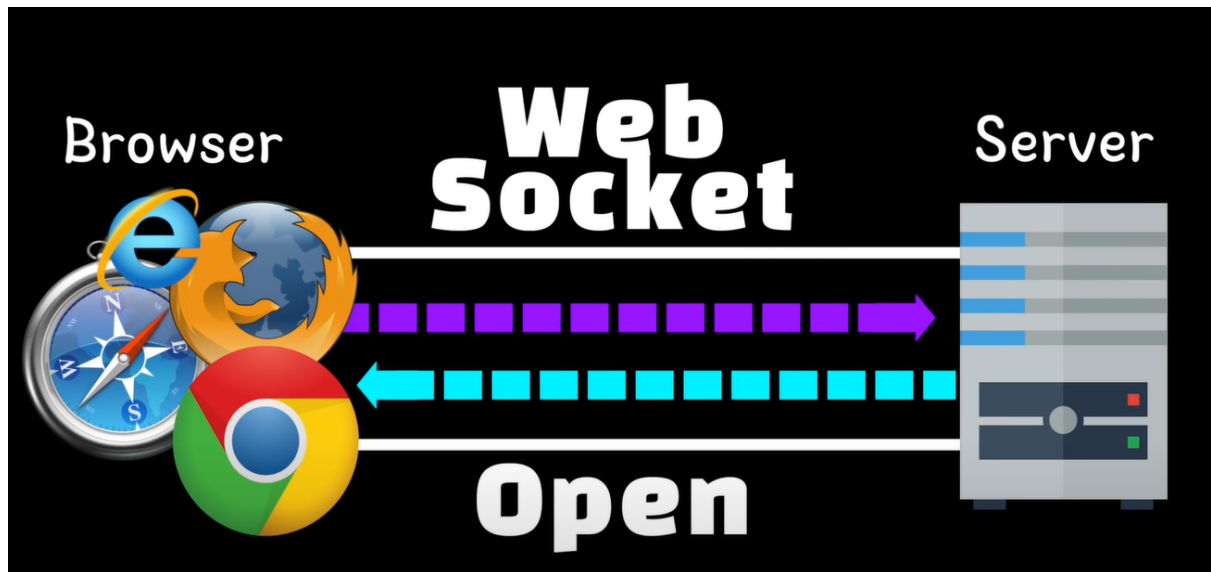


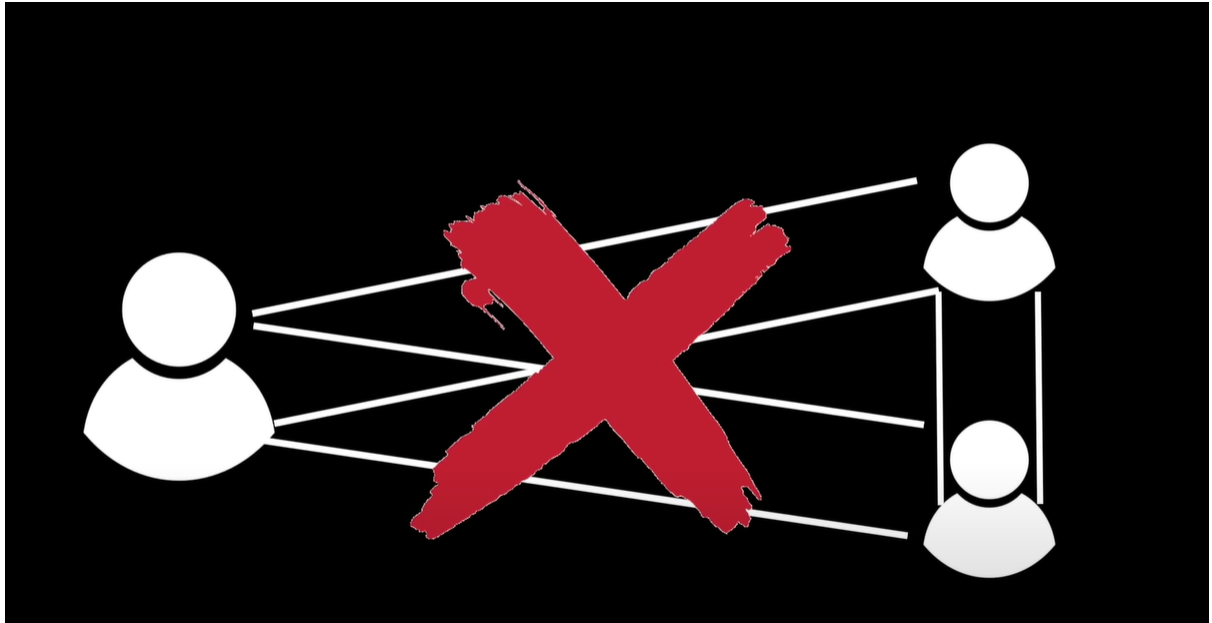
김나연

WebSocket

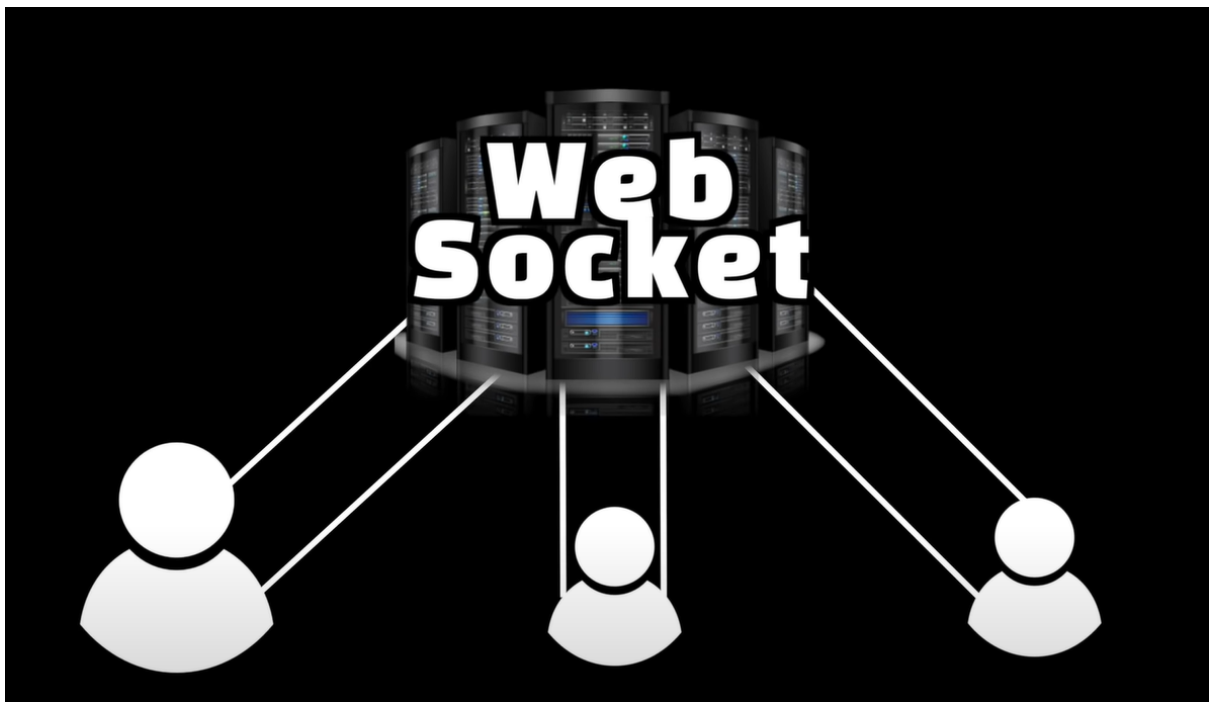
- connection : Open - Close(Request - Response X)



- 브라우저 는 서버 간 통신은 열려있고(Open) 이 통신은 원하는 순간까지 계속 열려있음
- 그렇기에 서버는 request를 기다릴 필요 없고, 원할 때 업데이트를 그냥 보낼 수 있음



- 만약 내가 채팅방에 입장하면 나는 친구들과 연결된게 아



- 모두 다 같은 웹 소켓 서버에 입장한 것
- 내가 메시지를 보내면 서버는 해당 메시지를 채팅방 사람들에게 보내주는 것

But 문제점

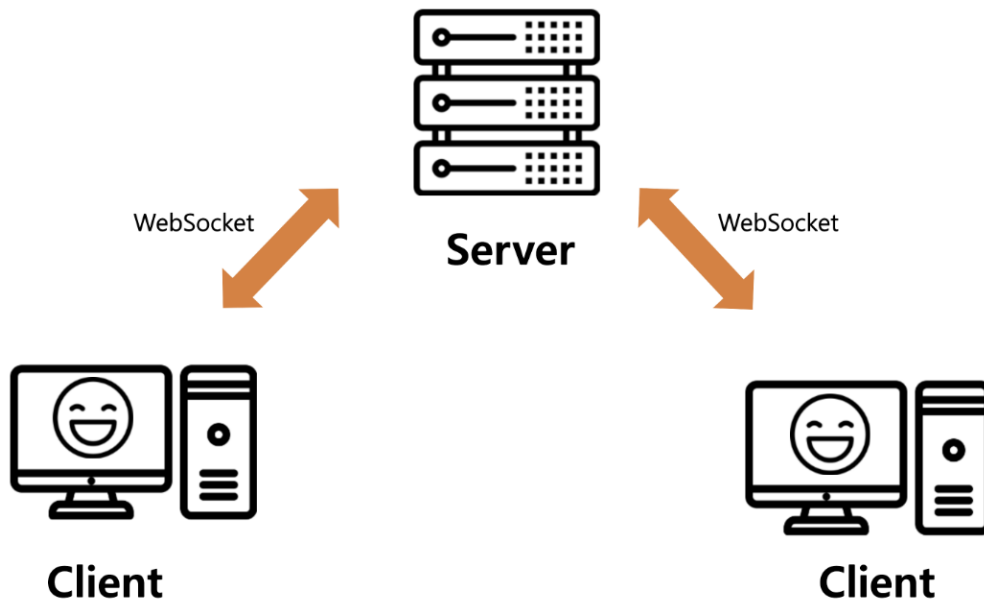
- 웹 소켓에서는 모든 통신을 추적하기 위해 서버의 메모리 파워가 중요함

- 그렇기에 유저가 많으면 많을수록 더 많은 메모리가 필요하고 서버에 더 많은 돈을 써야 함.
- 서버 빠르게 유지 → 메시지를 다른 사람에게 보내야하니까
- 그치만 서버에 많은 커넥션이 오고가면 딜레이가 생기게 됨.

WebRTC



- 브라우저끼리 연결
- P2P : 내가 메시지를 보내면 서버에 전달되는 것이 아닌 상대방에게 바로 전달됨.
- 단순 텍스트 뿐 아니라 영상, 오디오 등을 주고 받음.
- 모든 자료들을 서버를 통해 전달하지 않음.(그래서 빠름)
- 확장성에는 제약이 있지만 우리는 6명이니까 상관 없 ?



- 시그널링 서버를 통해 서로의 정보들을 전달받아 P2P 방식으로 연결

시그널링

- 서로 다른 네트워크에 있는 2개의 디바이스들을 통신하기 위해서는 각 디바이스들의 위치(IP) 발견 및 미디어 포맷협약이 필요함. 이 프로세스를 시그널링이라고 하고 각 디바이스들을 상호간에 동의된 서버에 연결시킴.

시그널링 서버

- 두 디바이스들 사이에 WebRTC 커넥션을 만들기 위해 인터넷 네트워크에서 그 둘을 연결 시키는 작업을 해줄 시그널링 서버가 필요함.
- P2P 연결을 위해선 상대방의 Public IP주소를 알아야함.
- but, 클라이언트가 방화벽이나 NAT 뒤에 있는 경우가 대부분이라 Public IP주소를 알기 어려움.
- 이를 해결하기 위해 WebRTC는 STUN 서버 사용을 권장함.
- 때론, NAT에 따라 정보를 가져오기 힘들 수 있어 이 경우엔 TURN 서버를 사용.

NAT

(Network Address Translation)

- Public IP 주소 하나로 여러 호스트가 인터넷을 사용하는 것(ex. 공유기)

OpenVidu

1. OpenVidu 배포 실행

```
docker run -p 4443:4443 --rm -e OPENVIDU_SECRET=MY_SECRET openvidu/openvidu-dev:2.28.0
```

2. 서버 애플리케이션 샘플 실행(자바)

```
1. 저장소 복제
git clone https://github.com/OpenVidu/openvidu-tutorials.git -b v2.28.0
cd openvidu-tutorials/openvidu-basic-java

2. 애플리케이션 실행
mvn spring-boot:run
```

3. 클라이언트 애플리케이션 튜토리얼 실행

```
npm install --location=global http-server
```


-
1. 서버 설정: OpenVidu를 사용하기 위해 먼저 서버를 설정해야 합니다. OpenVidu 서버는 OpenVidu를 관리하고 클라이언트와의 통신을 중재하는 역할을 수행합니다.
 2. 클라이언트 연결: 웹 브라우저나 모바일 앱에서 OpenVidu 클라이언트 라이브러리를 사용하여 OpenVidu 서버에 연결합니다. 클라이언트는 OpenVidu 서버와의 WebSocket 연결을 설정하고, 사용자의 오디오 및 비디오 스트림을 생성하여 전송할 수 있습니다.
 3. 세션 생성: 클라이언트는 OpenVidu 서버에 새로운 세션을 생성합니다. 세션은 비디오 및 오디오 통신의 단위이며, 여러 사용자 간의 통신을 관리합니다.
 4. 퍼블리셔(Publisher) 생성: 퍼블리셔는 사용자가 자신의 오디오 및 비디오 스트림을 생성하고 전송할 수 있게 해줍니다. 클라이언트는 퍼블리셔 객체를 생성하여 자신의 오디오

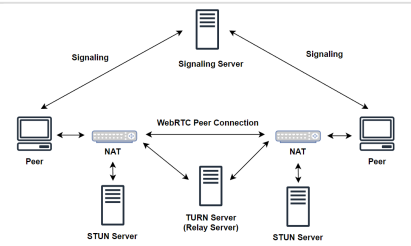
및 비디오를 스트리밍하고, 세션에 추가합니다.

5. 구독자(Subscriber) 생성: 구독자는 다른 사용자의 오디오 및 비디오 스트림을 수신할 수 있게 해줍니다. 클라이언트는 구독자 객체를 생성하여 다른 사용자의 스트림을 수신하고, 세션에 추가합니다.
6. 스트림 통신: 퍼블리셔는 자신의 오디오 및 비디오 스트림을 세션에 게시하고, 구독자는 세션에서 발행된 스트림을 수신합니다. 이를 통해 사용자들은 실시간으로 오디오 및 비디오를 주고받을 수 있습니다.

[WebRTC] ReactJS + EC2 + OpenVidu 환경에서 화상회의 구현


WebRTC (Web Real-Time Communication)란 별도의 플러그인이나 소프트웨어 없이 실시간으로 데이터(음성, 영상, 텍스트, 파일)를 서버(중계자) 없이 브라우저 간에 교환할 수 있도록 하는 기술이다. WebRTC 기술은 다

 <https://velog.io/@ohjinseo/ReactJS-EC2-OpenVidu-환경에서-화상회의-구현해보기>



Openvidu 커스터마이징 하기

openvidu는 Webrtc 에서 간단하게 사용해볼 수 있는 프레임워크다. 간단하게 사용하기에는 정말 쉽다. 하지만 openvidu를 가지고 커스터마이징을 하고, aws에 배포를 한다면 어마어마한 어려움과 함께 달려보낸 10

 <https://velog.io/@kimyunbin/Openvidu-커스터마이징-하기>

