

WebRTC, Kurento 관련 개념 정리


WebRTC?

- Web Real-Time Communication
- 서버를 최대한 거치지 않고 P2P(Peer-to-Peer Network)로 브라우저나 단말 간에 데이터를 주고받는 기술의 웹 표준

WebRTC API - Web API | MDN

WebRTC(Web Real-Time Communication)은 웹 애플리케이션과 사이트가 중간자 없이 브라우저 간에 오디오나 영상 미디어를 포착하고 마음대로 스트림할 뿐 아니라, 임의의 데이터도 교환할 수 있도록 하는 기술입니다. WebRTC를 구성하는 일련의 표준들은 플러그인이

 https://developer.mozilla.org/ko/docs/Web/API/WebRTC_API

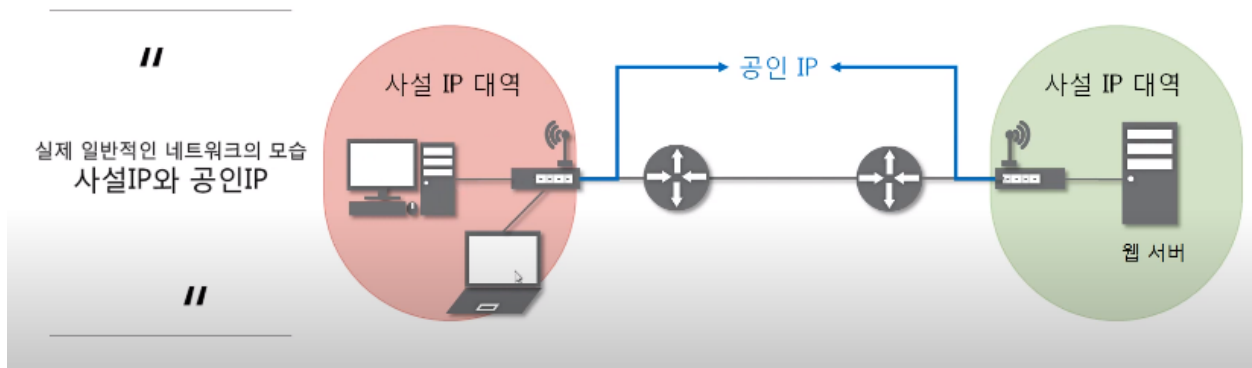
 mdn web docs

WebRTC 통신은 P2P에 최적화 되어 있음. 게다가 서버와 같은 중간자를 거치지 않고 브라우저 간 연결이 가능함.

말은 쉽지만 그렇다고 직접적인 통신이 바로 이뤄지지는 않음. 그 이유는 NAT라는 개념 때문!

NAT(Network Address Translation)

IP 패킷의 TCP/UDP 포트 숫자와 소스 및 목적지의 IP 주소 등을 재기록하면서 라우터를 통해 네트워크 트래픽을 주고 받는 기술



요게 무슨말이냐 하면 내 노트북에서 인터넷을 이용하려면 인터넷이 연결된 공유기에 연결되어 있어야하는데, 이때 공유기 밖으로 통신이 이뤄질 때는 노트북에 할당된 사설 IP가 아닌 공유기가 대표하는 공인 IP로 통신이 이뤄

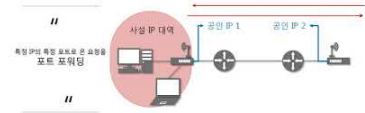
진다는 뜻이다. 때문에 수신 측은 출발지의 사설 IP를 알지 못하게된다.

이렇게 NAT로 네트워크가 형성되어 있기 때문에 인터넷에 연결된 모든 노트북(호스트)가 고유한 ip를 가져야할 필요가 없어지기 때문에 ip 주소도 절약되고 외부에는 사설 ip가 공유되지 않기 때문에 보안적으로도 안전하다는 장점이 있다.


[따라學IT] 10. NAT와 포트 포워딩 - 이론

 <https://www.youtube.com/watch?v=Qle5cfCcuEY&t=217s>

포트포워딩
포트포워딩이란?



[이해하기] NAT (Network Address Translation) - 네트워크 주소 변환 | STEVEN J. LEE

NAT (Network Address Translation - 네트워크 주소 변환)이란, IP 패킷에 있는 출발지 및 목적지의 IP 주소와 TCP/UDP 포트 숫자 등을 바꿔 재기록하면서 네트워크 트래픽을 주고 받게하는 기술입니다. 1. NAT 을 왜 쓰는가 - IP 주소 절약과 보안 (1) IP 주소 절약 NAT 기술을 이용하면, 하나의 공인 IP  <https://www.stevenjlee.net/2020/07/11/%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-nat-network-address-translation-%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC-%EC%A3%BC%EC%86%8C-%EB%B3%80%ED%99%98/>

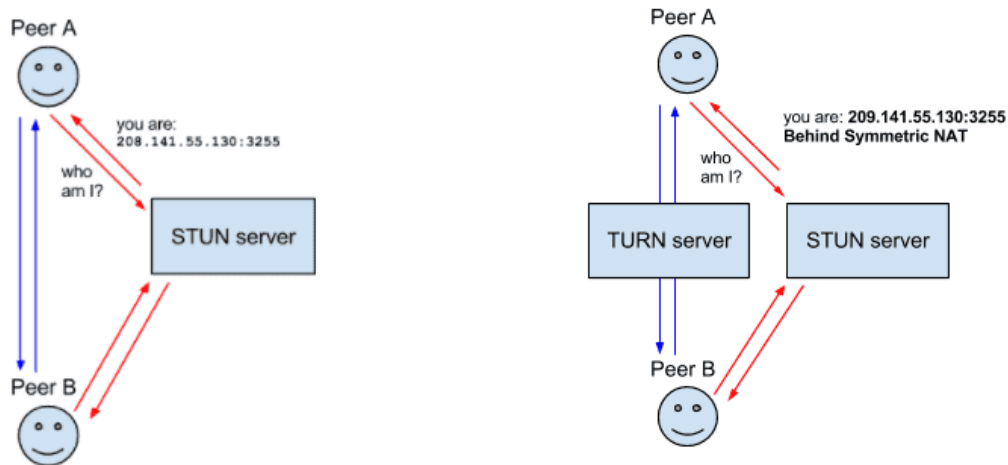


STUN, TURN

NAT이라는 개념은 분명 네트워크에 이점을 가져다준다. 하지만 WebRTC 뉴비인 우리에게는 걸림돌이 된다.

NAT으로 인해 공인 IP만 알뿐 사설 IP 정보는 알 수 없기 때문에 WebRTC로 통신하고자 하는 Peer들이 서로의 정확한 주소를 알지 못한다는 문제가 발생한다.

따라서 피어 간 연결 주소를 알기 위해서 STUN서버, STUN으로 해결하지 못하는 경우 TRUN 서버가 필요하다.



ICE(Interactive Connectivity Establishment)

두 단말이 서로 통신할 수 있는 최적의 경로를 찾을 수 있도록 도와주는 프레임워크.

ICE 프로세스를 사용하면 NAT가 통신을 위해 필요한 모든 포트를 열어두고 두 엔드 포인트 모두 다 연결할 수 있는 IP주소, 포트에 대한 완전한 정보를 갖게 된다.

요청하는 클라이언트와 미디어 서버 사이의 연결을 통해 미디어(비디오, 음성) 등을 주고 받을 수 있게된다.

STUN(Session Traversal Utilities for NAT)

공개 주소를 발견하거나 peer간의 직접 연결을 막는 등 라우터의 제한을 결정하면 ICE를 보완하는 프로토콜 → **단말이 자신의 공인 IP 주소와 포트를 확인하는 과정에 대한 프로토콜**

클라이언트는 인터넷을 통해 클라이언트의 공개 주소와 라우터의 NAT 뒤에 있는 클라이언트가 접근 가능한지에 대한 답변을 위한 요청을 STUN 서버에 보냄

하지만 라우터에 따라 방화벽 정책을 달리할 수도 있고, 이전에 연결된 적이 있는 네트워크만 연결할 수 있게 제한을 걸기도 하기 때문에 STUN 서버를 이용하더라도 정보를 알아낼 수 없을 수도 있다.

TURN(Traversal Using Relays around NAT)

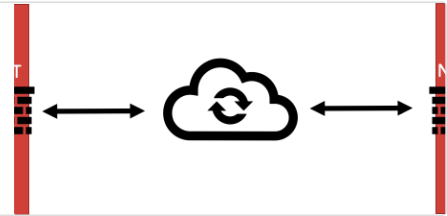
STUN이 해결 못하는 문제를 해결하는 것이 TURN 서버이다. TURN 방식은 네트워크 미디어를 중개하는 서버를 이용하는 방식이다.

중간에 서버를 한 번 거치기 때문에, 사실상 P2P 통신이 아니기도하고, 구조상 지연이 발생할 수 밖에 없지만 보안 정책이 엄격한 개인 NAT 내부에 위치한 브라우저와 통신을 할 수 있는 유일한 방법이기 때문에 최후의 수단으로 선택된다.

WebRTC란? (STUN과 TURN 서버의 이해) (2)

이전 글 복습 중간에 방화벽이 존재하거나 NAT 환경에 놓여 있는 경우에는 각 Peer에 대한 직접적인 시그널링이 불가능하다고 이야기하였다. 그렇기 때문에 시그널링을 하고 연결을 하기 위해서는 무언가 다른 방법이 필요하다. 이러한 방식을 UDP Hole Punching 방식이라

🔗 <https://andonekwon.tistory.com/59>



<https://wormwlrn.github.io/2021/01/24/Introducing-WebRTC.html>

Media Server?

WebRTC는 기본적으로 P2P 통신에 특화된 기술이다. 유튜브, 인스타 라이브와 같은 대규모의 서비스에서 1:N 스트리밍 혹은 다양한 기능을 추가하기에는 P2P만으로는 한계가 있다. (P2P에서 10만명에게 비디오를 전송하려면 10만번 파일을 올려야하기 때문)

그렇기 때문에 peer의 중간에 미디어 서버가 필요하고 미디어 서버 솔루션의 하나로 Kurento Media Server가 사용된다.

Kurento

Kurento is an Open Source Software WebRTC media server

🔗 <https://kurento.openvidu.io/>



Kurento Media Server(KMS)

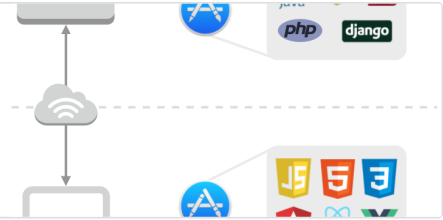
Kurento는 전체 WebRTC 스택의 기능적 구현을 제공하는 미디어 서버다. 쿠렌토 미디어 서버는 그룹간의 통신, 녹음, 방송, 시청각 흐름의 라우팅 기술을 지원하고 있다. 다른 미디어서버와 차별화된 기능으로 쿠렌토는 컴퓨터 비전, 음성 분석같은 고급 미디어 처리 기능도 제공한다.

Openvidu? : kurento와 같이 미디어 서버 역할을 수행한다. 사실 Openvidu는 WebRTC와 kurento 기반으로 동작한다고 한다. Openvidu는 모든 하위 수준의 작업들을 추상화시켜두었기 때문에 개발자가 복잡한 구현 사항에 대해 관여하지 않아도 될 정도로 간단한 API를 제공한다.

화상 미팅을 간단하게 구현할 수 있는 Kurento와 Openvidu 프레임워크

<https://2jinishappy.tistory.com/248?category=948597> WebRTC - 웹 브라우저 간 실시간 미디어 통신 기술 WebRTC: Web Real-Time Communication 웹 브라우저 간에 플러그인의 도움 없이 서로 통신할 수 있도록 설계된 API 2020년 이후 COVID-19로 인한 비대면 수

☞ <https://2jinishappy.tistory.com/335>



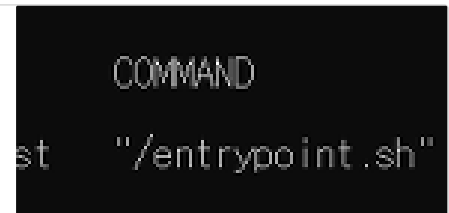
Docker로 KMS 실행시켜보기

무작정 시작해보고 싶다면 아래 블로그 참고!

[Kurento] 쿠렌토 서버 Docker로 실행시켜보기 (feat. 윈도우)

쿠렌토 서버 Docker로 실행시키기 윈도우 환경에서 로컬로 돌리게 되면 환경이 안맞는 부분도 발생할 수 있기 때문에 Docker를 사용하여 쿠렌토서버를 띄워보자 준비사항 Ubuntu 18.04 LTS Docker Docker에서 Kurento Media Server 실행시키기 우분투를 열고 명령어

☞ <https://gh402.tistory.com/44>



Kurento 튜토리얼

해당 레포지토리에는 각 서비스별 클라이언트 소스코드 예제가 수록되어 있다.

<https://github.com/Kurento/kurento-tutorial-java>

- [kurento-chroma](#) : 크로마 필터를 사용한 루프백의 WebRTC.
- [kurento-crowddetector](#) : 군중 감지기 필터가 있는 미러(루프백)의 WebRTC. 이 필터는 비디오 스트림에서 사람이 뭉치는 것을 감지
- [kurento-group-call](#) : WebRTC 다대다 화상 통화.
- [kurento-hello-world-recording](#) : Hello World(루프백의 WebRTC) 및 녹음.
- [kurento-hello-world-repository](#) : ~~This tutorial is not actively maintained.~~
- [kurento-hello-world](#) : Hello World(루프백의 WebRTC).
- [kurento-magic-mirror](#) : faceoverlay 필터(매직 미러)를 사용한 루프백의 WebRTC.
- [kurento-metadata-example](#) : 메타데이터를 사용하는 두 개의 필터가 있는 루프백의 WebRTC.
- [kurento-one2many-call](#) : WebRTC 일대다 화상 통화.
- [kurento-one2one-call-advanced](#) : WebRTC 일대일 화상 통화(녹화 및 필터링 포함).
- [kurento-one2one-call-recording](#) : WebRTC 일대일 화상 통화(녹음 포함).

- [kurento-one2one-call](#) : WebRTC 일대일 화상 통화.
- [kurento-platedetector](#) : 플레이트 검출기 필터가 있는 루프백의 WebRTC.
- [kurento-player](#) : WebRTC를 통한 비디오 재생.
- [kurento-pointerdetector](#) : 포인터 감지기 필터가 있는 루프백의 WebRTC.
- [kurento-rtp-receiver](#) : RTP 스트림을 재생
- [kurento-send-data-channel](#) : 데이터 채널을 통해 QR 코드에 대한 데이터를 전송하기 위해 필터 및 WebRTC에 연결된 플레이어.
- [kurento-show-data-channel](#) : 데이터 채널을 통해 수신된 데이터를 표시하는 필터가 있는 루프백의 WebRTC.

→ 우리 서비스에 필요한 기능이 있다면 찾아서 코드를 참고해보면 좋을 것 같음!