PIOLINK

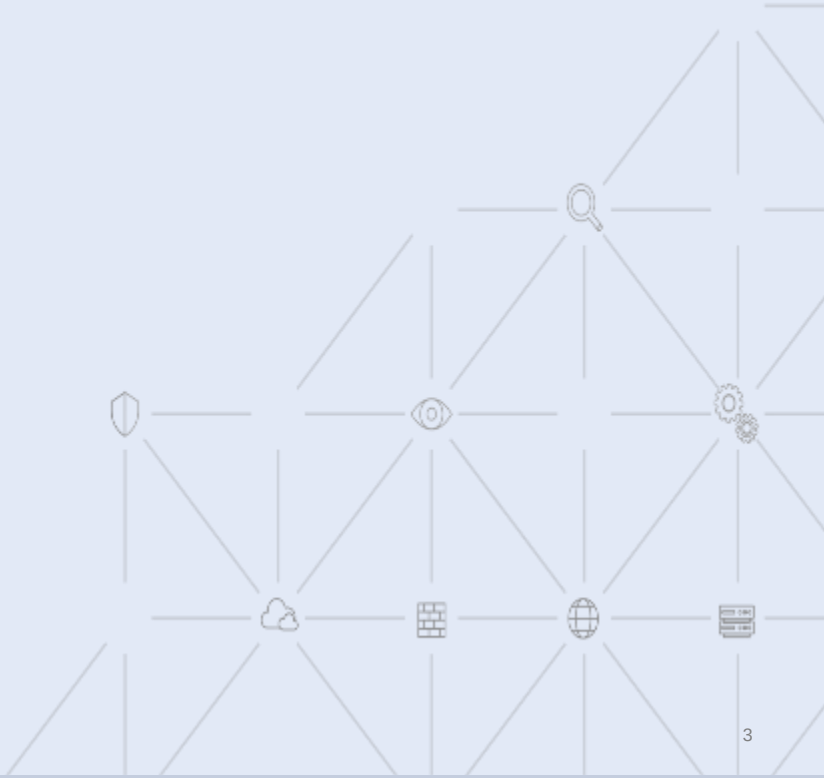
HTTP 2.0

파이오링크 | ADC사업실
2021.12.29

PIOLINK

목차

HTTP/2.0 등장 배경

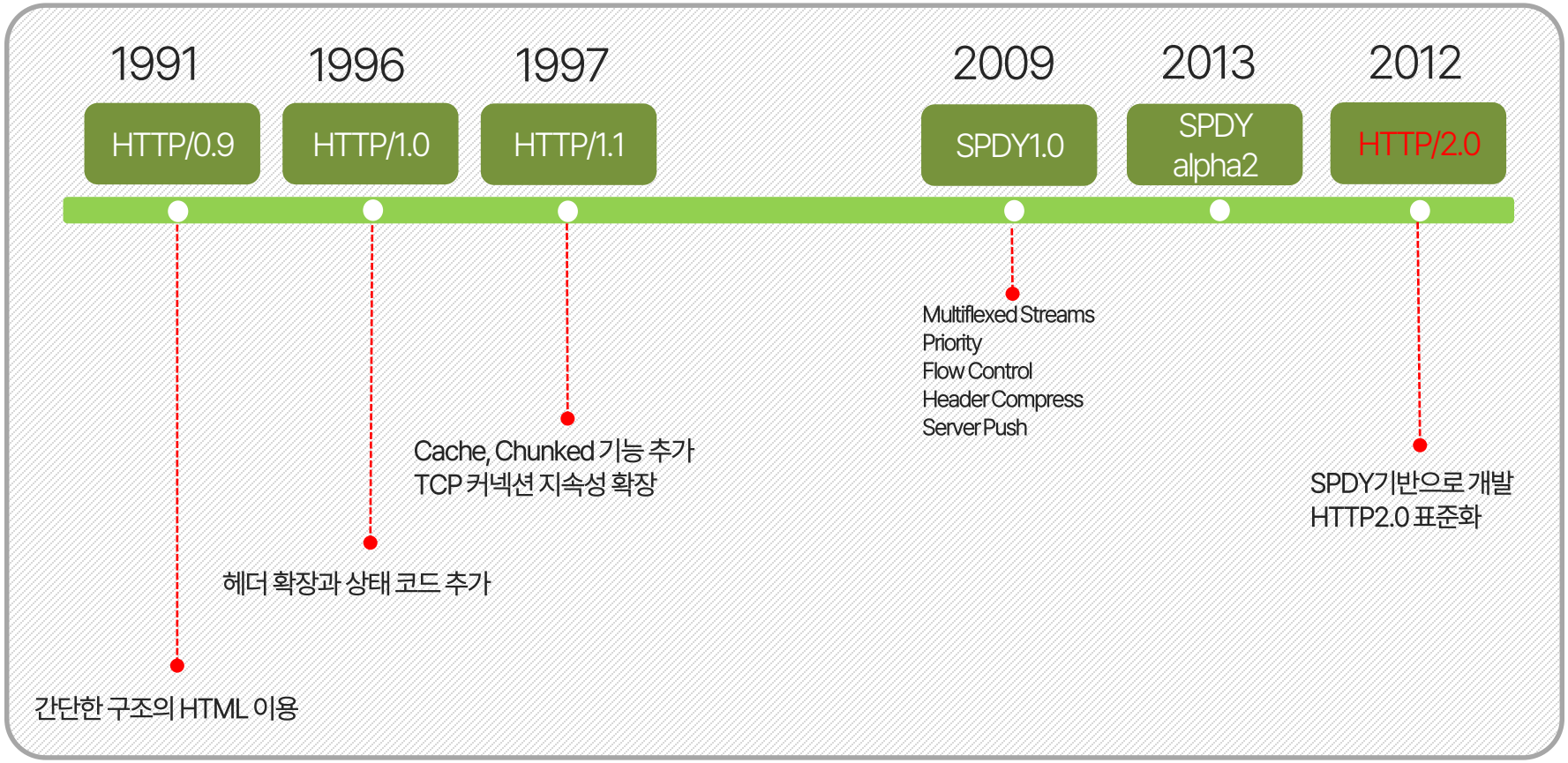


HTTP/2.0 등장배경

History

HTTP 역사

- HTTP/1.1 웹 페이지 로드 시간을 줄이기 위해 구글에서 SPDY 개발
- SPDY 비 표준이므로, HTTP2.0 표준안 발표

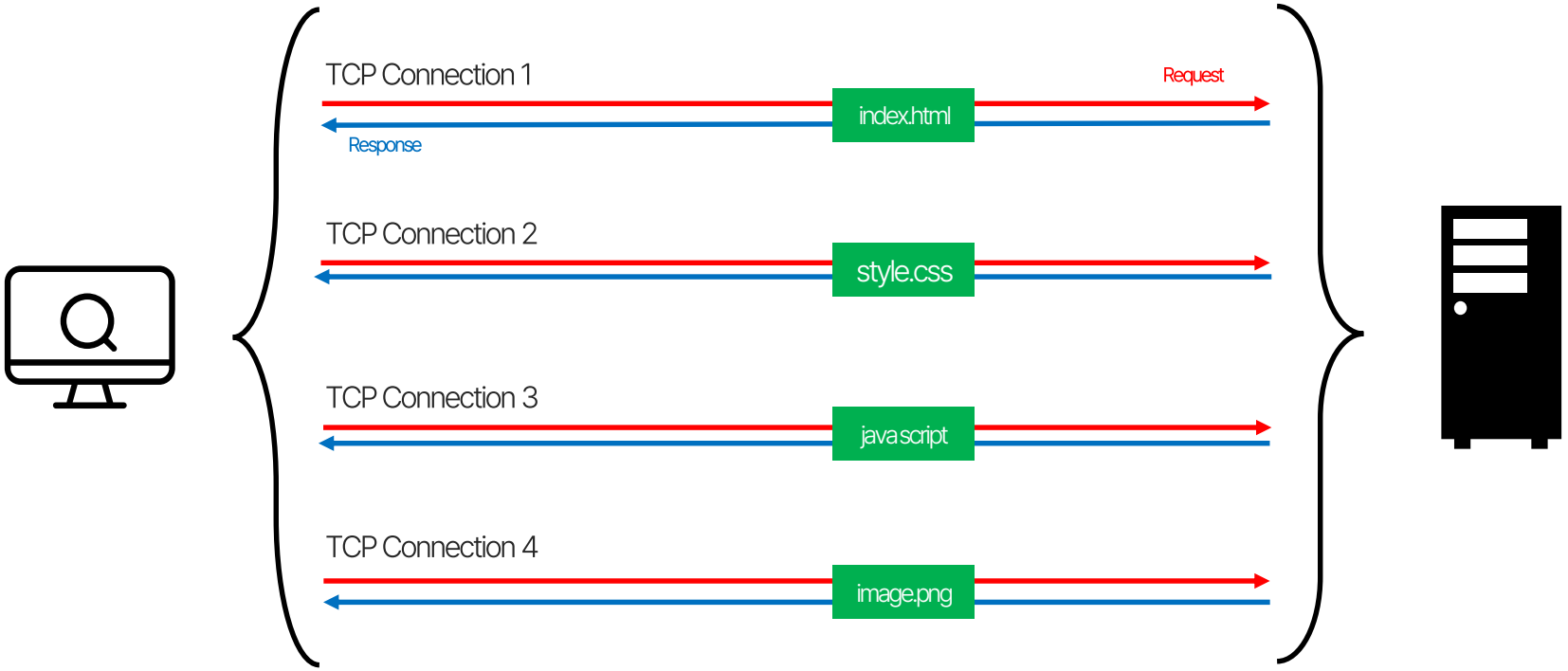


HTTP/2.0 등장배경

HTTP/1.1 Performance issue

HTTP/1.1 개선 사항과 한계

- HTTP 요청과 응답 처리 이후에 다음 HTTP 요청이 가능한 교환 방식의 구조.
  - Latency 감소를 위해 병렬 TCP 커넥션 연결
  - 파이프라인으로 HTTP 요청 시 패킷의 구분자 부재로 패킷 응답 지연 발생 시 성능 이슈 발생



HTTP/1.1 메시지 요청 방식

HTTP/2.0 등장배경

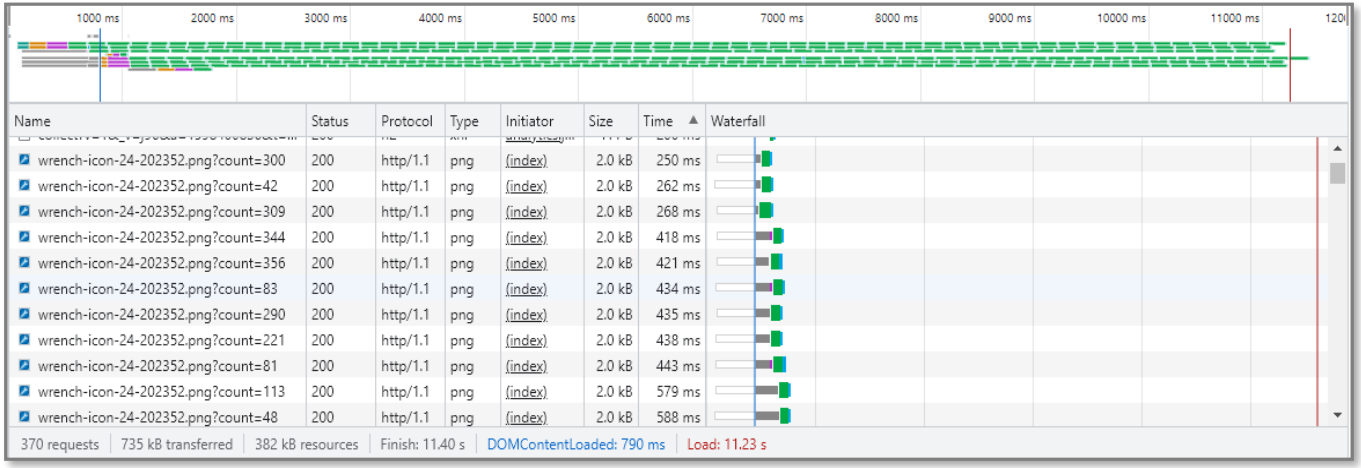
Waterfall

Waterfall 다이어그램 비교

- HTTPS: 순차적으로 리소스 요청 후 페이지 생성되는 방식

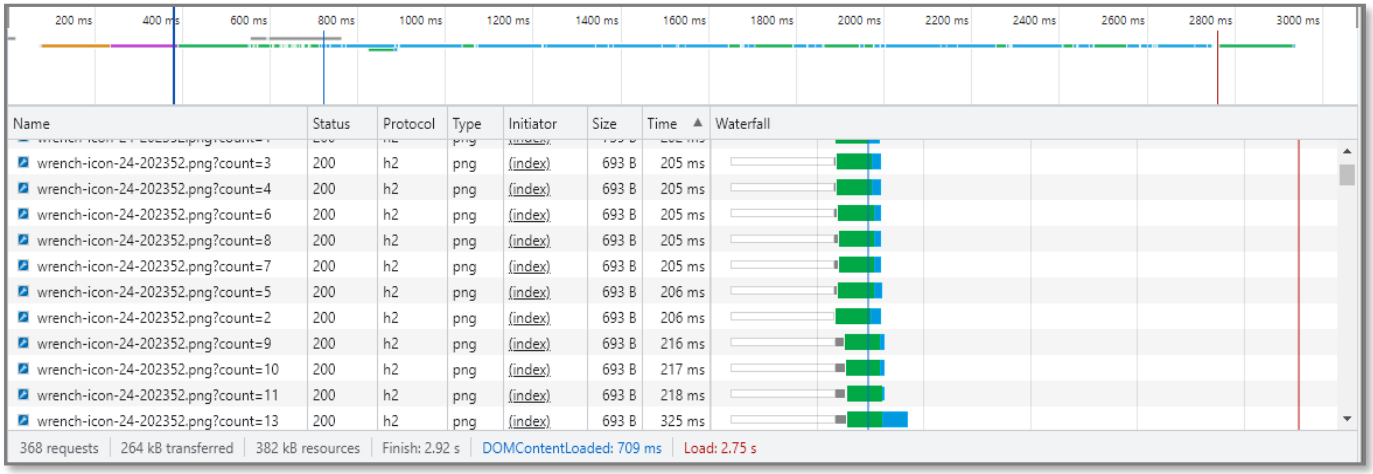
11.144s

Click HTTP button to compare
2.015s - HTTPS is 453% slower than HTTP/2

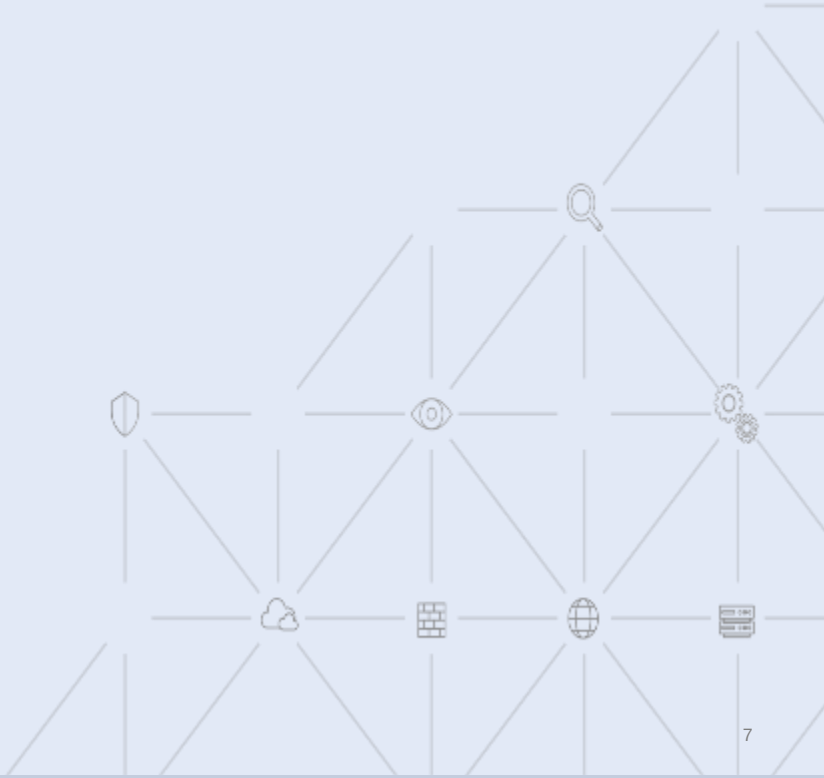


- HTTP2: 병렬 스트림으로 리소스 요청, Latency 감소

2.557s



HTTP/2.0 특징



HTTP/2.0 특징

HTTP/2.0 Communication

HTTP2 협상

- HTTPS 통신 방식
  - HTTPS 스킴에서 HTTP2 통신을 하려면, SSL 협상 과정 중 ALPN을 Extention하여 프로토콜 지원 여부 확인 과정이 필요합니다.

CLIENT HELLO

```
Extension: application_layer_protocol_negotiation (len=14)
    Type: application_layer_protocol_negotiation (16)
    Length: 14
    ALPN Extension Length: 12
  ALPN Protocol
      ALPN string length: 2
      ALPN Next Protocol: h2
      ALPN string length: 8
      ALPN Next Protocol: http/1.1
```

- 클라이언트가 지원하는 프로토콜을 CLIENT HELLO 패킷으로 전송

SERVER HELLO

```
Extension: application_layer_protocol_negotiation (len=5)
    Type: application_layer_protocol_negotiation (16)
    Length: 5
    ALPN Extension Length: 3
  ALPN Protocol
      ALPN string length: 2
      ALPN Next Protocol: h2
```

- 서버가 사용할 프로토콜을 선택하여 SERVER HELLO 패킷으로 응답

HTTP/2.0 특징

HTTP/2.0 Communication

HTTP2 협상

- HTTP Clear 텍스트 방식
  - h2c 헤더 업그레이드 후 HTTP2 통신이 가능합니다.
  - 헤더 업그레이드 경우, 브라우저에서 지원하지 않기 때문에 서버to서버 환경에서 사용합니다.

클라이언트 요청 패킷(서버)

```
GET / HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: <base64url encoding of HTTP/2 SETTINGS payload>
```

- 클라이언트에서 헤더 업그레이드 요청

서버 응답 패킷(서버)

```
[root@localhost ~]# curl --http2 -I localhost
HTTP/1.1 101 Switching Protocols
Upgrade: h2c
Connection: Upgrade

HTTP/2 200
date: Sun, 00 Jan 1900 00:00:00 GMT
server: Apache/2.4.37 (centos) OpenSSL/1.1.1k
last-modified: Thu, 29 Aug 2019 08:26:18 GMT
etag: W/"85-5913d427bde80"
accept-ranges: bytes
content-length: 133
content-type: text/html; charset=UTF-8
```

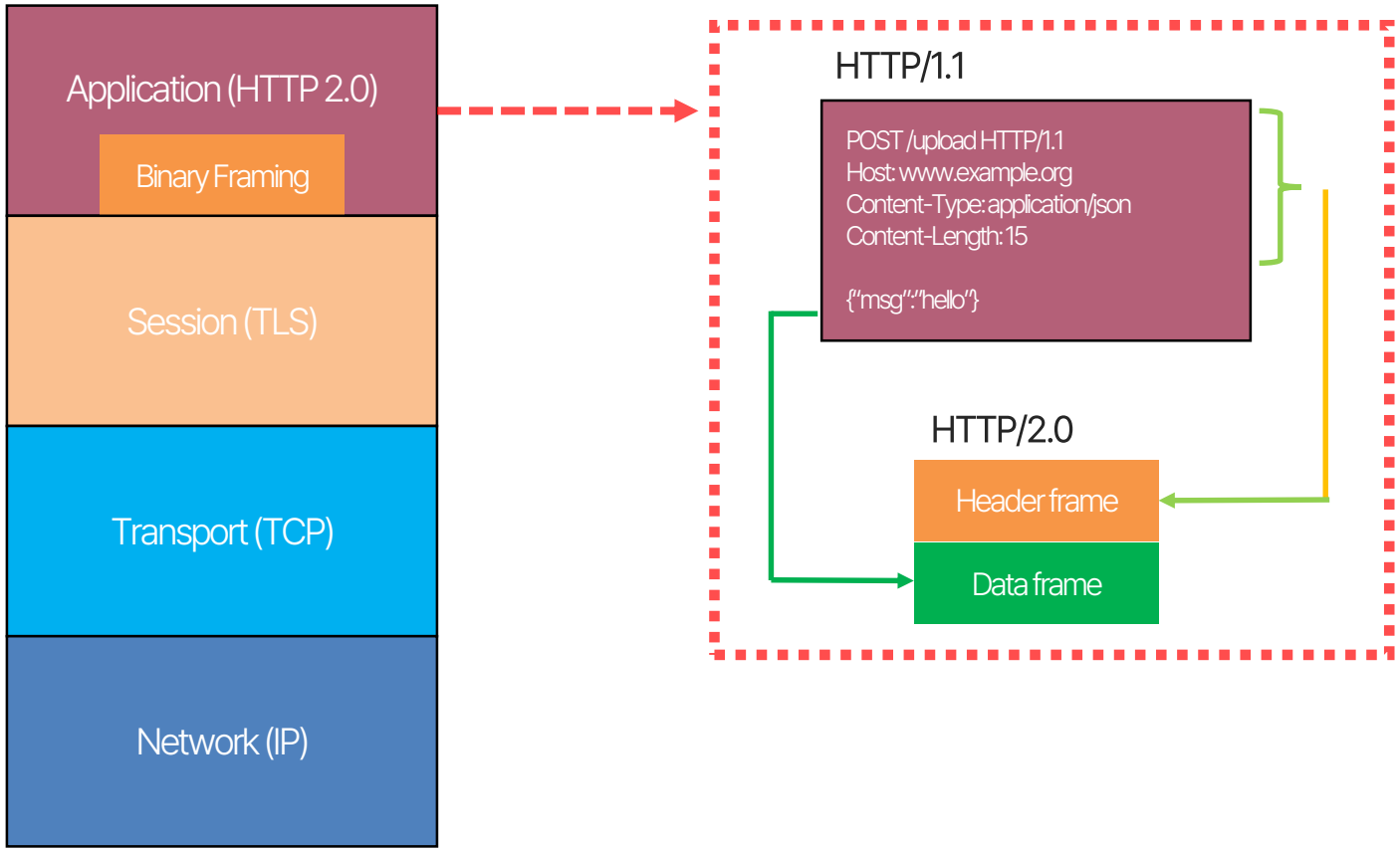
- Switching Protocols 응답 이후 HTTP/2.0 통신
- 서버가 HTTP2 미 지원 시 HTTP/1.1 200 OK 응답

HTTP/2.0 특징

Binary Format

Binary Format 특징

- 평문 구조에서 서버가 인식하기 쉬운 바이너리 포맷 형태로 변경.
- HTTP/1.1은 하나의 패킷에 헤더와 바디를 구분하여 전달했지만, HTTP2는 헤더/데이터 프레임으로 나누어서 전송.

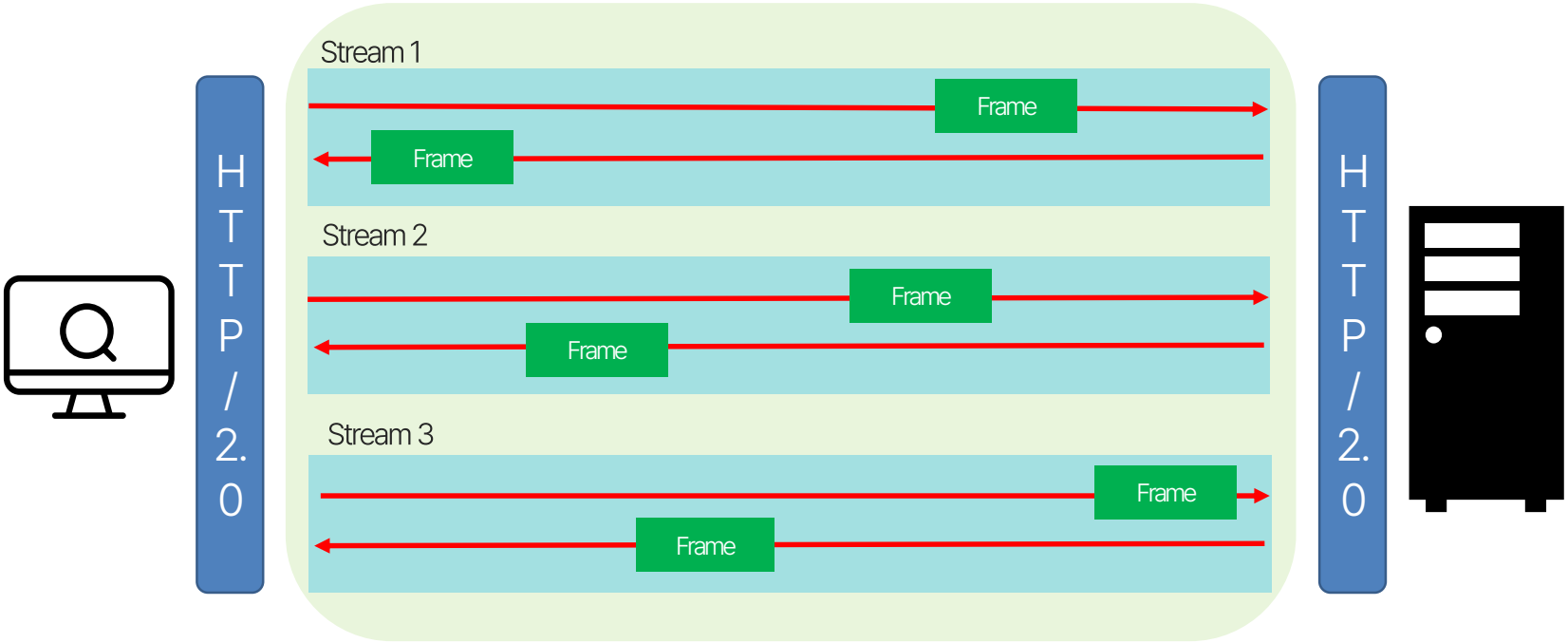


HTTP/2.0 특징

MultiFlexed Streams

MultiFlexed Streams 특징

- 스트림은 클라이언트와 서버 사이에 독립적으로 연결되어 프레임을 양방향 통신 합니다.
- HTTP2.0 커넥션에서 여러 스트림 생성이 가능하고 클라이언트와 서버는 스트림 상태를 공유합니다.



HTTP/2.0 특징

STREAM Priority

```
[  0.519] Connected
The negotiated protocol: h2
[  0.822] recv SETTINGS frame <length=6, flags=0x00, stream_id=0>
          (niv=1)
          [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]
[  0.822] recv WINDOW_UPDATE frame <length=4, flags=0x00, stream_id=0>
          (window_size_increment=2147418112)
[  0.822] send SETTINGS frame <length=12, flags=0x00, stream_id=0>
          (niv=2)
          [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]
          [SETTINGS_INITIAL_WINDOW_SIZE(0x04):65535]
[  0.822] send SETTINGS frame <length=0, flags=0x01, stream_id=0>
          ; ACK
          (niv=0)
[  0.822] send PRIORITY frame <length=5, flags=0x00, stream_id=3>
          (dep_stream_id=0, weight=201, exclusive=0)
[  0.822] send PRIORITY frame <length=5, flags=0x00, stream_id=5>
          (dep_stream_id=0, weight=101, exclusive=0)
[  0.822] send PRIORITY frame <length=5, flags=0x00, stream_id=7>
          (dep_stream_id=0, weight=1, exclusive=0)
[  0.822] send PRIORITY frame <length=5, flags=0x00, stream_id=9>
          (dep_stream_id=7, weight=1, exclusive=0)
[  0.822] send PRIORITY frame <length=5, flags=0x00, stream_id=11>
          (dep_stream_id=3, weight=1, exclusive=0)
[  0.822] send HEADERS frame <length=42, flags=0x25, stream_id=13>
          ; END_STREAM | END_HEADERS | PRIORITY
          (padlen=0, dep_stream_id=11, weight=16, exclusive=0)
          ; Open new stream
          :method: GET
          :path: /
          :scheme: https
          :authority: www.tunetheweb.com
          accept: */*
          accept-encoding: gzip, deflate
          user-agent: nghttp2/1.31.1
[  0.973] recv SETTINGS frame <length=0, flags=0x01, stream_id=0>
          ; ACK
          (niv=0)
```

- STREAM 우선순위 지정은 클라이언트에서 스트림 ID의 의존성 여부와, 가중치를 설정하여 우선 처리가 필요한 스트림ID를 서버에게 전달합니다.
- 우선순위로 인해 Latency가 발생되지 않도록 스트림의 우선순위는 무시될 수 있습니다.

HTTP/2.0 특징

Flow control

Flow Control

- WINDOW_UPDATE 프레임 패킷
  - WINDOW_UPDATE 프레임은 데이터를 한번에 수용할 수 있는 피어의 Capability를 확인합니다.
  - WINDOW SIZE는 서버가 DATA 프레임을 전송할 때 마다 감소하며, 클라이언트가 WINDOW_UPDATE(window_size_increment) 송신할 때 마다 최대 WINDOW_SIZE(2^31-1byte)을 유지합니다.
  - WINDOW_SIZE로 인해 CONTROL 프레임이 거부되지 않아야합니다. (DATA 프레임에만 Flow Control 적용)

```
[  0.299] Connected
The negotiated protocol: h2
[  0.605] recv SETTINGS frame <length=6, flags=0x00, stream_id=0>
          (niv=1)
          [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]
[  0.605] recv WINDOW_UPDATE frame <length=4, flags=0x00, stream_id=0>
          (window_size_increment=2147418112)
[  0.605] send SETTINGS frame <length=12, flags=0x00, stream_id=0>
          (niv=2)
          [SETTINGS_MAX_CONCURRENT_STREAMS(0x03):100]
          [SETTINGS_INITIAL_WINDOW_SIZE(0x04):65535]
[  0.605] send SETTINGS frame <length=0, flags=0x01, stream_id=0>
          ; ACK
          (niv=0)
```

- DEFAULT WINDOW_SIZE(SETTINGS_INITIAL_WINDOW_SIZE): 65536
- Flow Contorl 비 활성화 불가

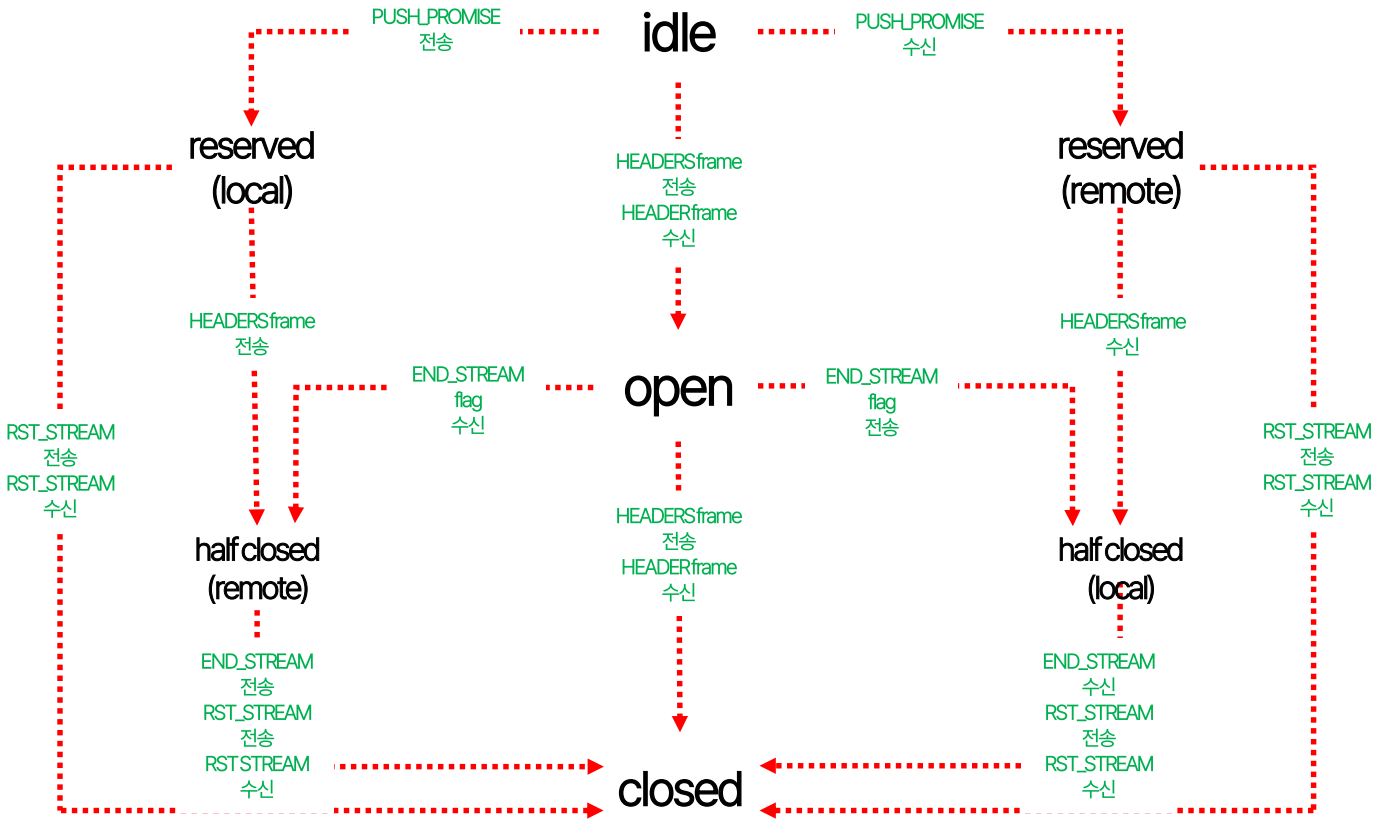
HTTP/2.0 특징

Stream Life Cycle

STREAM Status

- 상태 변화

  - 모든 스트림의 시작은 " idle " 상태에서 시작하고, 송/수신되는 프레임에 따라서 상태 변화가 이루어집니다. 변경된 상태에 따라서 수신 받을 수 있는 프레임 또한 정해져 있습니다. (스트림 상태에 따라서 다른 프레임 송/수신 받을 시 에러 발생)
  - HEADERS 프레임으로 스트림이 " open " 되고, 마지막 프레임 전송을 의미하는 END_STREAM flag 를 송신하면 " half closed " 상태가 됩니다.
    이후 피어에게 END_STREAM flag를 수신 받게 되면 스트림은 " closed " 됩니다.

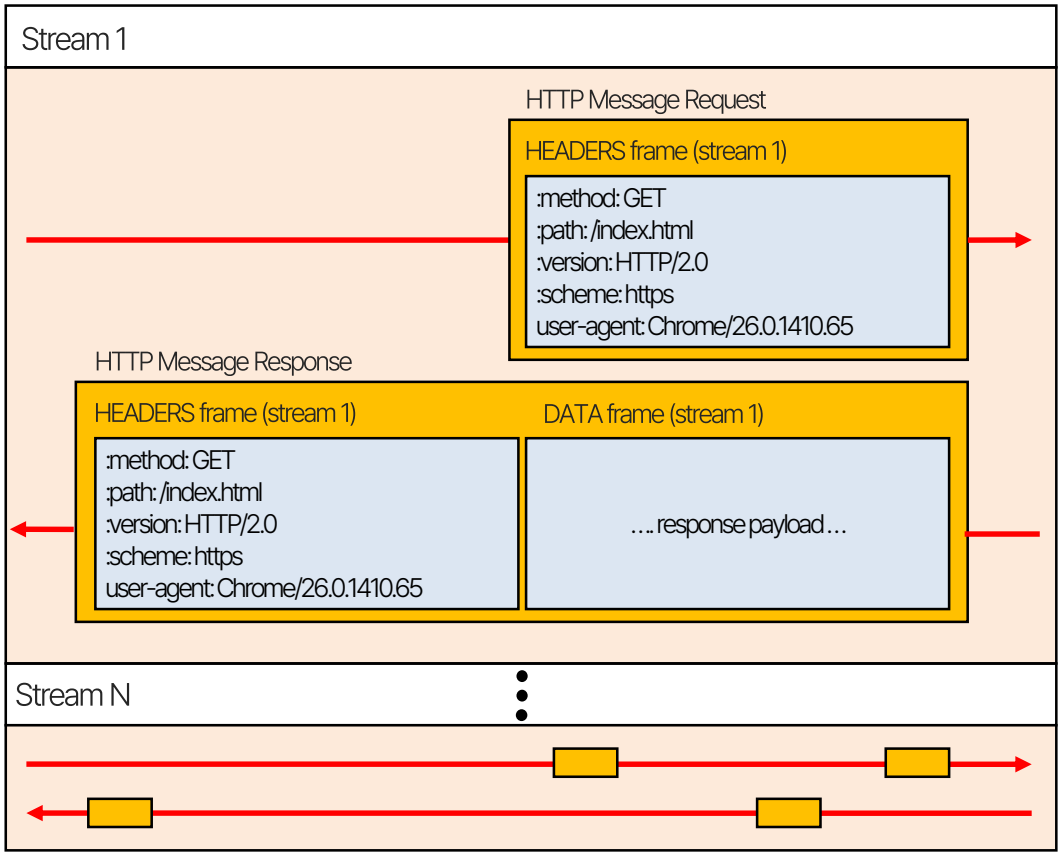


HTTP/2.0 특징

HTTP Frames

FRAMES

- FRAMES 전송 방식
  - GET Method HTTP 메시지는 하나의 HADERS 프레임과 0개 이상의 CONTITNUATION 프레임을 전송하며, 응답으로 HEADERS 프레임과 DATA 프레임을 전송 받습니다.
  - POST Method HTTP 메시지는 하나의 HEADERS 프레임과 0개 이상의 CONTINUATION 프레임, 하나 이상의 DATA 프레임을 전송합니다.
  - 전송되는 FRAME은 STREAM 생성 시 설정한 FRAME SIZE 보다 큰 경우 조각화해서 전송합니다.

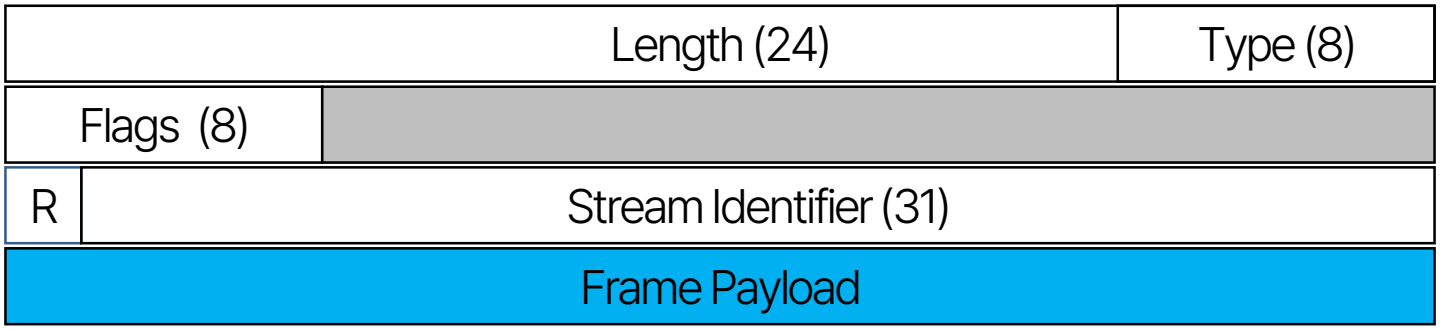


HTTP/2.0 특징

HTTP Frames

Frame Format

- FRAMES Format 구조



- Length: 24bytes
  - 프레임 페이로드 최대 길이 지정 *default 2^14(16,384)
- Type: 8byte
  - 프레임 종류
- Flags: 8byte
  - 프레임 타입 별로 피어에게 특정 상태 정보를 전달하기 위해 사용 ex) END_STREAM
- Stream ID: 31bits
  - Stream ID가 0인 경우, 커넥션과 연관된 프레임 ID ex) SETTINGS..
  - Client 스트림 ID 홀수 단위
  - Server 스트림 ID 짝수 단위 ex) PUSH_PROMISE..
- Frame Payload
  - 데이터 전송

HTTP/2.0 특징

HTTP Frames

Frames Type

- Type Description

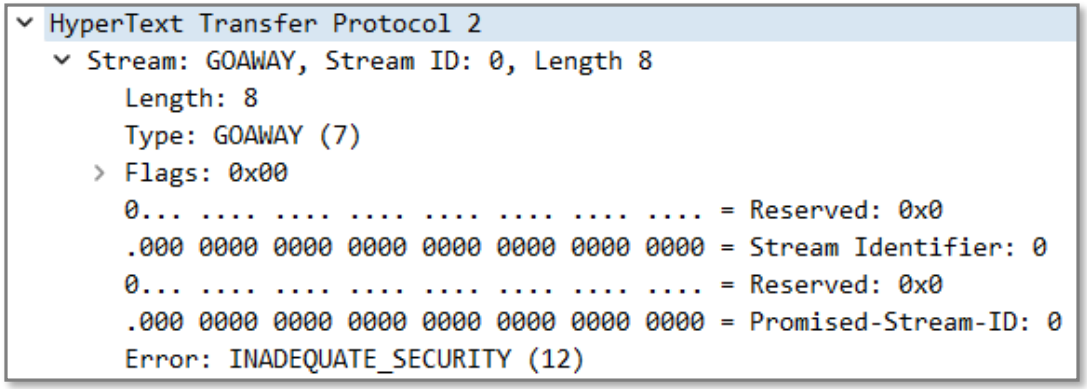
Type	Description
DATA(0)	• 데이터 전송 • 전송되는 프레임이 마지막 프레임인 경우 END_STREAM Flag 설정
HEADERS(1)	• 스트림 연결 시 사용 • END_STREAM flag가 없는 경우 반드시 동일한 스트림에 대한 CONTIUATION 프레임 전송 • 다른 유형의 프레임을 수신 시 PROTOCOL_ERROR 발생
Priority(2)	• 스트림에 대한 우선순위 지정
RST_STREAM(3)	• 스트림 정상 종료 또는 에러가 발생 시 에러코드와 함께 스트림 종료
SETTINGS(4)	• 커넥션 관련 설정 시 사용 • 스트림 ID=0이 아닌 경우 PROTOCOL_ERROR • SETTINGS의 응답으로 ACK Flag 사용
PUSH_PROMISE(5)	• 페이지 렌더링 후 필요한 리소스를 사전에 전송하기 위해 사용 (비 활성화 가능)
PING(6)	• RTT 를 계산 및 liveness 체크
GOAWAY(7)	• 커넥션 정상 종료 및 에러 발생 시 에러코드와 함께 커넥션 종료
WINDOW_UPDATE(8)	• Flow Control 사용
CONTINUATION(9)	• 헤더 블록이 조각화된 경우 사용

HTTP/2.0 특징

Frames Definition

GOWAY Frame

- GOWAY 프레임은 정상 종료 또는 에러가 발생한 경우, 커넥션을 종료하기 위해 사용됩니다.
  - Receiver가 처리했거나 처리 가능한 마지막 스트림 ID 정보를 포함하며, 에러 코드를 통해 커넥션 종료에 대한 확인 가능.



RST_STREAM

- RST_STREAM은 스트림에 대한 종료 및 에러가 발생한 경우를 의미합니다.

HTTP/2.0 특징

Frames Definition

ERROR 코드

- GOWAY / RST_STREAM 에러 코드

Error code	Description
NO_ERROR	• 커넥션 정상 종료
PROTOCOL_ERROR	• 불특정 프로토콜 감지
INTERNAL_ERROR	• 예상하지 못한 내부 오류
SETTINGS_TIMEOUT	• SETTINGS 프레임에 대한 ACK 못 받음
STREAM_CLOSED	• half-closed 상태에서 프레임을 수신하여 정상 종료
FRAME_SIZE_ERROR	• 잘못된 FRAME 사이즈 수신
REFUSED_STREAM	• 애플리케이션 처리 전에 스트림 거부
CANCEL	• 스트림 필요하지 않을 시 피어에게 알림
COMPRESSION_ERROR	• 헤더 압축 유지 불가
CONNECT_ERROR	• 비정상 종료
ENHANCE_YOUR_CALM	• 과도한 로드 요구
INADEQUATE_SECURITY	• 최소 보안 요구사항 미 충족
HTTP_1_1_REQUIRED	• HTTP1/1 사용 필요

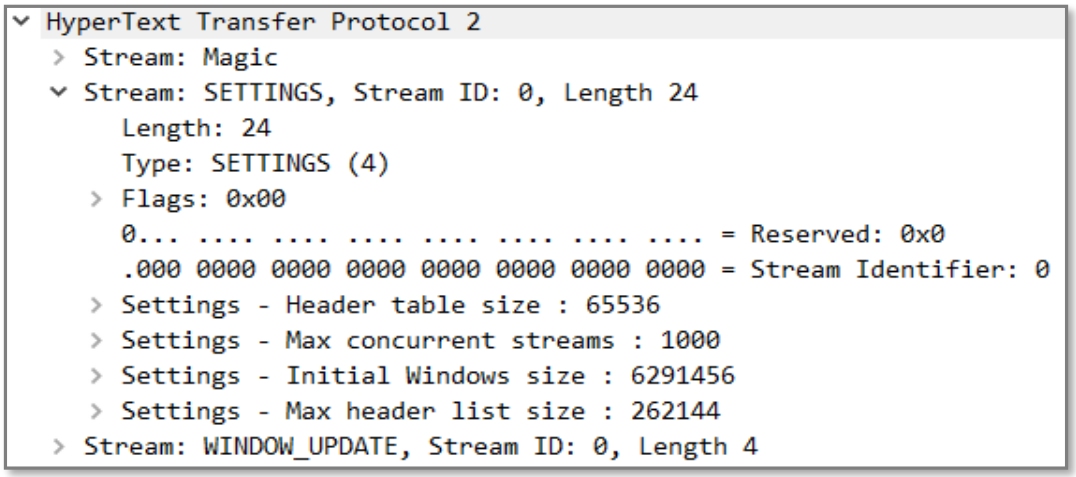
HTTP/2.0 특징

Frames Definition

SETTINGS 프레임

- 커넥션 관련 설정으로 HTTP2 통신을 위한 파라미터를 전달합니다.
  - Header table size
    - 공유할 헤더 테이블의 사이즈 정의
  - Max concurrent streams
    - 스트림 생성 최대 개수
  - Initial Windows size
    - WINDOW SIZE default
  - Max header list size
    - 최대 헤더 리스트 사이즈

(SETTINGS 프레임 앞에 Magic 은 더미이며, 의미 없는 값)

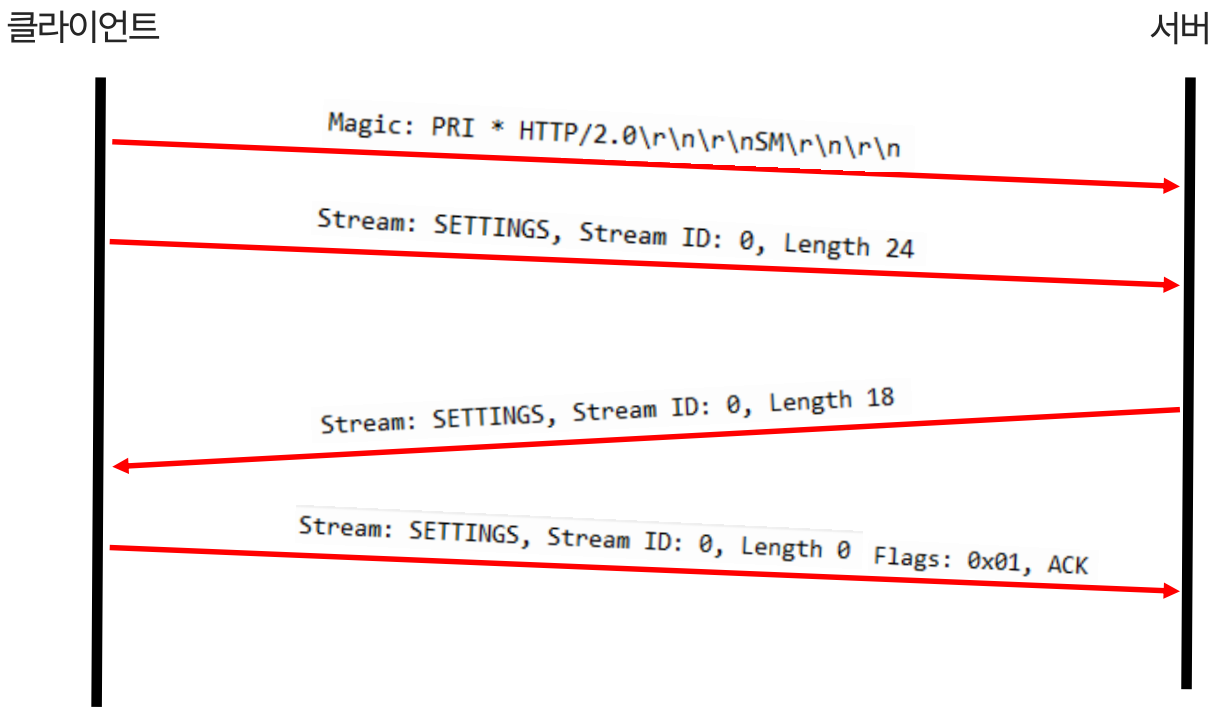


HTTP/2.0 특징

Frames Definition

SETTINGS 프레임

- SETTINGS 프레임 플로우
  - 클라이언트는 HTTP2 통신을 위해 더미 스트림(Magic) 전송 후 SETTINGS 프레임을 서버와 교환 후 동기화합니다.
  - 서버는 자신의 SETTINGS 프레임을 전송 후 클라이언트의 SETTINGS 프레임을 적용 및 동기화 처리 후 SETTINGS 프레임 Flag ACK 로 응답합니다.

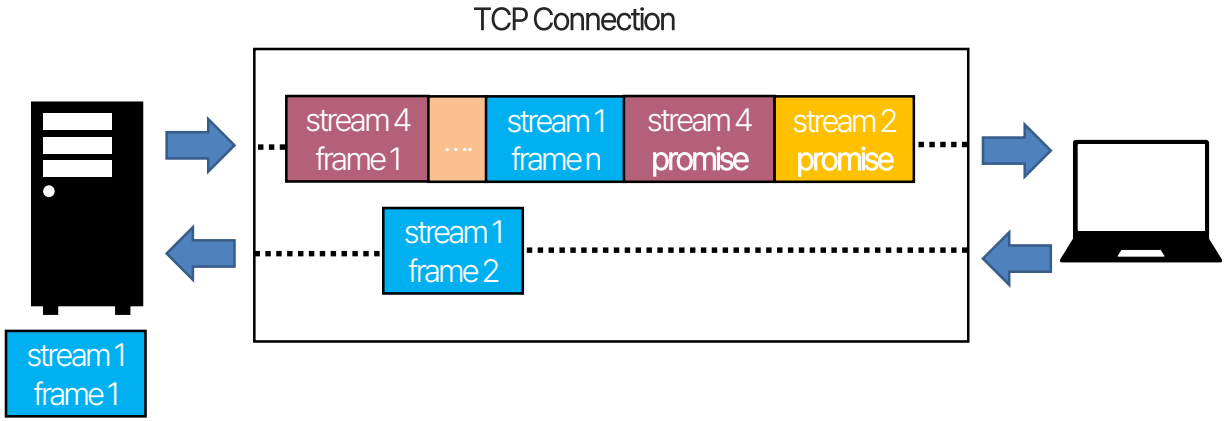


HTTP/2.0 특징

PUSH_PROMISE

PUSH_PROMISE 프레임

- PUSH_PROMISE 프레임은 SERVER_PUSH 프레임을 보내기 이전 서버가 클라이언트에게 어떤 스트림 ID로 SERVER_PUSH를 할지를 클라이언트에게 미리 Noti 하고, 스트림ID를 예약합니다.



HTTP/2.0 특징

PUSH_PROMISE

PUSH_PROMISE 프레임

- PUSH_PROMISE 전송
  - 서버가 클라이언트가 요청한 페이지에 렌더링 후 필요한 리소스를 미리 전달해주기 위해서 PUSH_PROMISE 프레임으로 스트림 ID=2, 4에 대한 정보를 클라이언트에게 Noti 합니다.

```
[  0.973] recv SETTINGS frame <length=0, flags=0x01, stream_id=0>
          ; ACK
          (niv=0)
[  0.975] recv (stream_id=13) :scheme: https
[  0.975] recv (stream_id=13) :authority: www.tunetheweb.com
[  0.975] recv (stream_id=13) :path: /assets/css/common.css
[  0.975] recv (stream_id=13) :method: GET
[  0.975] recv (stream_id=13) accept: */*
[  0.975] recv (stream_id=13) accept-encoding: gzip, deflate
[  0.975] recv (stream_id=13) user-agent: nghttp2/1.31.1
[  0.975] recv (stream_id=13) host: www.tunetheweb.com
[  0.975] recv PUSH_PROMISE frame <length=73, flags=0x04, stream_id=13>
          ; END_HEADERS
          (padlen=0, promised_stream_id=2)
[  0.975] recv (stream_id=13) :scheme: https
[  0.975] recv (stream_id=13) :authority: www.tunetheweb.com
[  0.975] recv (stream_id=13) :path: /assets/js/common.js
[  0.975] recv (stream_id=13) :method: GET
[  0.975] recv (stream_id=13) accept: */*
[  0.975] recv (stream_id=13) accept-encoding: gzip, deflate
[  0.975] recv (stream_id=13) user-agent: nghttp2/1.31.1
[  0.975] recv (stream_id=13) host: www.tunetheweb.com
[  0.975] recv PUSH_PROMISE frame <length=27, flags=0x04, stream_id=13>
          ; END_HEADERS
          (padlen=0, promised_stream_id=4)
```

HTTP/2.0 특징

SERVER_PUSH

SERVER_PUSH 프레임

- PUSH PROMISE 프레임으로 Noti 후 클라이언트에게 필요한 리소스(js,css,jpg 등)를 별도 클라이언트의 요청 없이 서버에서 미리 전달합니다.

```
[  0.976] recv DATA frame <length=1132, flags=0x01, stream_id=13>
          ; END_STREAM
[  0.979] recv (stream_id=2) :status: 200
[  0.979] recv (stream_id=2) date: Fri, 17 Dec 2021 09:24:55 GMT
[  0.979] recv (stream_id=2) server: Apache
[  0.979] recv (stream_id=2) expect-ct: max-age=60, report-uri="http://re
[  0.979] recv (stream_id=2) strict-transport-security: max-age=31536000;
[  0.979] recv (stream_id=2) x-frame-options: DENY
[  0.979] recv (stream_id=2) last-modified: Sun, 14 Mar 2021 08:17:46 GMT
[  0.979] recv (stream_id=2) accept-ranges: bytes
[  0.979] recv (stream_id=2) cache-control: max-age=10800, public
[  0.979] recv (stream_id=2) expires: Fri, 17 Dec 2021 12:24:55 GMT
[  0.979] recv (stream_id=2) vary: Accept-Encoding
[  0.979] recv (stream_id=2) content-encoding: gzip
[  0.979] recv (stream_id=2) x-content-type-options: nosniff
[  0.979] recv (stream_id=2) referrer-policy: no-referrer-when-downgrade
[  0.979] recv (stream_id=2) content-length: 5903
[  0.979] recv (stream_id=2) content-type: text/css; charset=utf-8
[  0.979] recv HEADERS frame <length=62, flags=0x04, stream_id=2>
          ; END_HEADERS
          (padlen=0)
          ; First push response header
```

```
[  0.979] recv (stream_id=4) :status: 200
[  0.979] recv (stream_id=4) date: Fri, 17 Dec 2021 09:24:55 GMT
[  0.979] recv (stream_id=4) server: Apache
[  0.979] recv (stream_id=4) expect-ct: max-age=60, report-uri="http://report.tun
[  0.979] recv (stream_id=4) strict-transport-security: max-age=31536000; include
[  0.979] recv (stream_id=4) x-frame-options: DENY
[  0.979] recv (stream_id=4) last-modified: Mon, 08 Feb 2021 07:45:17 GMT
[  0.979] recv (stream_id=4) accept-ranges: bytes
[  0.979] recv (stream_id=4) cache-control: max-age=10800, public
[  0.979] recv (stream_id=4) expires: Fri, 17 Dec 2021 12:24:55 GMT
[  0.979] recv (stream_id=4) vary: Accept-Encoding
[  0.979] recv (stream_id=4) content-encoding: gzip
[  0.979] recv (stream_id=4) x-content-type-options: nosniff
[  0.979] recv (stream_id=4) referrer-policy: no-referrer-when-downgrade
[  0.979] recv (stream_id=4) content-length: 1313
[  0.979] recv (stream_id=4) content-type: application/javascript; charset=utf-8
[  0.979] recv HEADERS frame <length=71, flags=0x04, stream_id=4>
          ; END_HEADERS
          (padlen=0)
```

- 스트림 id=2, 4 에서 SERVER_PUSH 프레임으로 서버에게 프레임을 수신 받은 패킷

HTTP/2.0 특징

Header

Pseudo-Header

- HTTP/2.0 에서 반드시 사용되는 헤더
  - 헤더 앞에 " : " 를 사용해서 일반 헤더 필드와 구분 필요.
  - pseudo-header는 사전에 정의되어 있는 헤더 외로 사용할 수 없습니다.
    - Request
      - :method:
      - :scheme:
      - :authority:
      - :path:
    - Response
      - :status:

HTTP/1.1 Request	HTTP/2 Request
GET /index.html HTTP/1.1 Host: www.site.com Referer:https://www.site.com/ Accept-Encoding:gzip	:method: GET :scheme: https :host: www.site.com :path: /index.html referer: https://www.site.com/ accept-encoding: gzip

Normal-Header

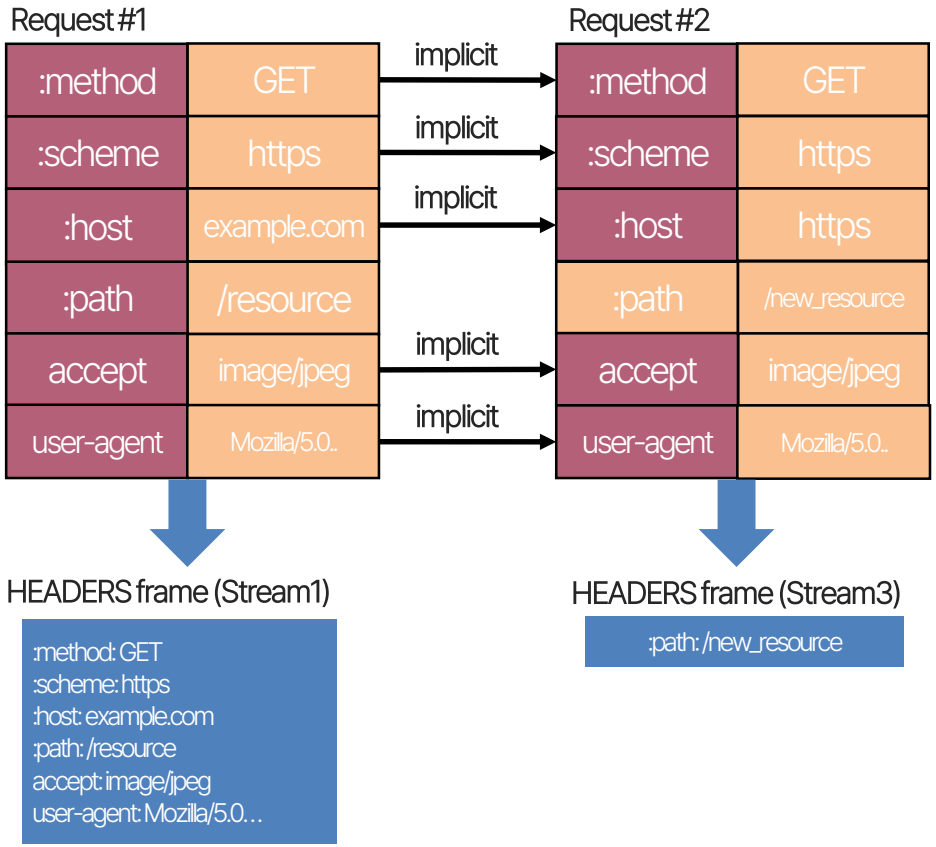
- pseudo-header 외 모든 헤더 필드는 Normal-Header (HTTP/1.1 동일)

HTTP/2.0 특징

Header Table

헤더 테이블 관리

- 내용
  - Dynamic table의 사이즈는 HTTP2 연결 후 SETTINGS 프레임 교환 단계에서 정의하며, 헤더를 인코딩하여 생성하고 교체 또는 제거합니다.
  - HEADER Table 은 피어 간 공유하고 관리하며 전송되는 헤더 필드는 인덱스 번호를 통해서 데이터를 줄일 수 있습니다.



HTTP/2.0 특징

Header Table

Static Table

- Static Table 목록
  - 자주 사용되는 헤더 필드(NAME/VALUE) 를 사전에 인덱스 값으로 지정해 놓은 테이블이며, 61개의 엔트리를 갖고 있습니다.

Index	Header Name	Header Value
1	:authority	
2	:method	GET
3	:method	POST
4	:path	/
5	:path	/index.html
6	:scheme	http
7	:scheme	https
8	:status	200
9	:status	204
10	:status	206
11	:status	304


```
▾ Header: :authority: 211.10.10.100
    Name Length: 10
    Name: :authority
    Value Length: 13
    Value: 211.10.10.100
    :authority: 211.10.10.100
    [Unescaped: 211.10.10.100]
    Representation: Literal Header F
    Index: 1
```

Dynamic Table

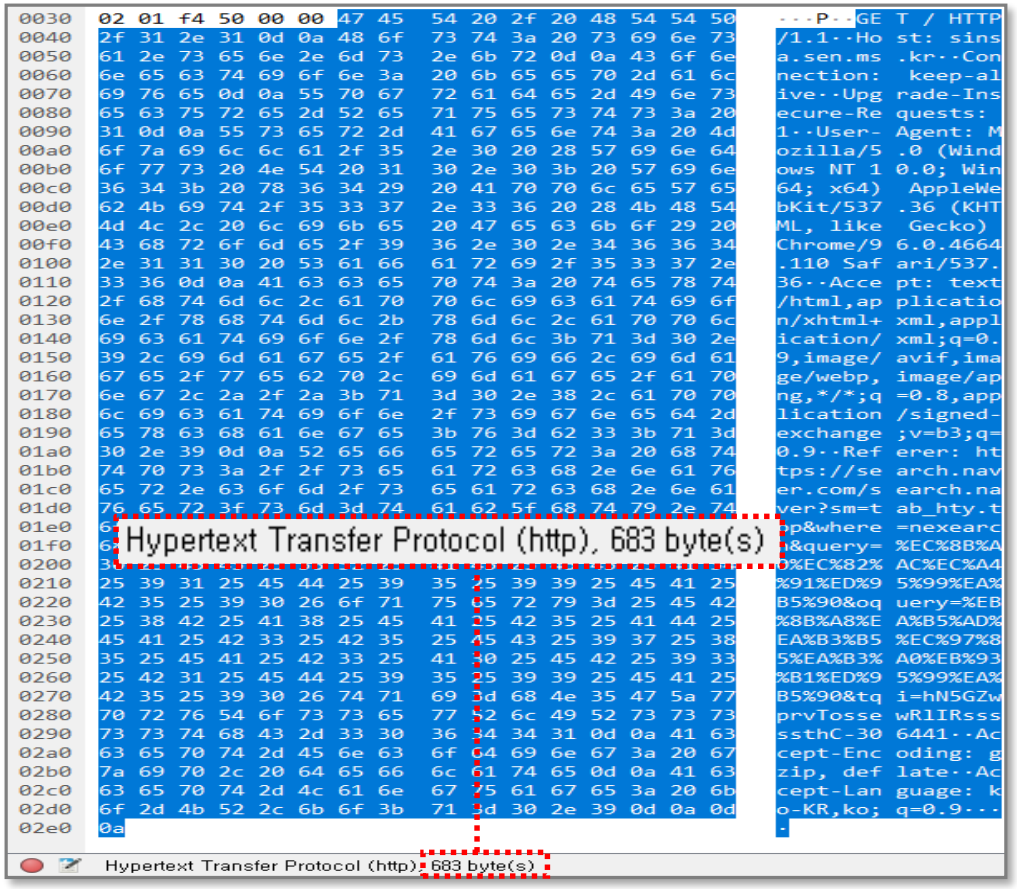
- Static Table 외 동적으로 발생하는 헤더를 관리
  - Static Table에 있는 헤더/값 외 헤더는 Dynamic 테이블에서 관리하며, 헤더필드는 Static Huffman 으로 인코딩 합니다.
  - SPDY에서 deflate으로 압축 했지만, CRIME 취약점으로 HTTP2 부터 Static Huffman 인코딩 방식으로 변경.

HTTP/2.0 특징

Header Compression

HTTP/1.1 HEADER

- HTTP/1.1 프로토콜에서 일반 텍스트의 헤더는 요청 당 평균 500-800 바이트의 오버헤드가 있고, 쿠키를 사용하게 되면 더 많은 오버헤드가 발생합니다.
- 웹 페이지의 발전으로 요구되는 헤더의 개수가 증가하고, 중복 헤더들로 인해 불 필요하게 대역폭 소모를 합니다.



- HTTP/1.1 GET 요청 패킷 사이즈

HTTP/2.0 특징

Header Compression

HTTP/2.0 HPACK

- HEADER COMPRESSION
  - 기존 NAME=VALUE 형태의 헤더필드 모음을 HPACK 알고리즘으로 압축한 뒤 피어에게 전송하며, 이러한 헤더 블록은 조각으로 하나 이상의 옥텟 시퀀스로 분할되어 전송 됩니다.

- HEADER DECOMPRESSION
  - 위와 같은 내용 때문에 헤더 프레임 전송은 순차적으로 전송 되어야하며, Receiver는 조각화된 헤더 블록을 재 조합 후 디코딩하여 헤더 리스트를 복원합니다.

HTTP/2.0 특징

Header Compression

HTTP/1.1

- HEADER 전송
  - 압축없이 ASCII 코드 형태로 전송되어 많은 바이트를 사용

No.	Time	Source	Destination	Protocol	Length	Info
37	2.494529	100.100.100.100	211.10.10.101	HTTP	685	GET /index_pask.jpg HTTP/1.1

```
Hypertext Transfer Protocol
  GET /index_pask.jpg HTTP/1.1\r\n
  Host: 211.10.10.101\r\n
  Connection: keep-alive\r\n
  sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96", "Google Chrome";v="96"\r\n
  sec-ch-ua-mobile: ?0\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Saf
```

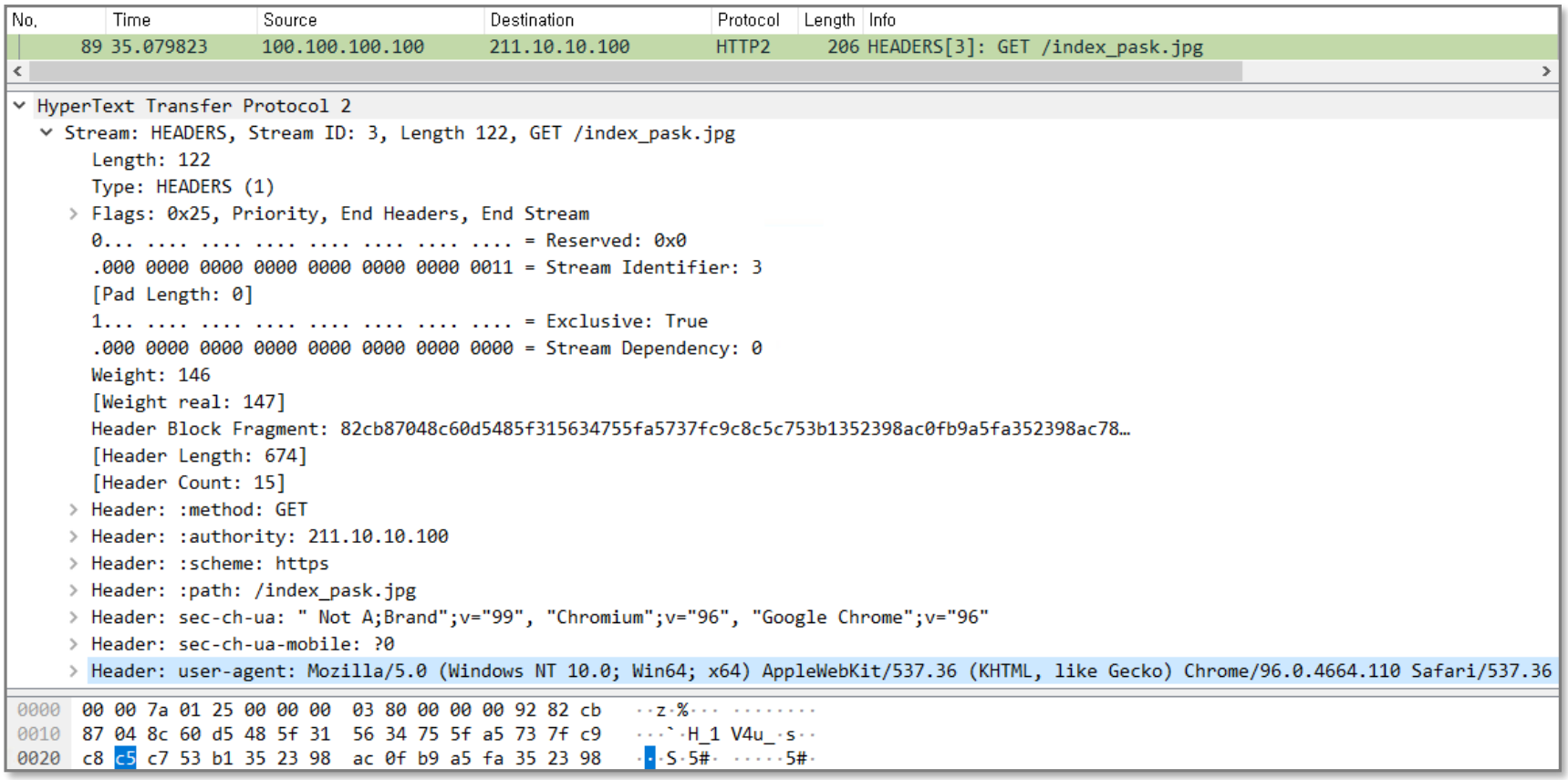
```
0090  76 3d 22 39 36 22 0d 0a  73 65 63 2d 63 68 2d 75   v="96"·· sec-ch-u
00a0  61 2d 6d 6f 62 69 6c 65  3a 20 3f 30 0d 0a 55 73   a-mobile : ?0··Us
00b0  65 72 2d 41 67 65 6e 74  3a 20 4d 6f 7a 69 6c 6c   er-Agent : Mozill
00c0  61 2f 35 2e 30 20 28 57  69 6e 64 6f 77 73 20 4e   a/5.0 (W indows N
00d0  54 20 31 30 2e 30 3b 20  57 69 6e 36 34 3b 20 78   T 10.0;  Win64; x
00e0  36 34 29 20 41 70 70 6c  65 57 65 62 4b 69 74 2f   64) Appl eWebKit/
00f0  35 33 37 2e 33 36 20 28  4b 48 54 4d 4c 2c 20 6c   537.36 ( KHTML, l
0100  69 6b 65 20 47 65 63 6b  6f 29 20 43 68 72 6f 6d   ike Geck o) Chrom
0110  65 2f 39 36 2e 30 2e 34  36 36 34 2e 31 31 30 20   e/96.0.4 664.110
0120  53 61 66 61 72 69 2f 35  33 37 2e 33 36 0d 0a 73   Safari/5 37.36··s
```

HTTP/2.0 특징

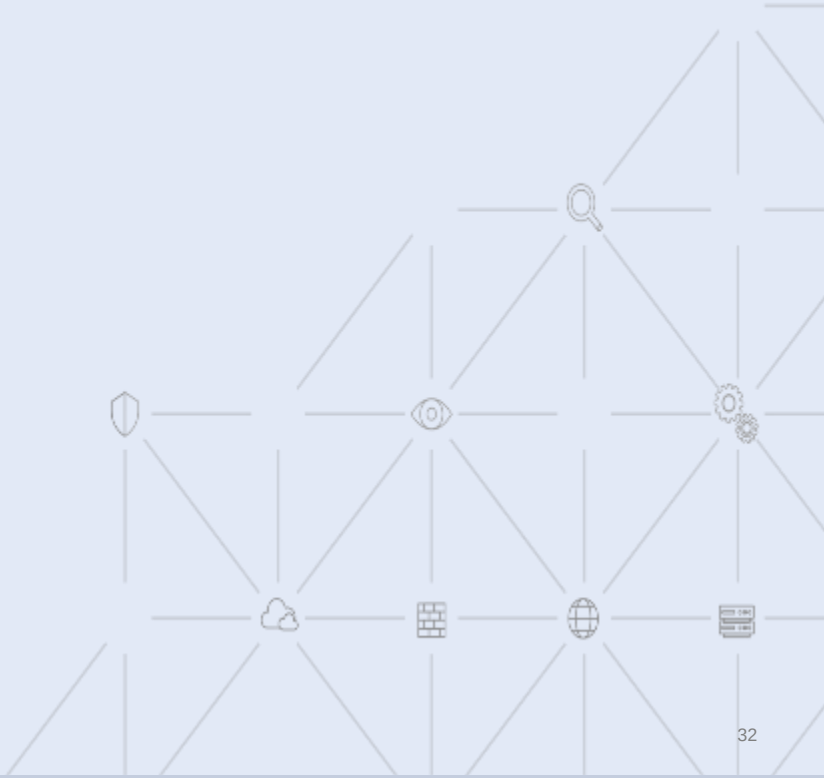
Header Compression

HTTP/2.0 압축

- HEADER COMPRESSION
  - user-agent 헤더는 압축 전송되어 1byte 인덱스 번호로 전송.



PAS-K 설정



PAS-K 설정

HTTP2.0

HTTP2 적용

- 주요 설정 정보
  - HTTP2 통신 활성화를 위해 advl7slb 기능으로 풀 프록시 설정이 필요합니다.
  - HTTP2 통신은 클라이언트 사이드에서 동작합니다.

■ RULE 생성

```
RULE:1
------------------------------------
 ID              :1
 Status          :enable
 Priority        :0
 Pattern         :
 Action          :group
 Error Action    :default
 Fail-Skip       :disable
 Snat-Addr-Type  :none
 Icap            :
 Urlmanip        :
 Headermanip     :
 Resource Cloaking   :
 Response Buffering  :
 Sure Connect    :
 Cache           :
 Compression     :
 Backend Timeout     :
 Send Complete Request   :disable
 Wait Complete Response  :disable
 Preserve HTTP Version   :disable

 SSL-Decrypt-Mirroring
   Status        :service

 --- Option ----------------------------------
 Group           :g_http2
 B-Grp           :
 Req-Scheme      :http
 Adapt Location Scheme   :enable
============================
```

■ Group 생성

```
GROUP: g_http2
-------------------------------------------
-
  Name          :g_http2

  LB Method
    Lb-Method     :rr

  Connection Pooling
    Status        :enable
    Mode          :per-client
    Max Reuse     :0
    Pool Size     :2048
    Timeout       :0

  Persist
    Type          :none
    Timeout       :1:00

  Surge Protection    :

  Real            :11
```

■ SSL Profile 적용

```
==================================
ADVL7SLB: http2_piolink
-------------------------------------------------------------
--  Name            :http2_test
  Priority          :100
  Host              :
  Status            :enable
  Cache             :
  SSL/TLS           :1
```

감사합니다.

(주) 파이오링크

(본사) 서울시 금천구 가산디지털2로 98, IT캐슬 1동 401호
대표전화 02 2025 6900 | www.PIOLINK.com

(주) 파이오링크

(본사) 서울시 금천구 가산디지털2로 98, IT캐슬 1동 401호
대표전화 02 2025 6900 | www.PIOLINK.com