HTTP

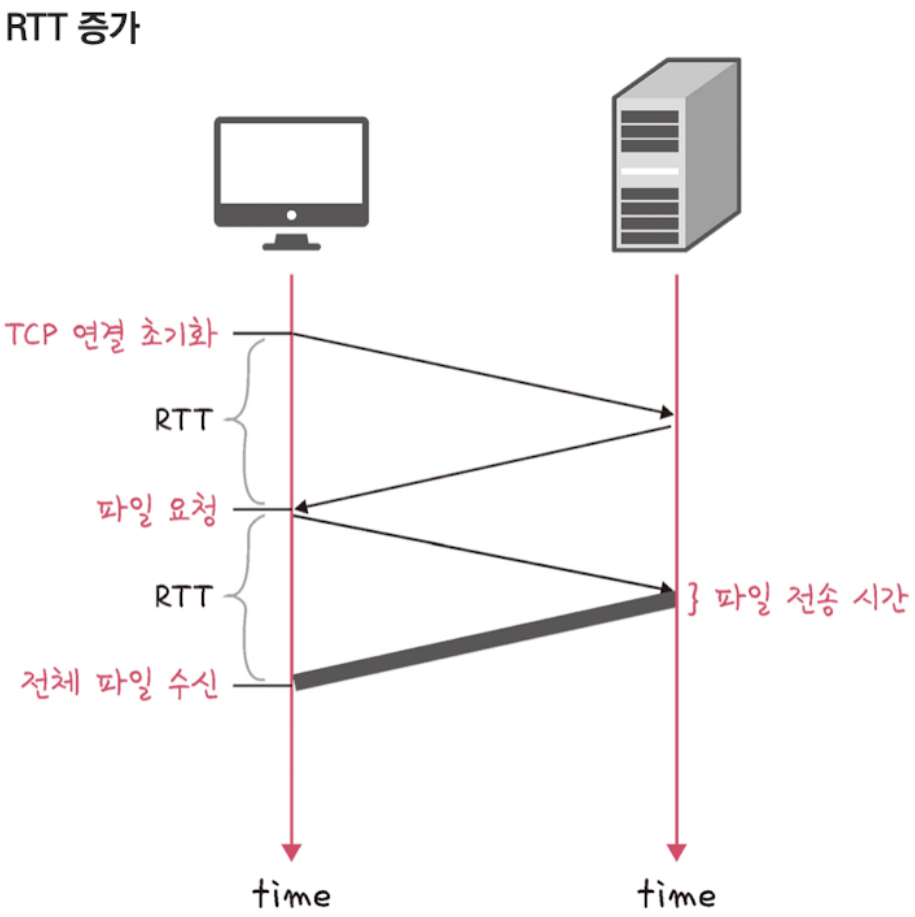
태그	네트워크 - HTTP

HTTP란?

전송 계층위의 애플리케이션 계층에서 동작하는 프로토콜로 클라이언트와 서버간에 정보를 주고받기 위한 통신규약이다.

HTTP/1.0

HTTP 1.0은 한 연결당 하나의 요청을 처리하도록 설계되었다. 각 요청마다 TCP 연결과정(3-way handshake)이 필요하기 때문에 RTT가 증가한다는 단점이 있다.



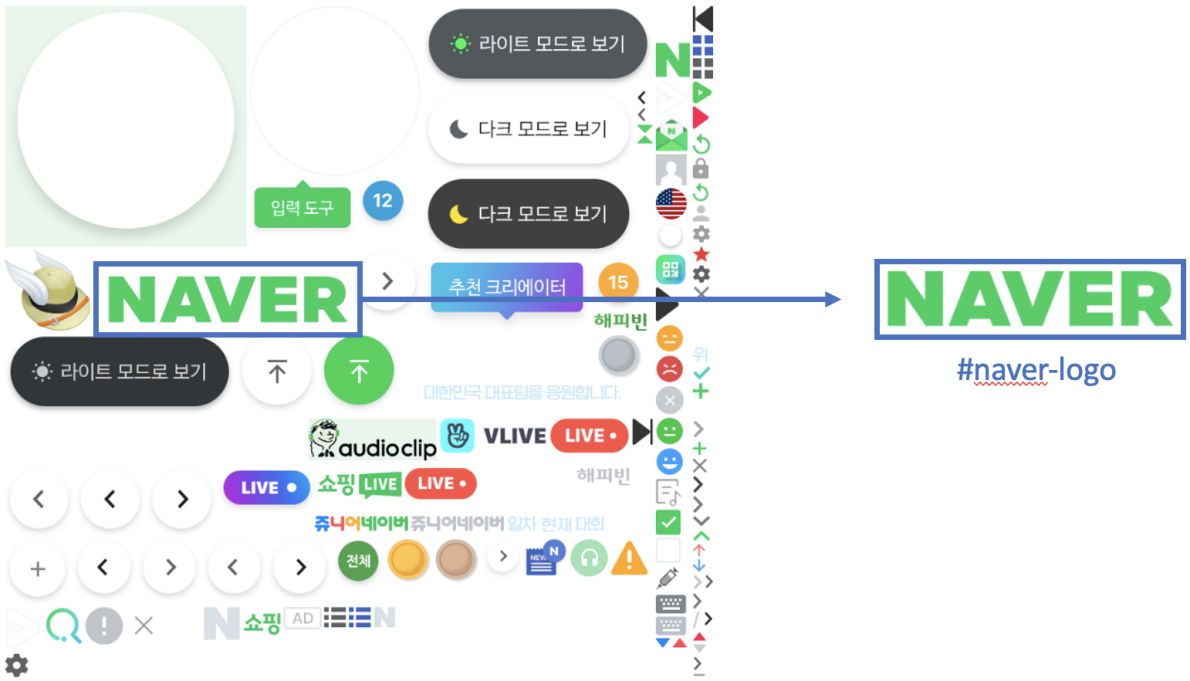
> **RTT(Round Trip Time)**
> 패킷이 목적지에 도달하고 나서 다시 출발지로 돌아오기까지 걸리는 시간이며 패킷 왕복시간을 말한다.

RTT 증가를 해결하기 위한 방법

매번 연결할 때마다 RTT가 증가하면서 서버에 부담이 많이가고 사용자 응답시간이 길어진다. 이를 해결하기 위해 이미지 스플리팅, 코드 압축, 이미지 Base64 인코딩을 사용했다.

이미지 스플리팅(이미지 스트라이프)

많은 이미지를 다운로드받게 되면 과부하가 걸리기 때문에 많은 이미지가 합쳐 있는 하나의 이미지를 다운로드받고, 이를 기반으로 background-image의 position을 이용하여 필요한 이미지만 화면에 표시하는 방법이다.



```
#naver-login {
  width: 180px;
  height: 50px;
  background-image: url(image/sprite.png)
  background-position: -10px -10px;
}
```

코드 압축

코드를 압축해서 개행 문자, 빈칸을 없애서 코드의 크기를 최소화하는 방법이다. 이를 통해 코드 용량을 줄일 수 있다.

```
// 기존코드
console.log("test");

for (var i = 0; i < 3; i++) {
  console.log("test");
}

console.log("test");
```

```
// 코드압축 적용
console.log("test");for(vari=0;i<3;i++){console.log("test");}console.log("test");
```

이미지 Base64 인코딩

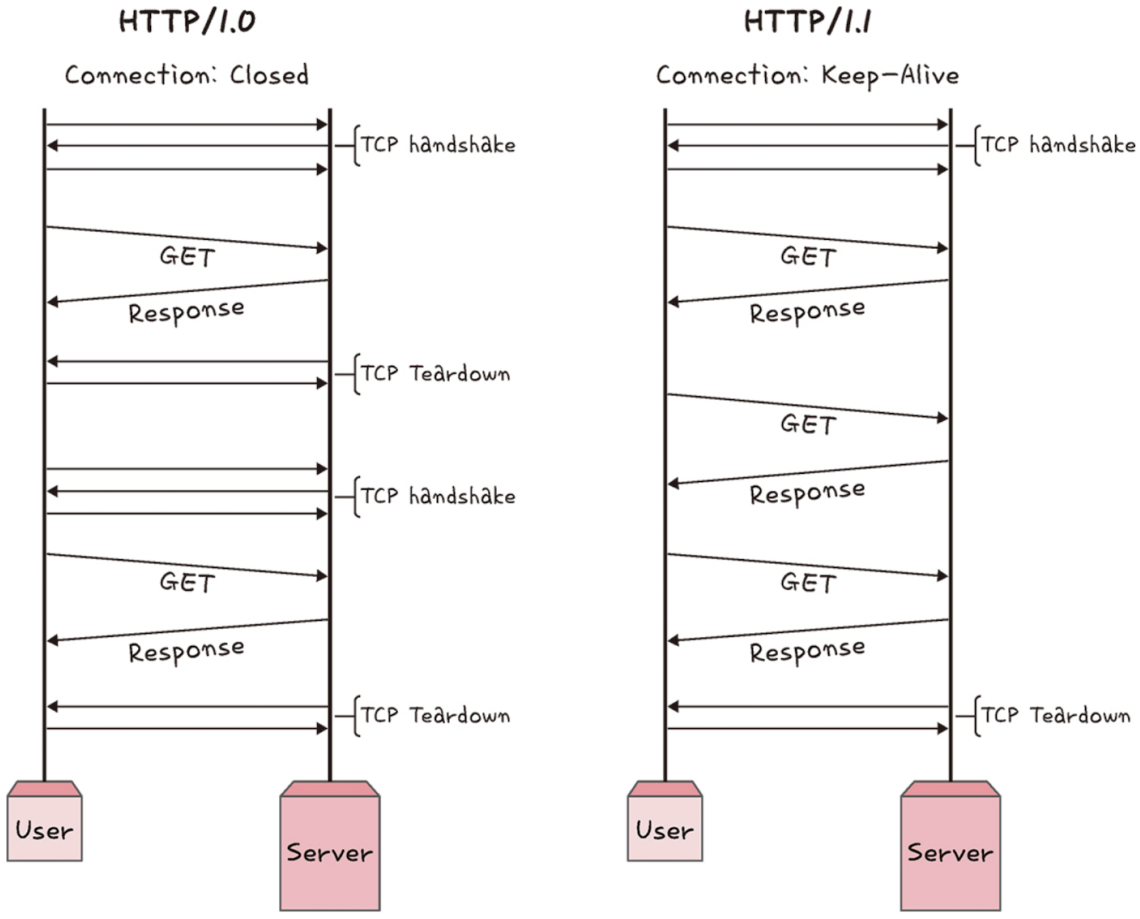
이미지 파일을 64진법으로 이루어진 문자열로 인코딩하는 방법이다. 이 방법을 사용하면 서버와의 연결을 열고 이미지에 대해 서버에 HTTP 요청을 할 필요가 없어진다는 장점이 있지만 Base64 문자열로 변환할 경우 37% 정도 크기가 더 커지는 단점이 있다.

> **인코딩**
> 정보의 형태나 형식을 다른 형태나 형식으로 변환하는 처리방식

HTTP/1.1

HTTP/1.0에서 발전한 형태로 매번 TCP 연결을 하는것이 아니라 한 번 TCP를 초기화 한 이후에 keep-alive라는 옵션으로 여러 개의 파일을 송수신할 수 있다. (HTTP/1.0 에서도 keep-alive 옵션이 있었지만 표준화가 되어있지 않았고 HTTP/1.1 부터 표준화 되어 기본 옵션으로 설정되었다.)

하지만, HTTP/1.1은 문서 안에 포함된 다수의 리소스(이미지, 동영상, js 파일 등)를 처리하는 과정에서 몇가지 단점을 가지고 있다.



[HTTP/1.1의 단점]

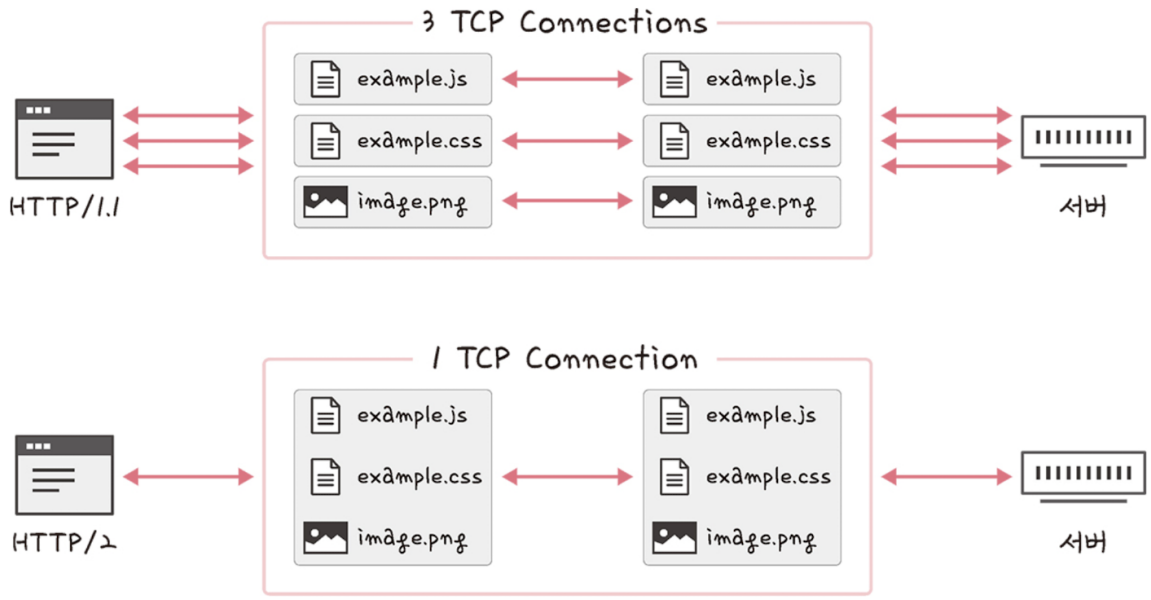
- HOL Blocking이 발생한다. HOL Blocking은 네트워크에서 같은 큐에 있는 패킷이 그 첫번째 패킷에 의해 지연될 때 발생하는 성능 저하 현상을 말한다.

- 헤더구조가 무겁다. HTTP/1.1의 헤더에는 쿠키 등 많은 메타데이터가 들어 있고 압축이 되지 않아 무겁다.

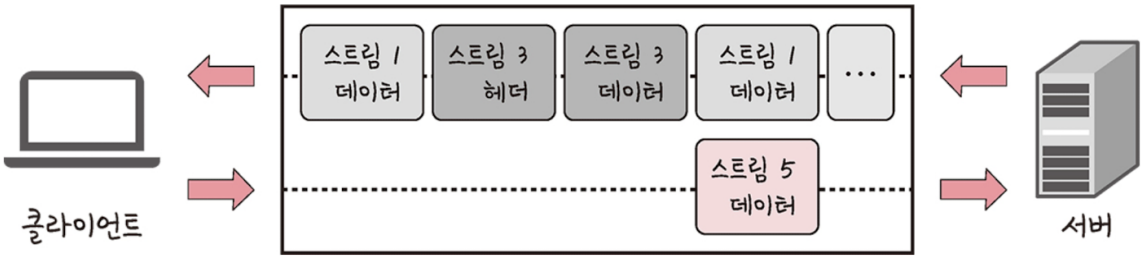
HTTP/2

HTTP/2는 HTTP1.x 보다 지연 시간을 줄이고 응답 시간을 더 빠르게 할 수 있으며 멀티플렉싱, 헤더압축, 서버푸시, 요청의 우선순위 처리를 지원한다.

멀티플렉싱



하나의 TCP 커넥션만으로 여러 개의 스트림을 사용하여 해당 웹 페이지의 모든 요청과 응답 데이터를 송수신하는것을 의미한다. 이를 통해 특정 스트림의 패킷이 손실되었다고 하더라도 해당 스트림이외의 스트림은 영향을 받지 않는다.



멀티 플렉싱은 병렬적인 스트림들을 통해 데이터를 전달하는데 스트림 내의 데이터들도 여러개로 나누어져 있다. 애플리케이션에서 받아온 메시지를 독립된 프레임으로 조각내어 서로 송수신한 이후 다시 조립하여 데이터를 주고받는 과정을 거친다.

이런 과정을 통해 HTTP/1.x에서 발생하는 문제인 HOL Blocking을 해결한다.

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

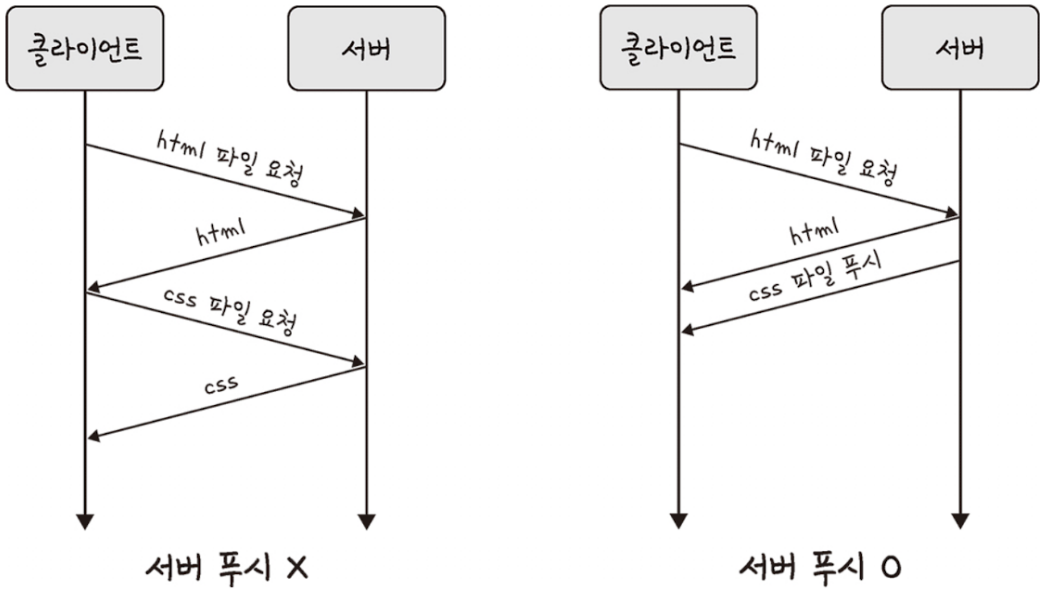
헤더 압축

HTTP/1.x에는 헤더의 크기가 크다는 문제가 있었다. 이를 HTTP/2에서는 헤더 압축을 통해 해결하는데 허프만 코딩 압축 알고리즘을 사용하는 HPACK 압축 형식을 가진다.

> **허프만코딩**
> 문자열을 문자 단위로 쪼개 빈도수를 세어 빈도가 높은 정보는 적은 비트 수를 사용하여 표현하고 빈도가 낮은 정보는 비트 수를 많이 사용하여 표현해서 전체 데이터의 표현에 필요한 비트양을 줄이는 원리

서버 푸시

HTTP/1.1 에서는 클라이언트가 서버에 요청을 해야 파일을 다운로드 받을 수 있었다면 HTTP/2는 클라이언트의 요청 없이 서버가 바로 리소스를 푸시할 수 있다.



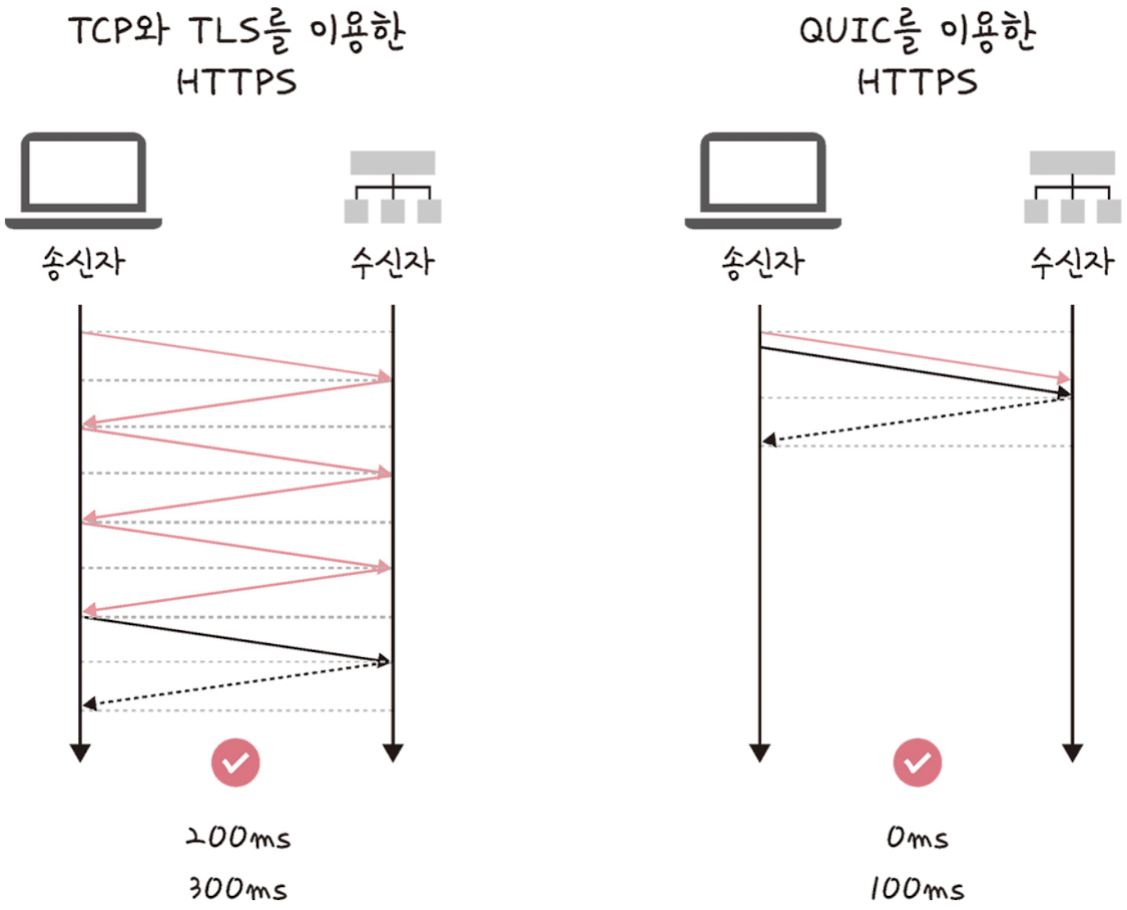
html에는 css나 js 파일이 포함되기 마련인데 html을 읽으면서 그 안에 들어 있던 css 파일을 서버에서 푸시하여 클라이언트에 먼저 전달할 수 있다.

HTTP/3

HTTP/3는 HTTP/1.1 및 HTTP/2와 함께 WWW에서 정보를 교환하는데 사용되는 HTTP의 세번째 버전이다. TCP 위에서 돌아가는 HTTP/2 와는 달리 HTTP/3는 QUIC이라는 계층 위에서 돌아가며 TCP 기반이 아닌 UDP 기반으로 동작한다.

[HTTP/3의 장점]

- 멀티플렉싱
- 초기 연결 설정 시 지연 시간 감소



> **HTTP/3 초기 연결 설정 시 지연시간 감소**
> QUIC은 TCP를 사용하지 않기 때문에 통신 시작시 연결과정(3-way handshake)과정을 거치지 않는다. QUIC는 첫 연결 설정에 1-RTT만 소요된다. 즉, 클라이언트가 서버에 어떤 신호를 한 번 주고, 서버도 응답하면 바로 통신을 시작할 수 있다.
> 참고로, QUIC는 순방향 오류 수정 메커니즘(FEC, Forward Error Correction)이 적용되어 있어 전송한 패킷이 손실되었다면 수신 측에서 에러를 검출하고 수정하는 방식을 통해 열악한 네트워크 환경에서도 낮은 패킷 손실률을 자랑한다.

HTTPS

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

SSL/TLS

SSL/TLS는 전송 계층에서 보안을 제공하는 프로토콜이다. 클라이언트와 서버가 통신 할 때 SSL/TLS를 통해 제 3자가 메시지를 도청하거나 변조하지 못하도록 하는 기능을 제공한다.

SSL/TLS는 보안 세션을 기반으로 데이터를 암호화하며 보안 세션이 만들어 질 때 인증 메커니즘, 키 교환 암호화 알고리즘, 해싱 알고리즘이 사용된다.

> **보안 세션**
> 보안이 시작되고 끝나는 동안 유지되는 세션을 말하고, SSL/TLS는 핸드셰이크를 통해 보안 세션을 생성하고 이를 기반으로 상태 정보등을 공유한다.

> **세션**
> 운영체제가 어떠한 사용자로부터 자신의 자신 이용을 허락하는 일정한 기간을 의미한다. 즉, 사용자는 일정 시간동안 응용 프로그램, 자원 등을 사용할 수 있다.