HTTP

HTTP는 전송 계층 위에 있는 애플리케이션 계층으로서 웹 서비스 통신에 사용된다.

HTTP/1.0부터 시작해서 발전을 거듭하여 지금은 HTTP/3 이다.

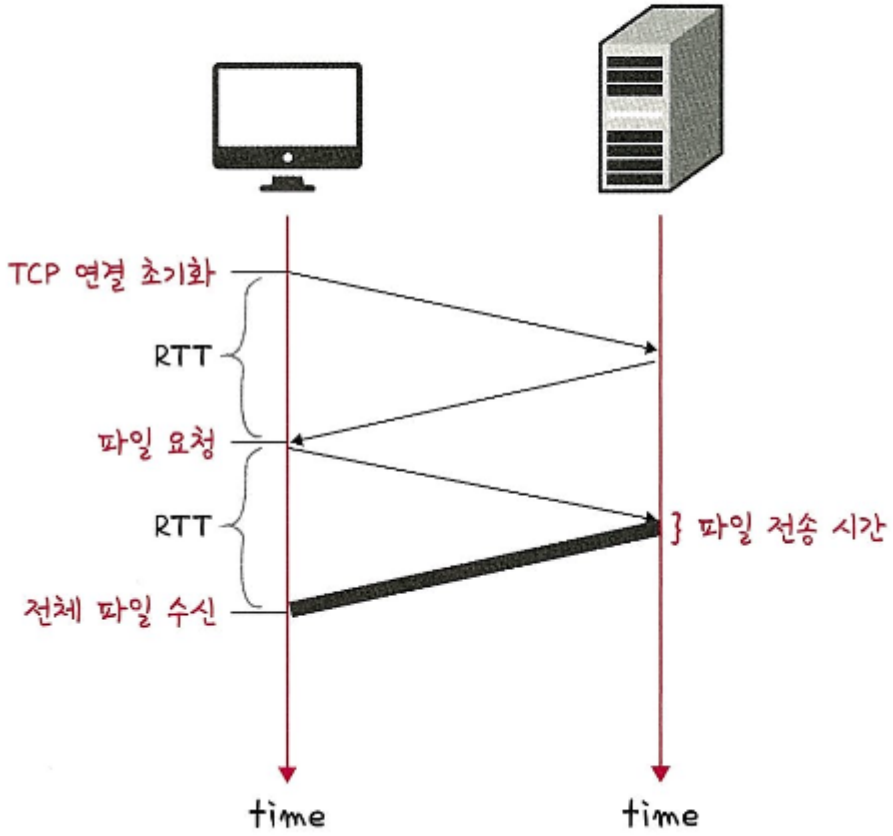
2.5.1 HTTP.1.0

- HTTP/1.0은 기본적으로 한 연결당 하나의 요청을 처리하도록 설계되었다.
- 이는 RTT 증가를 불러오게 됐다.

RTT 증가

RTT 증가

▼ 그림 2-54 RTT 증가



- 서버로부터 파일을 가져올때마다 TCP의 3-웨이 핸드셰이크를 계속해서 열어야하기 때문에 RTT가 증가하는 단점이 있다.

```
RTT : 패킷이 목적지에 도달하고 나서 다시 출발지로 돌아오기 까지 걸리는 시간이
패킷 왕복시간
```

RTT의 증가를 해결하기 위한 방법

- 매번 연결할 때마다 RTT가 증가하니 서버에 부담이 많이 가고 사용자 응답시간이 길어졌다.
- 이를 해결하기 위해 스플리팅, 코드 압축, 이미지 Base 64 인코딩을 사용하곤 했다.

이미지 스플리팅

- 많은 이미지를 다운로드 받게 되면 과부하가 걸리기 때문에 많은 이미지가 합쳐 있는 하나의 이미지를 다운로드 받고, 이를 기반으로 background-image의 position을 이용하여 이미지를 표기하는 방법이다.

코드 압축

- 코드 압축은 코드를 압축해서 개행 문자, 빈칸을 없애서 코드의 크기를 최소하하는 방법이다.

이미지 Base 64 인코딩

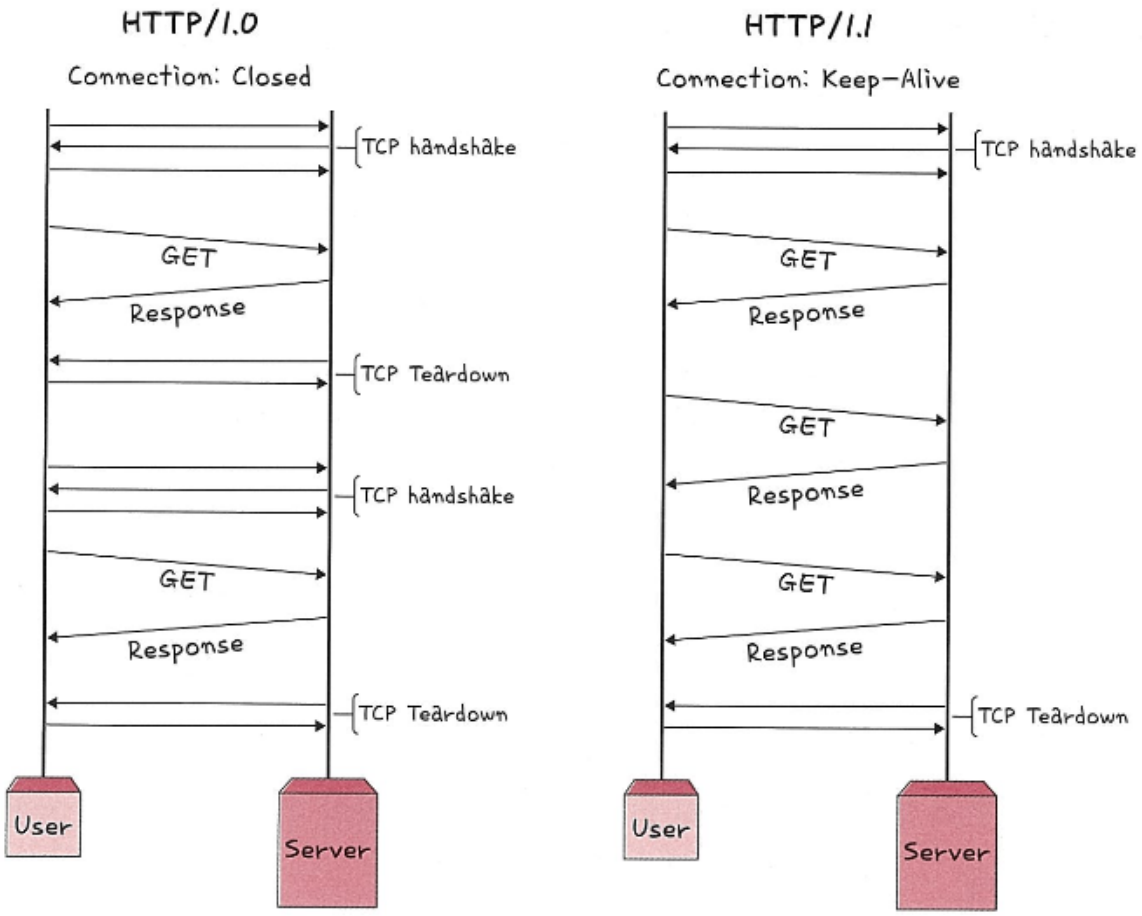
- 이미지 파일을 64진법으로 이루어진 문자열로 인코딩하는 방법이다.
- 이 방법을 사용하면 서버와의 연결을 열고 이미지에 대해 서버에 HTTP 요청을 할 필요가 없다는 장점이다.
- 하지만 Base 64 문자열로 변환할 경우 37%정도 크기가 더 커지는 단점이 있다.

```
인코딩
  - 정보의 형태나 형식을 표준화, 보안, 처리속도향상,
    저장 공간 절약 등을 위해 다른 형태나 형식으로 변환하는 처리 방식
```

2.4.2 HTTP/1.1

- HTTP/1.0 에서 발전한 것이 바로 HTTP/1.1 이다.
- 매번 TCP를 연결하는 것이 아니라 한 번 TCP 초기화를 한 이후에 keep-alive라는 옵션으로 여러 개의 파일을 송수신 할 수 있게 바뀌었다.
- 참고로 HTTP/1.0 에서도 keep-alive가 있었지만 표준화가 되어 있지 않았고 HTTP/1.1 부터 표준화가 되어 기본 옵션으로 설정되었다.

▼ 그림 2-55 HTTP/1.0과 HTTP/1.1의 비교

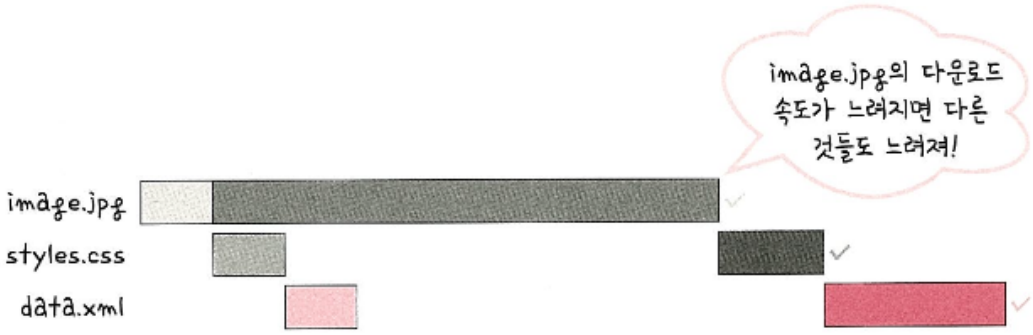


- 다음 그림처럼 한 번 TCP 3-웨이 핸드셰이크가 발생하면 그 다음부터 발생하지 않는 것을 볼 수 있다.
- 하지만 문서 안에 포함된 다수의 이미지(이미지, css, script)를 처리하려면 요청할 리소스 개수에 비례해서 대기 시간이 길어지는 단점이 있다.

HOL Blocking

- HOL Blocking(Head Of Line Blocking)은 네트워크에서 같은 큐에 있는 패킷이 그 첫 번째 패킷에 의해 지연될때 발생하는 성능 저하 현상을 말한다.

▼ 그림 2-56 HOL Blocking



- 예를 들어 앞의 그림처럼 image.jpg와 style.css, data.xml을 다운로드 받을 때 보통은 순차적으로 잘 받아지지만 image.jpg가 느리게 받아진다면 그 뒤에 있는 것들이 대기하게 되며 다운로드가 지연되는 상태가 된다.

무거운 헤더 구조

HTTP/1.1의 헤더에는 쿠키 등 많은 메타데이터가 들어 있고 압축이 되지 않아 무거웠다.

2.5.3 HTTP/2

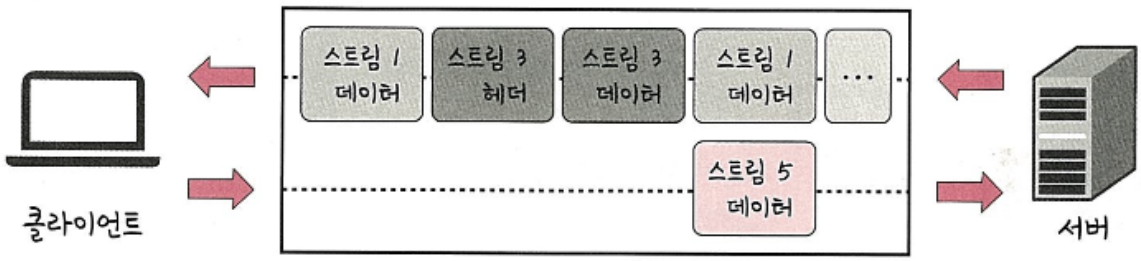
- HTTP/2 는 SPDY 프로토콜에서 http/1.x 버전보다 지연 시간을 줄이고 응답시간을 더 빠르게 할 수 있으며 멀티 플랙싱, 헤더 압축, 서버 푸시, 요청의 우선순위 처리를 지원하는 프로토콜이다.

멀티 플렉싱

- 멀티 플렉싱이란 여러 개의 스트림을 사용하여 송수신하다는 것이다,
- 이를 통해 특정 스트림의 패킷이 손실되었다고 하더라도 해당 스트림에만 영향을 미치고 나머지 스트림은 멀쩡하게 동작할 수 있다

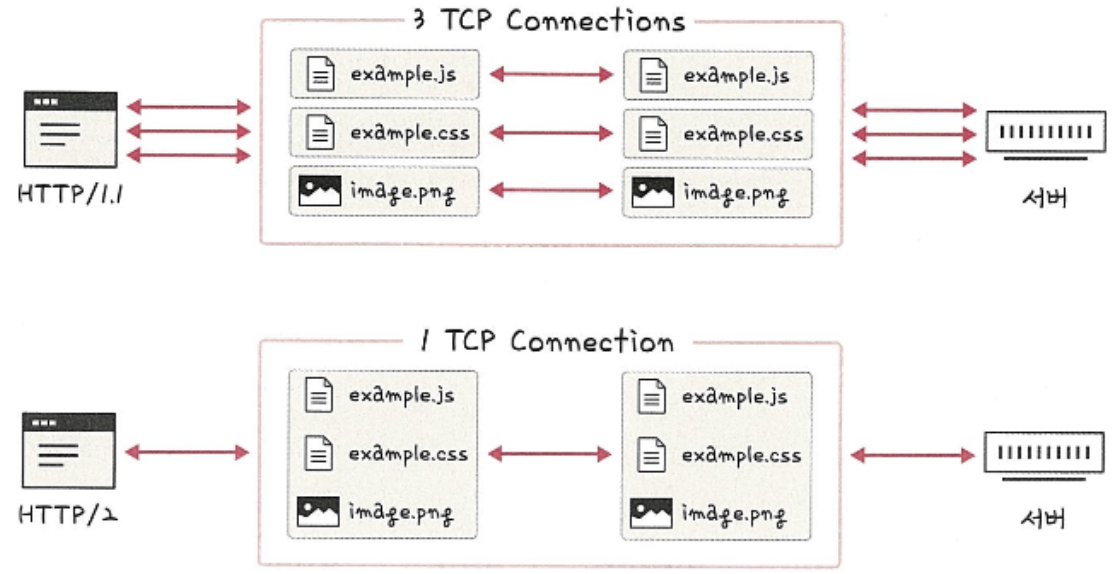
```
스트림(stream)
- 시간이 지남에 따라 사용할 수 있게 되는 일련의 데이터 요소를 가리키는 데이터 흐름
```

▼ 그림 2-57 멀티플렉싱



- 앞의 그름은 하나의 연결 내 여러 스트림을 캡처한 모습이다.
- 병렬적인 스트림들을 통해 데이터를 서빙하고 있다 또한, 스트림 내의 데이터들도 쪼개져 있다.
- 애플리케이션에서 받아온 메시지를 독립된 프레임으로 조각내어 서로 송수신한 이후 다시 조립하여 데이터를 주고 받는다.

▼ 그림 2-58 멀티플렉싱



- 이를 통해 단일 연결을 사용하여 병렬로 여러 요청을 받을 수 있고 응답을 줄 수 있다.
- 이렇게 되면 http/1.x에서 발생하는 문제인 HOL Blocking 문제를 해결할 수 있다.

헤더 압축

- HTTP/1.x 에는 크기가 큰 헤더라는 문제가 있다.

- 이를 HTTP/2에서는 헤더 압축을 써서 해결하는데, 허프만 코딩 압축 알고리즘을 사용하는 HPACK 압축 형식을 가진다.

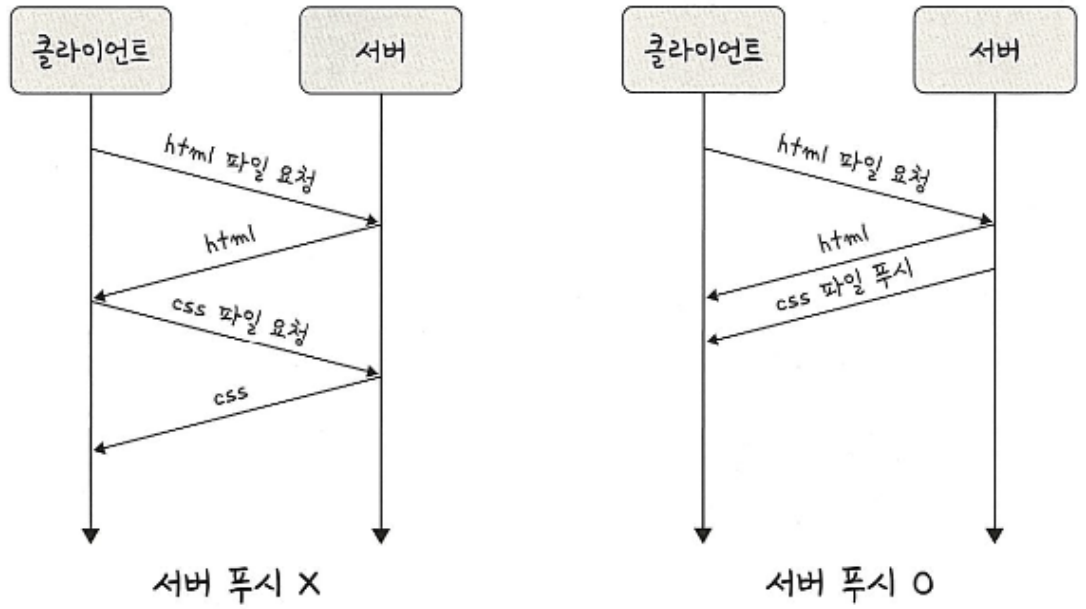
허프만 코딩

- 허프만 코딩은 문자열을 문자 단위로 쪼개 빈도수를 세어 빈도가 높은 정보는 적은 비트수를 사용하여 표현하고, 빈도가 낮은 정보는 비트 수를 많이 사용하여 표현해서 전체 데이터의 표현에 필요한 비트양을 줄이는 원리이다.

서버푸시

- HTTP/1.1 에서는 클라리언트가 서버에 요청을 해야 파일을 다운로드 받을 수 있었다면, HTTP/2는 클라이언트 요청 없이 서버가 바로 리소스를 푸시할 수 있다.

▼ 그림 2-60 서버 푸시



- html에는 css 나 js 파일이 포함되기 마련인데, html을 읽으면서 그 안에 들어 있던 css 파일을 서버에서 푸시하여 클라이언트에 먼저 줄 수 있다.

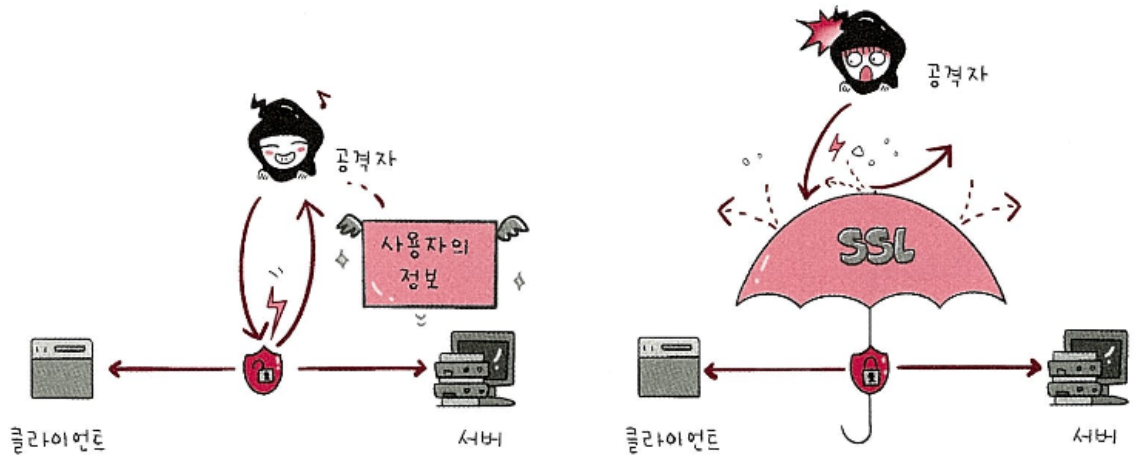
2.5.4 HTTPS

- HTTP/2는 HTTPS 위에서 동작한다. HTTPS는 애플리케이션 계층과 전송 계층 사이에 신뢰 계층인 SSL/TLS 계층을 넣은 신뢰할 수 있는HTTP 요청을 말한다.
- 이를 통해 통신을 암호화 한다.

SSLTLS

- SSL(Secure Socket Layer)은 SSL 1.0 부터 시작해서 2.0, 3.0 TLS(Transport Layer Security Protocol) 1.0 TLS 1.3 까지 버전이 올라가며 마지막으로 TLS로 명칭이 변경되었으나, 보통 이를 합쳐 SSL/TLS로 많이 부른다.
- SSL/TLS 은 전송 계층에서 보안을 제공하는 프로토콜이다.
- 클라이언트와 서버가 통신을 할때 SSL/TSL를 통해 제 3자가 메시지를 도청하거나 변조하지 못하도록 한다.

▼ 그림 2-61 SSL/TLS를 이용한 인터셉터 방지



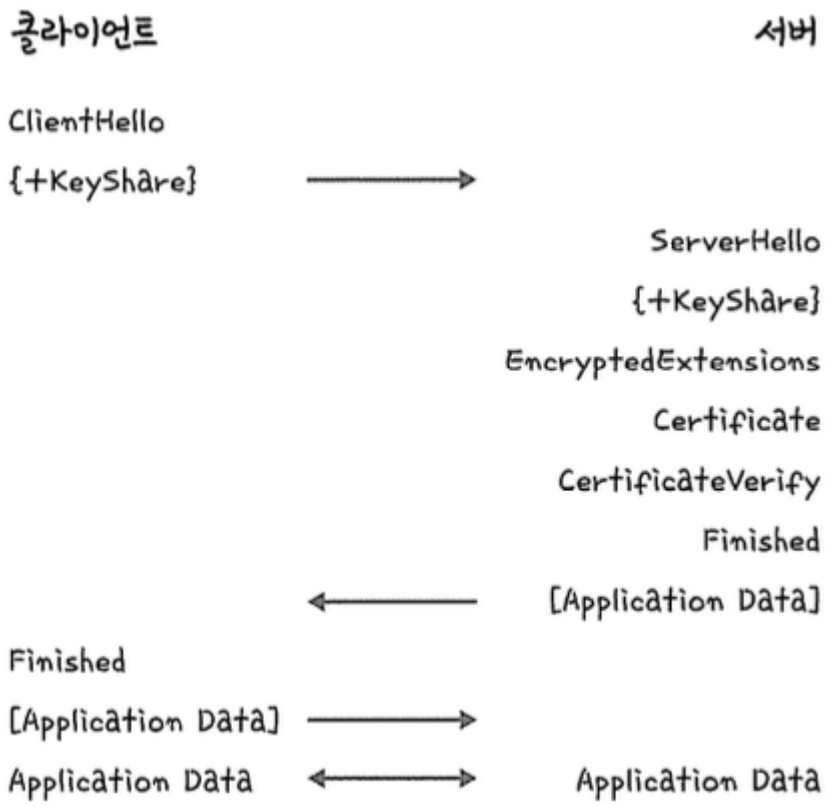
- SSL/TLS를 통해 공격자가 서버인 척 하면서 사용자 정보를 가로채는 네트워크 상의 인터셉터를 방지할 수 있다.
- SSL.TLS는 보안 세션을 기반으로 데이터를 암호화 하며 보안 세션이 만들어 질 때 인증 메커니즘, 키 교환 암호화 알고리즘, 해싱 알고리즘이 사용된다.

보안 세션

- 보안 세션이란 보안이 시작되고 끝나는 동안 유지되는 세션을 말하고, SSL/TLS는 핸드셰이크를 통해 보안 세션을 생성하고 이를 기반으로 상태 정보 등을 공유한다.

```
세션
  - 운영체제가 어떠한 사용자로부터 자신의 자산 이용을 허락하는 일정한 기간을 뜻한다.
  - 즉 사용자는 일정 시간동안 응용 프로그램, 자원 등을 사용할 수 있다.
```

▼ 그림 2-62 TLS의 핸드셰이크



- 클라이언트와 서버와 키를 공유하고 이를 기반으로 인증, 인증 확인 등의 작업이 일어나는 단 한번의 1-RTT가 생긴 후 데이터를 송수신 하는 것을 볼 수 있다.
- 클라이언트에서 사이퍼 슈트를 서버에 전달하면 서버는 받은 사이퍼 슈트의 암호화 알고리즘 리스트를 제공할 수 있는지 확인한다.
- 제공할 수 있다면 서버에서 클라이언트로 인증서를 보내는 인증 메커니즘이 시작되고 이후 해싱 알고리즘 등으로 암호화된 데이터의 송수신이 시작된다.

사이퍼 슈트

- 사이퍼 슈트는 프로토콜, AEAD 사이퍼 모드, 해싱 알고리즘이 나열된 규약을 말하며 다섯개가 있다.

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_CCM_SHA256
- TLS_AES_128_CCM_8_SHA256

AEAD 사이퍼 모드

- AEAD는 데이터 암호화 알고리즘이며 AES_128_GCM 등이 있다.

해싱 알고리즘

- 해싱 알고리즘은 데이터를 추정하기 힘든 더 작고, 섞여 있는 조각으로 만드는 알고리즘이다.
- SSL/TLS 는 해싱 알고리즘으로 SHA-256 알고리즘과 SHA-384 알고리즘을 쓴다.

SHA-256 알고리즘

- SHA-256 알고리즘은 해시 함수의 결괏값이 256비트인 알고리즘이며 비트 코인을 비롯한 많은 블록체인 시스템에서도 쓰인다.
- SHA-256 알고리즘은 해싱을 해야 할 메시지에 1을 추가하는 등 전처리를 하고 전처리된 메시지를 기반으로 해시를 반환한다.

```
해시
  - 다양한 길이를 가진 데이터를 고정된 길이를 가진 데이터로 패밍한 값
해싱
  - 임의의 데이터를 해시로 바꿔주는 일이며 해시 함수가 이를 담당
해시 함수
  - 임의의 데이터를 입력으로 받아 일정한 길이의 데이터로 바꿔주는 함수
```

SEO에도 도움이 되는 HTTPS

- 구글은 SSL 인증서를 강조해왔고 사이트 내 모든 요소가 동일하다면 HTTPS 서비스를 하는 사이트가 그렇지 않은 사이트보다 SEO 순위가 높을 것이라고 공식적으로 발표 했다.
- seo(Search Engine Optimaization)는 검색엔진 최적화를 뜻하며 사용자들이 구글, 네이버 같은 검색엔진으로 웹 사이트를 검색했을 때 그 결과를 페이지 상단에 노출시켜 많은 사람이 볼 수 있도록 최적화하는 방법을 의미한다.

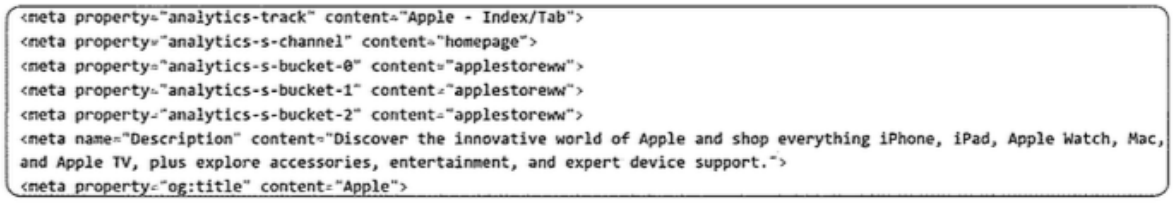
- 서비스를 운영한다면 SEO 관리는 필수이다. 이를 위한 방버으로 캐노니컬 설정, 메타 설정, 페이지 속도 개선, 사이트맵 관리 등이 있다.

캐노니컬 설정

```
<link rel="canonical" href="https://example.com/page2.php" />
```

메타 설정

html 파일의 가장 윗부분인 메타를 잘 성정해야한다.



- 최고의 웹 페이지라고 칭송받는 애플의 사이트는 앞의 그림처럼 메타를 설정 한다.

페이지 속도 개선

- 구글의 PageSpeedInsights로 가서 자신의 서비스에 대한 리포팅을 주기적으로 받으며 관리해야한다.

사이트맵 관리

- 사이트맵을 정기적으로 관리하는 것은 필수이다.
- 사이트맵 제너레이터를 사용하거나 직접 코드를 만들어 구축해도 된다.
- 사이트맵은 다음과같은 xml 파일을 말한다.

출력

```
<?xml version="1.0" encoding="utf-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
<url>
<loc>http://kundol.co.kr/</loc>
<lastmod>수정날짜</lastmod>
<changefreq>daily</changefreq>
<priority>1.1</priority>
</url>
</urlset>
```

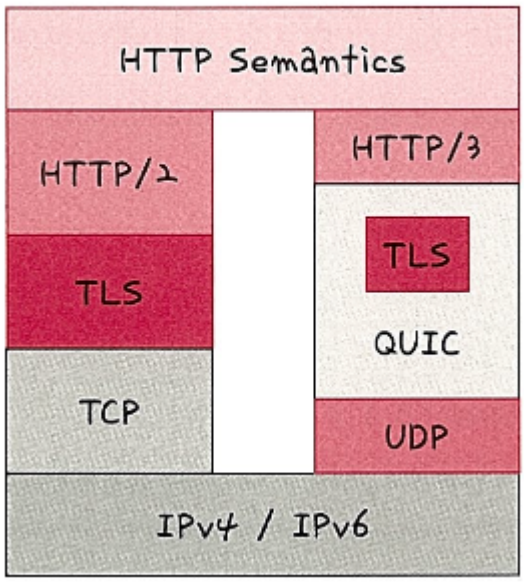
HTTPS 구축 방법

- HTTPS 구축 방법은 크게 세 가지이다. 직접 ca에서 구매한 인증키를 기반으로 서비스를 구축하거나, 서버 앞단의 HTTPS를 제공하는 로드밸런서를 두거나, 서버 앞단에 HTTPS를 제공하는 CDN을 둬서 구축한다.

2.5.5 HTTP

- TCP위에서 돌아가는 HTTP/2와 달리 HTTP/3는 QUIC이라는 계층 위에서 돌아가며, TCP 기반이 아닌 UDP기반으로 돌아간다.
-

▼ 그림 2-68 UDP 기반으로 돌아가는 HTTP/3

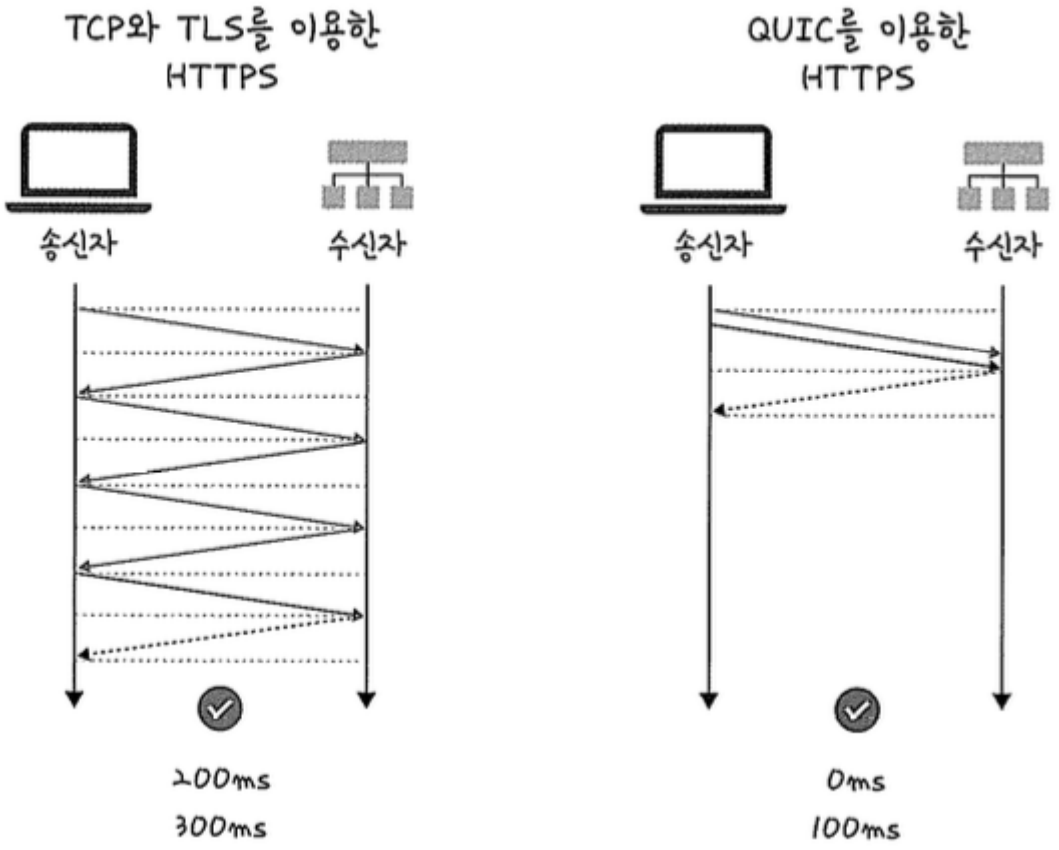


- 또한 http/2에서 장점이었던 멀티플렉싱을 가지고 있으며 초기 연결 설정 시 지연 시간 감소라는 장점이 있다

초기 연결 설정 시 지연 시간 감소

- QUIC는 TCP를 사용하지 않기 때문에 통신을 시작할때 번거러운 3-웨이 핸드셰이크 과정을 거치지 않아도 된다.

▼ 그림 2-69 RTT의 감소



- QUIC는 첫 연결 설정에 1-RTT만 소요 된다. 클라이언트가 서버에 어떤 신호를 한 번 주고, 서버도 거기에 응답하기만 하면 바로 본 통신을 시작할 수 있다.
- QUIC는 순반향 오류 수정 메커니즘(FEC)이 적용되었다.
- 이는 전송한 패킷이 손실되었다면 수신 측에서 에러를 검출하고 수정하는 방식이며 열악한 네트워크 환경에서도 낮은 패킷 손실률을 자랑한다.