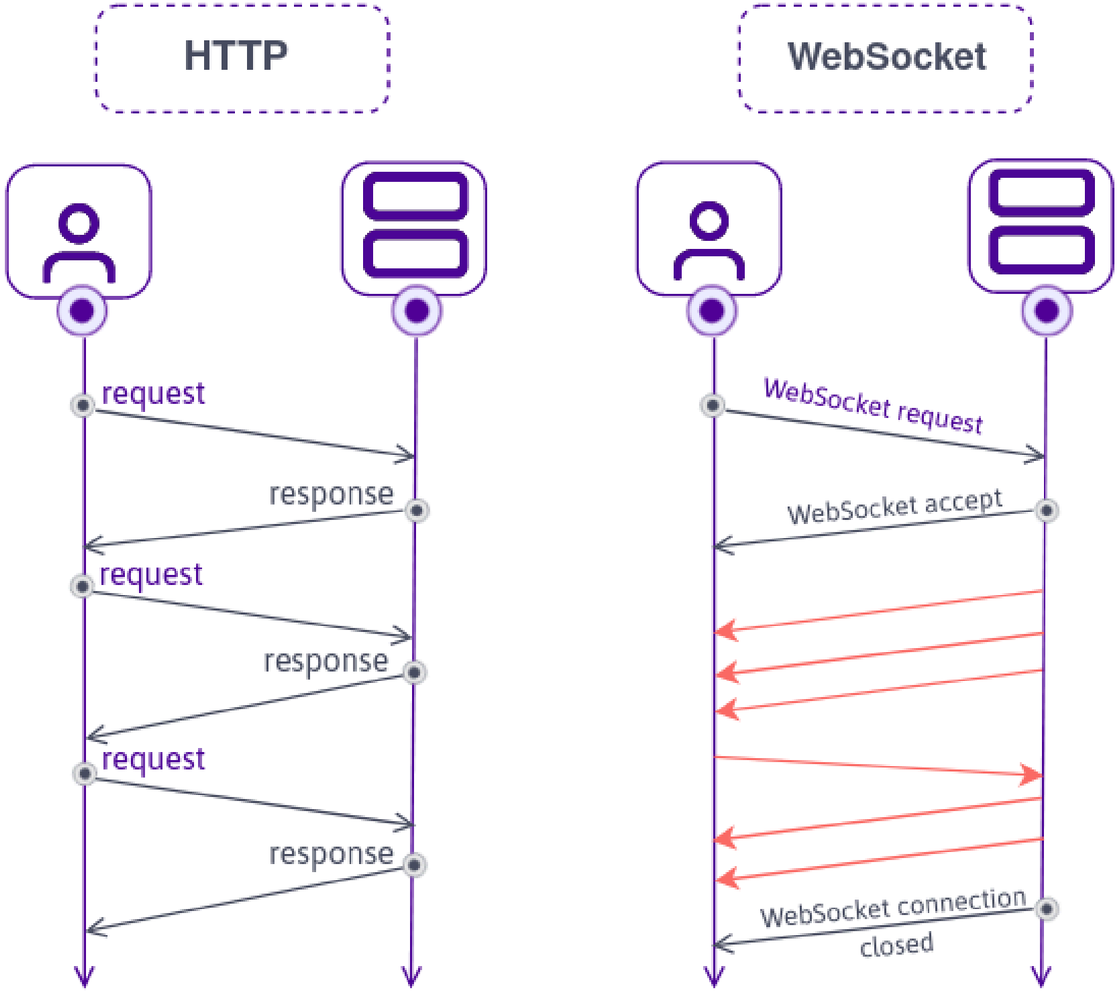


fd :File Descriptor
sd :Socket Descriptor
IP :Internet Protocol
TCP :Transmission Control Protocol
UDP: User Datagram Protocol

웹소켓?

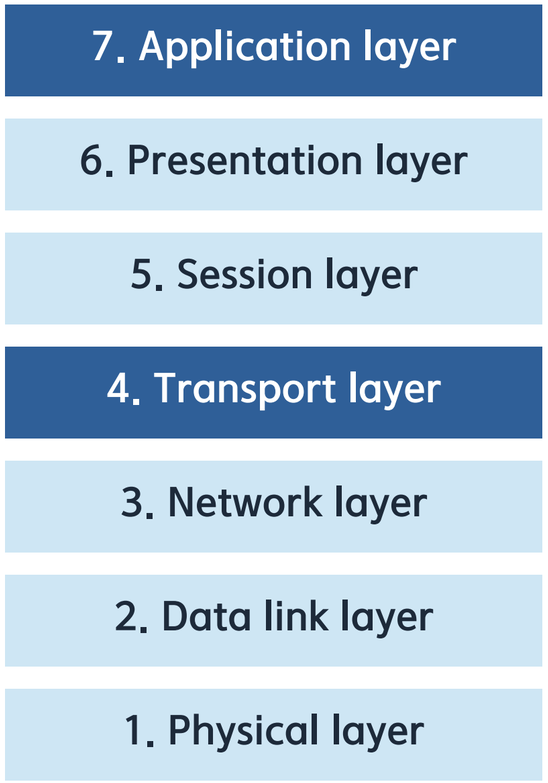
웹소켓

클라이언트와 서버 간의
실시간 **양방향 통신**을 위해
설계된 프로토콜


HTTP
request
response
request
response
request
response
WebSocket
WebSocket request
WebSocket accept
WebSocket connection
closed

소켓과 웹소켓의 차이

동작 계층 차이



TCP에 의존하지만 HTTP에 기반

--- WebSocket

인터넷 프로토콜에 기반

--- TCP socket, UDP socket

소켓과 포트의 차이

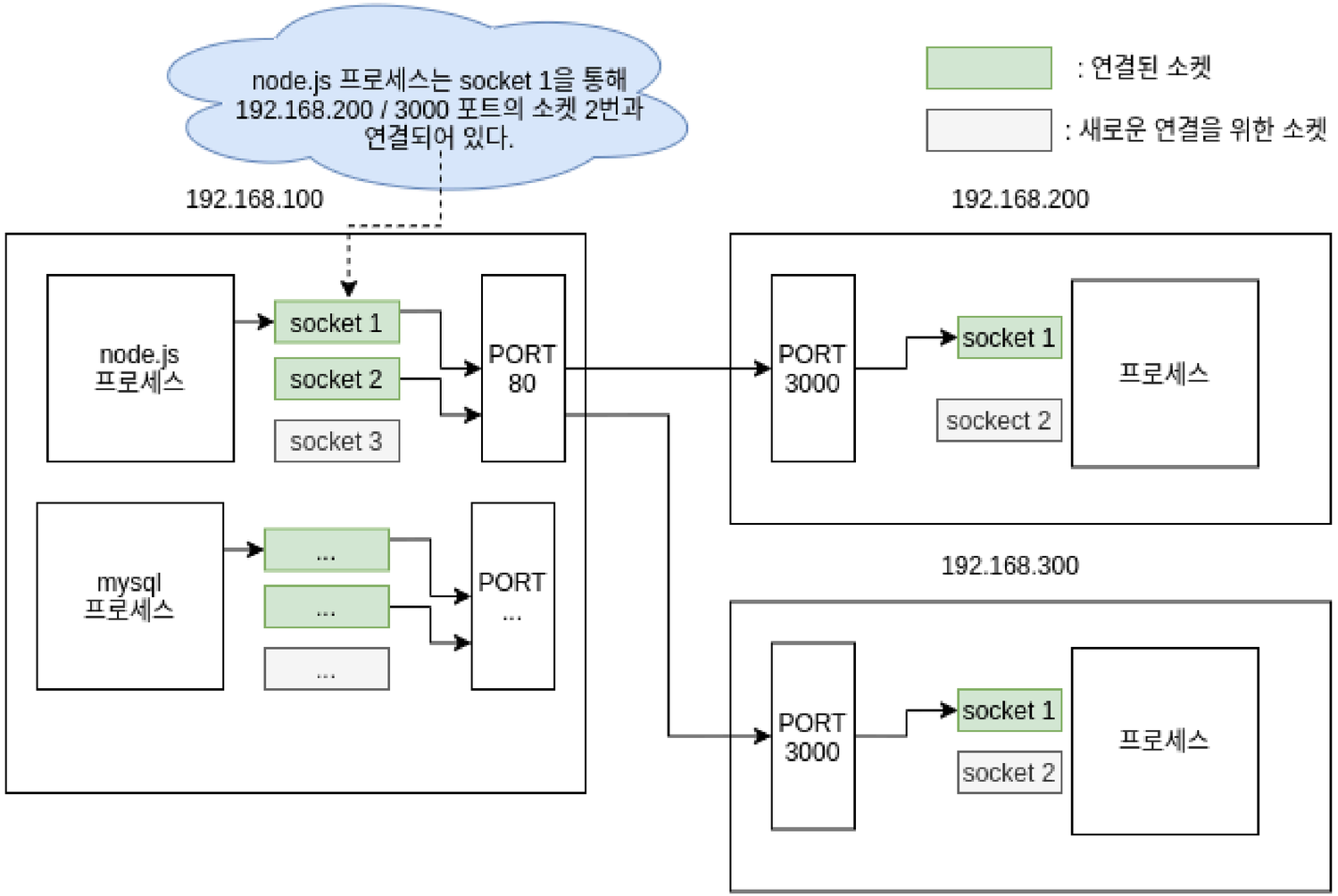
소켓과 포트의 차이

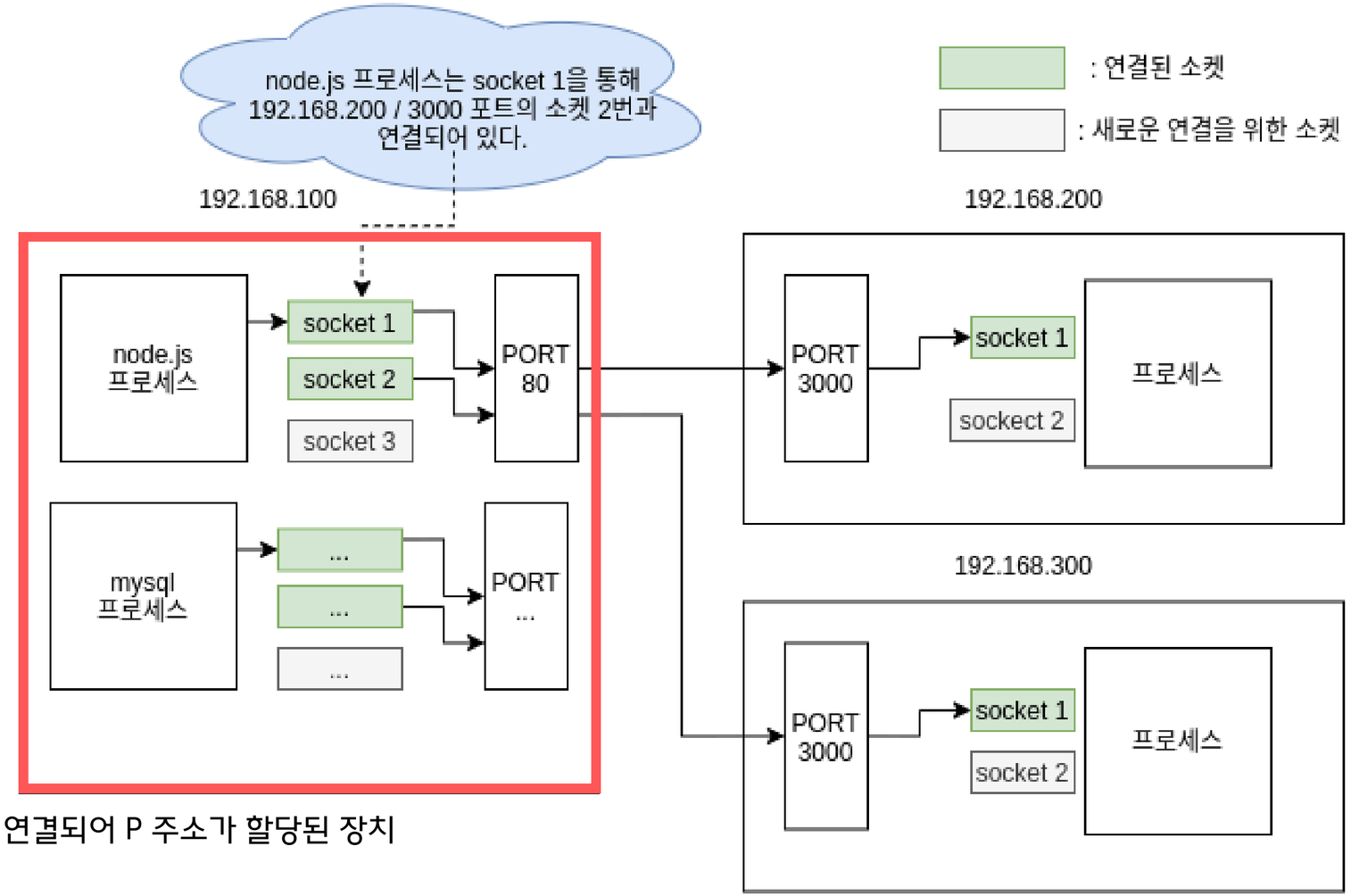
소켓

프로세스간에 소통을 하기 위한 통로

포트

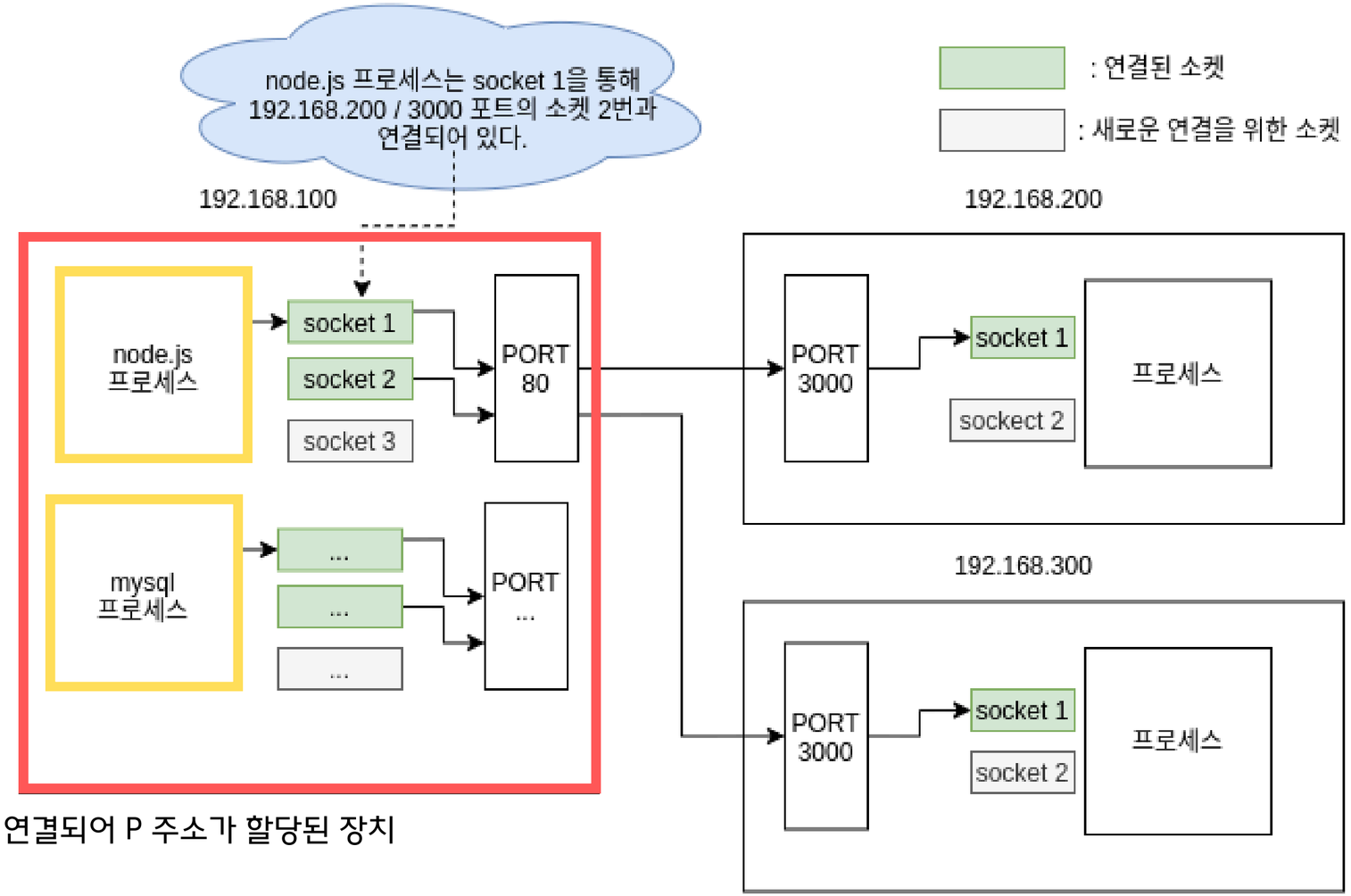
소켓을 연결하기 위한 통로


node.js 프로세스는 socket 1을 통해
192.168.200 / 3000 포트의 소켓 2번과
연결되어 있다.
: 연결된 소켓
: 새로운 연결을 위한 소켓
192.168.100
192.168.200
node.js
프로세스
socket 1
socket 2
socket 3
PORT
80
PORT
3000
socket 1
sockect 2
프로세스
mysql
프로세스
...
...
...
PORT
...
192.168.300
PORT
3000
socket 1
socket 2
프로세스



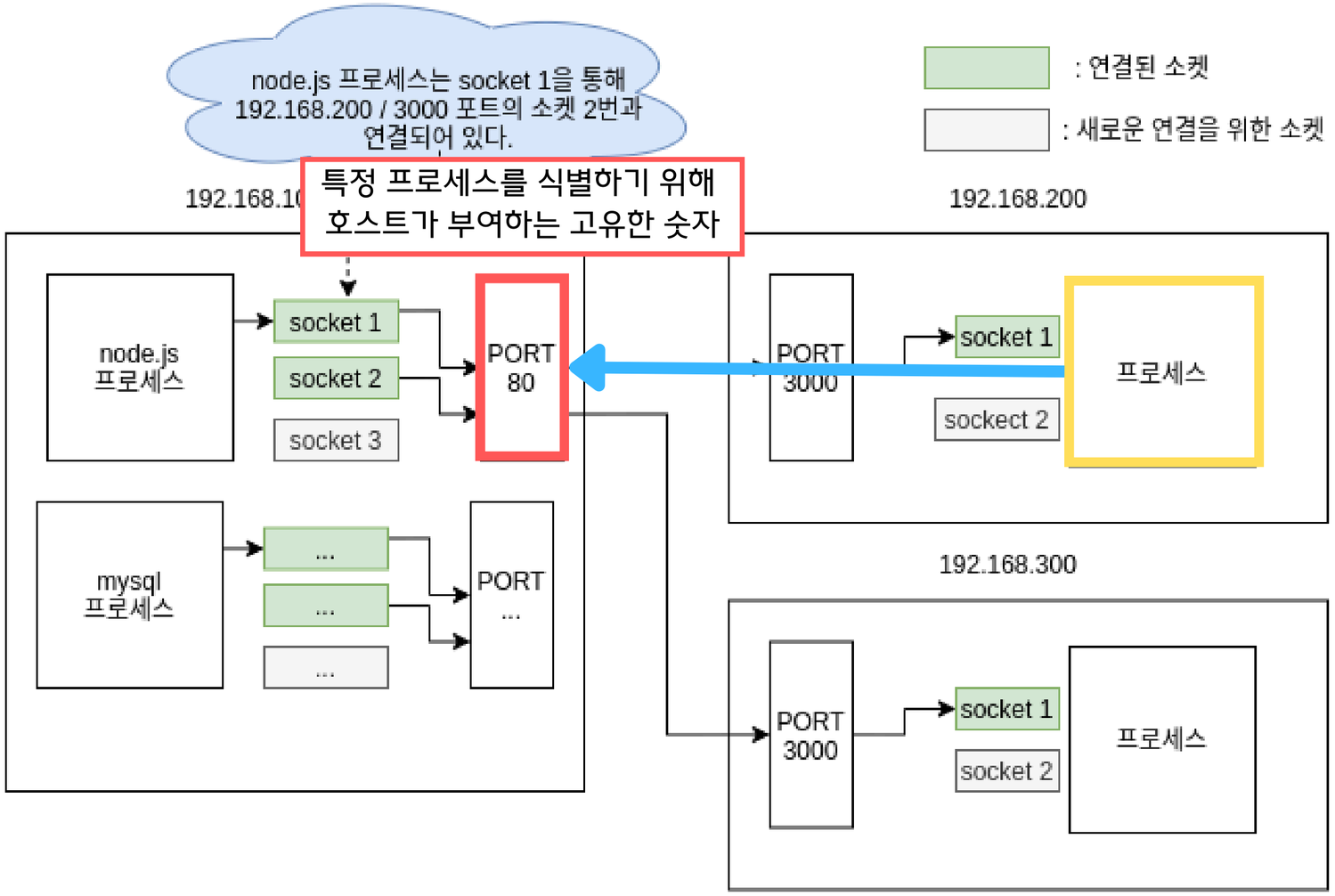
Host

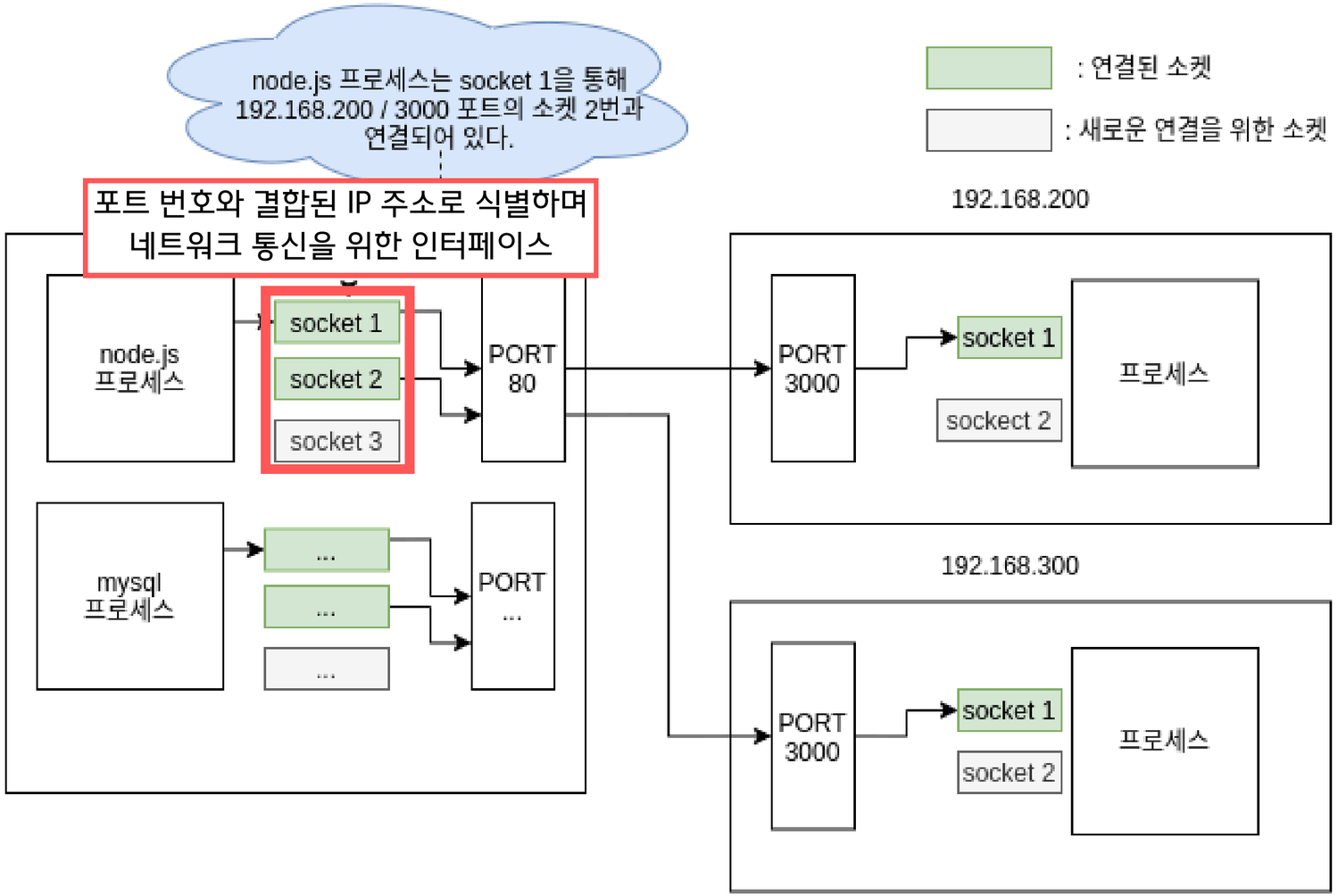
네트워크에 연결되어 P 주소가 할당된 장치



Host

네트워크에 연결되어 P 주소가 할당된 장치


node.js 프로세스는 socket 1을 통해
192.168.200 / 3000 포트의 소켓 2번과
연결되어 있다.
: 연결된 소켓
: 새로운 연결을 위한 소켓
특정 프로세스를 식별하기 위해
호스트가 부여하는 고유한 숫자
192.168.1
192.168.200
node.js
프로세스
socket 1
socket 2
socket 3
PORT
80
PORT
3000
socket 1
sockect 2
프로세스
mysql
프로세스
...
...
...
PORT
...
192.168.300
PORT
3000
socket 1
socket 2
프로세스


node.js 프로세스는 socket 1을 통해
192.168.200 / 3000 포트의 소켓 2번과
연결되어 있다.
포트 번호와 결합된 IP 주소로 식별하며
네트워크 통신을 위한 인터페이스
: 연결된 소켓
: 새로운 연결을 위한 소켓
192.168.200
node.js
프로세스
socket 1
socket 2
socket 3
PORT
80
mysql
프로세스
...
...
...
PORT
...
PORT
3000
socket 1
sockect 2
프로세스
192.168.300
PORT
3000
socket 1
socket 2
프로세스

여러 소켓이 있다고 할 때,
그 소켓의 포트 번호는 모두 다른가요?

여러 소켓이 있다고 할 때,
그 소켓의 포트 번호는 모두 다른가요?

모두 다를 수도 있고, 포트 번호가 같은 소켓이 있을 수도 있다.
포트 번호는 하나의 호스트내에서 고유하지만,
하나의 프로세스는 여러 개의 소켓을 열 수 있으므로
같은 IP, 같은 포트수를 가지고 있다 하더라도 여러 개의 소켓이 존재할 수 있습니다.

사용자의 요청이 무수히 많아지면,
소켓도 무수히 생성되나요?

사용자의 요청이 무수히 많아지면,
소켓도 무수히 생성되나요?

무수히 많은 소켓이 생성되는지 여부는
서버의 용량, 응용 프로그램의 설계,
사용되는 기술의 특성에 따라 달라집니다.

감사합니다