한국정보과학회 정보통신소사이어티 2022 추계 단기강좌

Opensource Softwarized Transport Protocols

컴퓨터네트워크 수업용도로 편집됨

2022. 10. 31

이 성 원 교수

경희대학교 소프트웨어융합학과

경희대학교
KYUNG HEE UNIVERSITY

Big Picture (1/2)

Socket API	Kernel mode
HTTP/1.1	Application layer
ZMQ	Application layer
WebRTC	Application layer
HTTP/2	Application layer
gRPC	Application layer
QUIC & HTTP/3	Application layer

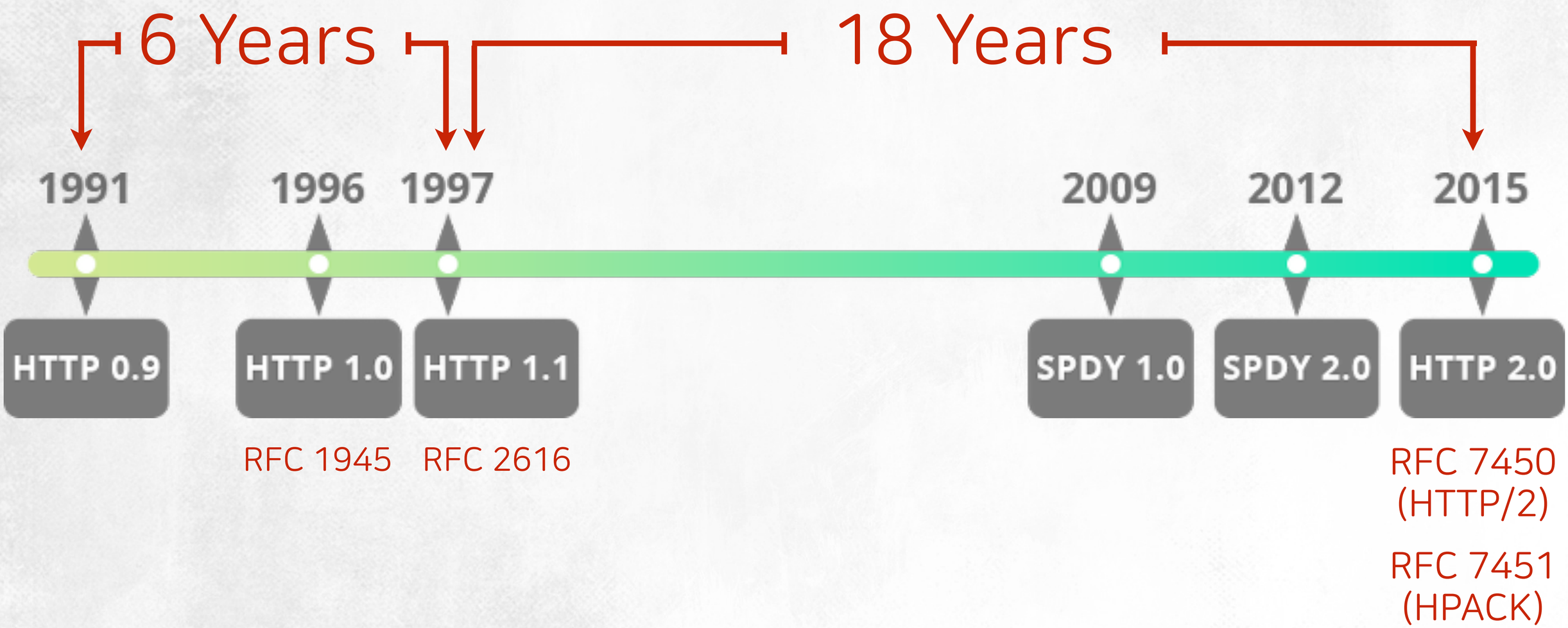
Big Picture (1/2)

Socket API	1983	(4.2BSD Unix OS)
HTTP/1.1	1997	(표준)
ZMQ	2007	(Opensource)
WebRTC	2011	(Opensource)
HTTP/2	**2015**	**(SPDY 〉〉 표준)**
gRPC	2016	(Opensource)
QUIC & HTTP/3	**2021**	**(QUIC 〉〉 표준)**

Paradigm Shift

- Out of Kernel
  - Move to application layer
  - Flexibility and Programmability
- Power-shift from Standard to Opensource
  - Opensource driven
  - Experimental >> Opensource >> Standard
- Out of TCP
  - UDP based new transport protocol
  - Creativity and R&D capability

HTTP history



Reference: https://datatracker.ietf.org/doc/html/rfc7540

Google SPDY (Proprietary Solution)

- SDPY 연혁
  - 2009년 구글의 내부 프로젝트로 공개
  - 2010년 크롬 브라우저의 오픈소스 엔진에 탑재
  - 2011년 구글의 모든 서비스에 도입
  - 2012년 아파치 서버용 구글 SPDY 패치 제공 & Nginx의 지원 공개
  - 2012년 IETF 단체를 통한 표준화 추진
  - 2015년 HTTP/2가 RFC 7540 표준화 되면서, SPDY 중단

Reference: https://en.wikipedia.org/wiki/SPDY

IETF RFC 7540[HTTP/2]/7541[HPACK] Standard

From: draft-ietf-httpbis-http2-17 — Proposed Standard
Updated by: 8740 — IPR declarations
Errata exist

Internet Engineering Task Force (IETF)
Request for Comments: 7540
Category: Standards Track
ISSN: 2070-1721

M. Belshe
BitGo
R. Peon
Google, Inc
M. Thomson, Ed.
Mozilla
May 2015

Hypertext Transfer Protocol Version 2 (HTTP/2)

Abstract

This specification describes an optimized expression of the semantics of the Hypertext Transfer Protocol (HTTP), referred to as HTTP version 2 (HTTP/2). HTTP/2 enables a more efficient use of network resources and a reduced perception of latency by introducing header field compression and allowing multiple concurrent exchanges on the same connection. It also introduces unsolicited push of representations from servers to clients.

This specification is an alternative to, but does not obsolete, the HTTP/1.1 message syntax. HTTP's existing semantics remain unchanged.

주요 특징

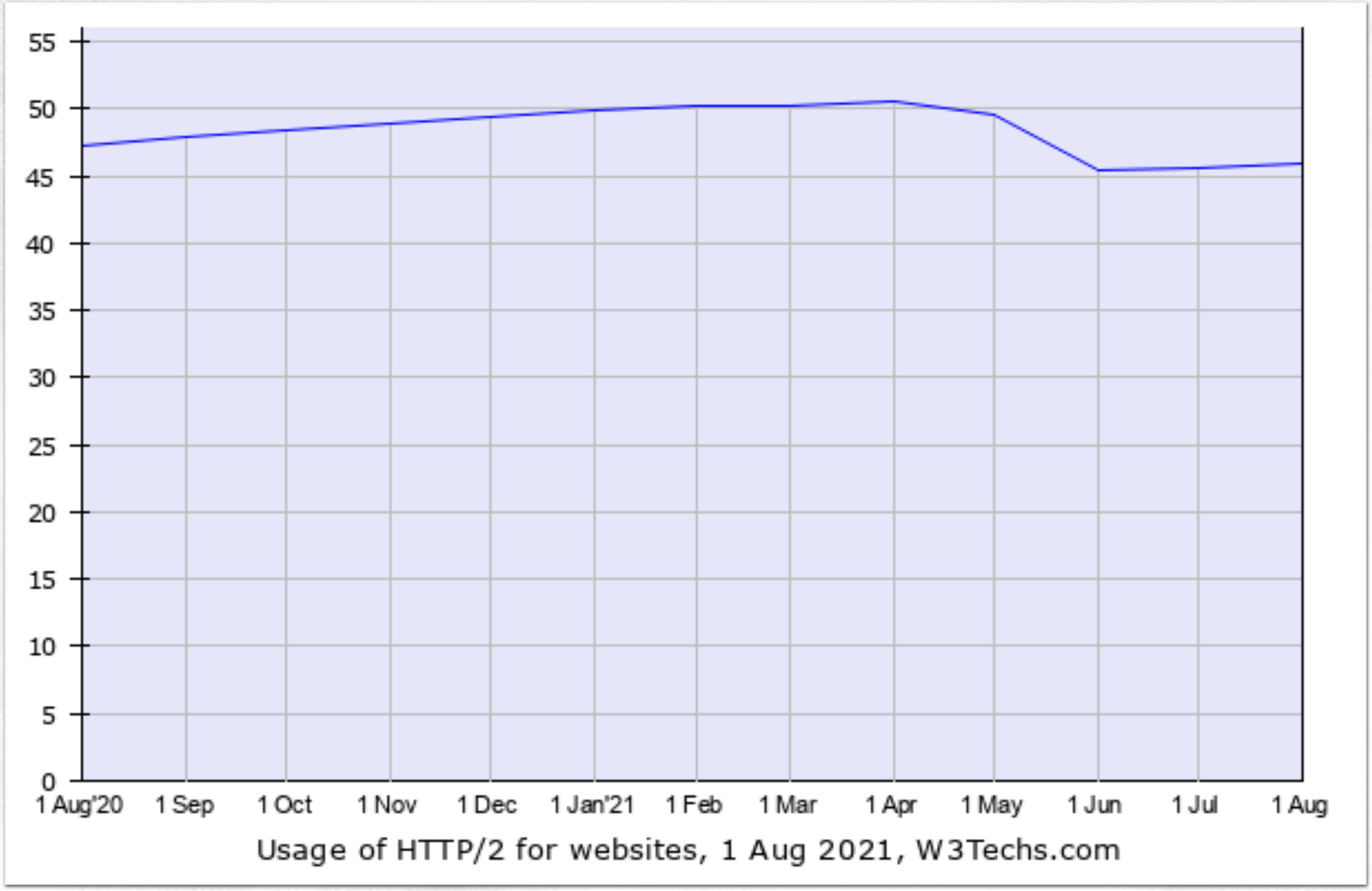
Skip [Lecture.10 참조]

- 줄어든 지연 시간
- 응답 다중화 지원
- HTTP 헤더 필드 압축 통한 프로토콜 오버헤드 최소화
- 요청 별 우선순위 지정을 추가함
- 서버 푸시
- 기존 어플리케이션의 수정 없는 지원
  - HTTP 메서드, 상태 코드, URI 및 헤더 필드 등 지원

Reference: https://developers.google.com/web/fundamentals/performance/http2?hl=ko

HTTP 사용 현황

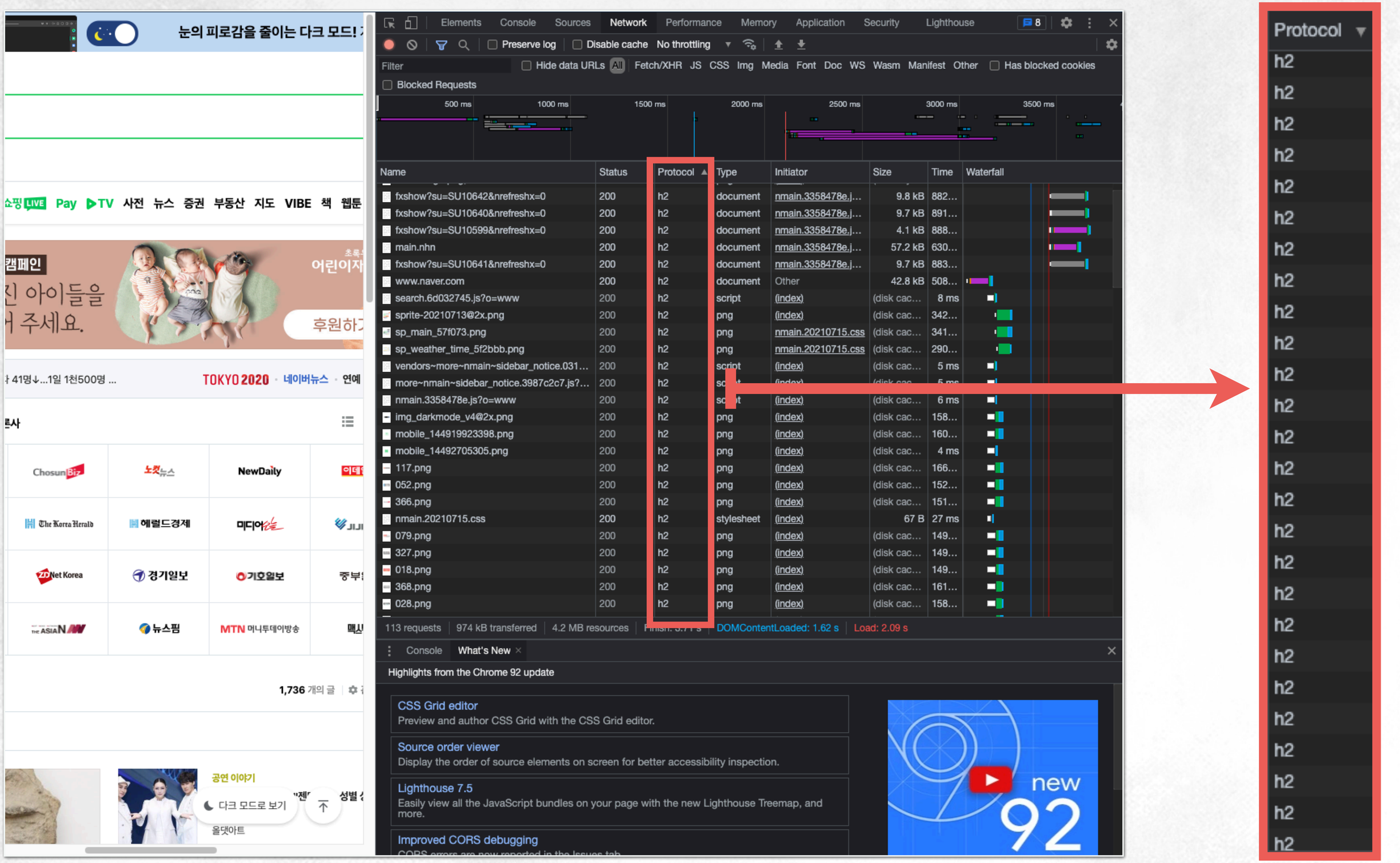
- 전 세계 웹사이트의 45.9%가 HTTP/2를 사용 중



Reference: https://w3techs.com/technologies/details/ce-http2

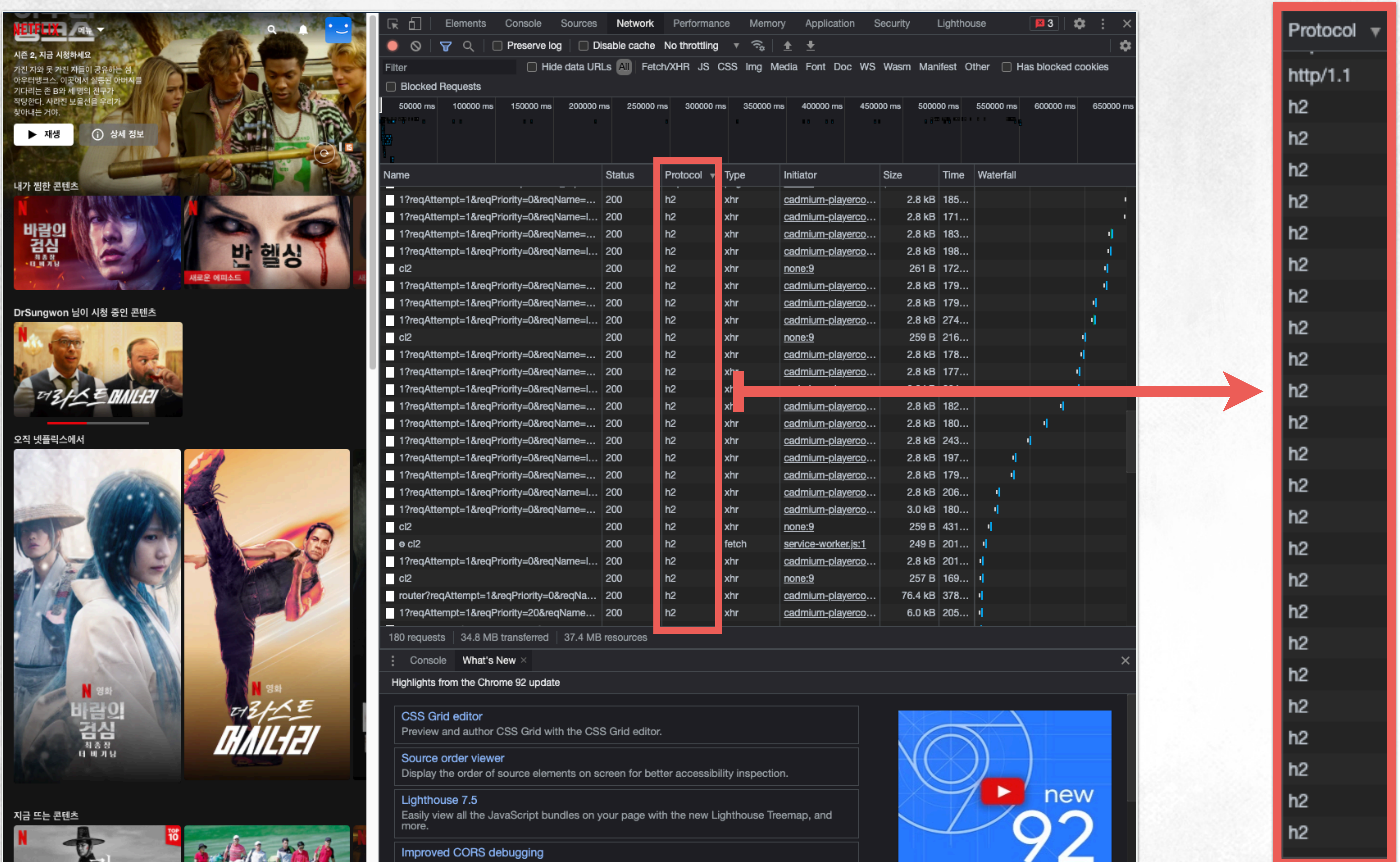
HTTP/2 이해

Naver.com



HTTP/2 이해

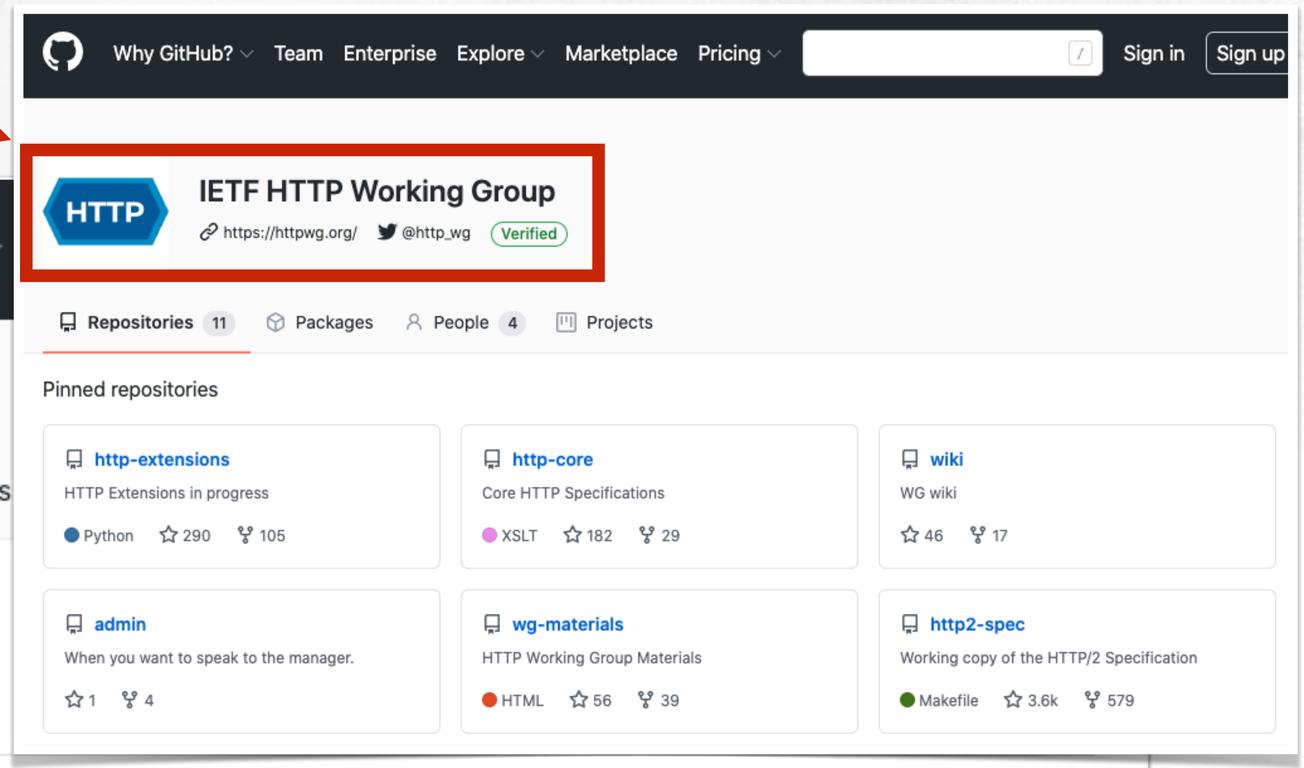
Netflix.com



HTTP/2 개발 환경

Programming Language

- IETF HTTP WG의 사이트에서 언어별 라이브러리를 확인 가능



HTTP/2 개발 환경

Programming Language

- IETF HTTP WG의 사이트에서 HTTP/2 개발 도구도 확인 가능

Tools

Otto van der Schaaf edited this page on 29 Oct 2019 · 24 revisions

This is a listing of tools for analysing, debugging and visualising HTTP/2. See also the Implementations listing.

- Curl supports HTTP/2[1] as of 7.43.0. See its documentation for details (including prerequisites).
- h2i is a command-line interactive client that lets you send H2 frames, translate H1 to H2, and generally figure out how the protocol works.
- h2load is a benchmarking / load generation tool for HTTP/2 and SPDY.
- Nighthawk a L7 (HTTP/HTTPS/HTTP2) performance characterization tool
- scapy allows to sniff, dissect, craft, send HTTP 2 packets.
- nghttp is a non-interactive command line HTTP/2 client that has plenty of debugging options, such as changing flow control window, dumping frames, HTTP Upgrade etc.
- nghttpd is a simple static file HTTP/2 server that is very handy to debug client side implementations.
- mitmproxy is an interactive console program that allows traffic flows to be intercepted, inspected, modified and replayed. Can be used programmatically (Python).
- is-http2 lets you quickly find out if a host supports H2 from the command line.
- jmeter Jmeter HTTP/2 sampler via - Netty 5 and netty-tcnative & hpack
- jmeter JMeter HTTP/2 Request sampler and View Result Tree Http2 listener plugin installable with JMeter Plugins Manager
- HTTP/2 Test is an online tool to check if a website supports HTTP/2.
- HTTP2.Pro is an online tool to check HTTP/2, ALPN, and Push support of web sites.
- Wireshark has a HTTP/2 decoder[2].
- h2c - A Simple HTTP/2 Command-Line Client
- h2spec - Conformance testing tool for HTTP/2 implementations
- http2fuzz is a semi-intelligent fuzzer for HTTP/2.
- WProf extracts dependencies of activities during a page load, to identify bottlenecks.
- LoadRunner emulates HTTP/2 browsers (but not server push) to generate artificial load on servers as an enterprise-grade framework with GUI analytics.
- Vegeta OS load tool that supports h2 through Go's (>= 1.6) net/http API
- HTTP/1 vs HTTP/2 speed test an online tool to test and compare speed of HTTP/2-ready websites, via HTTP/2 versus via HTTP/1 (disabling h2 support in Chrome)

Pages 17

Find a Page...

Home
Advice
Akamaighost
ALPN Status
ContinuationProposals
DosVectors
Encryption Terminology
EricssonMSP
F5
Fredschromium
gfe
Implementations
Microsoft HTTP 2 Prototype
Ops
Proxy User Stories
Show 2 more pages...

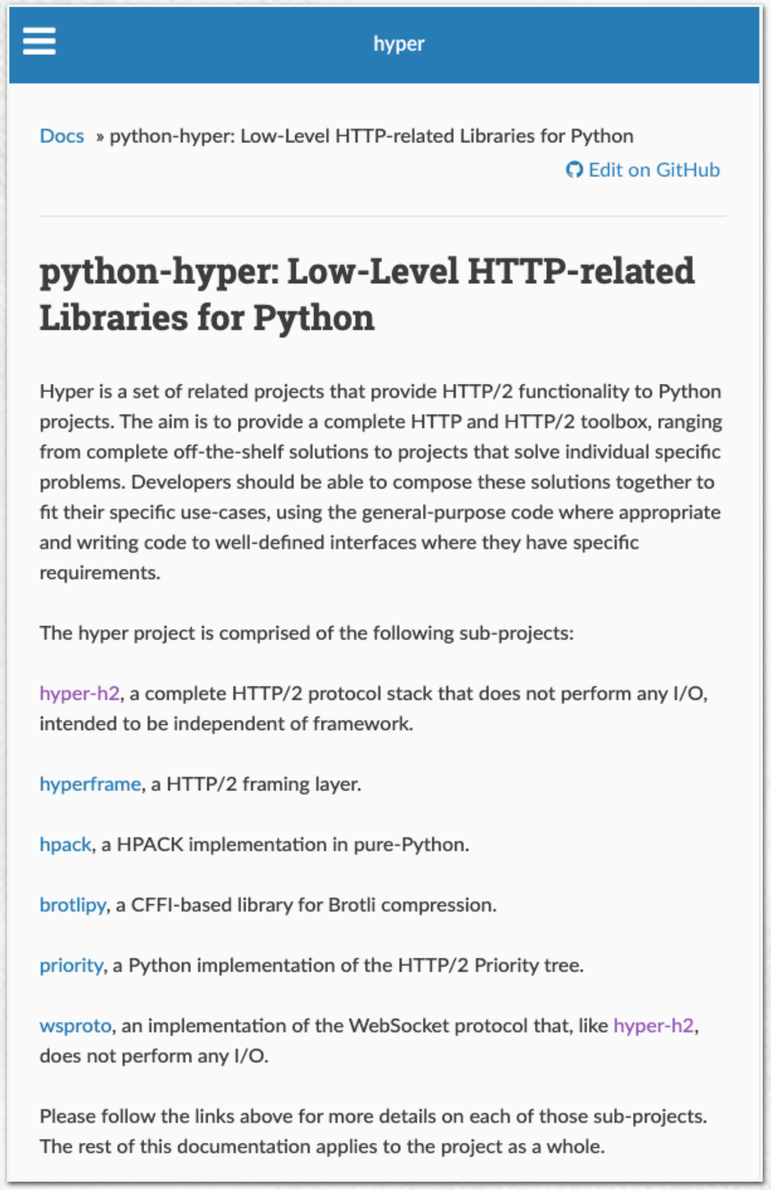
Clone this wiki locally

https://github.com/httpwg/h

HTTP/2 개발 환경

Programming Language

- Python : 표준 라이브러리 없음. 외부 라이브러리 활용 가능
  - Hyper : 대표적인 client / server 용 라이브러리



hyper

Docs » python-hyper: Low-Level HTTP-related Libraries for Python

Edit on GitHub

python-hyper: Low-Level HTTP-related Libraries for Python

Hyper is a set of related projects that provide HTTP/2 functionality to Python projects. The aim is to provide a complete HTTP and HTTP/2 toolbox, ranging from complete off-the-shelf solutions to projects that solve individual specific problems. Developers should be able to compose these solutions together to fit their specific use-cases, using the general-purpose code where appropriate and writing code to well-defined interfaces where they have specific requirements.

The hyper project is comprised of the following sub-projects:

hyper-h2, a complete HTTP/2 protocol stack that does not perform any I/O, intended to be independent of framework.

hyperframe, a HTTP/2 framing layer.

hpack, a HPACK implementation in pure-Python.

brotlipy, a CFFI-based library for Brotli compression.

priority, a Python implementation of the HTTP/2 Priority tree.

wsproto, an implementation of the WebSocket protocol that, like hyper-h2, does not perform any I/O.

Please follow the links above for more details on each of those sub-projects. The rest of this documentation applies to the project as a whole.

hyper

Docs » Hyper: HTTP/2 Client for Python

Edit on GitHub

Hyper: HTTP/2 Client for Python

Release v0.7.0.

HTTP is changing under our feet. HTTP/1.1, our old friend, is being supplemented by the brand new HTTP/2 standard. HTTP/2 provides many benefits: improved speed, lower bandwidth usage, better connection management, and more.

`hyper` provides these benefits to your Python code. How? Like this:

```
from hyper import HTTPConnection

conn = HTTPConnection('http2bin.org:443')
conn.request('GET', '/get')
resp = conn.get_response()

print(resp.read())
```

Simple. `hyper` is written in 100% pure Python, which means no C extensions. For recent versions of Python (3.4 and onward, and 2.7.9 and onward) it's entirely self-contained with no external dependencies.

`hyper` supports Python 3.4 and Python 2.7.9, and can speak HTTP/2 and HTTP/1.1.

Caveat Emptor!

Please be warned: `hyper` is in a very early alpha. You *will* encounter bugs when using it. In addition, there are very many rough edges. With that said, please try it out in your applications: I need your feedback to fix the bugs and file down the rough edges.

hyper-h2

Release 4.1.0+dev

https://buildmedia.readthedocs.org/media/pdf/hyper-h2/latest/hyper-h2.pdf

Feb 23, 2021

Programming Language

- Go : 2016년 부터 Standard 라이브러리로 제공

Golang

Go 1.6 is released

Andrew Gerrand
17 February 2016

Today we release Go version 1.6, the seventh major stable release of Go. You can grab it right now from the download page. Although the release of Go 1.5 six months ago contained dramatic implementation changes, this release is more incremental.

The most significant change is support for HTTP/2 in the net/http package. HTTP/2 is a new protocol, a follow-on to HTTP that has already seen widespread adoption by browser vendors and major websites. In Go 1.6, support for HTTP/2 is enabled by default for both servers and clients when using HTTPS, bringing the benefits of the new protocol to a wide range of Go projects, such as the popular Caddy web server.

HTTP/2

Go 1.6 adds transparent support in the `net/http` package for the new HTTP/2 protocol. Go clients and servers will automatically use HTTP/2 as appropriate when using HTTPS. There is no exported API specific to details of the HTTP/2 protocol handling, just as there is no exported API specific to HTTP/1.1.

Programs that must disable HTTP/2 can do so by setting `Transport.TLSNextProto` (for clients) or `Server.TLSNextProto` (for servers) to a non-nil, empty map.

Programs that must adjust HTTP/2 protocol-specific details can import and use `golang.org/x/net/http2`, in particular its ConfigureServer and ConfigureTransport functions.

Reference: https://blog.golang.org/go1.6 & https://golang.org/doc/go1.6

QUIC?

Skip [Lecture.10 참조]

- QUIC 연혁
  - QUIC은 TCP를 대체하는 범용 목적의 전송 계층 통신 프로토콜로서, 구글의 짐 로스킨드가 처음 설계하였고, 2012년 구현 및 적용되었으며, 2013년 실험 확대로서 공개 발표됨
  - 국제 인터넷 표준화 기구에 기술되었고, 2021년 5월 IETF RFC9000(+RFC9001/9002/8989)으로 정식 표준화 됨
  - 이미 구글 크롬에서부터 구글 서버에 이르는 모든 연결의 절반 이상에 사용되고 있으며, 크롬/MS 엣지/모질라 파이어폭스/사파리에서 지원함
- QUIC 이름
  - 당초 구글은 "Quick UDP Internet Connections”로 제안했으나, IETF에서 ‘QUIC’을 고유 명사로 지정함

Reference: https://ko.wikipedia.org/wiki/QUIC

QUIC - The Transport Protocol

QUIC 관련 표준 및 개발 정보 사이트

QUIC is an IETF Working Group that is chartered to deliver the next transport protocol for the Internet.

See our contribution guidelines if you want to work with us.

Upcoming Meetings

IETF 111, July 26-30, 2021

Our Documents

Our current documents cover different aspects of the 'core' QUIC protocol, define a mapping of HTTP semantics to it, and cover selected other topics.

- **RFCs**
  - **RFC 8999 - Version-Independent Properties of QUIC** - HTML / TXT / PDF
  - **RFC 9000 - QUIC: A UDP-Based Multiplexed and Secure Transport** - HTML / TXT / PDF
  - **RFC 9001 - Using TLS to Secure QUIC** - HTML / TXT / PDF
  - **RFC 9002 - QUIC Loss Detection and Congestion Control** - HTML / TXT / PDF
- **In-progress documents**
  - **HTTP/3** - Editors' Draft / WG Draft / Repo / Open Issues
  - **QPACK** - Editors' Draft / WG Draft / Repo / Open Issues
  - **Load Balancers** - Editors' Draft / WG Draft / Repo / Open Issues
  - **Operations** -
    - **Applicability** Editors' Draft / WG Draft / Repo / Open Issues
    - **Manageability** Editors' Draft / WG Draft / Repo / Open Issues
  - **Extensions** -
    - **Datagram** Editors' Draft / WG Draft / Repo / Open Issues
    - **Version Negotiation** Editors' Draft / WG Draft / Repo / Open Issues
    - **Greasing the QUIC bit** Editors' Draft / WG Draft / Repo / Open Issues
    - **Acknowledgement Frequency** Editors' Draft / [WG Draft - TBD] / Repo / Open Issues
    - **qlog**
      - **Main logging Schema** Editors' Draft / WG Draft / Repo / Open Issues
      - **QUIC event definitions** Editors' Draft / WG Draft / Repo / Open Issues
      - **HTTP/3 and QPACK event definitions** Editors' Draft / WG Draft / Repo / Open Issues

Implementing QUIC

We have a number of experimental implementations in progress. The current goal for interop testing is the 21st Implementation Target.

Implementers should join the quicdev Slack to coordinate testing; contact the WG chairs for an invitation. Note that discussions on Slack are considered IETF contributions under "Note Well".

Automated, continuous interop testing is performed for participating QUIC implementations. Implementers are encouraged to join this effort by making compatible Docker images of their implementations available.

See Also

- Working Group materials – agendas, minutes, etc.
- Related Activities – QUIC extensions and applications that *might* happen

← 표준 문서

← 개발 정보

QUIC 관련 표준

- QUICA UDP Based Multiplexed and Secure Transport
  - RFC 9000: https://www.rfc-editor.org/info/rfc9000
- Using TLS to Secure QUIC
  - RFC 9001: https://www.rfc-editor.org/info/rfc9001
- QUIC Loss Detection and Congestion Control
  - RFC 9002: https://www.rfc-editor.org/info/rfc9002
- Version-Independent Properties of QUIC
  - RFC 8999: https://www.rfc-editor.org/info/rfc8999

QUIC 관련 라이브러리 개발 현황

구현체	라이선스	언어	설명
Chromium	무료	C++	This is the source code of the Chrome web browser and the reference gQUIC implementation. It contains a standalone gQUIC and QUIC client and server programs that can be used for testing. Browsable source code. This version is also the basis of LINE's stellite and Google's cronet.
QUIC Library (mvfst)	MIT 허가서	C++	mvfst (Pronounced move fast) is a client and server implementation of IETF QUIC protocol in C++ by Facebook.
LiteSpeed QUIC Library (lsquic)	MIT 허가서	C	This is the QUIC and HTTP/3 implementation used by LiteSpeed Web Server and OpenLiteSpeed.
ngtcp2	MIT 허가서	C	This is a QUIC library that's crypto library agnostic and works with OpenSSL or GnuTLS. For HTTP/3, it needs a separate library like nghttp3.
Quiche	BSD 허가서	Rust	Socket-agnostic and exposes a C API for use in C/C++ applications.
quicly	MIT 허가서	C	This library is the QUIC implementation for the H2O web server.
quic-go	MIT 허가서	Go	This library provides QUIC support in Caddy web server. Client functionality is also available.
Quinn	아파치 라이선스	Rust	
Neqo	아파치 라이선스	Rust	This implementation from 모질라 is planned to be integrated in Necko, a network library used in the Firefox web browser
aioquic	BSD 허가서	Python	This library features an I/O-free API suitable for embedding in both clients and servers.
picoquic	BSD 허가서	C	A minimal implementation of QUIC aligned with the IETF specifications
pquic	MIT 허가서	C	An extensible QUIC implementation that includes an eBPF virtual machine that is able to dynamically load extensions as plugins
MsQuic	MIT 허가서	C	A cross platform QUIC implementation from 마이크로소프트 designed to be a general purpose QUIC library.
QUANT	BSD 허가서	C	Quant supports traditional POSIX platforms (Linux, MacOS, FreeBSD, etc.) as well as embedded systems.
quic	BSD 허가서	Haskell	This package implements QUIC based on Haskell lightweight threads.

C

C++

Go

Haskell

Java

JavaScript

Objective-C

Python

RUST

Swift

화석화 (Ossification) 탈피 (1/2)

Skip [Lecture.10 참조]

인터넷은 네트워크의 네트워크다. 이 네트워크의 네트워크가 의도대로 동작하게 하는 장비가 인터넷 여러 곳에 설치되어 있다. 우리는 이러한 기기(네트워크에 분포된 박스)를 미들박스라고 부른다. 이 박스는 전통적인 네트워크 데이터 전송의 주요 요소인 두 엔드포인트 사이에 있다.

박스는 여러 가지 다른 목적이 있지만 무언가 동작하게 하려면 이 박스가 해당 위치에 있어야 한다고 누군가 생각했기 때문에 곳곳에 깔린 것이라 생각한다.

미들박스는 네트워크 사이에서 IP 패킷을 라우팅하고, 악성 트래픽을 차단하고, NAT(Network Address Translation) 역할을 하고, 성능을 향상시키고, 통과하는 트래픽을 감시하는 등의 역할을 한다.

이러한 박스는 의무를 다하려면 자신들이 모니터링하고 수정하는 네트워크 및 프로토콜에 대해 반드시 알아야 한다. 박스는 이런 목적으로 소프트웨어를 실행한다. 그 소프트웨어를 항상 자주 업그레이드하는 것은 아니다.

화석화 (Ossification) 탈피 (2/2)

Skip [Lecture.10 참조]

박스는 인터넷이 유지되도록 연결하는 구성 요소이지만 종종 최신 기술을 따라잡지 못하는 경우가 있다. 보통 네트워크 중간 영역은 전 세계의 클라이언트나 서버 같은 엣지 만큼 빠르게 바뀌지 않는다.

어떤 프로토콜을 검사하기를 원하는지, 그리고 어떤것이 괜찮고 안 괜찮은지 알고있는 박스들은 과거에 배포되었고 그 당시의 프로토콜 기능 셋을 가지고 있다. 이전에는 몰랐던 새로운 기능이나 동작의 변경 사항이 추가되면 박스가 이를 잘못된 것이나 허용하지 않는 것으로 판단할 위험이 있다. 그래서 이러한 트래픽은 사용자가 해당 기능을 사용하고 싶지 않을 정도로 지연되거나 중단될 수 있다.

이를 "프로토콜 고착화(ossification)"라고 부른다.

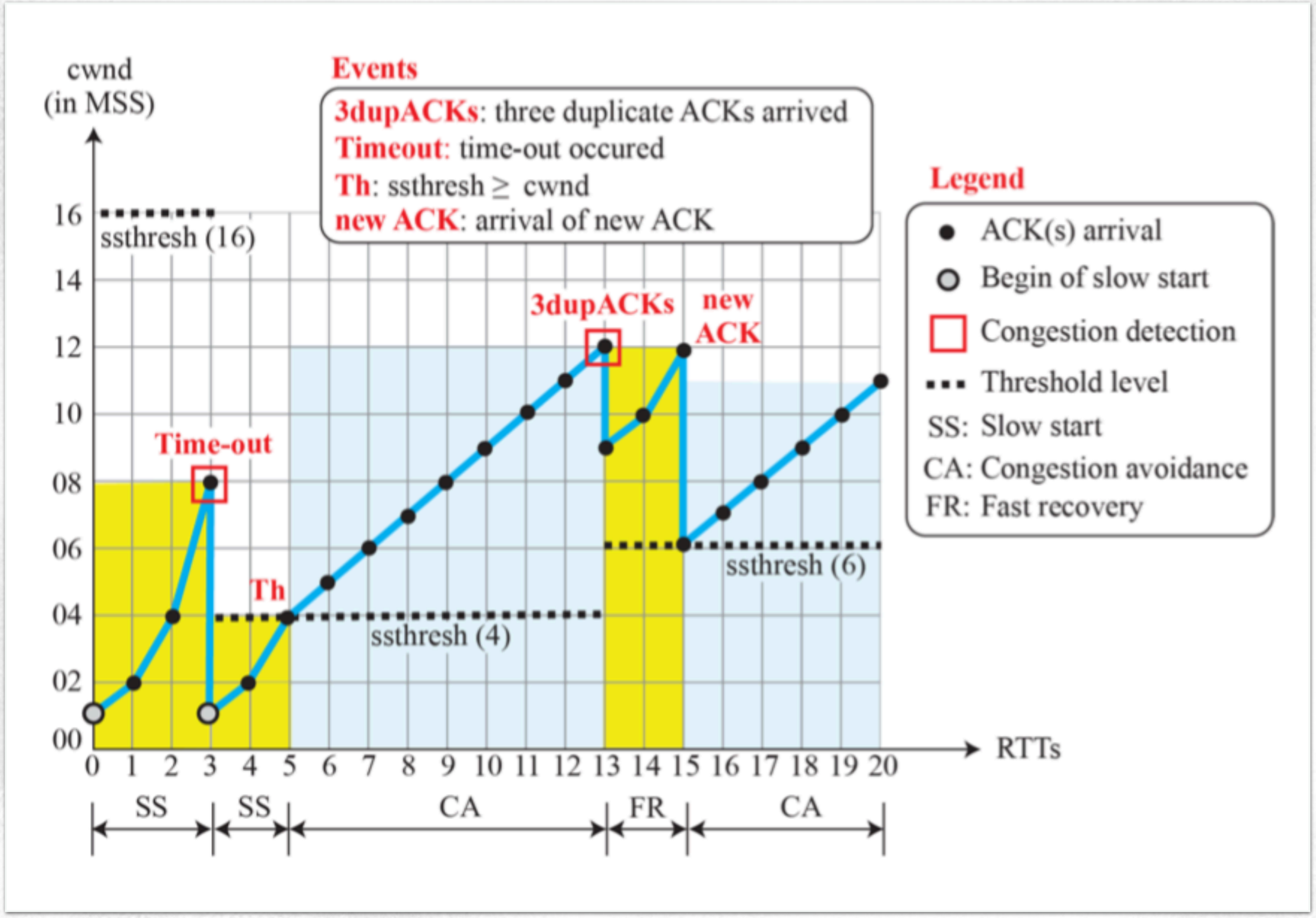
TCP를 변경하는 것도 고착화 문제를 겪는다. 클라이언트와 원격 서버 사이에 있는 박스 중 일부는 알지 못하는 새로운 TCP 옵션을 발견하고 이 옵션이 무엇인지 몰라서 해당 연결을 차단해버릴 것이다. 프로토콜의 상세 내용을 탐지하도록 허용한 경우 시스템은 프로토콜이 보통 어떻게 동작하는지 배우게 되고 시간이 지나면 프로토콜을 변경할 수 없게 된다.

유일하게 고착화를 "방지하는데" 효과적인 방법은 미들박스가 지나가는 프로토콜에서 많은 것을 볼 수 없도록 통신을 최대한 암호화하는 것이다.

TCP의 현대 사회에 어울리지 않는 동작 (1)

Skip [Lecture.10 참조]

- Slow start
- Multiplicative decrease based congestion control



Reference: Behrouz A. Forouzan, "Data Communications and Networking, 5th Edition", McGraw-Hill Companies, Inc.

TCP의 현대 사회에 어울리지 않는 동작 (2)

Skip [Lecture.10 참조]

HTTP/2를 사용하는 일반적인 브라우저는 TCP 연결 한개로 수십, 수백 개의 병렬 전송을 한다.

HTTP/2로 통신하는 두 엔드포인트 사이 네트워크 어딘가에서 하나의 패킷이 빠지거나 없어진다면 없어진 패킷을 다시 전송하고 목적지를 찾는 동안 전체 TCP 연결이 중단되게 된다. 즉, TCP는 "체인"이기 때문에 한 링크가 갑자기 사라지면 그 링크 이후에 와야 하는 모든 것들이 기다려야 한다는 뜻이다.

체인 메타포를 사용해서 이 연결을 통해 두 스트림을 보내는 경우를 그린 그림을 보자. 빨간색 스트림과 녹색 스트림이 있다.

체인에서 링크는 다른 색으로 보여준다

이는 TCP에 기반을 둔 head of line 블로킹이 된다!

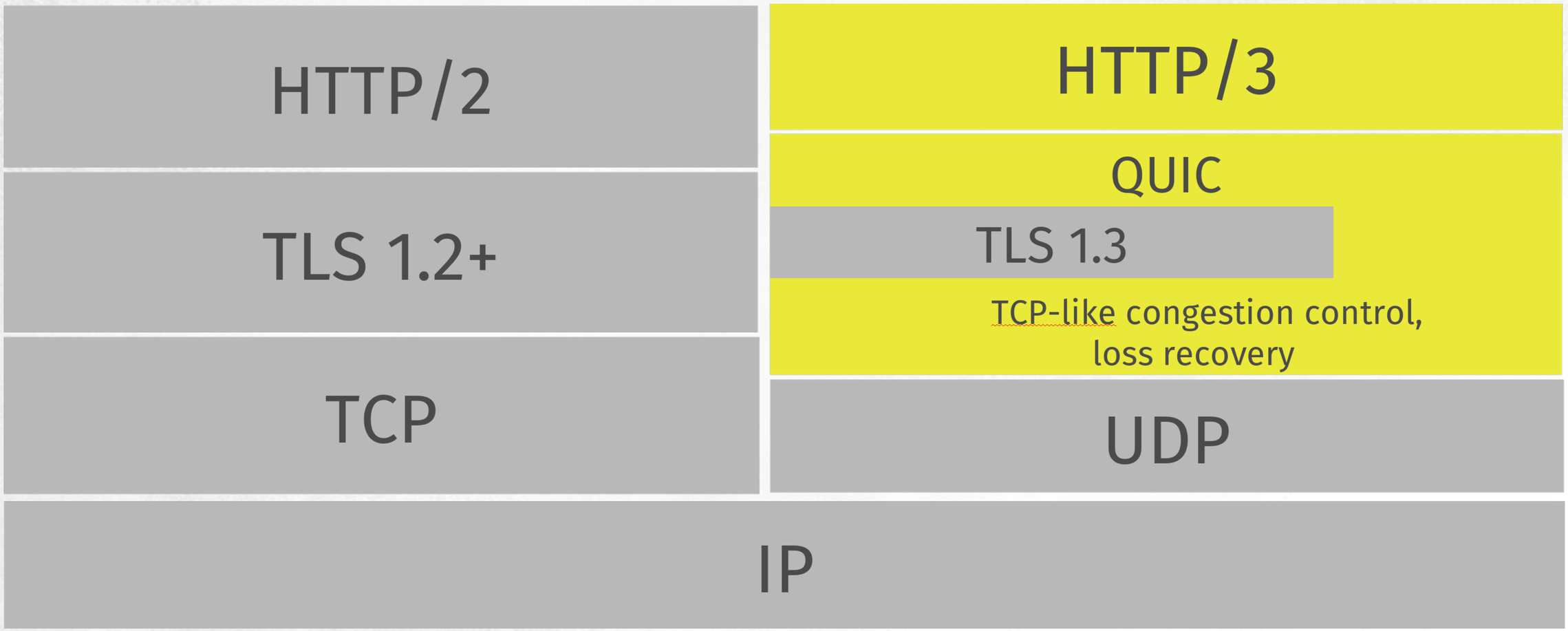
패킷 손실률이 증가하면 HTTP/2의 성능도 저하된다. 테스트를 통해 패킷 손실률이 2%(상기하자면, 이는 끔찍한 네트워크 품질이다) 일때 HTTP/1 사용자가 더 나은 것으로 입증했다. 그 이유는 HTTP/1은 보통 손실된 패킷을 분배하는데 6개의 TCP 연결을 갖고 있어서 손실된 패킷이 없는 다른 연결은 계속 사용할 수 있기 때문이다.

TCP를 사용하는 한 이 이슈를 고치는 것은 (가능할 수도 있지만) 쉽지 않다.

Skip [Lecture.10 참조]

UDP based NEW Transport Protocol (1/2)

- UDP를 기반으로 하여 새롭게 만들어진 전송 프로토콜임



Reference: https://http3-explained.haxx.se/

Skip [Lecture.10 참조]

UDP based NEW Transport Protocol (2/2)

- QUIC은 UDP 위에 새로운 계층을 추가함으로써 신뢰성을 제공함. 추가된 계층은 TCP에 존재하는 패킷 재전송, 혼잡 제어, 속도 조정 및 다른 기능들을 제공함
  - UDP는 데이터 전송의 신뢰성을 보장하지 않음
- QUIC은 (TCP와 같은) 범용의 전송 프로토콜로써 다른 애플리케이션 프로토콜이 그 위에서 사용할 수 있음
  - 첫 애플리케이션 계층 프로토콜은 HTTP/3(h3)임
  - 전송 계층은 연결과 스트림을 지원함

Reference: https://http3-explained.haxx.se/

HTTP/3 is 'HTTP over QUIC'

- HTTP is the First & Primary service of QUIC
- HTTP/2 vs HTTP/3 (HTTP over QUIC)
  - In QUIC the streams are provided by the transport itself, while in HTTP/2 the streams were done within the HTTP layer
  - QUIC streams are slightly different than HTTP/2 streams (thus, header compression protocol is also slightly different)

Reference: https://http3-explained.haxx.se/

HTTP/2와 HTTP/3의 유사점

Skip [Lecture.10 참조]

- 스트림을 제공함
- 서버 푸시를 지원함
- 헤더 압축을 제공함
  - QPACK과 HPACK은 설계상 비슷함
- 스트림을 이용해서 하나의 연결을 통해 멀티플랙싱을 제공함
- 스트림에 우선순위를 정함

HTTP/1.1 vs HTTP/2 vs HTTP/3

HTTP/1.1 ASCII over TCP

HTTP/2 Binary multiplexed over TCP

HTTP/3 Binary over Multiplxed QUIC

Reference: https://files.gotocon.com/uploads/slides/conference_36/1626/original/http3-gotopia-2020.pdf

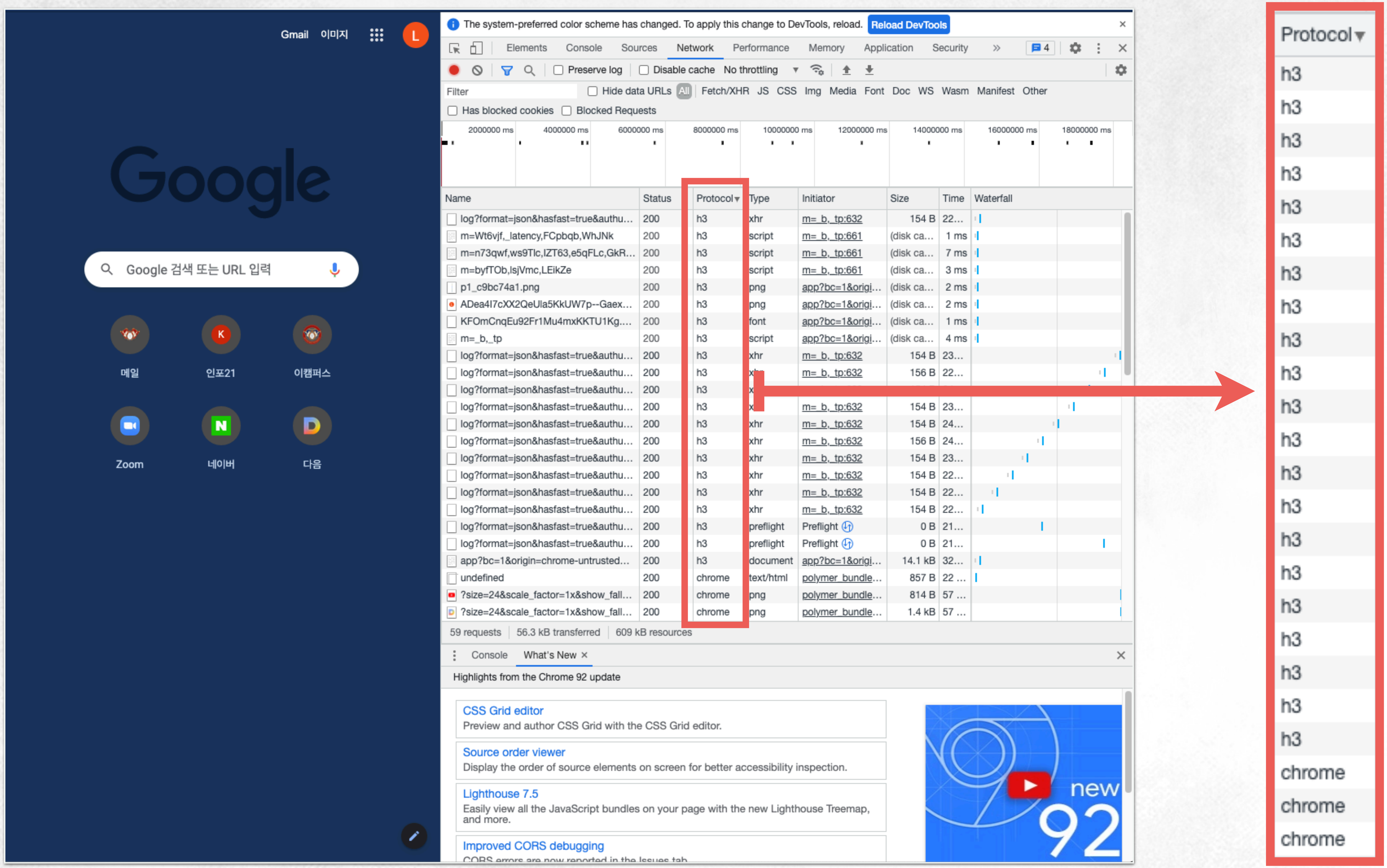
2021.05 Standard Finalized !!!

- Sites
  - Google
  - YouTube
  - Instagram
  - Facebook
  - Cloudflare

Reference: https://files.gotocon.com/uploads/slides/conference_36/1626/original/http3-gotopia-2020.pdf

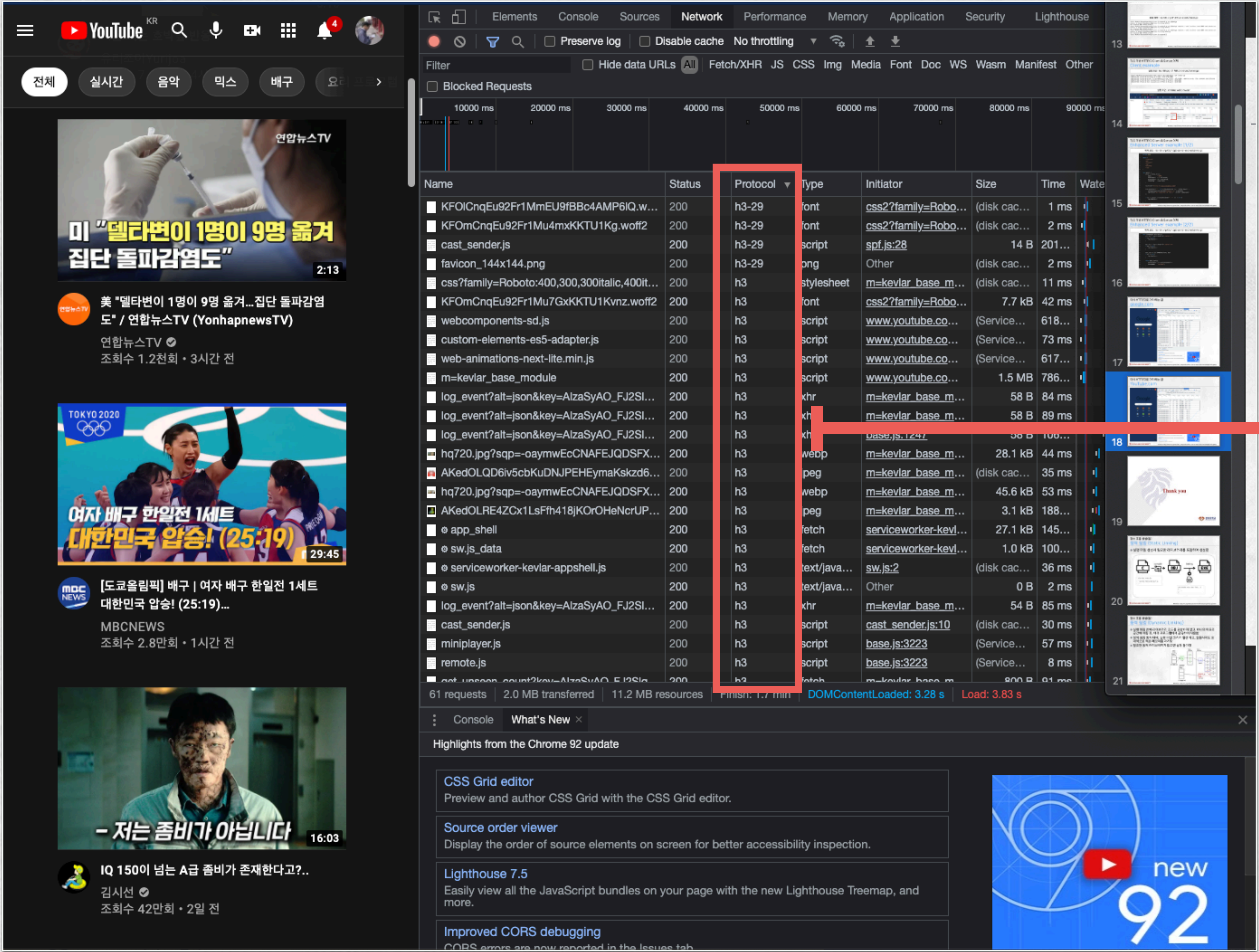
QUIC & HTTP/3 현황

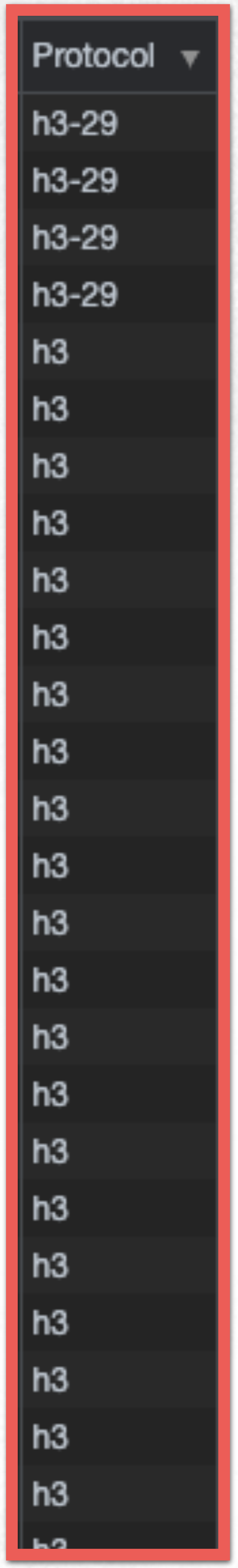
2021.05 Standard Finalized !!!



QUIC & HTTP/3 현황

2021.05 Standard Finalized !!!

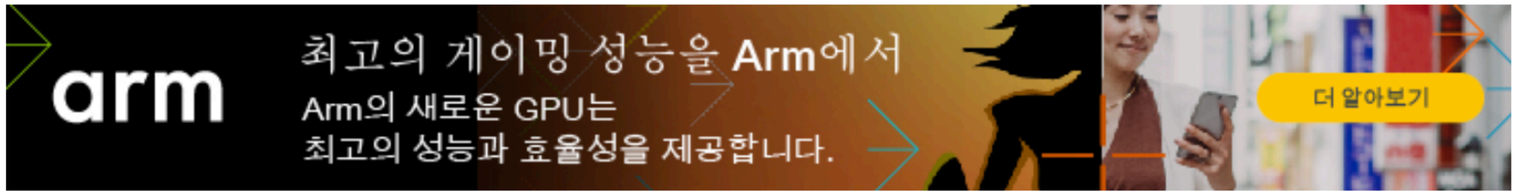




QUIC & HTTP/3 현황

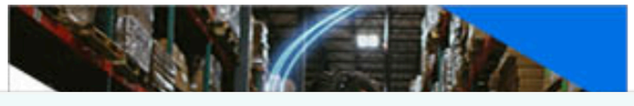
Finally in NAVER (2022.11)

ZDNET Korea 뉴스 이슈진단+ 지스타2022 반도체 교육 컨퍼런스 칼럼・연재 포토・영상



NG

네이버, HTTP/3 도입…"음영지역서도 안정적 검색 가능"

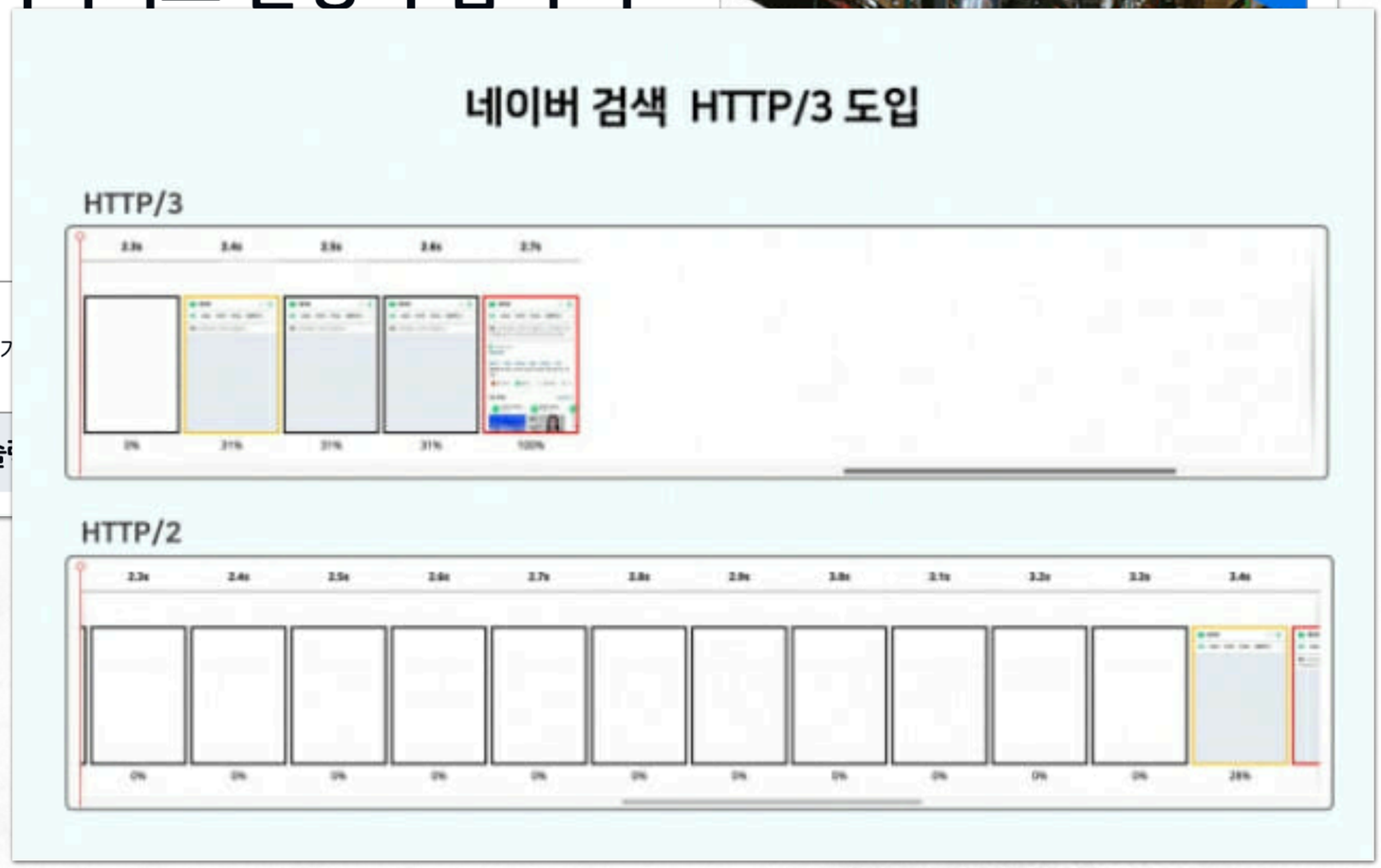


| 이용자에게 빠르고 안정적인 검색 환경 제공 목표

인터넷 | 입력 :2022/11/15 10:55

김성현 기자 | ✉ N 기자 페이지 구독 기자의 다른기사 보기

[웨비나] 11월 22일 Slack 추가기능 데모 및 무신사 피플 슬



2021.05 Standard Finalized !!!

- Server
  - Nginx support for HTTP/3 is being worked on. A technology preview of nginx with HTTP/3 support was released in June 2020
  - LiteSpeed Web Server (and OpenLiteSpeed) supports HTTP/3 draft 32
  - The Caddy web server has experimental support for HTTP/3 as of 2.0 beta 17
  - Cloudflare distributes a patch for nginx that integrates the quiche HTTP/3 library into it
  - Apache ? ==> Plan is not released

Reference: https://en.wikipedia.org/wiki/HTTP/3

QUIC & HTTP/3 현황

2021.05 Standard Finalized !!!

Browser

Browser support for HTTP/3

Browser	Version implemented (disabled by default)		Version shipped (enabled by default)	
Chrome	Stable build (79)	December 2019	87[16]	April 2020[17]
Firefox	Stable build (72.0.1)	January 2020	88[5]	April 2021[18]
Safari	Safari Technology Preview 104	April 2020		
Edge			87	April 2020

2021.05 Standard Finalized !!!

Programming Library

Library support for HTTP/3

Name	Client	Server	Programming language	Company	Repository
http	Yes	Yes	Rust		https://github.com/hyperium/http
quiche	Yes	Yes	Rust	Cloudflare	https://github.com/cloudflare/quiche
neqo	Yes	Yes	Rust	Mozilla	https://github.com/mozilla/neqo
quinn	Yes	Yes	Rust		https://github.com/quinn-rs/quinn
proxygen	No	Yes	C++	Facebook	https://github.com/facebook/proxygen#quic-and-http3
Cronet	Yes	Yes	C++	Google	https://github.com/chromium/chromium/tree/master/net/quic
lsquic	Yes	Yes	C	LiteSpeed	https://github.com/litespeedtech/lsquic
nghttp3	Yes	Yes	C		https://github.com/ngtcp2/nghttp3
h2o	No	Yes	C		https://github.com/h2o/h2o
libcurl[20][21]	Yes	No	C		https://github.com/curl/curl
MsQuic[22]	Yes	Yes	C	Microsoft	https://github.com/microsoft/msquic
Quic.NET	Yes	Yes	C#		https://github.com/Vect0rZ/Quic.NET
Flupke	Yes	No	Java		https://bitbucket.org/pjtr/flupke
aioquic	Yes	Yes	Python		https://github.com/aiortc/aioquic
quic-go	Yes	Yes	Go		https://github.com/lucas-clemente/quic-go
http3	Yes	Yes	Haskell		https://github.com/kazu-yamamoto/http3

QUIC & HTTP/3 현황

2021.05 Standard Finalized !!!

Python Programming Library

README.rst

aioquic

docs passing | pypi v0.9.15 | python 3.6 | 3.7 | 3.8 | 3.9 | license BSD | tests passing | coverage 100% | code style black

What is `aioquic`?

`aioquic` is a library for the QUIC network protocol in Python. It features a minimal TLS 1.3 implementation, a QUIC stack and an HTTP/3 stack.

QUIC standardisation is not finalised yet, but `aioquic` closely tracks the specification drafts and is regularly tested for interoperability against other QUIC implementations.

To learn more about `aioquic` please read the documentation.

Why should I use `aioquic`?

`aioquic` has been designed to be embedded into Python client and server libraries wishing to support QUIC and / or HTTP/3. The goal is to provide a common codebase for Python libraries in the hope of avoiding duplicated effort.

Both the QUIC and the HTTP/3 APIs follow the "bring your own I/O" pattern, leaving actual I/O operations to the API user. This approach has a number of advantages including making the code testable and allowing integration with different concurrency models.

Features

- QUIC stack conforming with RFC 9000
- HTTP/3 stack conforming with draft-ietf-quic-http-34
- minimal TLS 1.3 implementation
- IPv4 and IPv6 support
- connection migration and NAT rebinding
- logging TLS traffic secrets
- logging QUIC events in QLOG format
- HTTP/3 server push support

Languages

Python 96.1% C 3.9%

QUIC & HTTP/3 현황

2021.05 Standard Finalized !!!

C++ Programming Library

Product Solutions Open Source Pricing Search / Sign in Sign up

facebook / proxygen Public

Notifications Fork 1.5k Star 7.6k

Code Issues 28 Pull requests 13 Actions Projects Security Insights

main 4 branches 303 tags Go to file Code

About

A collection of C++ HTTP libraries including an easy to use HTTP server.

afrind and facebook-github-bot Fix off-by-one in control stream A... ... 183c823 19 minutes ago 11,277 commits

Proxygen: Facebook's C++ HTTP Libraries

Support Ukraine linux passing mac passing

This project comprises the core C++ HTTP abstractions used at Facebook. Internally, it is used as the basis for building many HTTP servers, proxies, and clients. This release focuses on the common HTTP abstractions and our simple HTTPServer framework. Future releases will provide simple client APIs as well. The framework supports HTTP/1.1, SPDY/3, SPDY/3.1, HTTP/2, and HTTP/3. The goal is to provide a simple, performant, and modern C++ HTTP library.

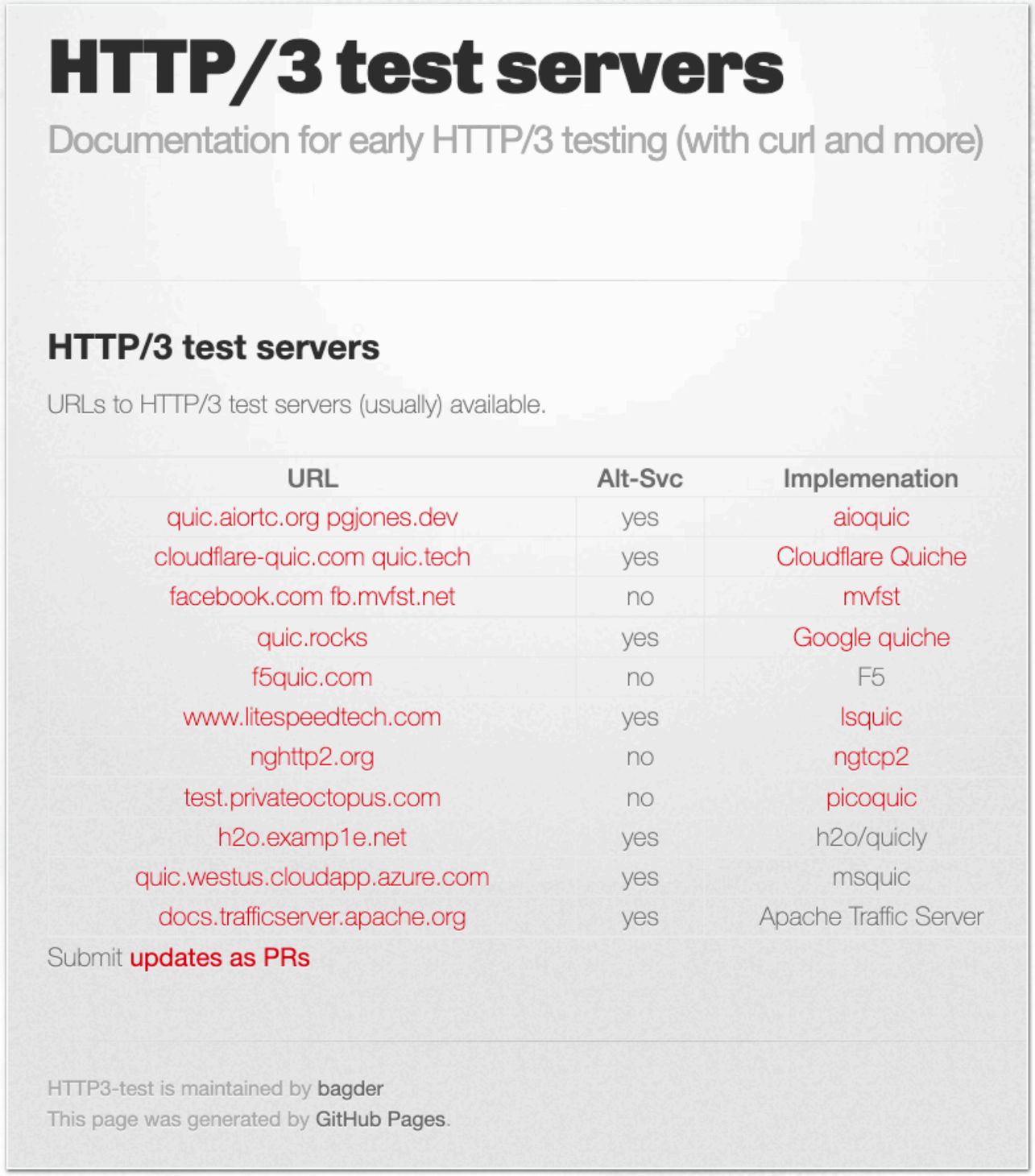
We have a Google group for general discussions at https://groups.google.com/d/forum/facebook-proxygen.

The original blog post also has more background on the project.

Reference: https://github.com/facebook/proxygen

2021.05 Standard Finalized !!!

- Test Sites (Servers)



HTTP/3 test servers

Documentation for early HTTP/3 testing (with curl and more)

HTTP/3 test servers

URLs to HTTP/3 test servers (usually) available.

URL	Alt-Svc	Implemenation
quic.aiortc.org pgjones.dev	yes	aioquic
cloudflare-quic.com quic.tech	yes	Cloudflare Quiche
facebook.com fb.mvfst.net	no	mvfst
quic.rocks	yes	Google quiche
f5quic.com	no	F5
www.litespeedtech.com	yes	lsquic
nghttp2.org	no	ngtcp2
test.privateoctopus.com	no	picoquic
h2o.examp1e.net	yes	h2o/quicly
quic.westus.cloudapp.azure.com	yes	msquic
docs.trafficserver.apache.org	yes	Apache Traffic Server

Submit **updates as PRs**

HTTP3-test is maintained by **bagder**
This page was generated by **GitHub Pages**.

Reference: https://bagder.github.io/HTTP3-test/

2021.05 Standard Finalized !!!

- Tools
  - curl: experimentally supported

curl HTTP/3 command line — @bagder

```
$ curl --http3 https://example.com/
HTTP/3 200
date: Wed, 09 Oct 2019 11:16:06 GMT
content-type: text/html
content-length: 10602
set-cookie: crazy=d8bc7e7; expires=Thu, 08-Oct-22
11:16:06 GMT; path=/; domain=example.com;
alt-svc: h3-32=":443"; ma=86400
```


Reference: https://files.gotocon.com/uploads/slides/conference_36/1626/original/http3-gotopia-2020.pdf

Non Socket API

- QUIC에는 표준 API가 없음
- QUIC에서는 기존 라이브러리 구현체 중 하나를 선택하고 그 API를 따라야 함. 이는 애플리케이션을 어떤 부분에서 하나의 라이브러리에 "락인(locked in)"시킴. 다른 라이브러리로 바꾼다는 것은 다른 API를 뜻하므로 많은 작업이 필요할 것임
- 또한 QUIC이 보통 사용자 영역에서 구현되므로 소켓 API를 쉽게 확장할 수 없고 기존의 TCP나 UDP 기능과 비슷하게 보일 수 없음. QUIC을 사용한다는 것은 소켓 API와는 다른 API를 사용한다는 것을 의미함

Reference: https://http3-explained.haxx.se/

Kernel & Code Optimization

- Google과 Facebook은 QUIC의 대규모 배포가 TLS에서 HTTP/2로 서비스 할때보다 같은 트래픽 기준으로 거의 2배의 CPU가 필요하다 보고함
  - Linux의 UDP 부분은 TCP 스택만큼 최적화되어 있지 않음. 이는 전통적으로 지금처럼 고속 전송에 사용해오지 않았기 때문임
  - 하드웨어가 TCP와 TLS의 부담을 덜어주고 있지만 UDP에 대해서 그렇게 하는 경우는 드물고 기본적으로 QUIC에 대해서는 아예 존재하지 않음
- 현재 업계와 학계에서의 QUIC 프로토콜 성능 개선을 위한 연구가 활발하게 논의중인 상황임

Reference: https://http3-explained.haxx.se/

What's NEXT?

Thank you

경희대학교
KYUNG HEE UNIVERSITY