2025.10.16

HTTP Version

network

HTTP Version

CONTENTS

01

HTTP 1.0 이전

World Wide Web (1989)

HyperText System

- HTTP
- HTML
- World Wide Web (브라우저)
- httpd (웹 서버)

HTTP 0.9 (1991)

```
(Connection 1 Establishment - TCP Three-Way Handshake)
Connected to xxx.xxx.xxx.xxx

(Request)
GET /my-page.html

(Response in hypertext)
<HTML>
A very simple HTML page
</HTML>
```

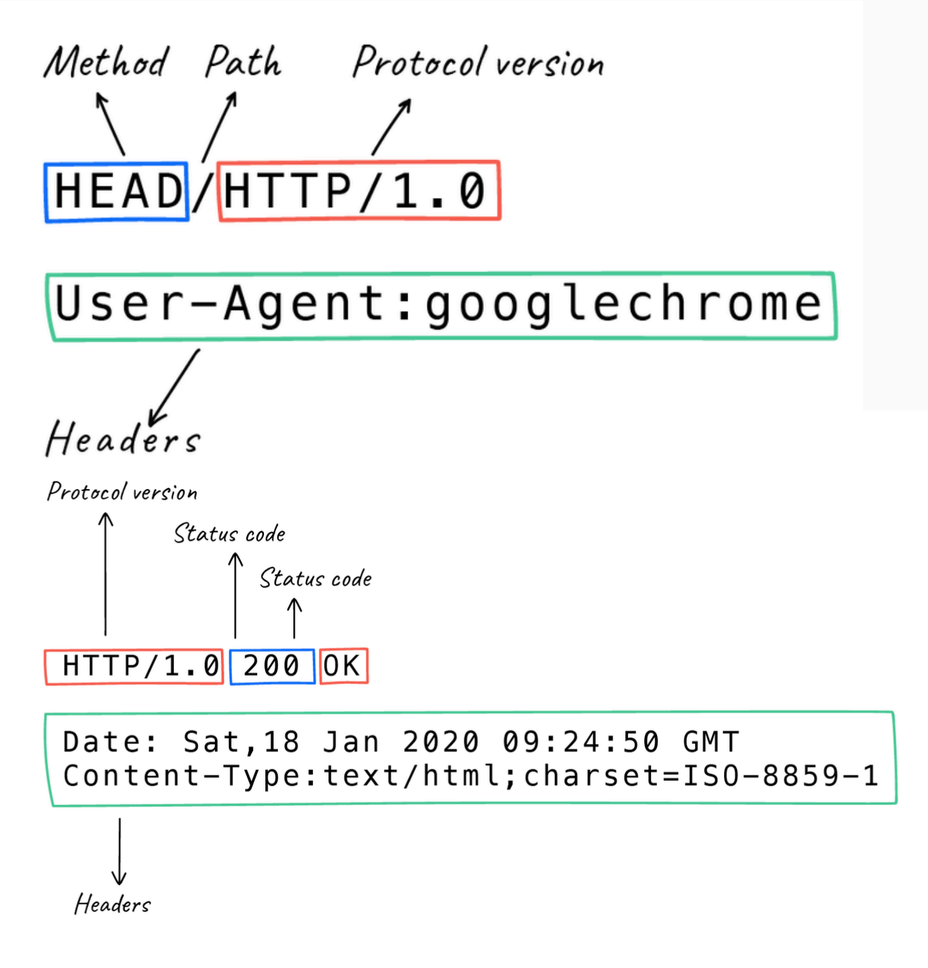
특징

- GET 메소드만 사용 가능
- 응답은 HTML 파일만 가능
- 상태코드 없음
- HTTP 헤더 없음

02

HTTP 1.0

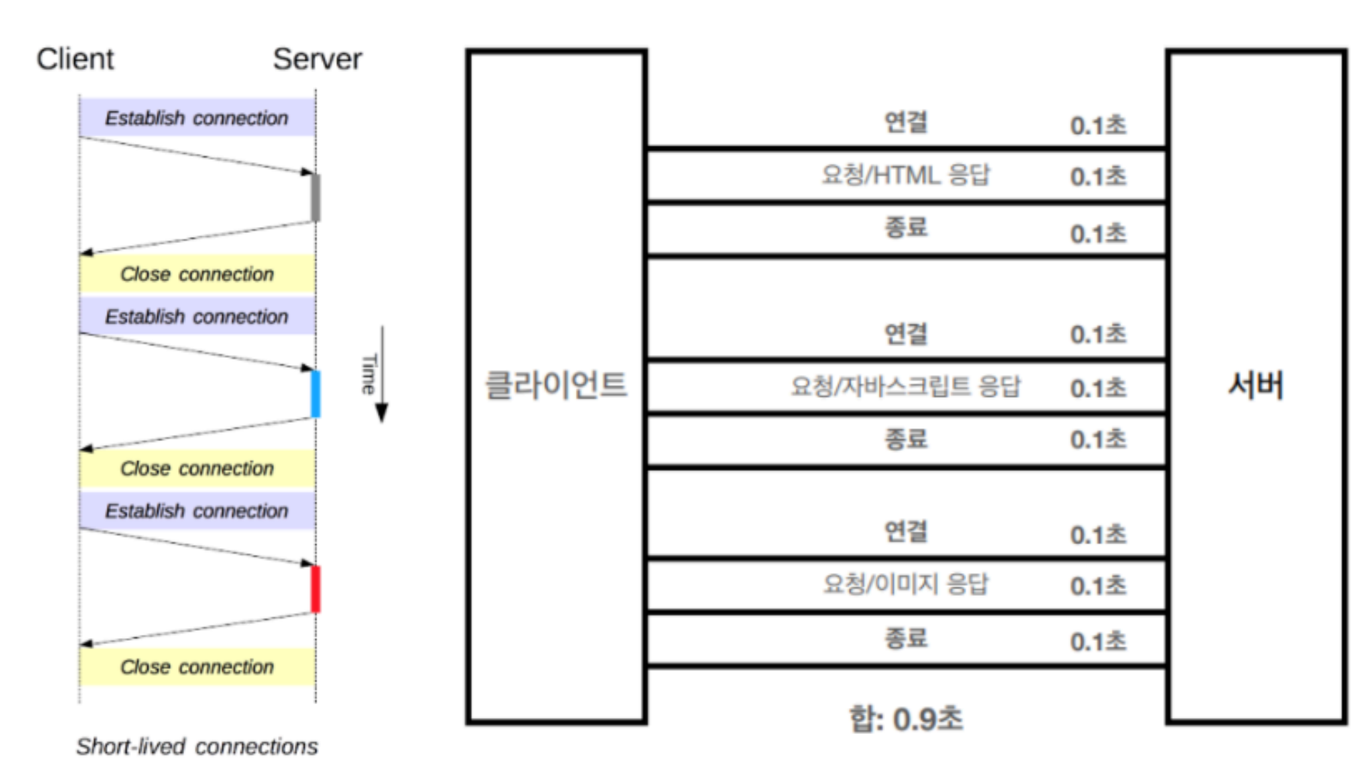
HTTP 1.0 (1996)



특징

- 버전 정보 명시
- HTTP 헤더 도입
  - Content-Type, User-Agent, Server
- 상태 코드 도입
- POST, HEAD 메소드 사용

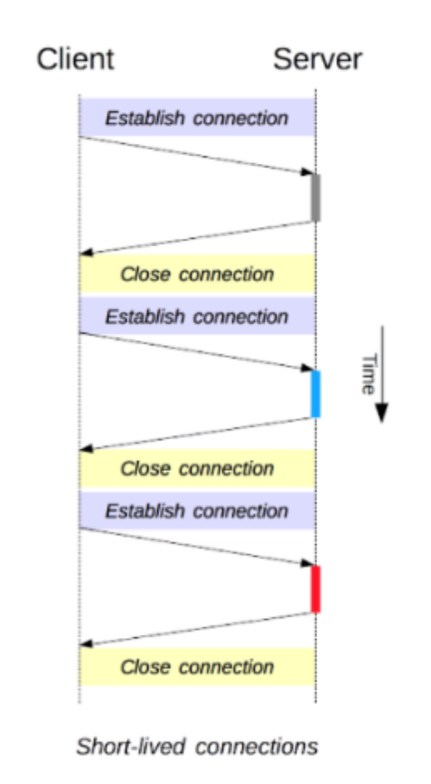
HTTP 1.0 (1996)

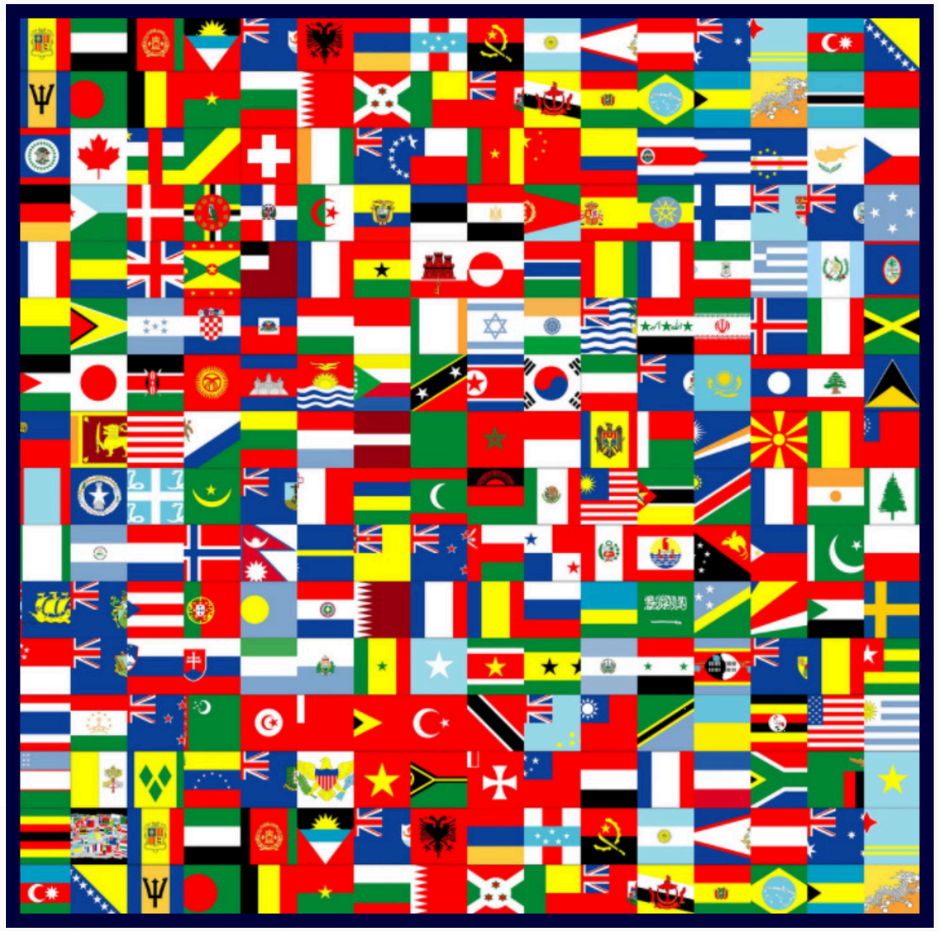


Short-lived Connections

- 매 요청마다 연결을 하고 끊는 방식

HTTP 1.0 (1996)





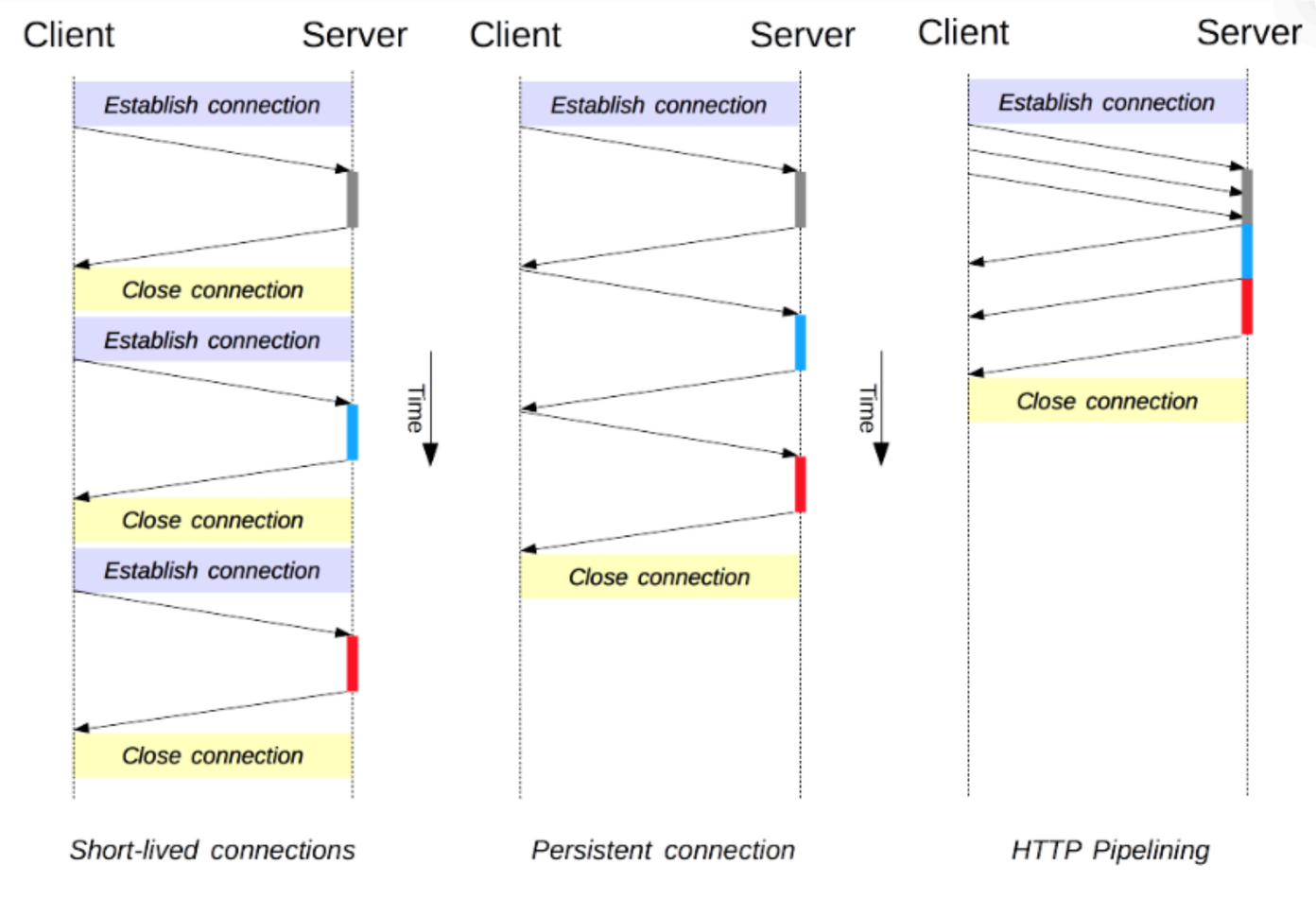
Short-lived Connections

- 매 요청마다 연결을 하고 끊는 방식
- 사진이 100장이 있는 웹 페이지라면?

03

HTTP 1.1

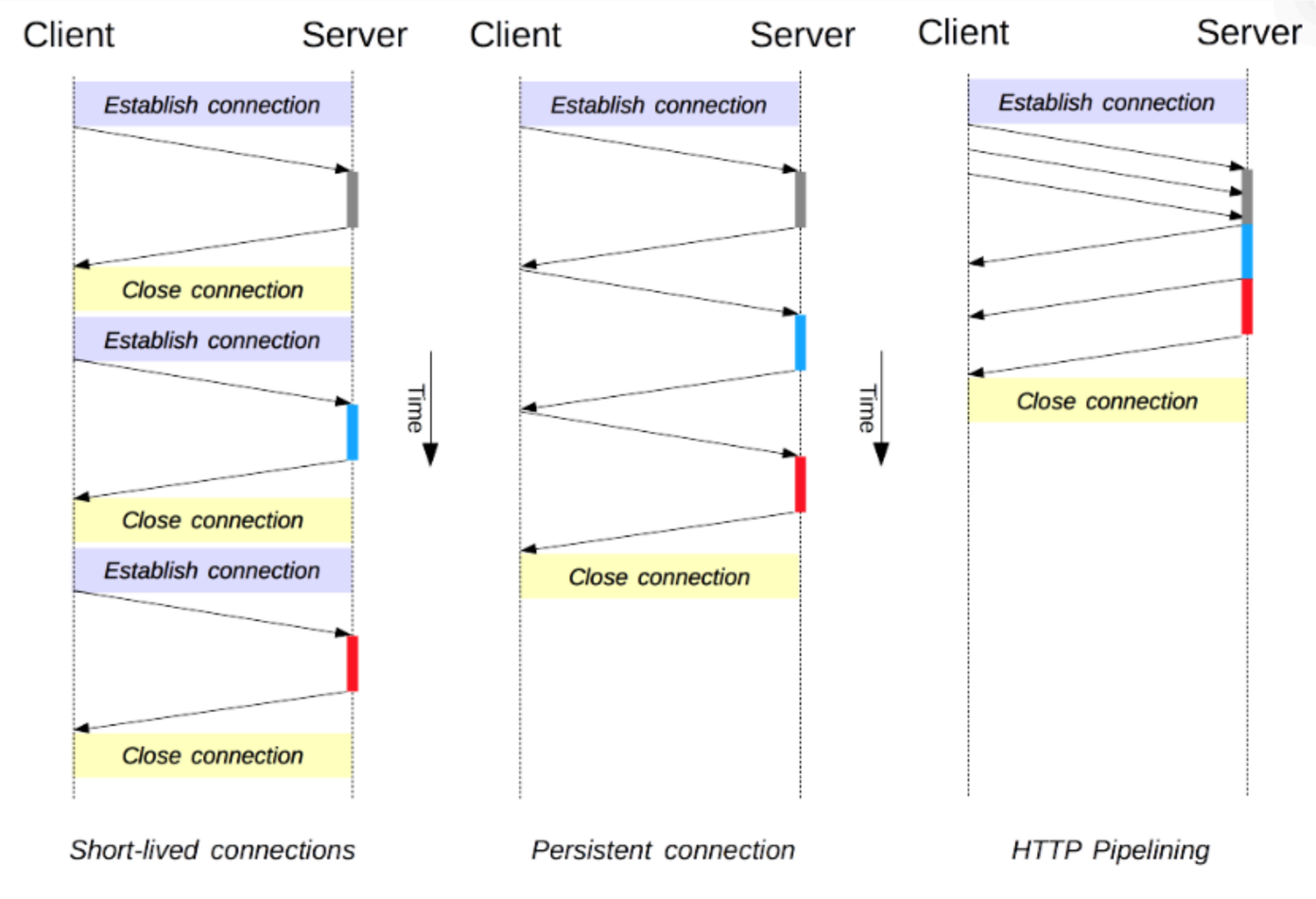
HTTP 1.1 (1997)



Persistent Connection

- 한번 맺은 연결을 계속 재사용
- 서버 부하 감소

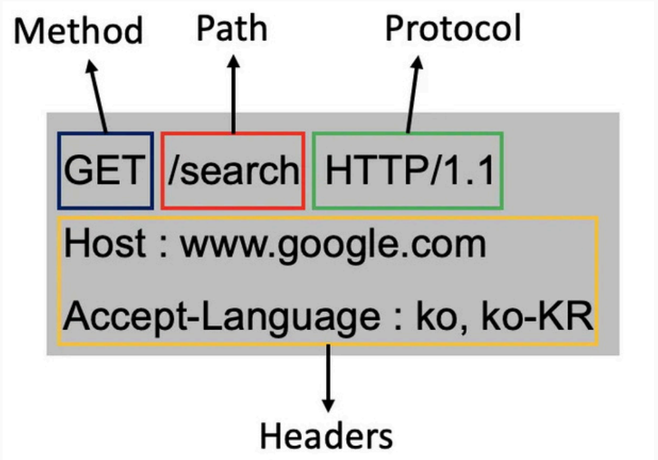
HTTP 1.1 (1997)

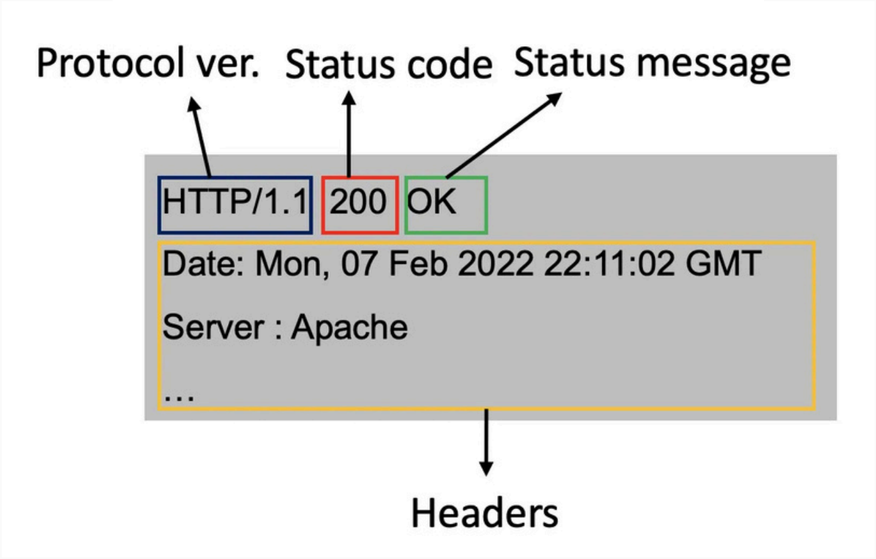


Pipelining

- 이전 요청의 응답을 기다리지 않고 연속해서 여러 요청을 보냄
- 지연시간 감소

HTTP 1.1 (1997)

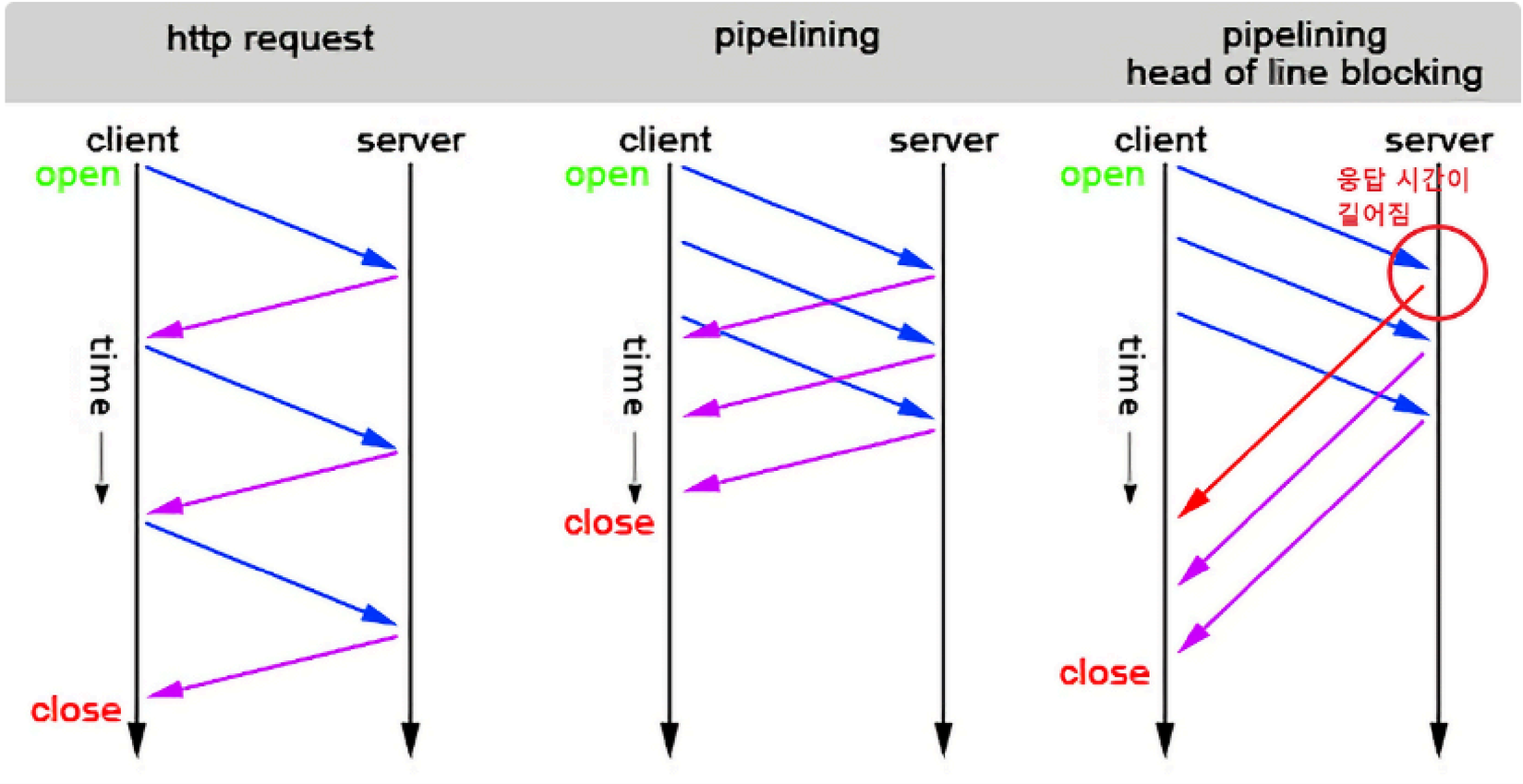




특징

- HOST 헤더 필수
  - 하나의 서버에서 도메인 이름의 웹 사이트 운영
  - 가상 호스팅 서비스
- PUT, PATCH, DELETE, OPTION 메서드 가능
- ‘Cache-Control’ 헤더 도입

HTTP 1.1 (1997)



HOL(Head-of-Line) Blocking

- TCP는 요청과 응답이 순차적으로 처리되는 방식
- 만약 먼저 보낸 요청이 처리되지 않는다면? 병목 현상 발생

HTTP 1.1 (1997)

Request #1

:method	GET
:scheme	https
:host	example.com
:path	/resource
accept	image/jpeg
user-agent	Mozilla/5.0 ...

Request #2

:method	GET
:scheme	https
:host	example.com
:path	/new_resource
accept	image/jpeg
user-agent	Mozilla/5.0 ...

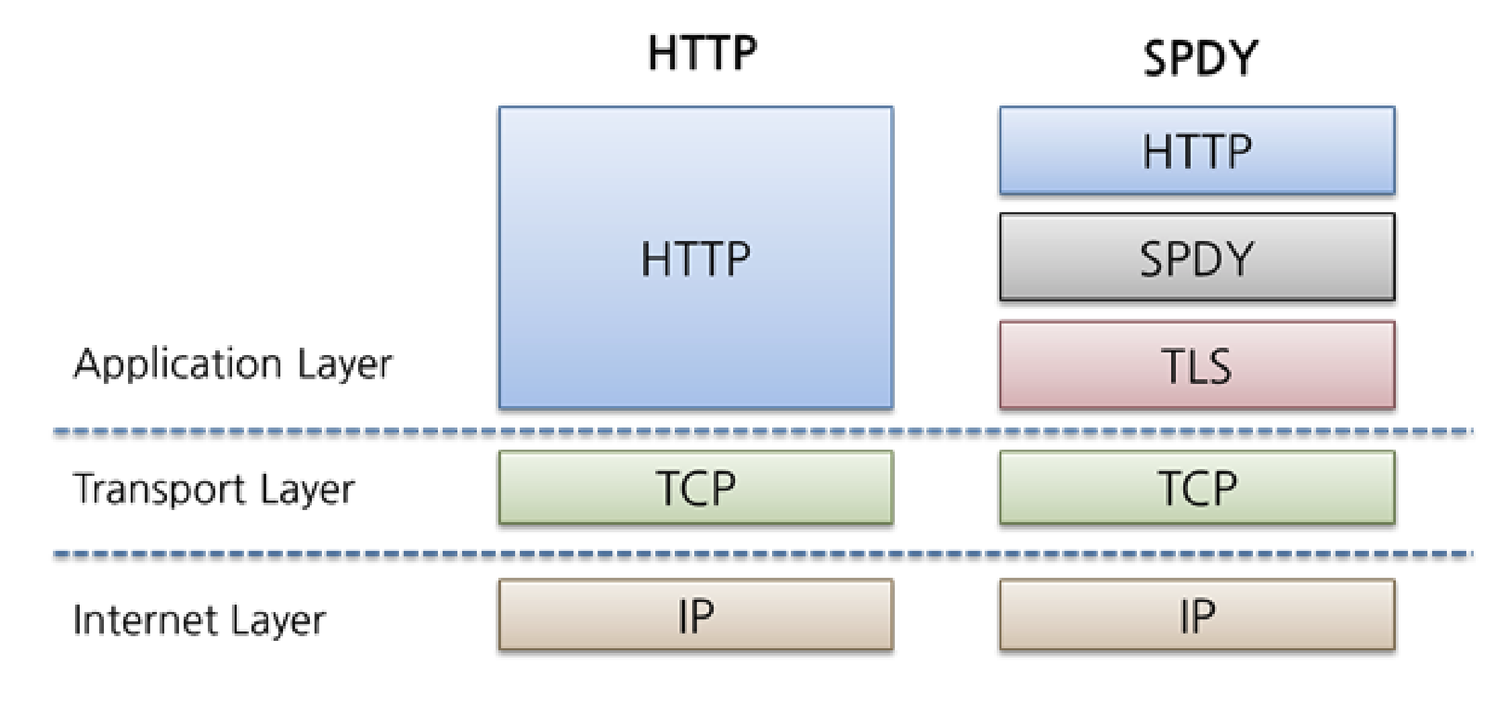
헤더의 중복

- 거의 동일한 내용의 헤더가 그대로 전송
- 전송하는 내용보다 헤더 값이 더 큰 경우도 있음

04

HTTP 2

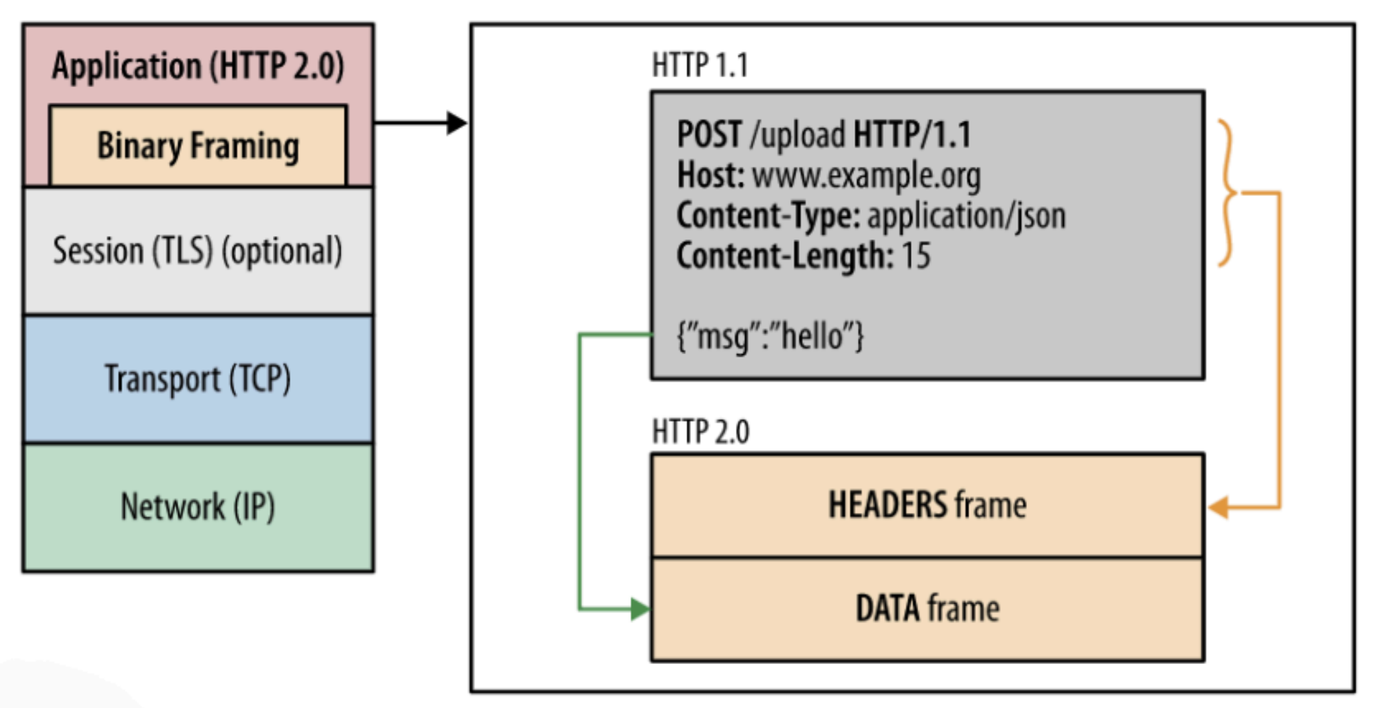
HTTP 2 (2015)



SPDY(2009)

- HTTP/1.1의 속도 문제를 해결하기 위해 등장
- HTTP를 대체하는 것이 아닌 HTTP가 전송 계층을 통해 전송되는 방식을 재정의하는 프로토콜

HTTP 2 (2015)

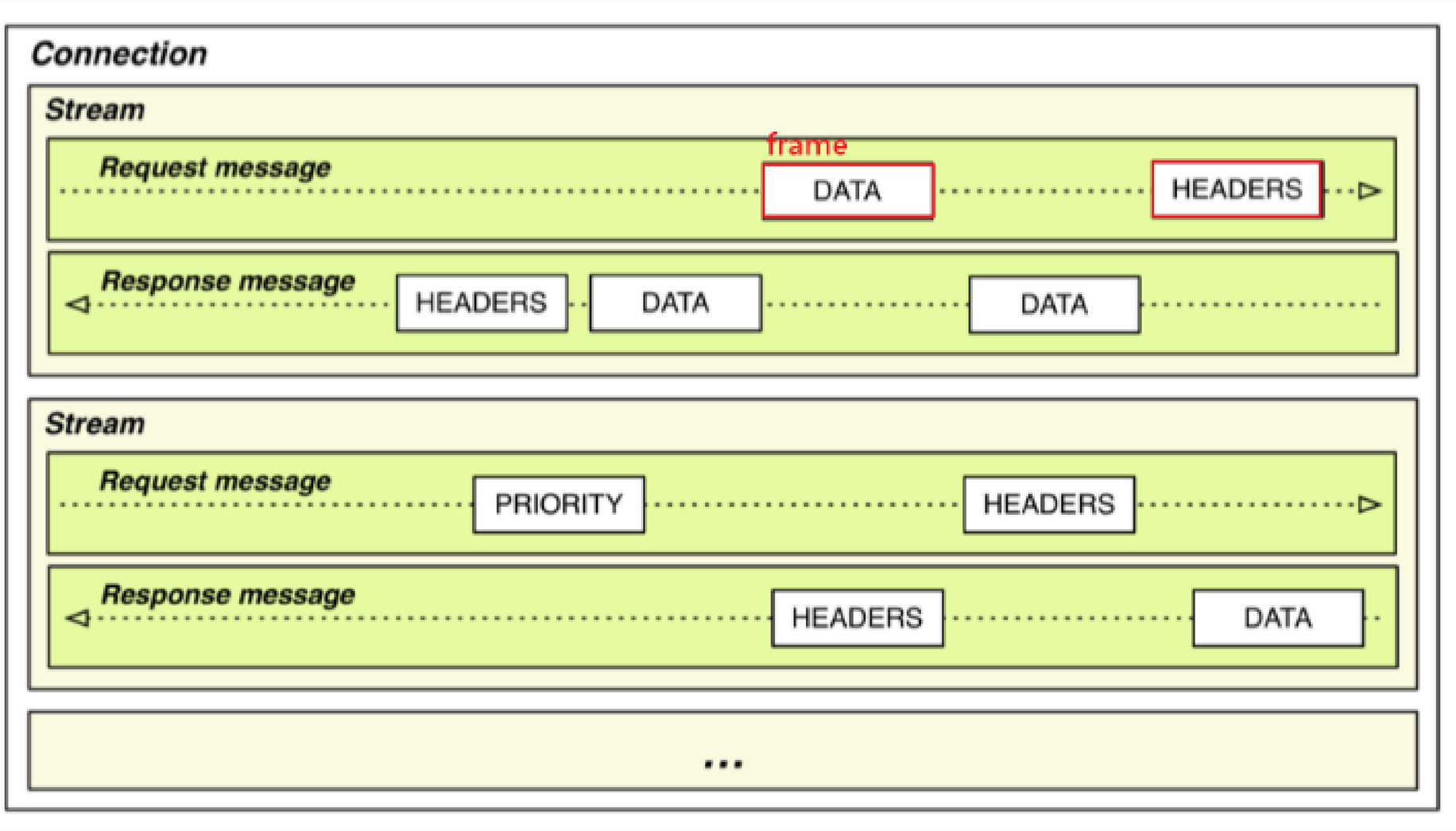


binary framing

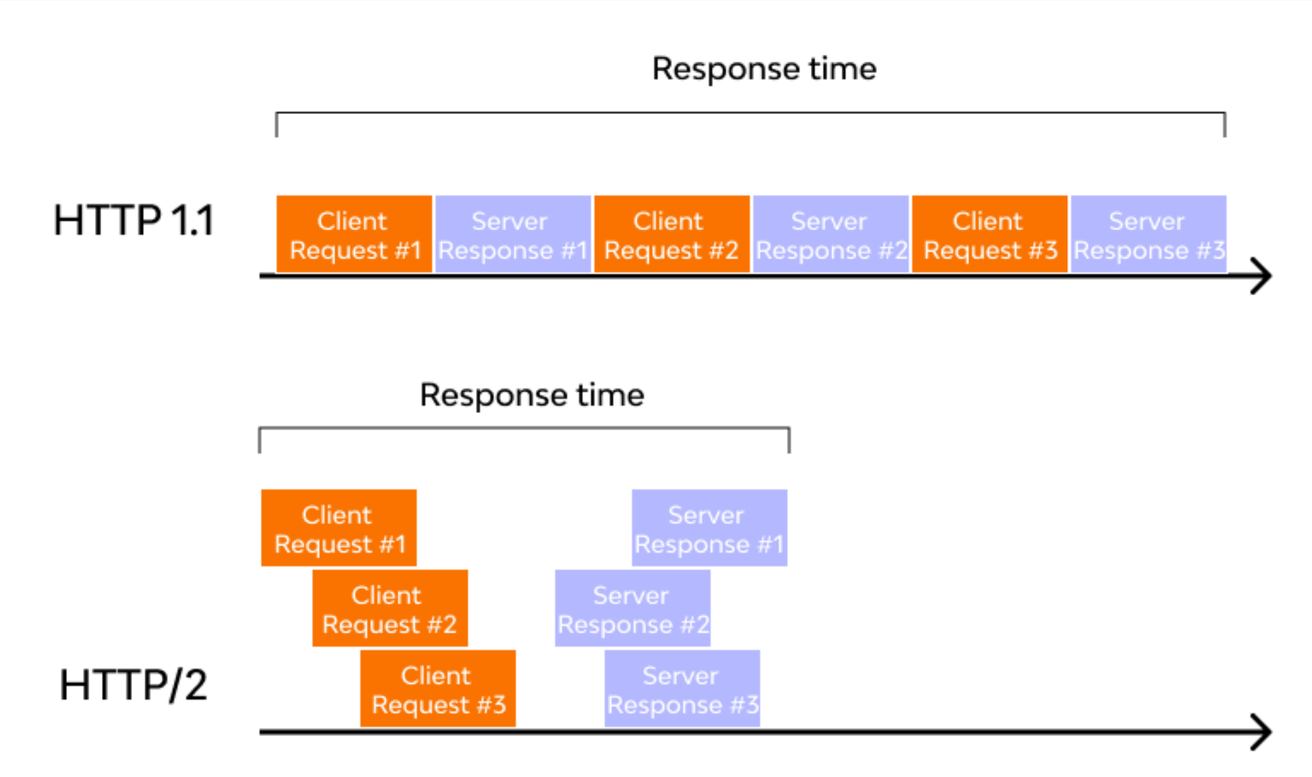
- 텍스트가 아닌 이진 프로토콜 구조
- 모든 메시지를 Frame으로 나누어 전송

- 파싱 속도 향상 및 오류 감소

HTTP 2 (2015)


Connection
Stream
Request message
frame
DATA
HEADERS
Response message
HEADERS
DATA
DATA
Stream
Request message
PRIORITY
HEADERS
Response message
HEADERS
DATA
...

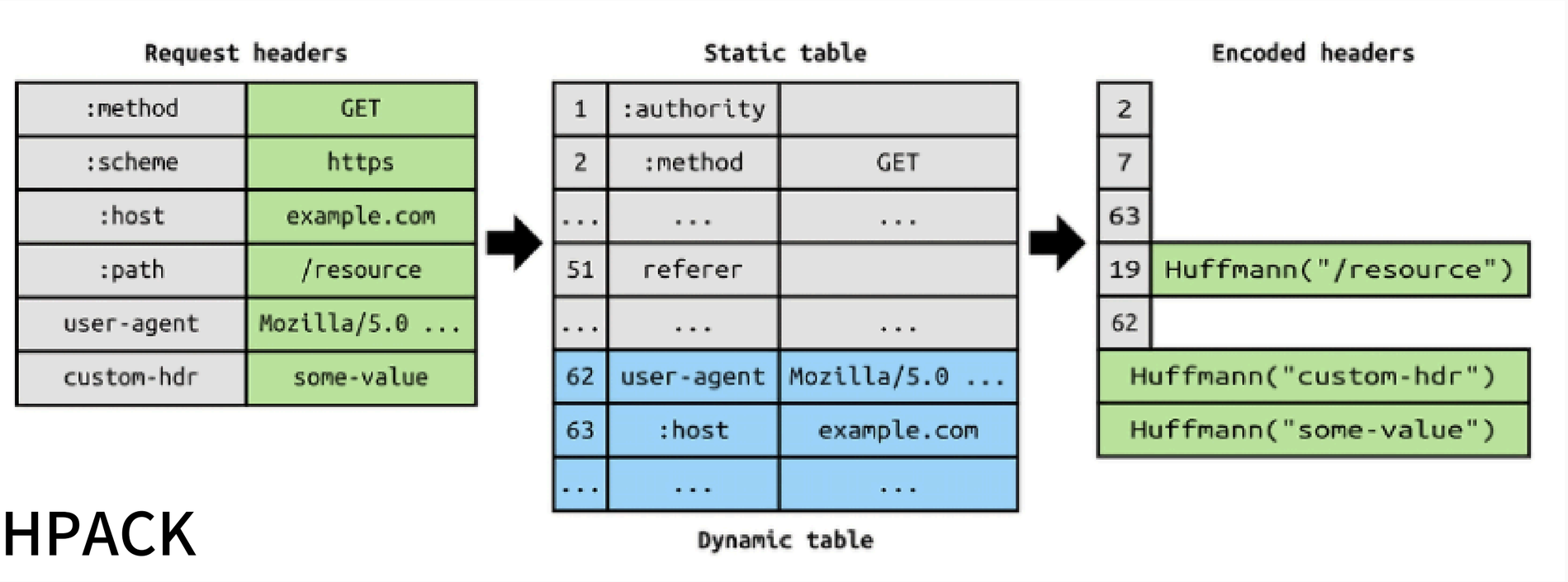
HTTP 2 (2015)



Multiplexing

- 하나의 Connection 안에서 독립적인 Stream을 만들어 요청과 응답
- 특정 응답이 지연된다면?

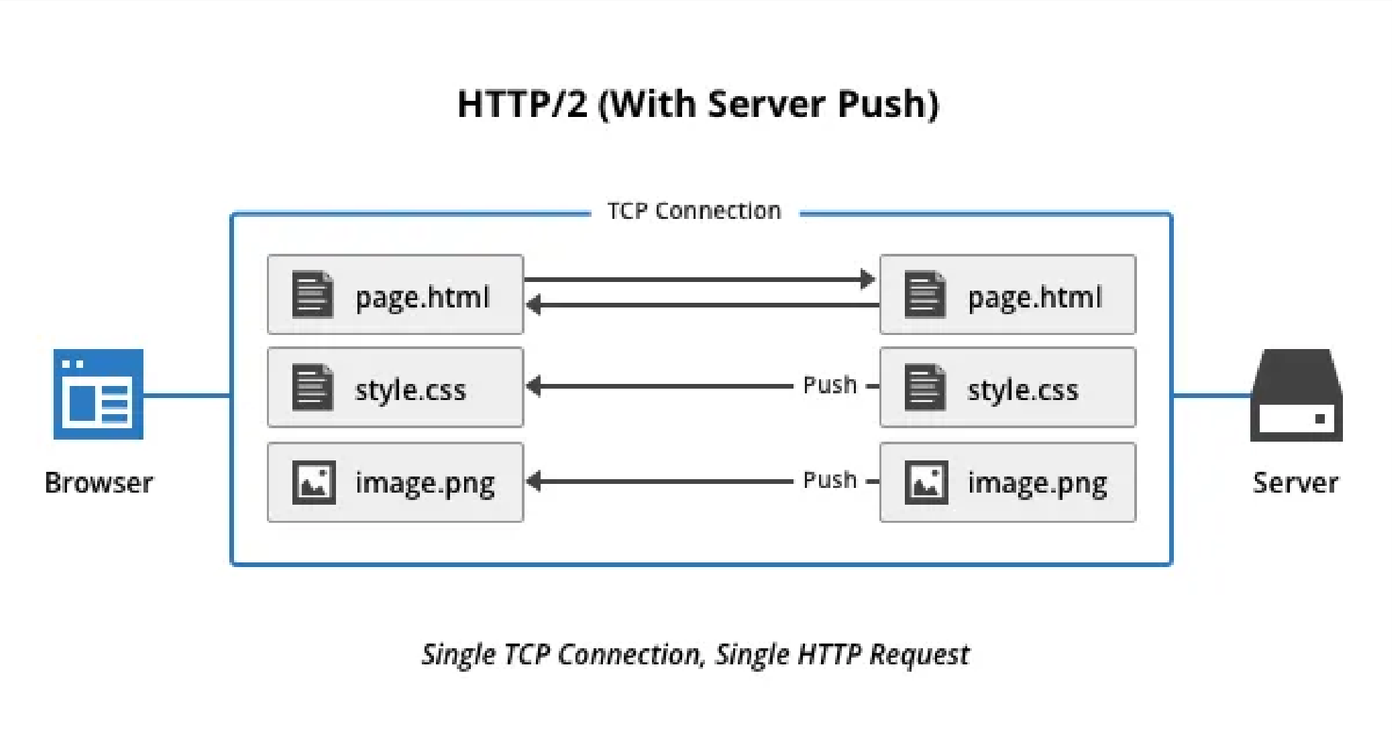
HTTP 2 (2015)



HPACK

- 정적 테이블은 자주 사용되는 값 저장
- 한 번 보낸 헤더는 동적 테이블에 저장
- 인덱스 번호를 대체하여 전송
- 허프만 코딩으로 압축

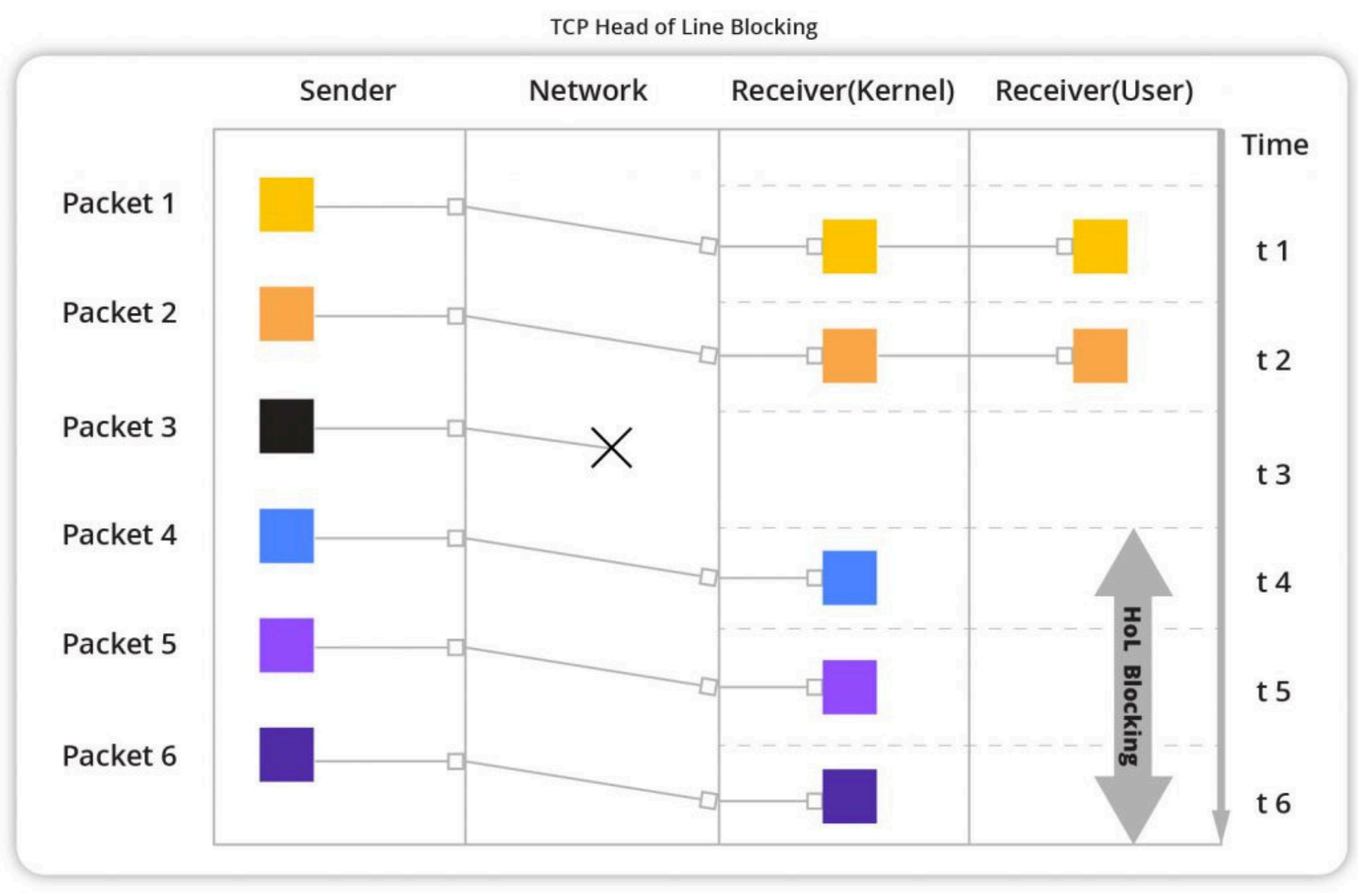
HTTP 2 (2015)



Server Push

- 클라이언트 요청하지 않아도 알아서 보냄

HTTP 2 (2015)



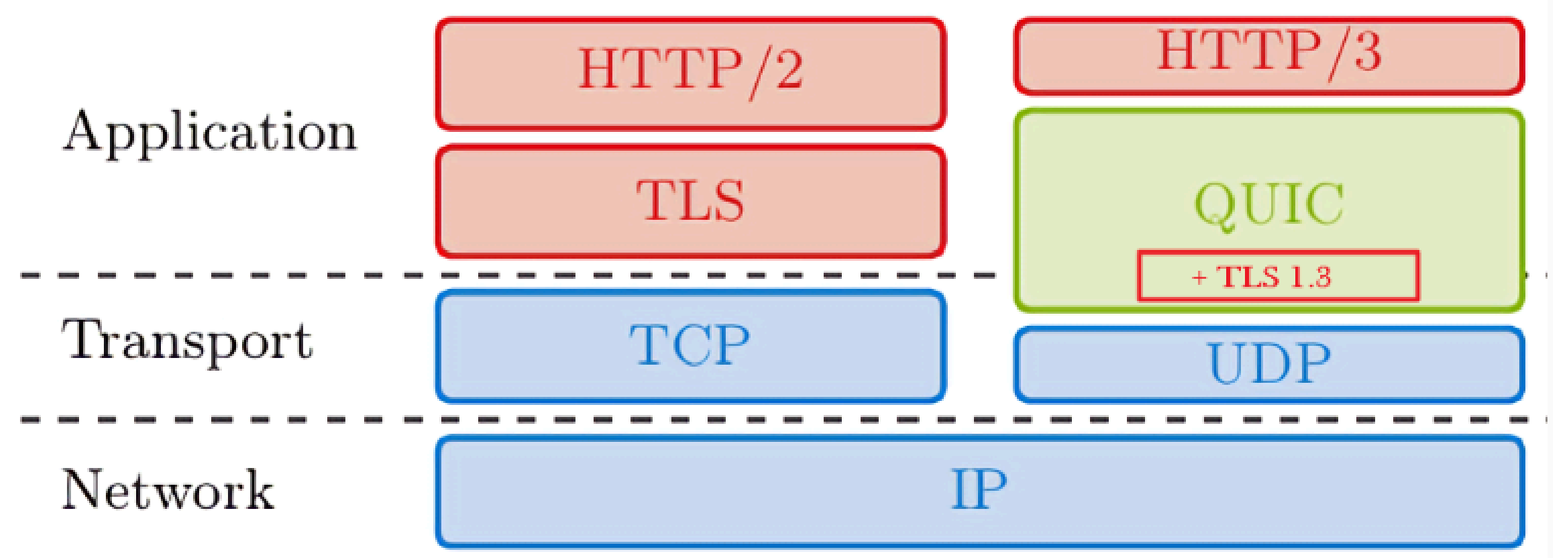
TCP-level HOL Blocking

- 병렬 처리로 동작하지만 결국 하나가 유실되면 모든 스트림이 멈춤
- TCP의 한계점

05

HTTP 3

HTTP 3 (2019)



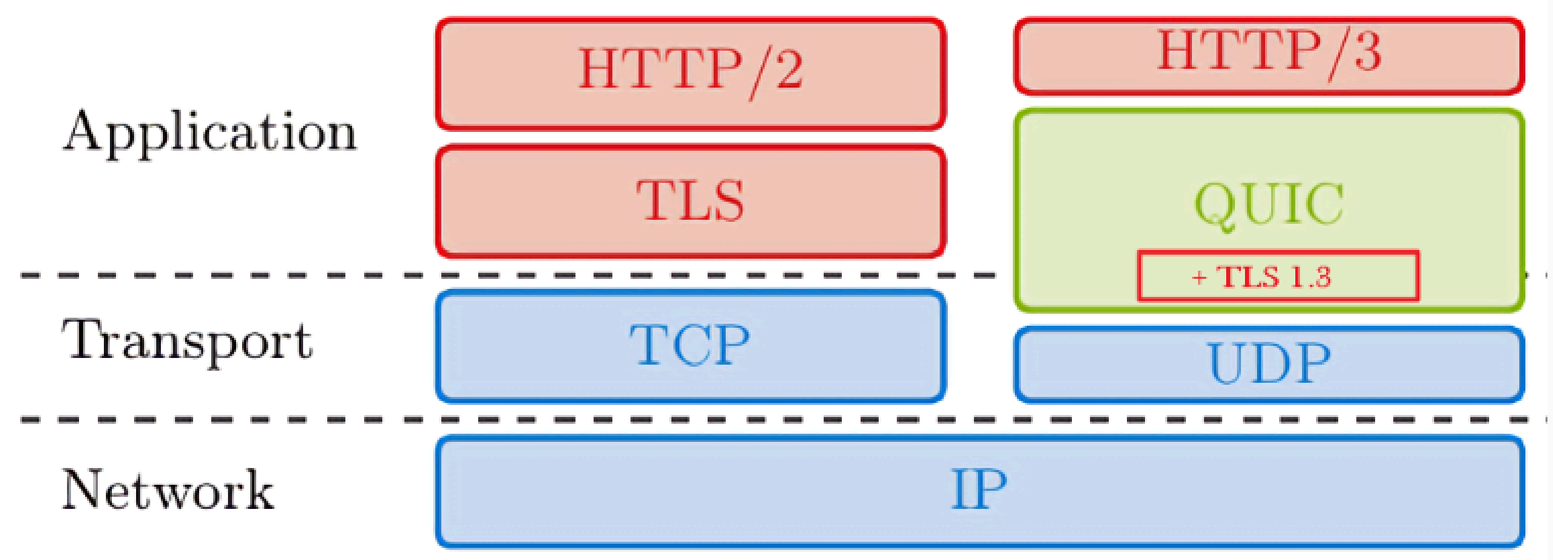
TCP + TLS

- 신뢰성 보장
- 흐름 제어
- 혼잡 제어
- 스트림 내 순서 보장

QUIC(Quick UDP Internet Connections)

- TCP의 HOL Blocking 문제를 해결하기 위해 구글이 개발
- UDP 기반의 전송 계층 프로토콜

HTTP 3 (2019)



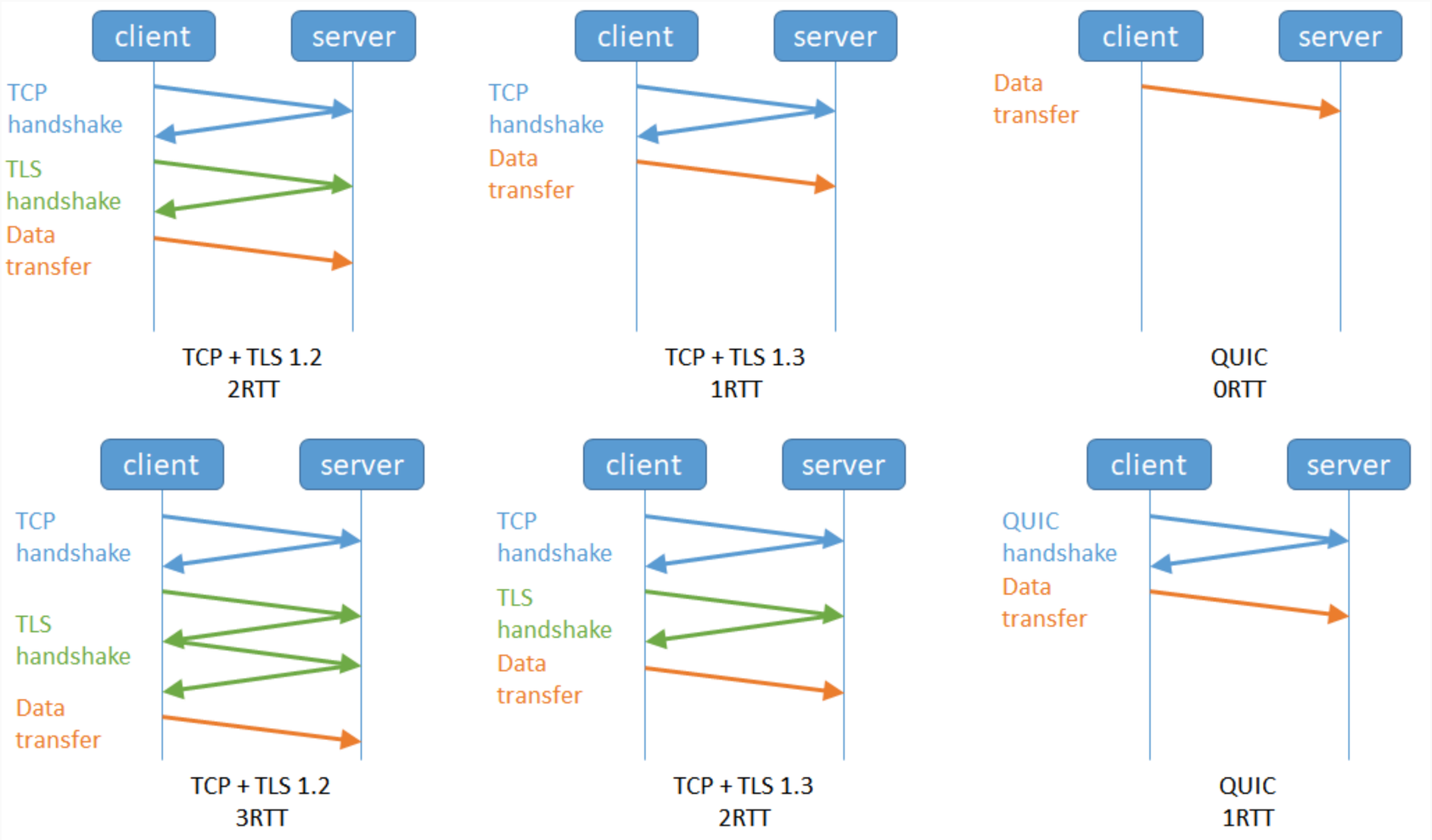
QUIC(Quick UDP Internet Connections)

- TCP의 HOL Blocking 문제를 해결하기 위해 구글이 개발
- UDP 기반의 전송 계층 프로토콜

UDP 사용 이유

- 커널 대신 애플리케이션 레벨의 UDP
- 기존 장비의 새로운 프로토콜 차단
- 연결 속도가 빠름

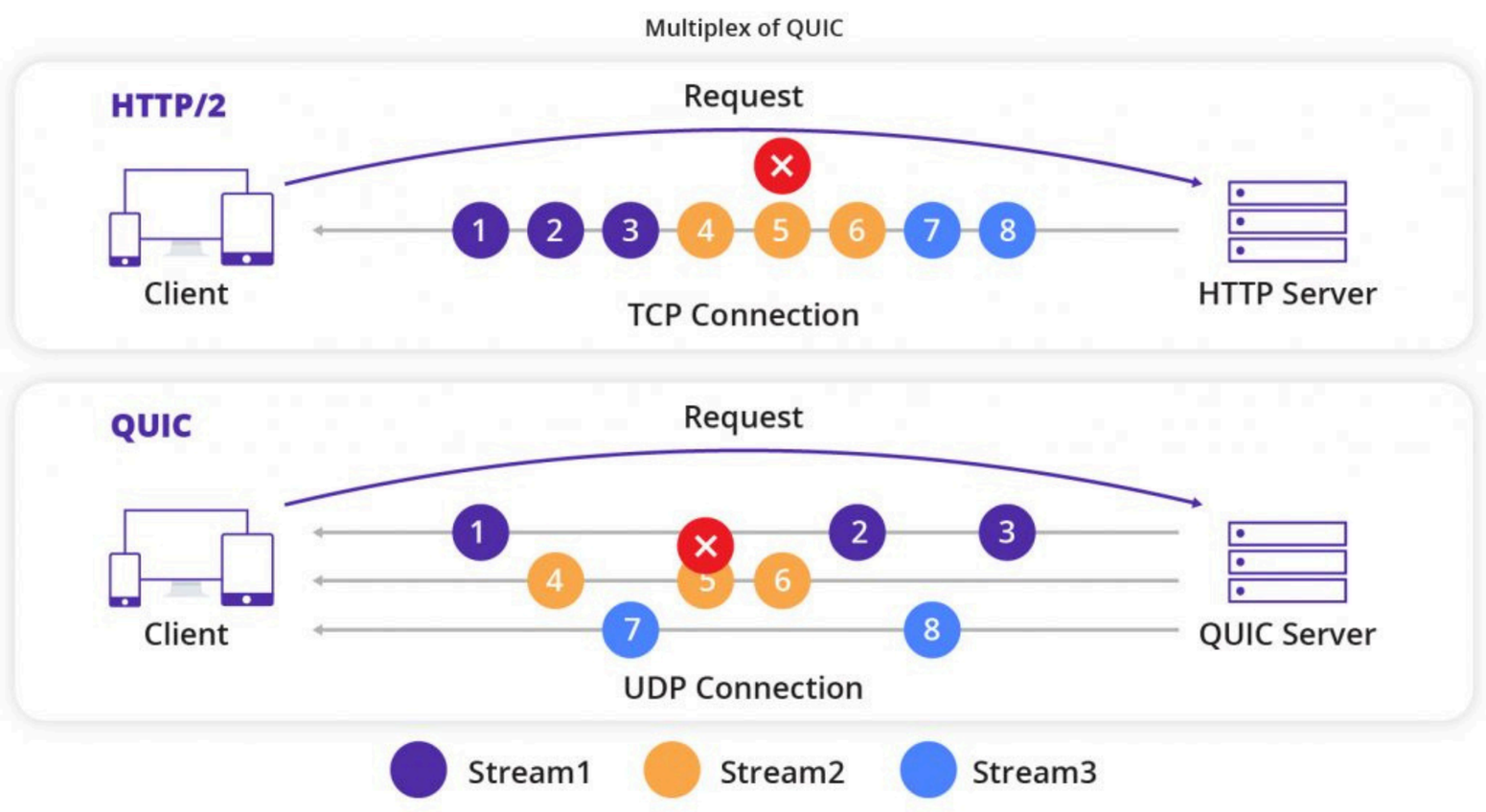
HTTP 3 (2019)



빠른 연결 설정

- 연결에 필요한 RTT가 적음
- QUIC에 TLS 인증서가 포함됨

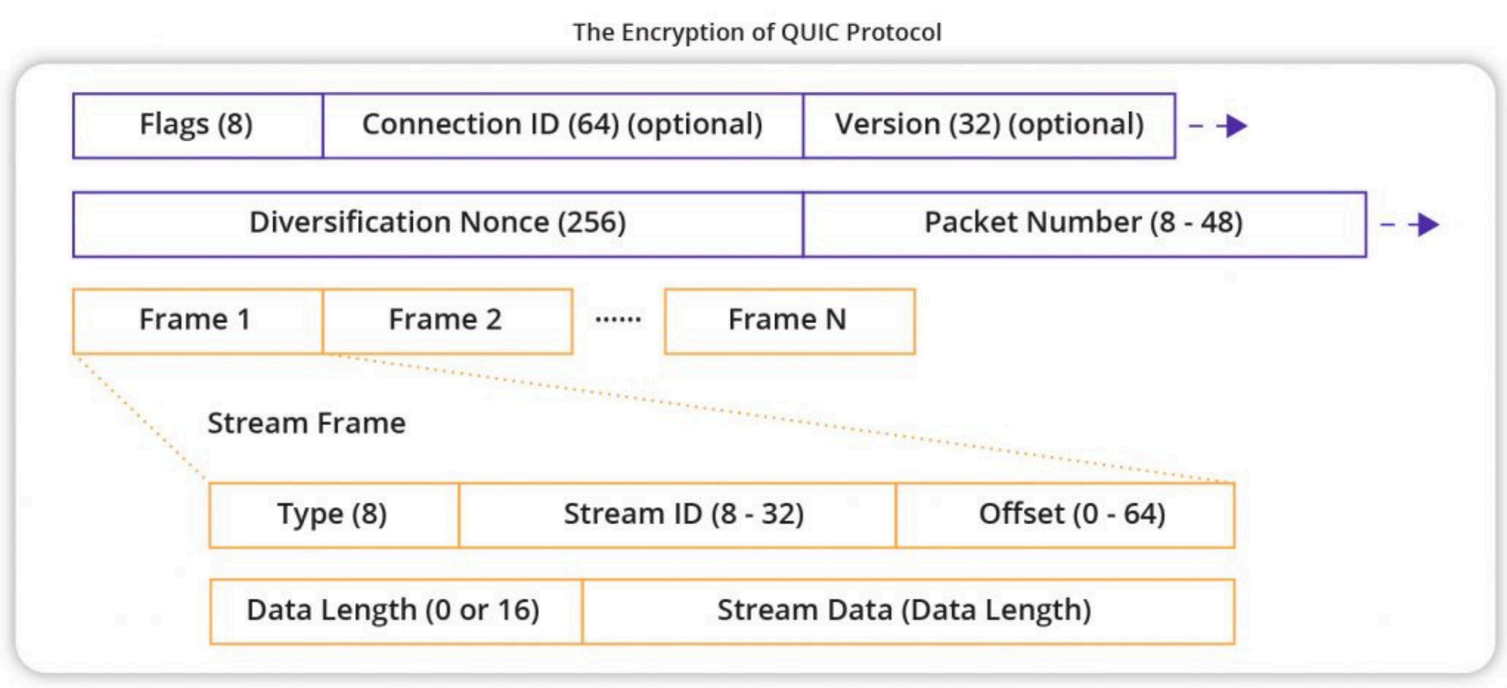
HTTP 3 (2019)



독립적인 스트림

- 한 스트림에서 패킷이 유실되더라도 다른 스트림과는 상관 없음
- 같은 스트림내에서만 순서 보장

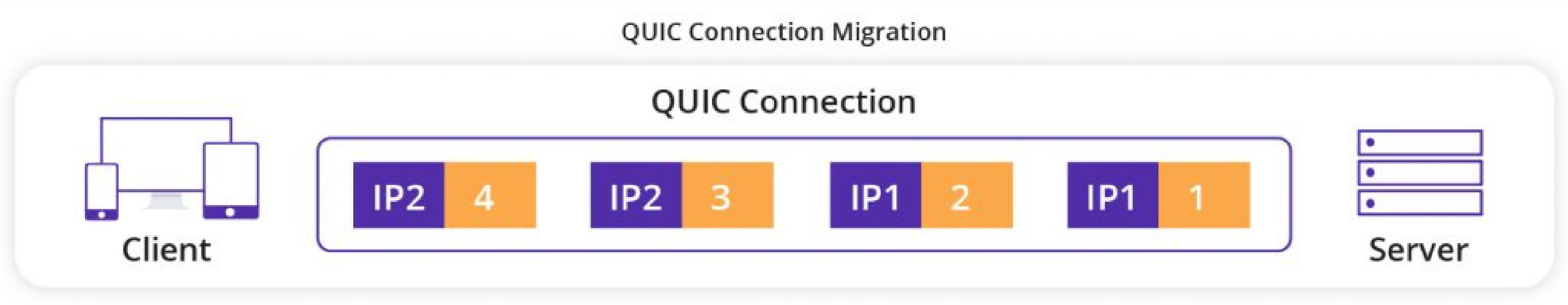
HTTP 3 (2019)



보안 기능

- TLS 1.3 암호화로 암호화된 연결 보장
- 헤더 영역도 암호화됨

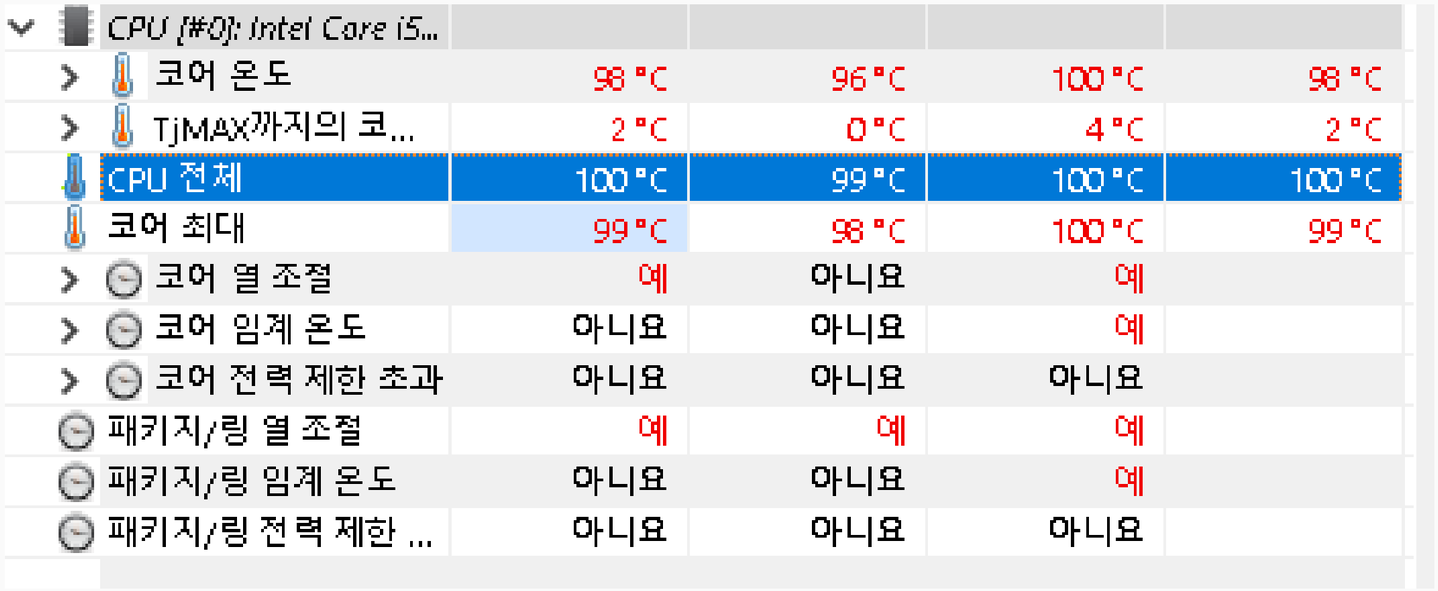
HTTP 3 (2019)



Connection ID

- IP 주소와 포트가 바뀌어도 Connection ID 이용 시 연결 유지 가능
- Connection ID를 이용해 클라이언트와 서버가 서로를 식별하고 동일한 연결을 인지

HTTP 3 (2019)



CPU [#0]: Intel Core i5...				
코어 온도	98 °C	96 °C	100 °C	98 °C
TjMAX까지의 코...	2 °C	0 °C	4 °C	2 °C
CPU 전체	100 °C	99 °C	100 °C	100 °C
코어 최대	99 °C	98 °C	100 °C	99 °C
코어 열 조절	예	아니요	예	
코어 임계 온도	아니요	아니요	예	
코어 전력 제한 초과	아니요	아니요	아니요	
패키지/링 열 조절	예	예	예	
패키지/링 임계 온도	아니요	아니요	예	
패키지/링 전력 제한 ...	아니요	아니요	아니요	

CPU 사용량

- 사용자 공간에서 QUIC를 처리하기에 CPU 사용량이 높음
- 항상 암호화 처리를 하기에 CPU 사용량이 높음