HTTP

HTTP (Hypertext Transfer Protocol)

클라이언트와 서버 간 통신을 위한 통신 프토토콜

애플리케이션 계층으로써 웹 서비스 통신에 사용됨

HTML과 같은 하이퍼미디어 문서를 전송함

HTTP/1.0 부터 시작해서 현재는 HTTP/3 까지 발전을 거듭하였음

HTTP/1.0

기본적으로 한 연결당 하나의 요청을 처리하도록 설계되었음

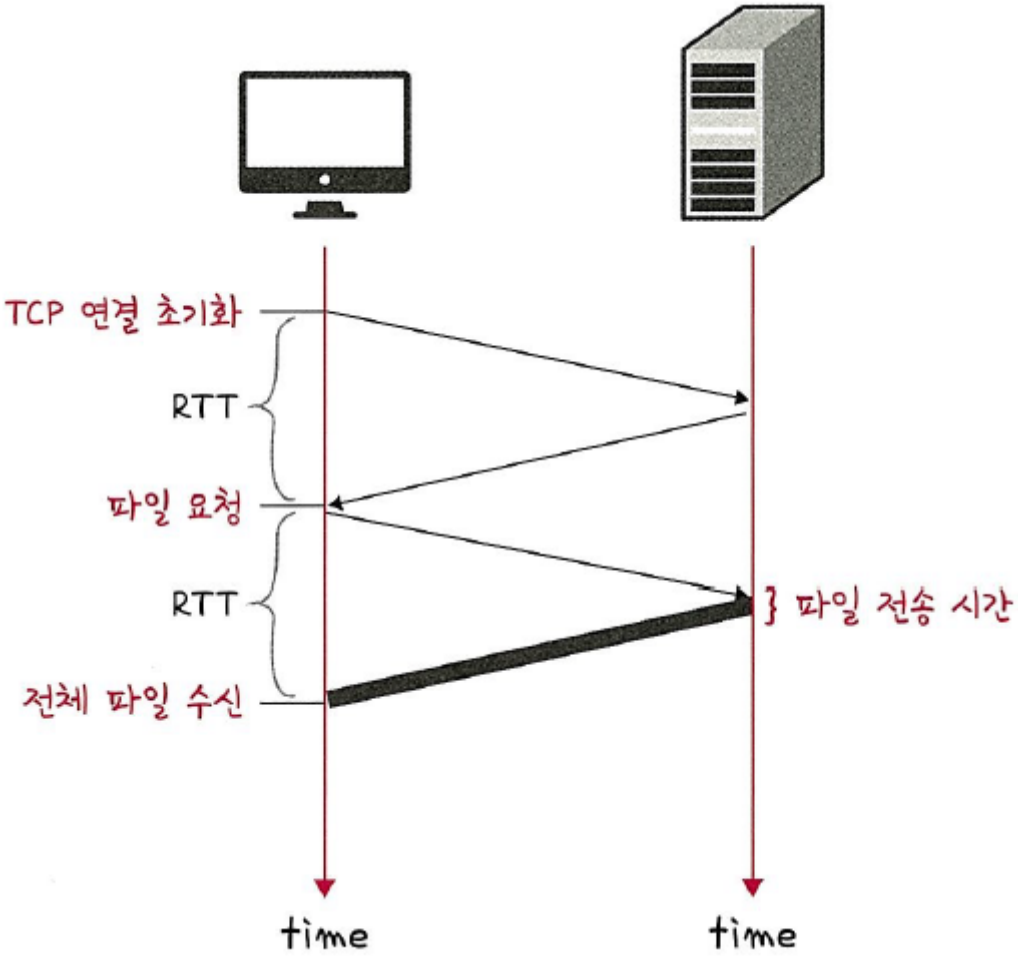
RTT 증가를 불러오게 됨

RTT (Round Trip Time)

패킷이 목적지에 도달하고 나서 해당 패킷에 대한 응답이 출발지로 다시 돌아오기까지의 시간

즉, 패킷 왕복 시간을 말함

RTT 증가



서버로부터 파일을 가져올 때마다 TCP 의 3-way handshake 를 계속해서 열어야하므로 RTT 가 증가하는 단점이 있음

RTT 증가 해결

매번 연결시 마다 RTT 가 증가함에따라 서버에 부담이 많이 가고 사용자 응답 시간이 길어짐

이를 해결하기 위해 이미지 스플리팅, 코드 압축, 이미지 Base 64 인코딩을 사용

이미지 스플리팅

많은 이미지를 다운로드 받게 되면 과부하가 걸리므로, 많은 이미지가 합쳐 있는 하나의 이미지를 다운로드 받고 이를 기반으로 backgroun-image 의 position 을 이용하여 이미지를 표기하는 방법

```
#icons>li>a {
  background-image: url("icons.png");
  width: 25px;
  display: inline-block;
  height: 25px;
  repeat: no-repeat;
}

#icons>li:nth-child(1)>a {
  background-position: 2px -8px;
}

#icons>li:nth-child(2)>a {
  background-position: -29px -8px;
}
```

위 코드와 같이 background-image, background-position 등을 기반으로 이미지를 설정함

코드 압축

코드를 압축해서 개행문자, 빈칸을 없애서 코드의 크기를 최소화 하는 방법

javscript 파일 중 한줄로 되어있는 파일들 (ex. filename_min.js)

이미지 Base64 인코딩

이미지 파일을 64진법으로 이루어진 문자열로 인코딩하는 방법

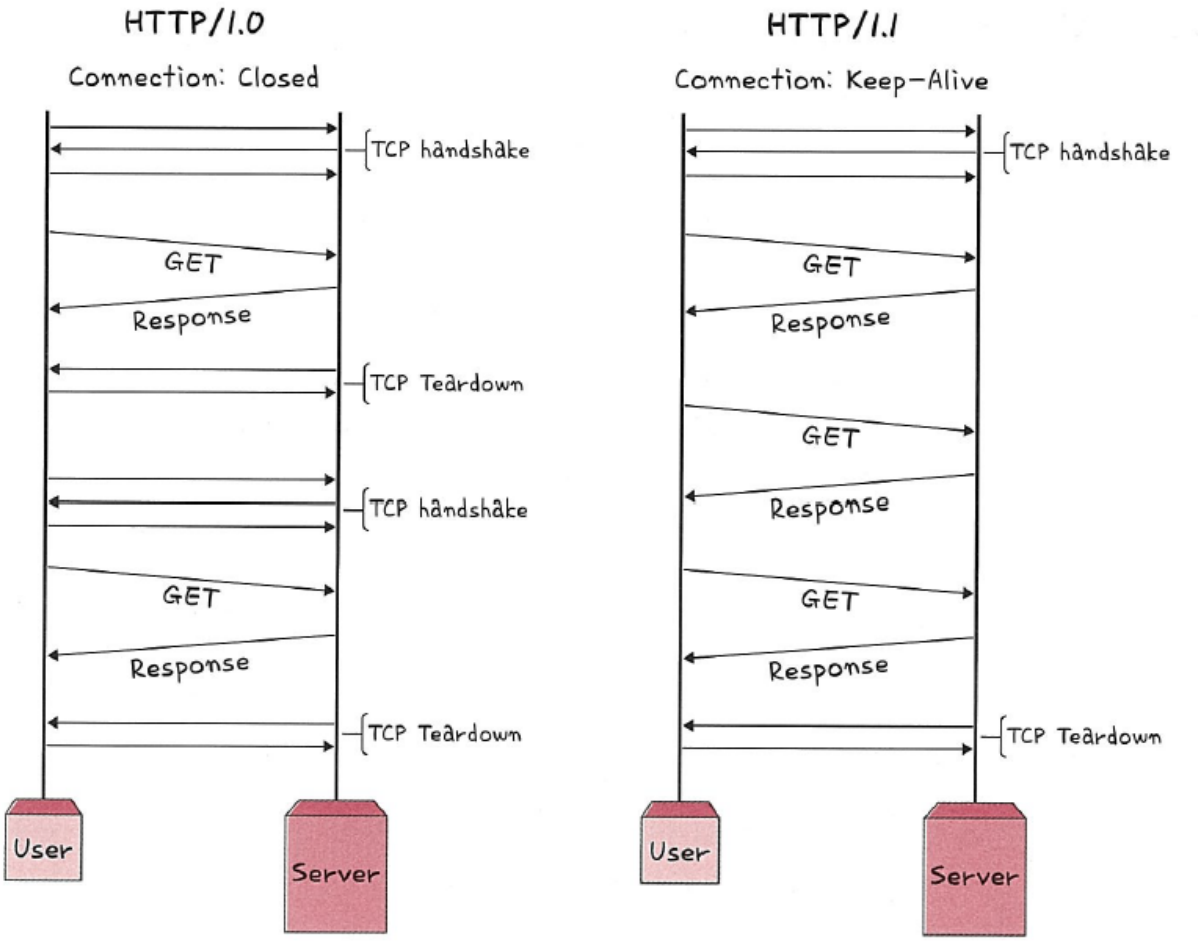
서버와의 연결을 열고 이미지에 대해 서버에 HTTP 요청을 할 필요가 없다는 장점이 있음

37% 정도 크기가 더 커지는 단점이 있음

HTTP/1.1

매번 TCP 연결을 하는것이 아니라 한번 TCP 초기화를 한 이후에 keep-alive 라는 옵션으로 여러개의 파일을 송수신할 수 있게 바뀜

HTTP/1.0 에도 없던것은 아니나, HTTP/1.1 부터 표준화가 되어 기본 옵션으로 설정되었음



위 그림과 같이 한번 TCP 3-way handshake 발생 시 그 다음부터 발생하지 않는 것을 볼 수 있음

문서 안에 포함된 다수의 리소스 (이미지, css, js 파일) 처리 시요청할 리소스 개수에 비례해서 대기 시간이 길어지는 단점이 있음

HOL (Head Of Line) Blocking

네트워크에서 같은 큐에 있는 패킷이 그 첫번째 패킷에 의해 지연될 때 발생하는 성능 저하 현상

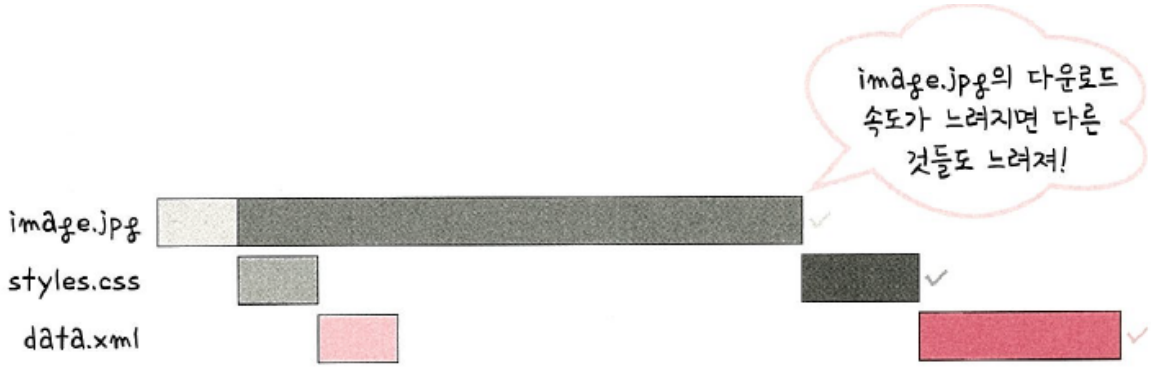


image.jpg 와 style.css, data.xml 을 다운로드 받을 때 보통은 순차적으로 잘 받아지나, image.jpg 가 느리게 받아질 경우 그 뒤에 있는 파일들이 대기하게되며 다운로드가 지연되는 상태가 됨

무거운 헤더 구조

HTTP/1.1의 헤더는 쿠키 등 많은 메타 데이터가 들어 있고 압축이 되지 않아 무거움

HTTP/2

SPDY 프로토콜에서 파생된 HTTP/1.x 보다 지연시간을 줄이고 응답시간을 더 빠르게 할 수 있으며

멀티플렉싱, 헤더 압축, 서버 푸시, 요청의 우선순취 처리를 지원하는 프로토콜

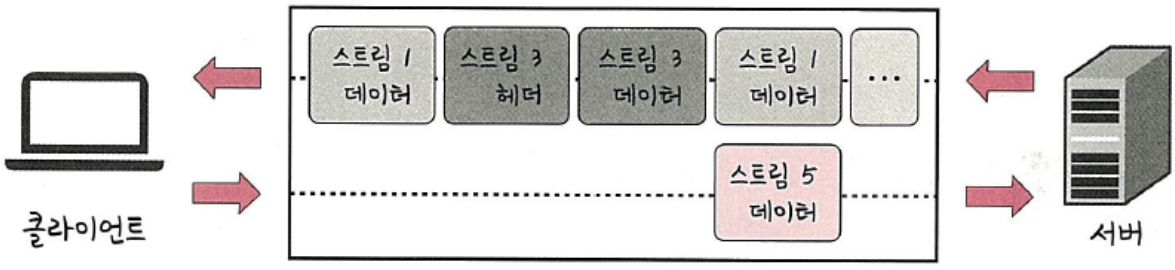
멀티플렉싱 (Multiplexing)

여러개의 스트림을 사용하여 송수신한다는 것

이를 통해 특정 스트림의 패킷이 손실되었다고 하더라도 해당 스트림에만 영향을 미치고 나머지 스트림은 멀쩡하게 동작할 수 있음

스트림 (stream)

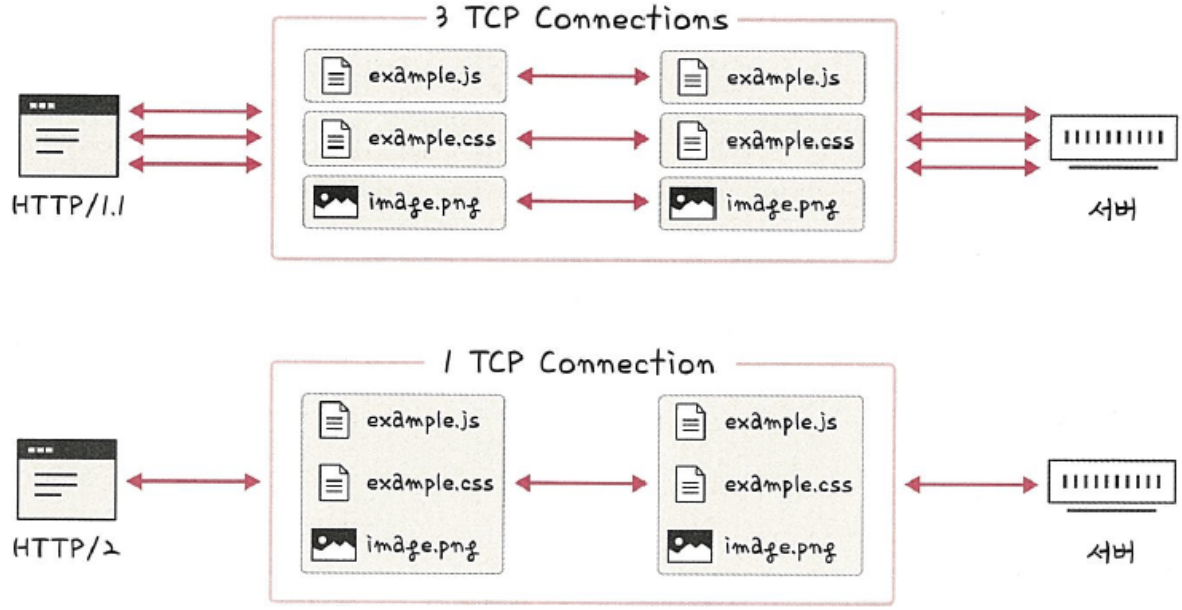
시간이 지남에 따라 사용할 수 있게 되는 일련의 데이터 요소를 가리키는 데이터 흐름



위 그림은 병렬적인 스트림들을 통해 데이터를 서빙하고 있음

스트림 내의 데이터들도 쪼개져 있음

애플리케이션에서 받아온 메시지를 독립된 프레임으로 조각내어 서로 송수신한 이후 다시 조립하며 데이터를 주고받음



이를 통해 단일 연결을 사용하여 병렬로 여러 요청을 받을 수 있고 응답을 줄 수 있음

HTTP/1.x 에서 발생하는 문제인 HOL Blocking 을 해결할 수 있음 (병렬처리)

헤더 압축

HTTP/1.x 의 문제점인 무거운 헤더 구조 해결

허프만 코딩 압축 알고리즘을 사용하는 HPACK 압축 형식을 가짐

허프만 코딩 (huffman coding)

문자열을 문자 단위로 쪼개 빈도수를 세어 빈도가 높은 정보는 적은 비트 수를 사용하여 표현하고

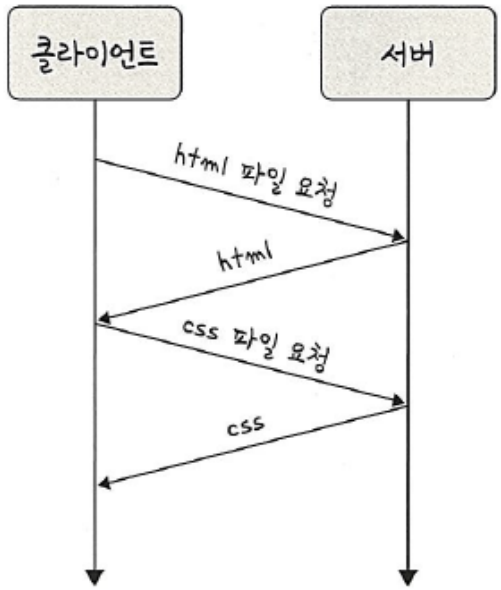
빈도가 낮은 정보는 비트 수를 많이 사용하여 표현

⇒ 전체 데이터의 표현에 필요한 비트양을 줄이는 원리

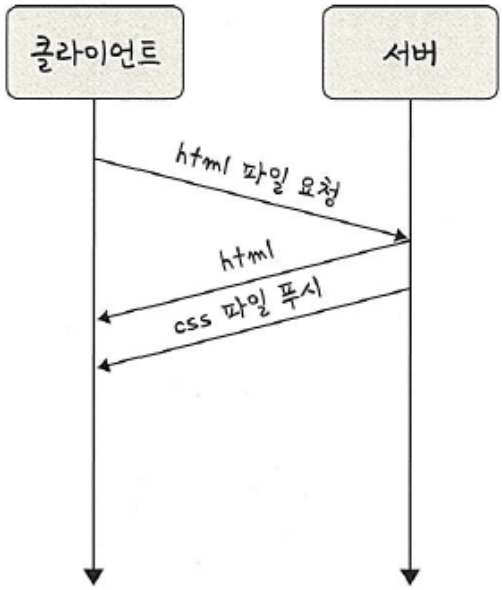
서버 푸시

HTTP/1.1 에서는 클라이언트가 서버에 요청을 해야 파일을 다운로드 받을 수 있었음

HTTP/2 는 클라이언트 요청 없이 서버가 바로 리소스를 푸시할 수 있음



서버 푸시 X



서버 푸시 O

HTML 에는 css 나 js 파일이 포함되는데, HTML 을 읽으면서 그 안에 있던 css 파일을 서버에서 푸시하여 클라이언트에 먼저 전달할 수 있음

HTTPS

애플리케이션 계층과 전송 계층 사이에 신뢰 계층인 SSL/TLS 계층을 넣은 신뢰할 수 있는 HTTP 요청

⇒ 통신을 암호화함

HTTP/2 는 HTTPS 위에서 동작함

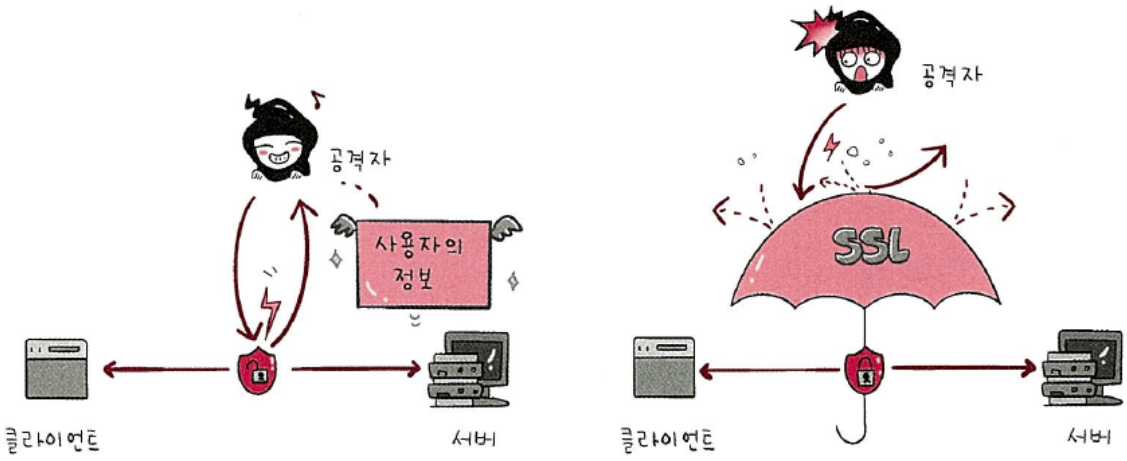
SSL/TLS

SSL (Secure Socket Layer) 은 SSL 1.0부터 시작해서 SSL2.0, SSL3.0, TLS (Transport Layer Security Protocol) 1.0, TLS 1.3 까지 버전이 올라가며, 마지막으로 TLS 로 변경되었음

보통 이를 합쳐 SSL/TLS 라고 함

전송 계층에서 보안을 제공하는 프로토콜

클라이언트와 서버가 통신할 때 SSL/TLS를 통해 제 3자가 메시지를 도청하거나 변조하지 못하도록함



위와 같이 공격자가 서버인 척하며 사용자 정보를 가로채는 네트워크 상의 인터셉터 방지 가능

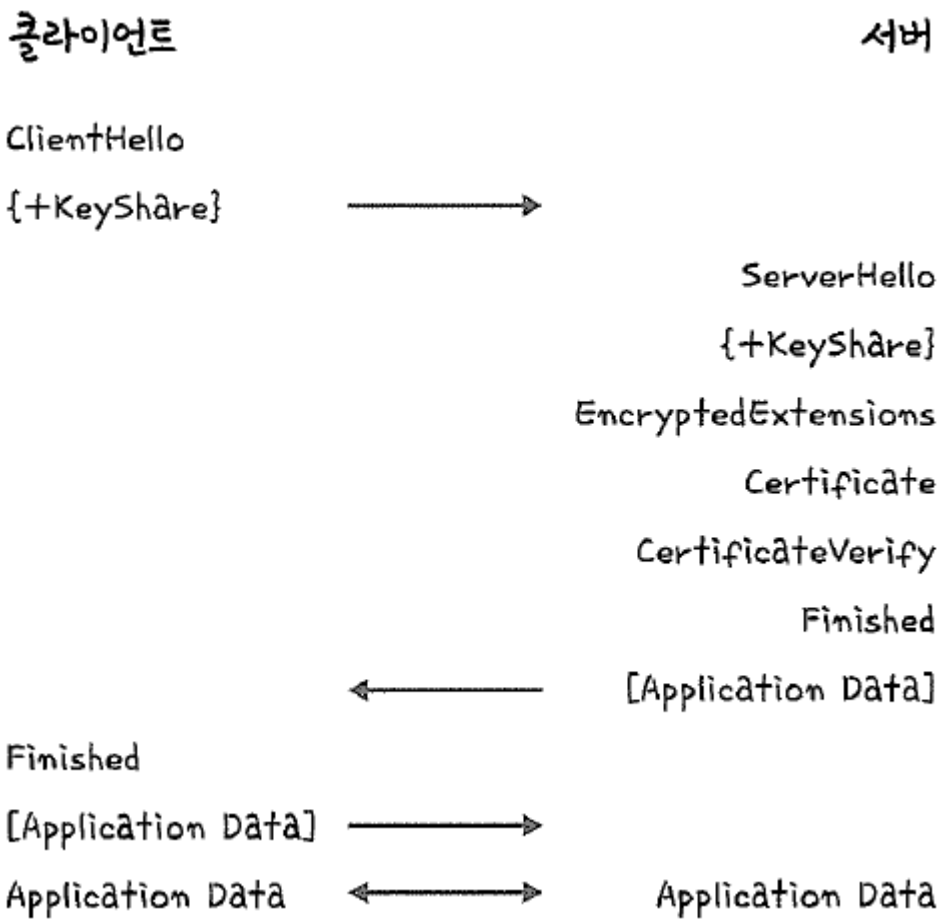
SSL/TLS는 보안 세션을 기반으로 데이터를 암호화하며, 보안 세션이 만들어질 때 인증 매커니즘, 키 교환 암호화 알고리즘, 해싱 알고리즘이 사용됨

보안 세션

보안이 시작되고 끝나는 동안 유지되는 세션

SSL/TLS는 handshake 를 통해 보안 세션을 생성하고 이를 기반으로 상태 정보 등을 공유

TLS 의 handshakek



클라이언트와 서버가 키를 공유하고 이를 기반으로 인증, 인증 확인 등의 작업이 일어나는 단 한번의 1-RTT 가 생긴 후 데이터를 송수신함

클라이언트에서 사이퍼 슈트 (cypher suites)를 서버에 전달하면 서버는 받은 사이퍼 슈트의 암호화 알고리즘 리스트를 제공가능한지 확인

제공 가능하다면 서버에서 클라이언트로 인증서를 보내는 인증 메커니즘이 시작됨

이후 해싱 알고리즘 등으로 암호화된 데이터의 송수신이 시작됨

사이퍼 슈트 (Cypher Suites)

프로토콜, AEAD 사이퍼 모드, 해싱 알고리즘이 나열된 규약. 다섯개가 있음

- 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

AEAD (Authenticated Encryption with Associated Data) 는 데이터 암호화 알고리즘

AES_128_GCM 등이 있음

- AES_128_GCM : 128비트의 키를 사용하는 표준 블록 암호화 기술과 병렬 계산에 용이한 암호화 알고리즘 GCM 이 결합된 알고리즘

인증 메커니즘

CA (Certificate Authorities) 에서 발급한 인증서를 기반으로 이루어짐

안전한 연결을 시작하는데 있어 필요한 공개키를 클라이언트에 제공하고 사용자가 접속한 서버가 신뢰 할 수 있는 서버임을 보장함

서비스 정보, 공개키, 지문, 디지털 서명 등으로 이루어져 있음

CA 는 아무 기업이나 할 수 있는 것이 아니고, 신뢰성이 엄격하게 공인된 기업들만 참여할 수 있음

CA 발급 과정

자신의 서비스가 CA 인증서를 발급받으려면 자신의 사이트 정보와 공개키를 CA에 제출해야함

이후 CA 는 공개키를 해시한 값인 지문(finger print) 을 사용하는 CA 의 비밀키 등을 기반으로 CA 인증서를 발급함

암호화 알고리즘

대수곡선 기반의 ECDHE(Elliptic Curve Diffie-Hellman Ephermeral) 또는 모듈식 기반의 DHE(Diffie-Hellman Ephermeral) 를 사용함

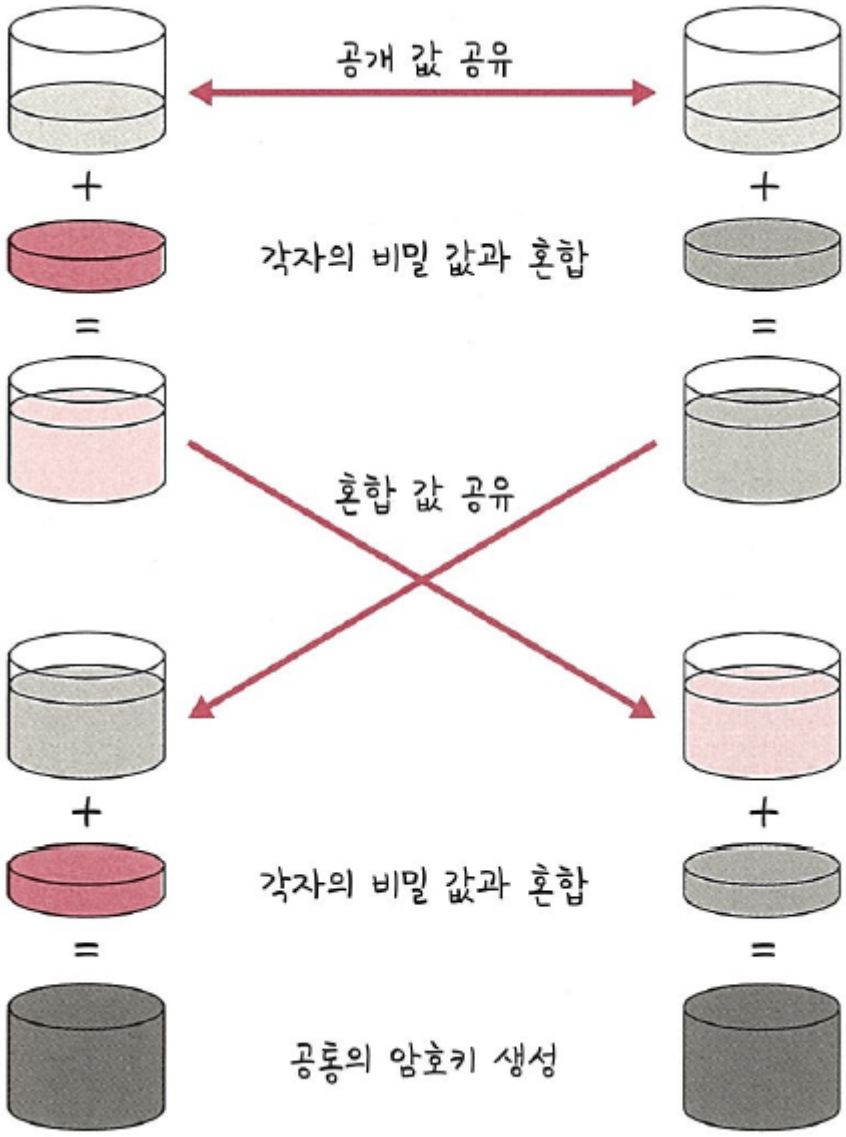
둘 다 디피-헬만 방식을 근간으로 만들어짐

디피-헬만 (Diffie-Helman) 키 교환 암호화 알고리즘

암호키를 교환하는 하나의 방법

$y = g^x mod p$

위 식에서 g 와 x 와 p 를 안다면 y 는 구하기 쉬우나, g 와 y 와 p 만 안다면 x를 구하기는 어렵다는 원리에 기반한 알고리즘



위 그림과 같이 처음에 공개값을 공유하고 각자의 비밀 값과 혼합한 후 혼합 값을 공유함

그 다음 각자의 비밀 값과 또 혼합함

이후에 공통의 암호키가 생성됨

클라이언트와 서버 모두 개인키와 공개키를 생성하고 서로에게 공개키를 보내고

공개키와 개인키를 결합하여 PSK(사전 합의된 비밀키)가 생성된다면

악의적인 공격자가 개인키 또는 공개키를 가지고도 PSK 가 없으므로 공격할 수 없음

해싱 알고리즘

데이터를 추정하기 힘든 더 작고, 섞여 있는 조각으로 만드는 알고리즘

SSL/TLS는 해싱알고리즘으로 SHA-256 알고리즘과 SHA-384 알고리즘을 사용함

SHA-256 알고리즘

해시함수의 결과값이 256비트인 알고리즘

비트코인을 비롯한 많은 블록체인 시스템에서도 사용

해싱을 해야할 메시지에 1을 추가하는 등 전처리를 하고 전처리된 메시지를 기반으로 해시를 반환함

TLS 1.3 의 경우 사용자가 이전에 방문한 사이트로 다시 방문할경우 SSL/TLS에서 보안세션을 만들때 걸리는 통신을 하지 않아도 되며, 이를 0-RTT라고함

SEO 에도 도움이 되는 HTTPS

SEO(Search Engine Optimization)

검색 엔진 최적화

사용자들이 검색엔진으로 웹 사이트를 검색했을 때 그 결과를 페이지 상단에 노출시켜 많은 사람이 볼 수 있도록 최적화하는 방법

SEO 를 위한 방법으로 캐노니컬 설정, 메타 설정, 페이지 속도 개선, 사이트맵 관리 등이 있음

캐노니컬 설정

`<link rel="canonical" href="www.example.com">`

위와 같이 link 에 캐노니컬을 설정해야함

메타 설정

HTML 파일의 가장 윗부분인 메타를 잘 설정해야함

아래는 애플의 메타 (최고의 웹 페이지라고 함)

```
<meta property="analytics-track" content="Apple - Index/Tab">
<meta property="analytics-s-channel" content="homepage">
<meta property="analytics-s-bucket-0" content="applestoreww">
<meta property="analytics-s-bucket-1" content="applestoreww">
<meta property="analytics-s-bucket-2" content="applestoreww">
<meta name="Description" content="Discover the innovative world of Apple and shop everything iPhone, iPad, Apple Watch, Mac,
and Apple TV, plus explore accessories, entertainment, and expert device support.">
<meta property="og:title" content="Apple">
```

페이지 속도 개선

사이트의 속도는 빨라야함

사이트맵 관리

사이트맵을 정기적으로 관리하는 것은 필수

사이트맵 제너레이터를 사용하거나 직접 코드를 만들어 구축해도됨

HTTPS 구축 방법

크게 세가지가 있음

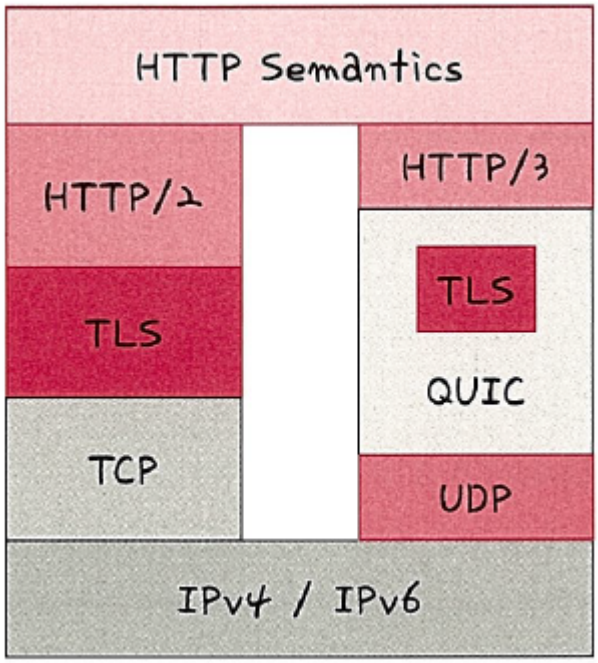
1. 직접 CA에서 구매한 인증키를 기반으로 HTTPS 서비스 구축
2. 서버 앞단의 HTTPS를 제공하는 로드밸런서를 통해 구축
3. 서버 앞단에 HTTPS를 제공하는 CDN을 통해 구축

HTTP/3

HTTP의 세번째버전

QUIC 라는 계층 위해서 동작함

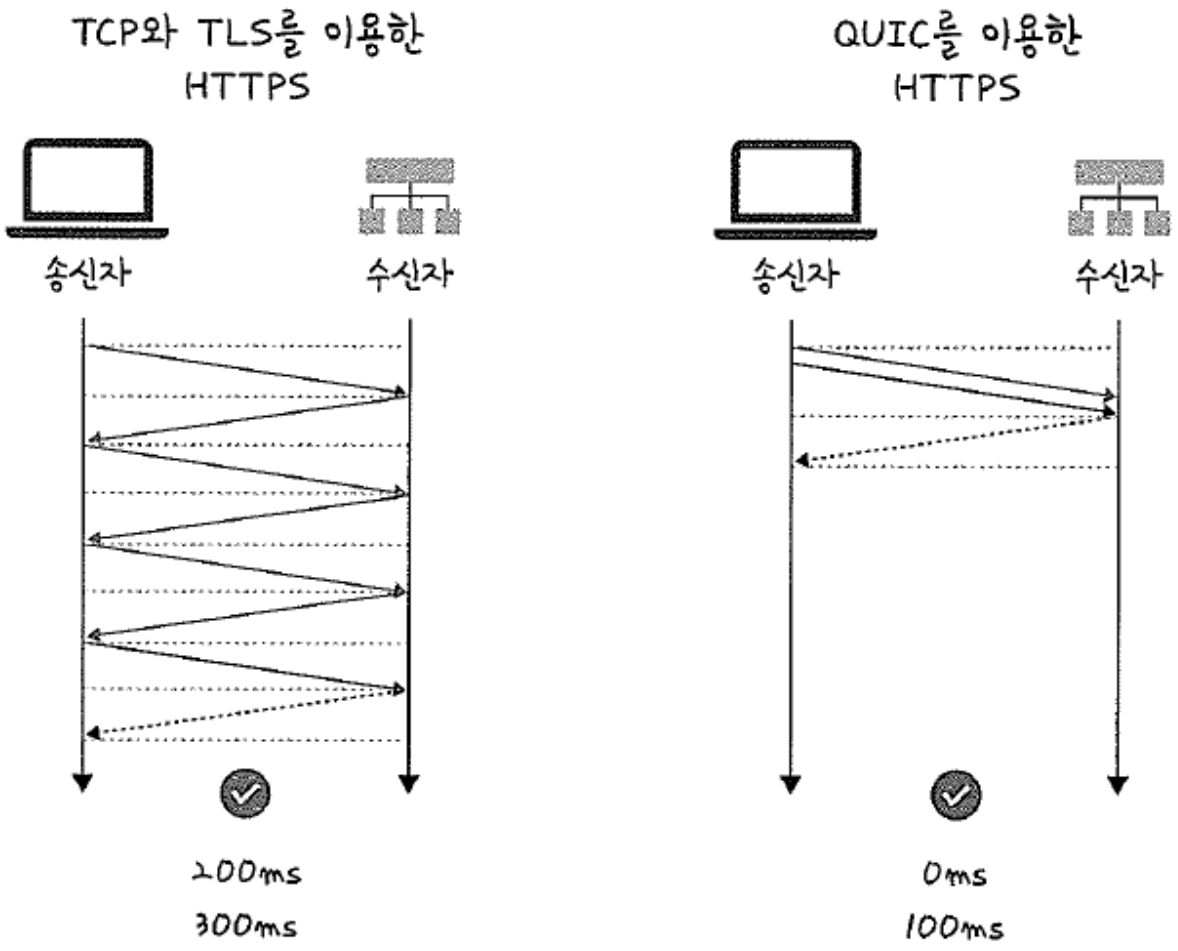
TCP 기반이 아닌 UDP 기반으로 동작함



HTTP/2 에서 장점이었던 멀티플렉싱을 가지고 있으며, 초기 연결 설정 시 지연시간 감소라는 장점이 있음

초기 연결 설정 시 지연 시간 감소

QUIC 는 TCP 를 사용하므로 통신 시작 시 번거로운 3-way handshake 과정을 거치지 않아도 됨



QUIC는 첫 연결 설정에 1-RTT만 소요됨

→ 클라이언트가 서버에 어떤 신호를 한번 주고, 서버도 응답하기만 하면 바로 본 통신을 시작할 수 있음

QUIC는 순방향 오류 수정 메커니즘 (FEC, Forward Error Correction)이 적용되었음

순방향 오류 수정 메커니즘 (FEC, Forward Error Correction)

전송한 패킷이 손실되었다면 수신 측에서 에러를 검출하고 수정하는 방식

열악한 네트워크 환경에서도 낮은 패킷 손실률을 보임