HTTP 커넥션 관리

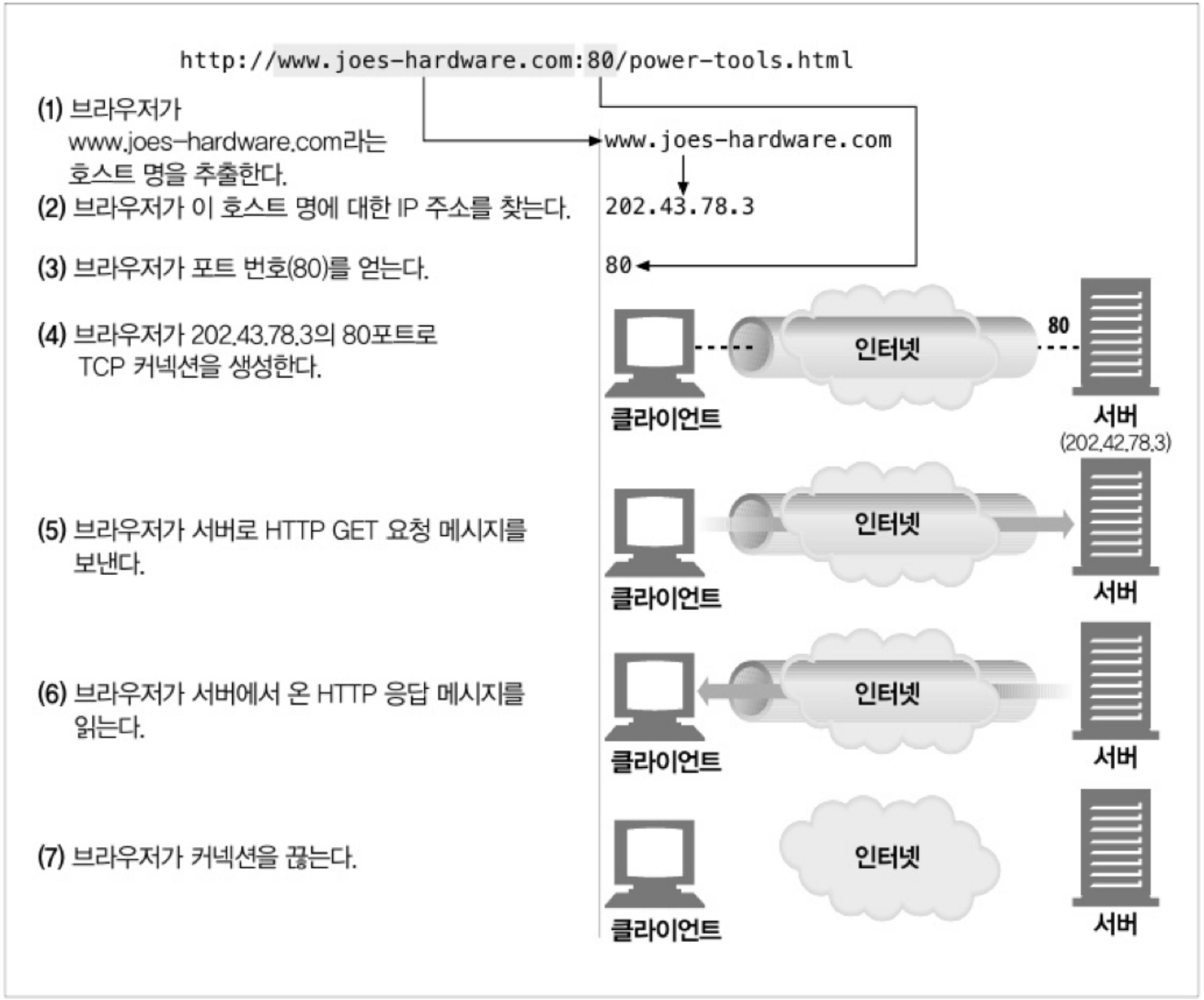
20210463 박연종

2024.04.17.

목차

땅다람쥐
ⓒ O'Reilly Media

HTTP 커넥션


http://www.joes-hardware.com:80/power-tools.html
(1) 브라우저가 www.joes-hardware.com라는 호스트 명을 추출한다.
(2) 브라우저가 이 호스트 명에 대한 IP 주소를 찾는다.
(3) 브라우저가 포트 번호(80)를 얻는다.
(4) 브라우저가 202.43.78.3의 80포트로 TCP 커넥션을 생성한다.
(5) 브라우저가 서버로 HTTP GET 요청 메시지를 보낸다.
(6) 브라우저가 서버에서 온 HTTP 응답 메시지를 읽는다.
(7) 브라우저가 커넥션을 끊는다.
www.joes-hardware.com
202.43.78.3
80
인터넷
80
클라이언트
서버
(202.42.78.3)
인터넷
클라이언트
서버
인터넷
클라이언트
서버
인터넷
클라이언트
서버

TCP 커넥션

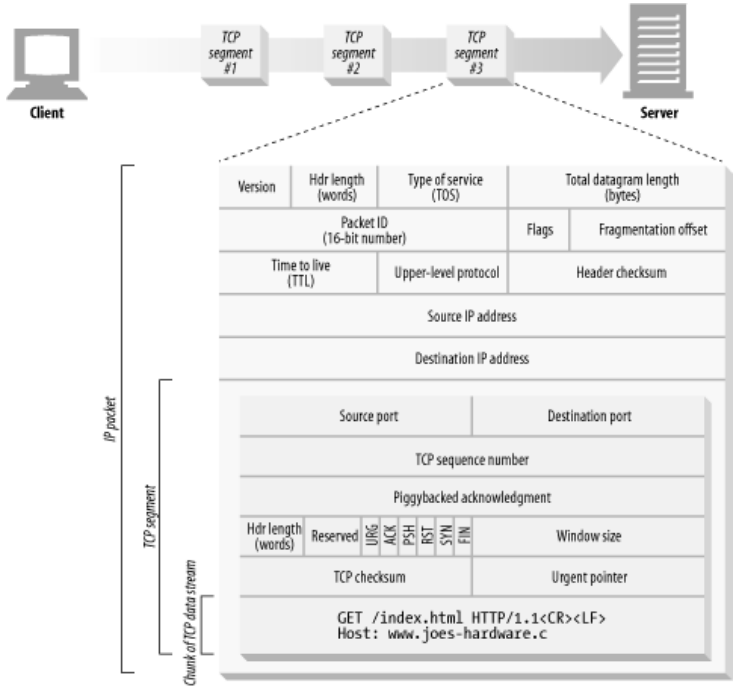
- 모든 HTTP 통신은 TCP/IP를 통해 이루어지며 몇 개의 규칙을 제외하고는 TCP 커넥션에 불과

신뢰할 수 있는 데이터 전송 통로, TCP

- TCP 커넥션 한쪽에 있는 바이트들은 반대쪽으로 충돌없이 순서에게 맞게 정확하게 전달됨

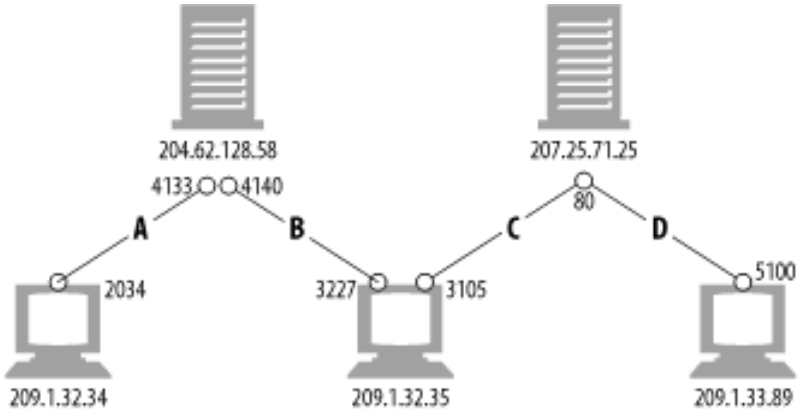
TCP 스트림은 세그먼트로 나뉘어 IP 패킷을 통해 전송

- HTTP 메시지 전송 시 현재 연결된 TCP 커넥션을 통해 메시지 내용을 순서대로 전송
- 전송 시 데이터 스트림을 세그먼트 단위로 나누고 이를 하나의 IP 주소에서 다른 IP 주소로 IP 패킷에 담아 전달



TCP 커넥션

- 컴퓨터는 포트 번호를 통해 여러 개의 TCP 커넥션 유지
- 네 가지 값으로 유일한 커넥션 생성



< 발신자 IP, 발신자 포트, 수신자 IP, 수신자 포트 >

- `IP` 각 호스트를 식별하기 위함
- `Port` 실행 중인 프로세스를 구분하기 위함
  서로 다른 프로세스는 서로 다른 포트 사용

TCP 커넥션 과정

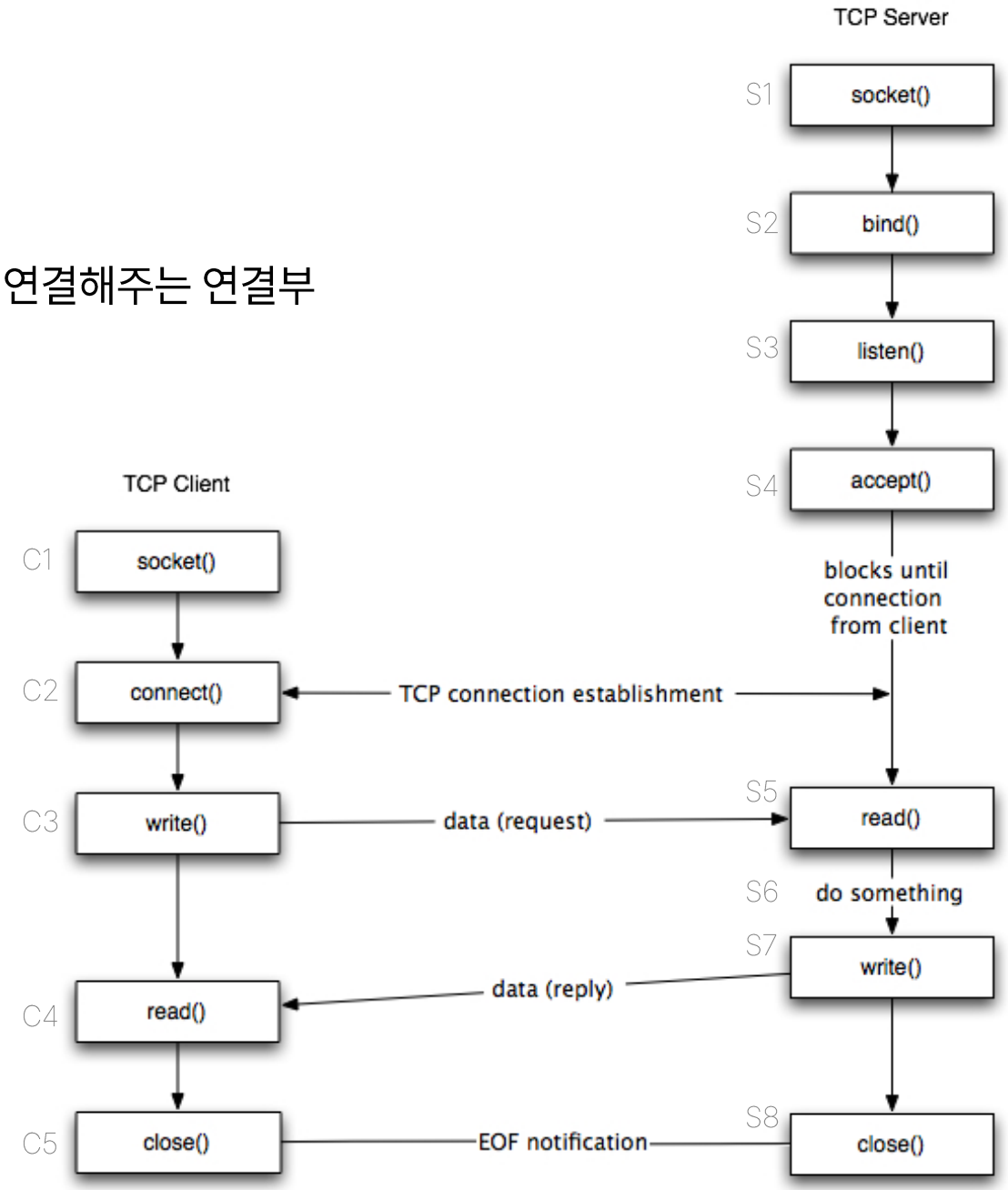
소켓 프로그램이 네트워크에서 데이터를 통신할 수 있도록 연결해주는 연결부

S1 새로운 듣기 소켓 생성

S2 듣기 소켓에 80번 포트 할당

S3 듣기 소켓을 연결 요청 대기 상태로 변경

S4 연결 소켓을 생성해 요청을 기다림



TCP 커넥션 과정

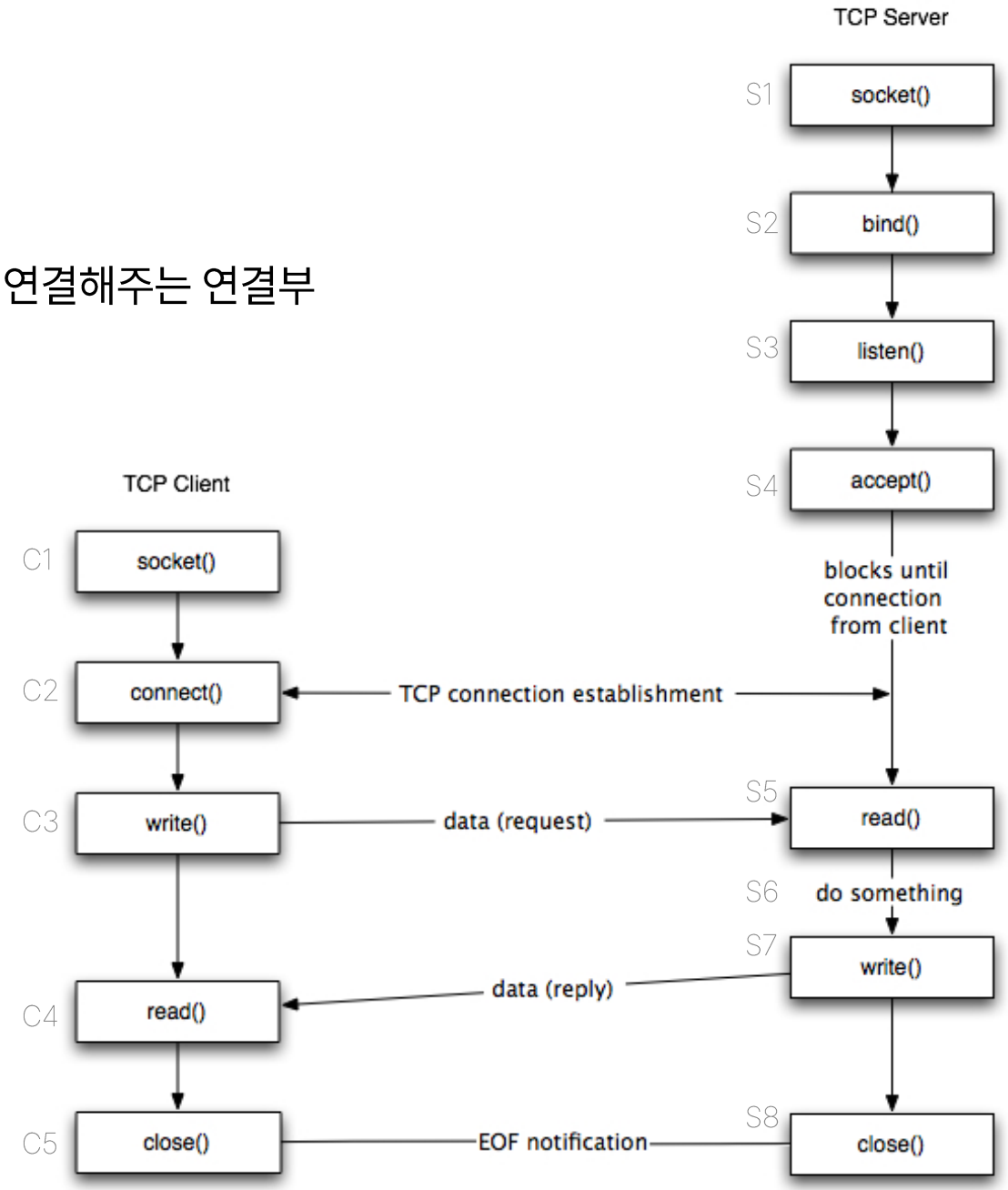
소켓 프로그램이 네트워크에서 데이터를 통신할 수 있도록 연결해주는 연결부

C1 서버의 IP 주소와 포트를 얻어 새로운 소켓 생성

C2 서버에 연결 요청

S5 요청을 읽음

S6 HTTP 요청 메시지를 처리



TCP 커넥션 과정

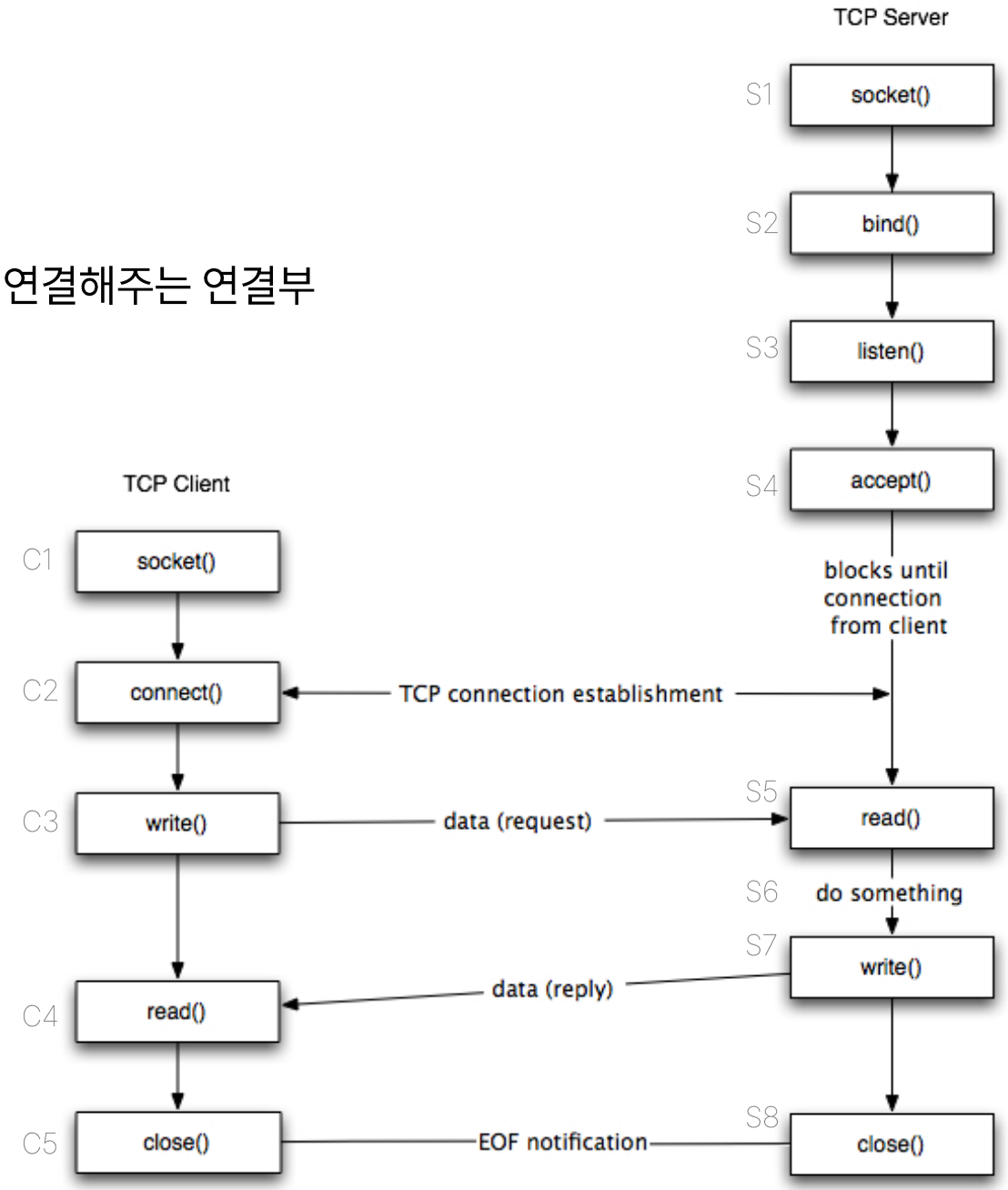
소켓 프로그램이 네트워크에서 데이터를 통신할 수 있도록 연결해주는 연결부

S7 HTTP 응답을 보냄

S8 커넥션을 종료

C4 HTTP 응답을 처리

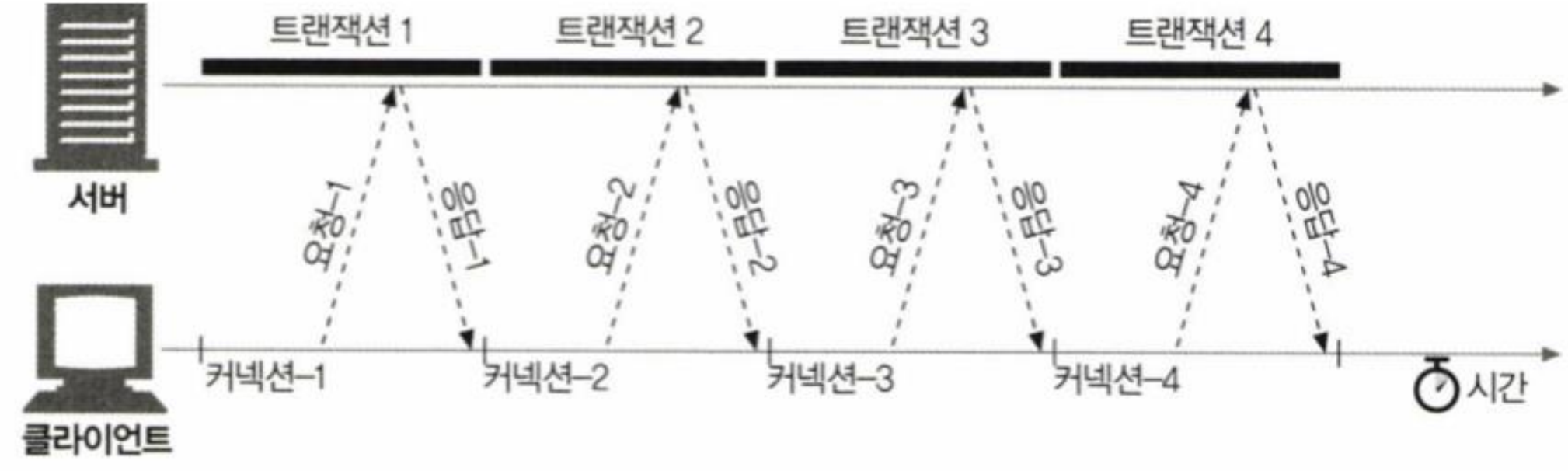
C5 커넥션을 종료



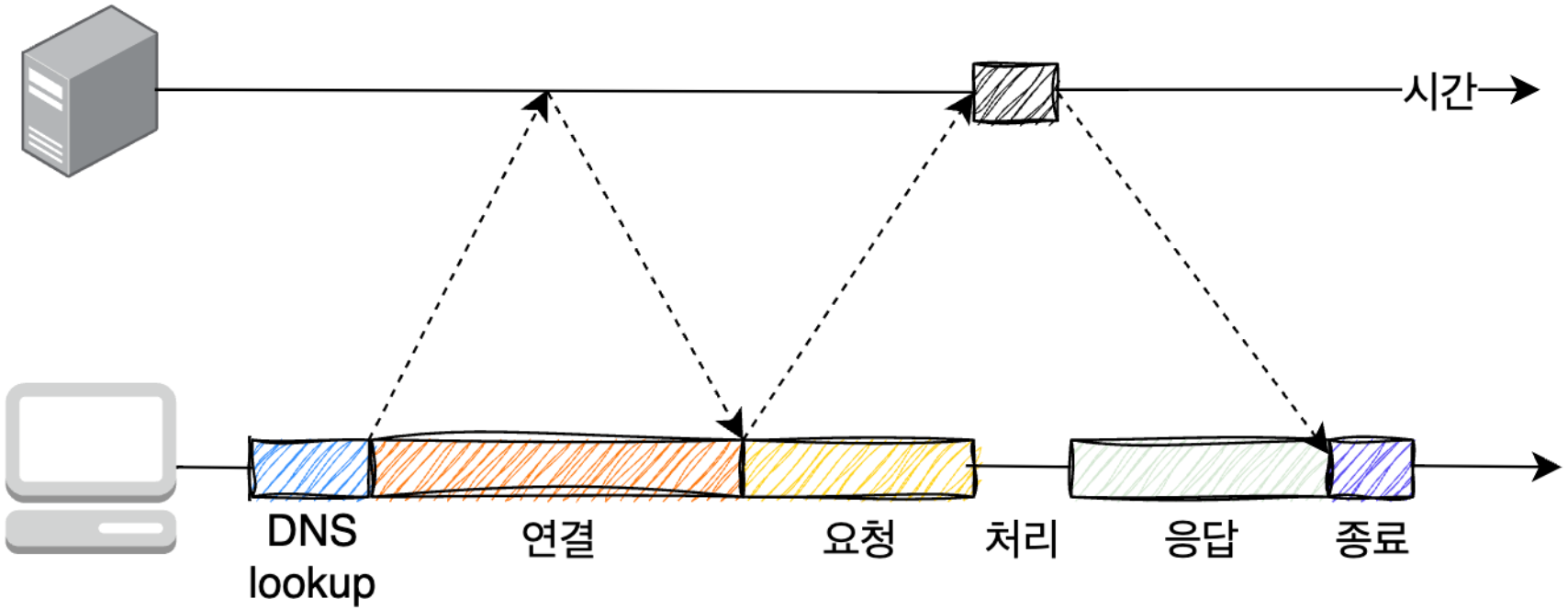
순차적인 트랜잭션 처리에 의한 지연

트랜잭션 데이터베이스 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위

i.e.) 사용자의 요청에 대한 서버 단에서의 처리 과정

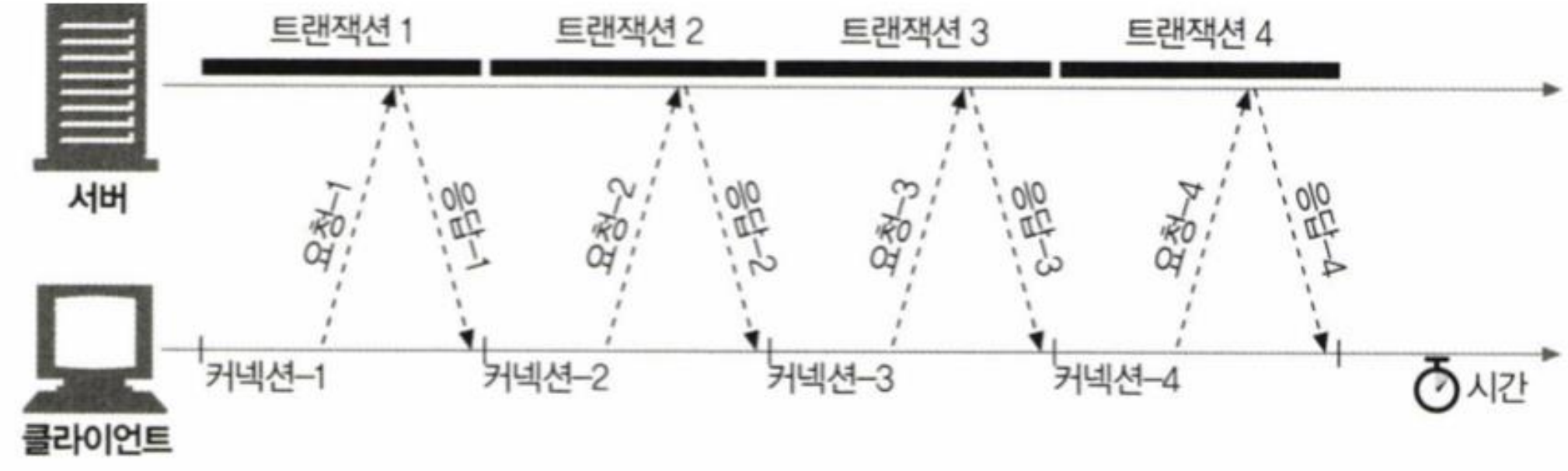


순차적인 트랜잭션 처리에 의한 지연



순차적인 트랜잭션 처리에 의한 지연

TCP의 느린 시작 패킷 전송에 사용 가능한 대역폭을 감지하고 네트워크 연결 속도의 균형을 맞추는 데 사용되는 알고리즘



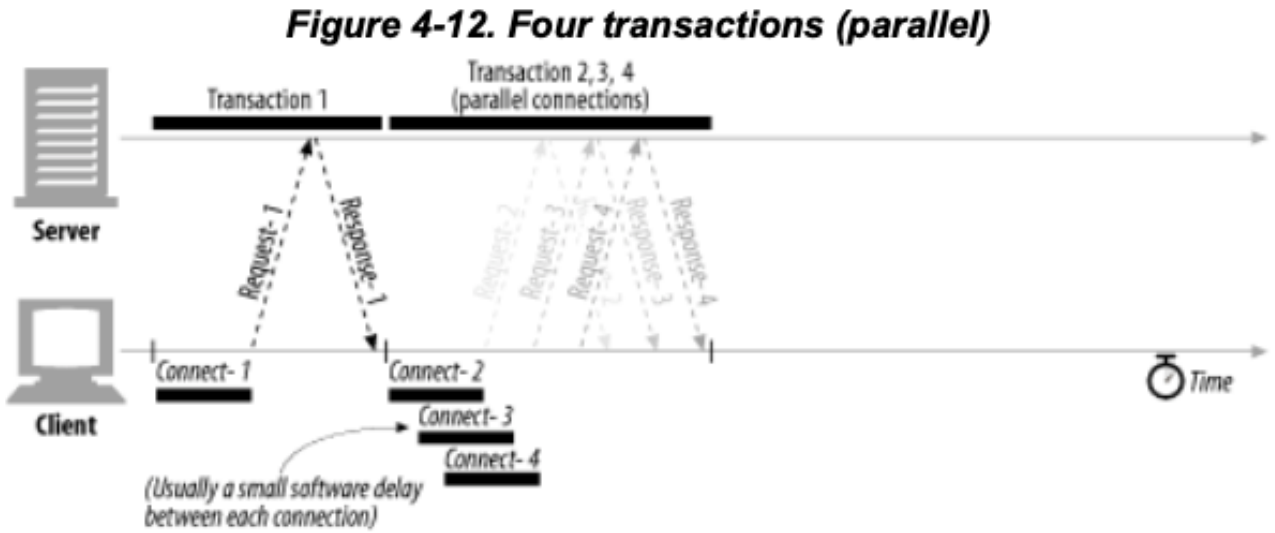
HTTP 커넥션 관리

D HTTP 커넥션 향상 기술

- **a** 병렬 커넥션
- **b** 지속 커넥션
- **c** HTTP 파이프라이닝
- **d** 멀티플렉싱

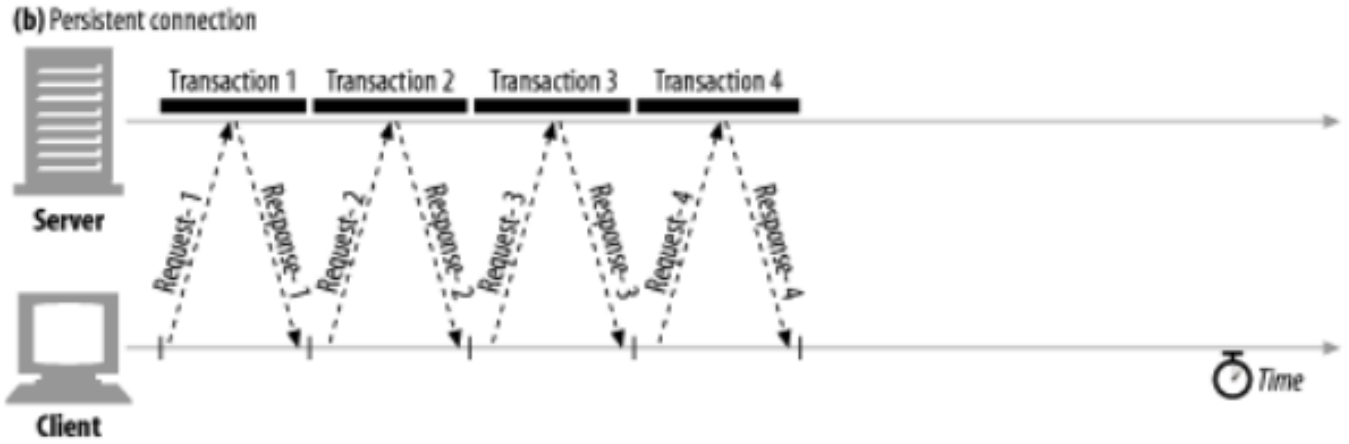
병렬 커넥션

- 네트워크 대역폭을 하나의 커넥션이 다 쓰는 것이 아닌 여러 개의 커넥션이 쓰도록 함
- 웹 페이지를 더 빠르게 내려 받을 수 있지만 항상 그런 것은 아님 (느린 네트워크 속도 환경에서의 병렬 커넥션)
- 사용자가 웹 페이지 접근 속도가 더 빠르다고 느낄 수 있음 (하나씩 보여짐 versus 동시에 보여짐)
- 다수의 커넥션은 메모리를 많이 소비하고 자체적인 성능 문제를 야기할 수 있음 (많은 사용자가 커넥션을 여러 개 맺으면 서버는...)



지속 커넥션

- 사용자는 보통 같은 사이트에 여러 개의 커넥션을 맺음 (사이트 지역성)
- 이미 맺어진 커넥션을 재사용하여 커넥션을 맺기 위한 준비 시간을 절약할 수 있음
- 이미 맺어진 커넥션은 TCP의 느린 시작 지연을 피할 수 있음



지속 커넥션

HTTP/1.0+에서의 지속 커넥션, keep-alive 커넥션

- HTTP 헤더의 Connection 부분에 keep-alive 옵션이 있으면 커넥션이 유지되기를 요청할 수 있음
- Connection 헤더를 이해하지 못하는 프록시로 인해 브라우저의 요청이 무시되고 커넥션은 유지되는 문제점 발생

HTTP/1.1에서의 지속 커넥션, 지속 커넥션

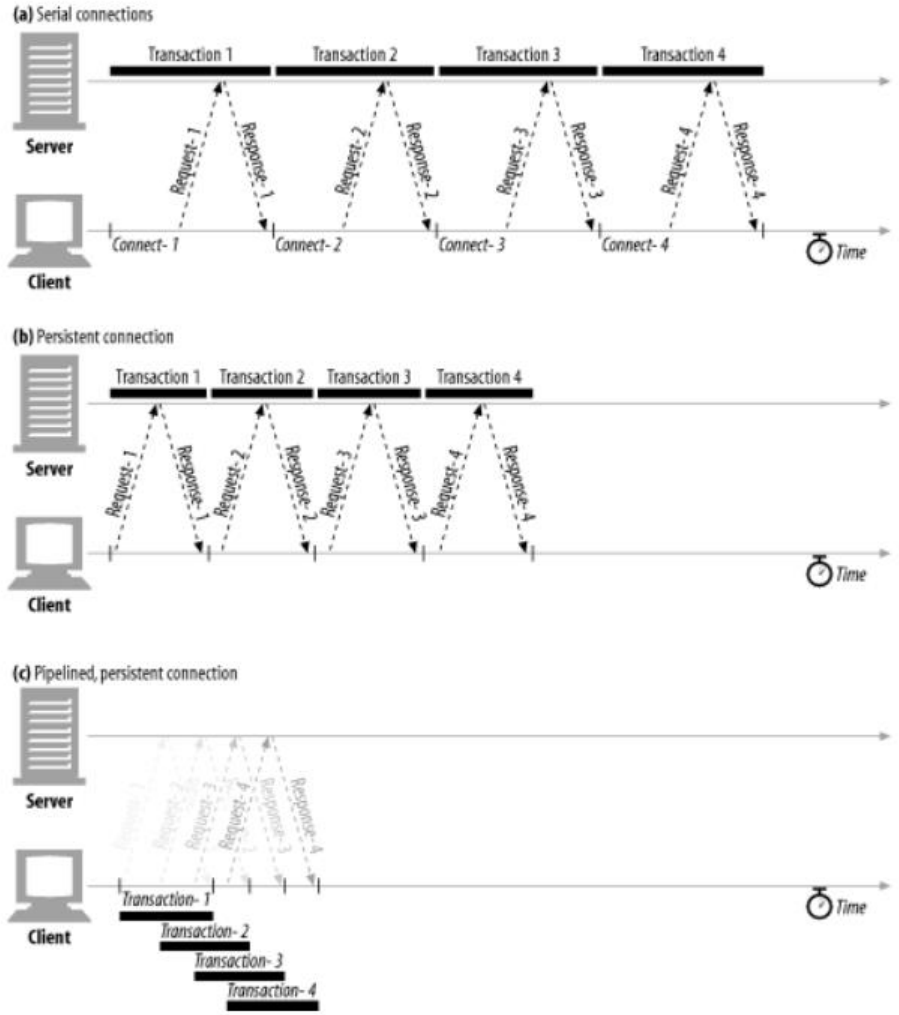
- 모든 커넥션을 지속 커넥션으로 취급하고 keep-alive 옵션을 없앰
- HTTP 헤더의 Connection 부분에 close 옵션이 있으면 커넥션을 끊을 수 있음
- 프록시는 커넥션 관련 기능의 클라이언트 지원 범위를 모르면 지속 커넥션을 맺어서는 안 됨

HTTP 파이프라이닝

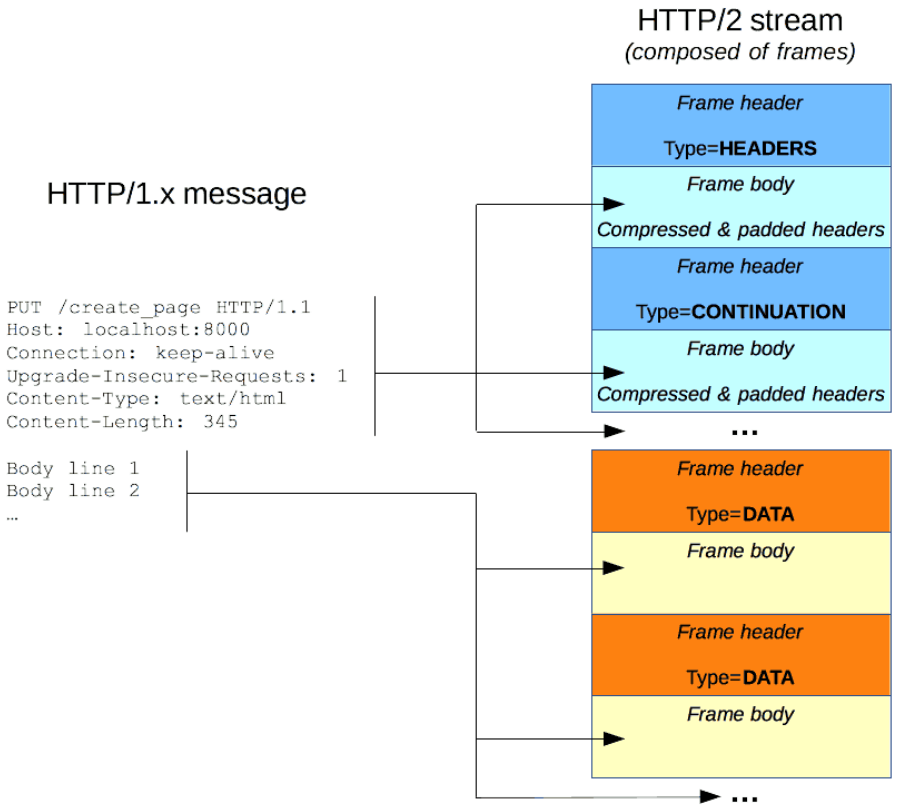
- 응답을 기다리는 중에 새로운 요청을 해 다음 요청까지의 대기 시간을 없애 네트워크 가동율을 높이고 성능을 향상시킴

- 서버에 변화가 생기는 요청을 파이프라인 커넥션으로 반복해서 보내면 문제가 발생할 수 있음
- 파이프라이닝은 요청만 먼저 보내지 응답은 요청한 순서대로 받아야 하므로 앞의 요청이 지연되면 뒤의 요청이 지연돼 응답 대기시간이 지연되는 문제 발생 (Head-of-line blocking)

HTTP/1.1의 커넥션 성능 향상 기법



멀티플렉싱

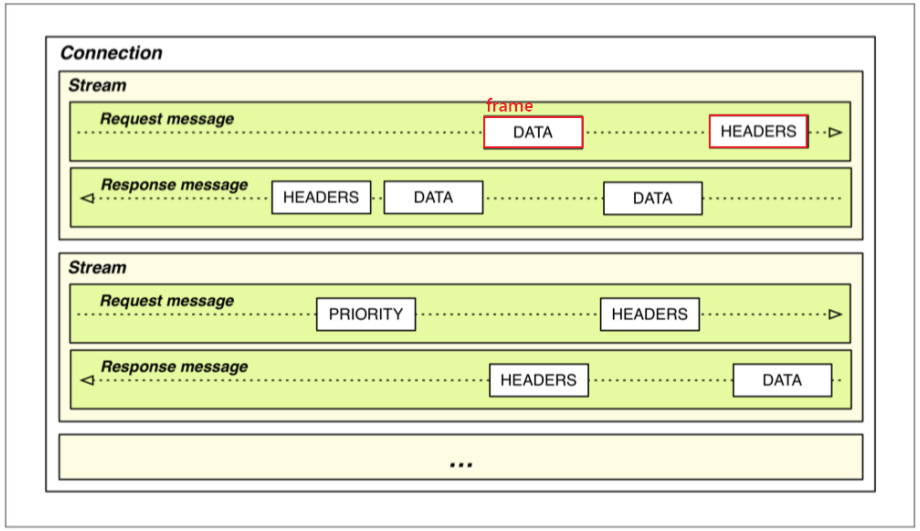


- HTTP 요청과 응답은 Status Line, Header와 Body로 이루어진 메시지라는 단위로 구성되었으나 새로운 단위가 추가됨

Frame 제일 작은 정보의 단위이며, Header 혹은 Data 둘 중 하나

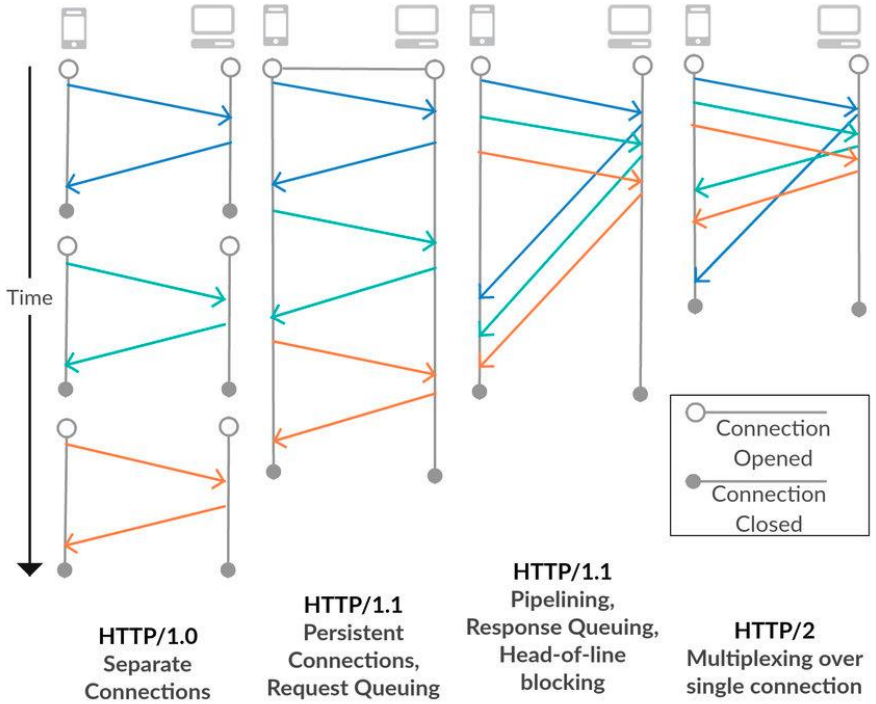
Message 요청 혹은 응답의 단위이며, 다수의 Frame으로 이루어짐

Stream 맺어진 커넥션을 통해 양방향으로 주고받는 다수의 Message



멀티플렉싱

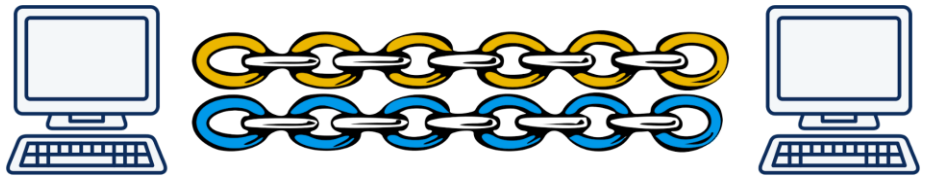
HTTP/2.0의 커넥션 성능 향상 기법



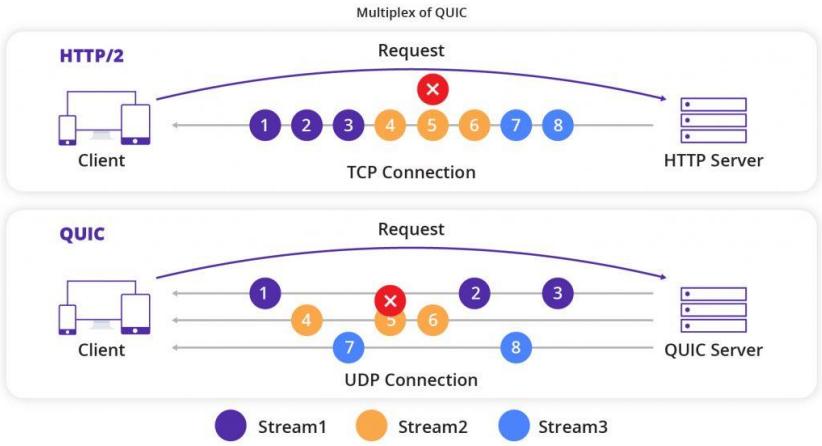
- 스트림을 사용해 하나의 커넥션으로 여러 개의 데이터를 전송할 수 있음
- HTTP/2.0에서는 기존 텍스트 기반(아스키 코드로 작성됨)의 하나의 HTTP 메시지를 바이너리 기반의 여러 개의 프레임으로 쪼갬
- 요청 순서 상관 없이 완료되는 대로 응답을 보냄
- 파이프라이닝에서의 HOL 문제가 해결됨

개선된 멀티플렉싱

HTTP/2.0에서의 멀티플렉싱



HTTP/3.0에서의 멀티플렉싱



HTTP/2.0에서의 멀티플렉싱

- TCP를 사용하므로 데이터를 가상회선 패킷 교환 방식을 이용해 연결 지향형임
- 스트림이 하나의 커넥션을 공유하고 번갈아 가며 패킷이 전달됨
- 중간에 패킷 손실 발생 시 다른 리소스가 지연되는 발생 (Head-of-line blocking)

HTTP/3.0에서의 멀티플렉싱

- UDP를 사용한 QUIC 프로토콜이 적용돼 데이터그램 패킷 교환 방식을 이용해 비연결 지향형임
- 독립적으로 스트림이 전송되므로 한 스트림의 손실이 다른 스트림에 영향을 주지 않게 됨

참고자료

- 데이빗 고울리, 브라이언 토티, 마조리 세이어, 세일루 레디, 안슈 아가왈 'HTTP 완벽 가이드' : O'Reilly, 인사이트
- 김정민 'TCP 커넥션이란 무엇인가' : Ratel의 개발일지
- kimmin-ko '[HTTP 완벽 가이드 4장] 커넥션 관리' : Focus on today
- 구나영 'HTTP 커넥션 관리' : mooongs.log
- Stefan Cho '[HTTP] 커넥션 관리에 대해서' : devstefancho.log
- 이인애 'HTTP Connection의 관리 (TCP 지연 방지)' : kel.log
- 여러 기여자 'HTTP/1.x의 커넥션 관리' : MDN Web Docs
- 신기영 'HTTP Connection 헤더 쉽게 알아보기' : Gidhub

땅다람쥐

감사합니다

땅다람쥐