Open Source Software Project

: DGU Chat Final Report

Prof. Dongho Kim

과제	최종보고서
제출	2017년 06월 11일 일요일
번호	3조
팀장	컴퓨터공학과 2012111938 석정한
팀원	컴퓨터공학과 2013112043 심희오
	컴퓨터공학과 2013112048 강정석
	컴퓨터공학과 2014113128 천훼

1. 과제 목표

공개소프트웨어의 중요성과 사회 발전에 미치는 영향을 이해한다. 또한, 기존의 공개소프트웨어를 기반으로 새로운 공개소프트웨어를 개발하는 전 과정을 경험한다. 공개소프트웨어 개발환경에서 공개소스를 기반으로 팀 단위 협업프로그래밍 과제를 수행하여 유용한 소프트웨어를 제작하고 이를 공개한다.

특히, 3조는 WebRTC (Web Real-Time Communication) 기술을 이용한 화상 채팅 웹 어플리케이션 개발과, Qt 프레임워크을 이용하여 WebRTC를 지원하지 않는 브라우저에서도 동작할 수 있게 멀티 플랫폼, 하이브리드 어플레케이션 개발을 목표로 한다.

마지막으로 개발된 소프트웨어와 개발 방법 및 사용법을 Github에 공개하여 사회에 기여한다.

2. 과제의 목적

프로그램을 개발할 때 필요한 지식과 기술들이 너무나 많고 개발 방법들이 매우 다양하다.

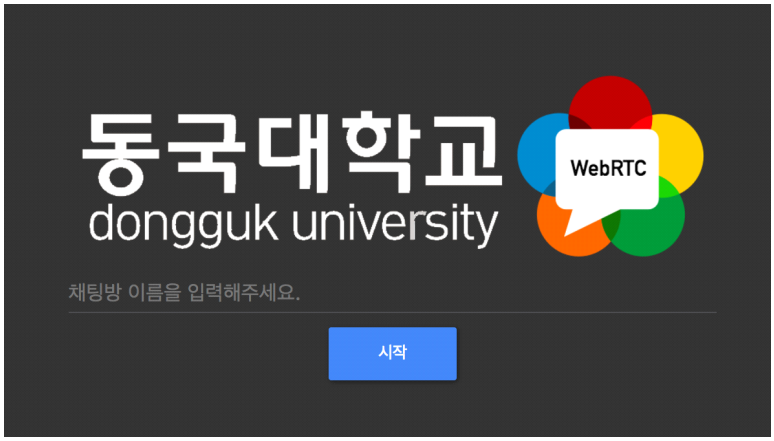
이 프로젝트의 예로 (화상 채팅 프로그램 개발 프로젝트) 서버 측에서는 클라이언트들의 데이터를 처리하는 소켓 프로그래밍 지식, 네트워크에 대한 이해, 네트워크 보안 등에 대해 알아야한다. 클라이언트 측에서는 해당 시스템 아키텍쳐와 운영체제를 고려하여 캠과 오디오를 접근하며 이를 서버측에 보내고 때론 서버 측에서 데이터를 받아 상대방의 모습을 화면에 출력해야한다. 특정 운영체제의 API를 사용해서 어플리케이션을 개발해도 되며, 다양한 GUI 프레임워크를 사용할수도 있다.

만약, 개발자가 서버와 클라이언트 모두 개발을 해야하거나, 서버나 클라이언트 프로그램들이 요구하는 지식의 일부를 가지고 있지 않을 때 어떻게 해결 할 수 있을까? 이 문제를 해결할 수 있는 방법은 바로 공개소프트웨어이다.

이번 과제는 공개소프트웨어를 이용해서 화상 채팅 어플리케이션을 개발한다. 이를 통해 얼마나 쉽고 빠르게 강력한 어플리케이션을 개발할 수 있는지 나타내어, 많은 이들이 공개소프트웨어의 필요성을 느끼게 하는 것이 목적이다.

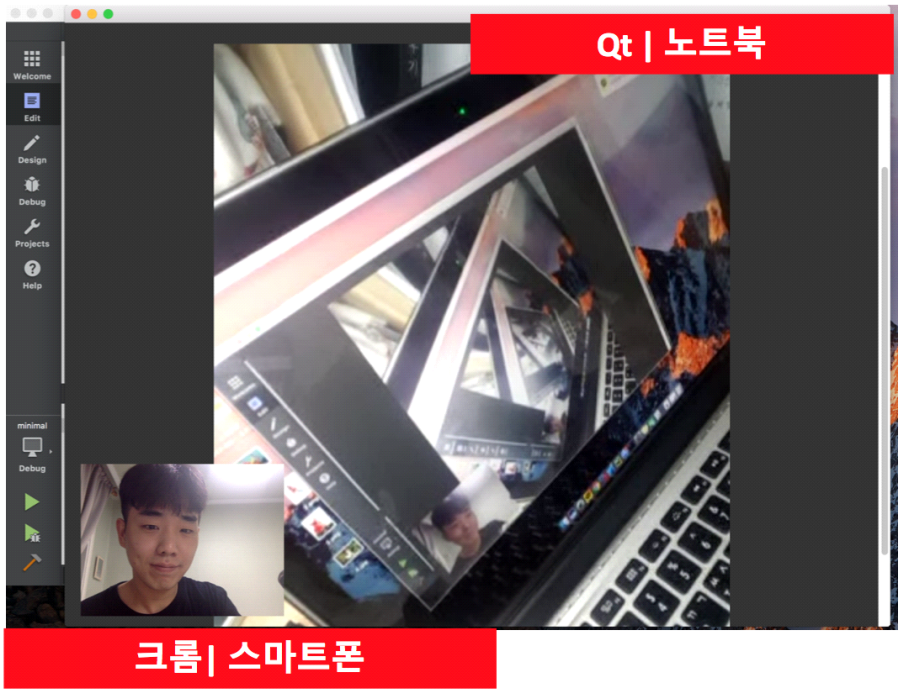
규모가 어느 정도 있는 소프트웨어를 개발할 때 여러 사람들과 함께 협업을 한다. 소프트웨어 개발을 위해 팀을 구성하고 Github를 통하여 협업한다. 이를 통해 소프트웨어 개발에서 협업의 중요성과 Github 편리함을 느끼는 것이 목적이다.

3. 최종 결과물 소개

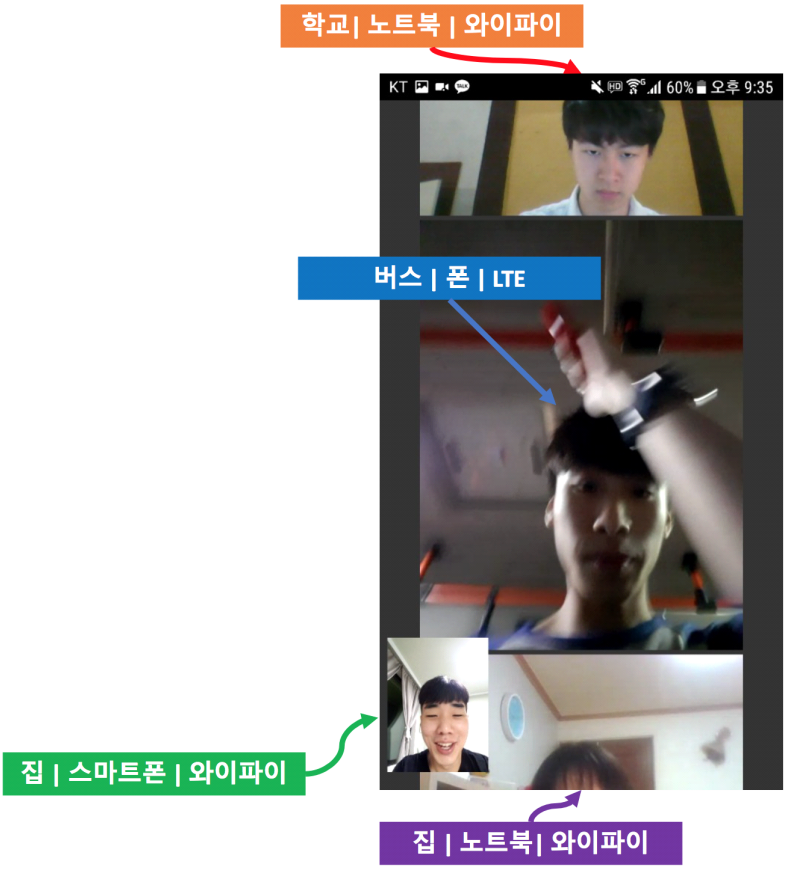


https://rudebono.github.io/dguchat/
접속 화면

채팅방 이름에 원하는 채팅방 이름을 적고 시작 버튼을 누르면 상대방을 기다린다. 상대방도 같은 채팅방 이름을 적고 시작 버튼을 누르면 채팅방에 참여가 된다.



1대1 화상채팅의 예이다. 클라이언트 A는 스마트폰에 크롬을 사용하였고, 클라이언트 B는 Mac OS X 노트북에 Qt 어플리케이션을 사용하여 접속하였다.



다대다 화상채팅의 예이다. 크롬 브라우저를 사용하되, 서로 다른 플랫폼과 네트워크 환경에서 화상채팅을 사용하였다.

WebRTC를 지원하는 브라우저를 사용한다면 서로 다른 플랫폼에서, 다른 네트워크 환경에서 화상채팅을 할 수 있다.

3. 사용된 기술들과 공개 소프트웨어

3조는 화상 채팅 프로그램을 개발하기 위해 WebRTC 기술과 Qt 프레임워크를 사용한다.

WebRTC 기술에 필요한 시그널링 서버 프로그램과 라이브러리를 공개 소프트웨어로 사용한다. 이 공개 소프트웨어는 Muaz Khan이 개발하고 Github에 공개한 RTCMultiConnection이다. (MIT 라이센스)

WebRTC를 지원하지 않는 브라우저에서도 동작할 수 있는 멀티 플랫폼, 하이브리드 어플레케이션 개발을 하기 위해 Qt 프레임워크 (LGPL 무료 라이센스)을 사용한다.

그 외, 시그널링 서버를 구성하기 위해 Heroku 클라우드 서비스를 이용하였으며, 웹 어플리케이션을 보관할 곳은 Github 페이지를 이용하였다. 해당 프로젝트에 대한 소개와 소스코드 설명 및 사용 방법은 Markdown 언어를 이용하여 깃허브에 기술 하였다.

3.1 WebRTC (Web Real Time Communication)

WebRTC는 웹 브라우저 간에 플러그인 도움 없이 상호 통신할 수 있도록 설계된 JavaScript API이다. WebRTC는 구글이 오픈소스화한 프로젝트에서 기원하였으며, 그 뒤로 국제 인터넷 표준화 기구(IETF)가 프로토콜 표준화 작업을, W3C가 API 정의를 진행하였으며, 음성 통화, 영상 통화, P2P 파일 공유, 데이터 통신 등으로 활용 될 수 있다.

WebRTC를 지원하는 웹 브라우저는 크롬, 파이어폭스, 오페라, 엣지 등이 있다. 마이크로소프트사의 인터넷 익스플로러와 애플의 사파리는 지원을 하지 않는다.

WebRTC의 통신 방법은 클라이언트 Alice와 Bob이 시그널링 서버를 통해 Session Description protocol (SDP)을 주고 받고 서버를 거치지 않고 서로 P2P (Peer to Peer) 통신을 시작하게 된다.

P2P 연결이 시작되면 RTCPeerConnection이 클라이언트에 직접 연결하기 위한 시도가 이루어진다. 하지만 실제 네트워크상의 복잡성과 보안상 문제로 인해 P2P 통신을 위해서라면 STUN, TURN 서버를 이용해야한다.

STUN 서버는 외부 네트워크 주소를 얻는데 사용되며, TURN 서버는 직접(P2P) 연결이 실패할 경우 트래픽을 중계하는데 사용된다.

자세한 내용은 https://webrtc.org/ 에서 확인할 수 있다.

3.2 RTCMultiConnection

클라이언트의 화면에 나타내는 웹화면을 Front-End 영역, 클라이언트에서 전달된 데이터 및 서버관리를 Back-End 영역이라고 정의를 한다면,

WebRTC를 이용해 채팅 웹 어플리케이션을 개발 시 채팅 방을 입장하거나 사용자들의 화상 정보를 웹 브라이저에 출력하는 영역을 Front-End 영역, 클라이언트들의 SDP 정보를 교환하는 영역을 Back-End 영역이라고 볼 수 있다. 그래서 우리는 Javascript WebRTC API과 HTML5, CSS3를 이용해서 Front-End를 구성하고 다양한 언어와 Socket 프로그래밍으로 Back-End를 구성할 수 있다.

Front-End 영역은 HTML5, CSS3, Javascript 지식을 어느정도 소유하고있다면 WebRTC API 명세서를 확인해서 충분히 구현을 할 수 있다. Back-End 영역에서는 클라이언트가 1대1 화상채팅만을 한다면 보다 간단하게 구현을 할 수 있지만, 만약 여러 클라이언트 모두 화상채팅을 한다고 하면 이 문제는 절대 간단하게 구현할 수 없다. 수십, 수백, 수천명의 데이터를 처리하고 관리하는 이 문제는 프로그래머의 능력에 따라 달라진다.

우리는 위 문제를 해결 하기 위해 Github에 공개된 RTCMultiConnection를 이용해서 Front-End와 Back-End 영역을 구현했다. RTCMultiConnection은 Muaz Khan이 개발 하였고, MIT 라이센스를 가지고 배포하고 있다.

RTCMultiConnection은 WebRTC API를 한번더 가공해서 RTCMultiConnection Signaling Server. Back-End에 맞게 Front-End Javascript API를 제공한다.

RTCMultiConnecton Signaling Server는 Node.js를 이용해서 다대다 WebRTC 통신을 지원하게 설계되어 있다.

우리는 공개소프트웨어인 RTCMultiConnection을 이용해서 시그널링 서버를 구축하고,

Javascript API를 이용하여 다대다를 지원하는 화상채팅 웹 어플리케이션을 개발하였다.

Muaz Khan은 RTCMultiConnection Signaling Server 또한 미리 구축하여 공개하였으나, 우리는 Muaz Khan의 공개된 시그널링 서버를 사용하지 않고, RTCMulticonnection을 사용하여 Heroku(https://www.heroku.com/)클라우드에 직접 서버를 구축하였다. 구축한 시그널링 서버의 주소는 http://dguchat.herokuapp.com/ 이다.

Front-End 부분은 Github Page에 구축하여 시그널링 서버의 트레픽을 분산 시켰다. https://rudebono.github.io/dguchat/ 에서 서비스를 이용할 수 있으며, 자세한 코드는 https://github.com/rudebono/dguchat/tree/gh-pages 에서 확인 가능하다.

3.3 Qt

Qt컴퍼니에서 개발한 GUI 프로그램 개발에 널리 쓰이는 크로스 플랫폼 프레임 워크이다. GUI 뿐만 아니라 콘솔 프로그램과 같은 비GUI 프로그램 개발에도 사용된다. 주로 C++을 기반으로 하고 있지만, 파이썬, 루비, C, perl, 파스칼과도 연동이 가능하다. 또한 Qt는 "write once, complie anywhere"을 표방하여 하나의 코드로도 다양한 플랫폼에서 빌드가 가능하도록 지원하고 있다. (유료, LGPL, GPL)

WebRTC를 이용해서 웹 어플리케이션을 개발 한다면, 클라이언트들은 필히 WebRTC를 지원하는 브라우저를 이용하여 서비스를 이용해야한다. 하지만 클아이언트들이 WebRTC를 지원하지 않는 브라우저가 없거나, 새로 브라우저를 설치를 하는 것에 대해 반대를 한다면 이 문제는 발생한다. 그렇다면 우리는 클라이언트들의 플랫폼 환경에서 추가 설치되는 브라우저 없이 WebRTC를 지원하는 어플리케이션을 개발하려면 어떻게 해야할까? 새로운 브라우저를 만들 되 필요한 기능만 남겨놓고 (WebRTC 웹 어플리케이션에 접근하고 서비스하는 것.) 다른 기능들을 빼버리면 된다. 하지만 브라우저를 개발한다는 것은 그리 쉬운일이 아니다. 네트워크, 언어처리, 오디오, 그래픽 처리 등 해결해야 문제들이 너무나 많다. 그래서 우리는 브라우저를 개발하기 위해 웹 브라우저 개발 프레임워크를 사용한다. 이 또한 공개 소프트웨어 이다.

공개된 브라우저 개발 프레임워크는 대표적으로 WebKit, Chromium이 있다. 이 프레임워크와 다양한 언어를 합하고 GUI API를 사용해서 클라이언트 플랫폼에 맞는 프로그램을 개발 할 수 있다. 하지만 클라이언트들의 플랫폼은 다양하기 때문에 여러 플랫폼을 하나하나씩 개발하는데 큰 문제가 있다. 그래서 크로스 멀티 플랫폼 프레임 워크인 Qt를 사용한다.

Qt는 qtWebkit, qtWebengine이란 모듈로 WebKit과 Chromium를 동시에 지원하며, 우리는 qtWebengine를 사용하여 하이브리드 어플리케이션을 개발하였다. 하지만 Qt LGPL, GPL 버전은 qtWebKit과 qtWebEngine를 정적링크로 배포를 할 수 없고, qt 라이브러리를 설치해야만 실행 파일을 실행할 수 있다. qt 라이브러리를 설치하는 번거러움이 웹 브라우저를 설치하는 방법보다 높아 소스코드만 공개하였다.

qt에 대한 자세한 내용은 https://www.qt.io/ , WebKit에 대한 자세한 내용은 https://webkit.org/ , Chromium에 대한 제세한 내용은 https://www.chromium.org/ 에서 확인 가능하다.

3.4 Cloud Service

웹 서비스를 개발할 때 문제가 되는게 바로 서버 문제이다. 만약 개인이 집에서 서버를 직접 구축을 한다고 하면 고려할 사항이 매우 많다. 서버의 하드웨어 구성, 운영체제, 네트워크, 보안, 프로그램 등이 있다. 기존에는 많은 개발자들이 호스팅 서비스를 이용해서 웹 서비스를 개발하였지만 이는 그리 효율적이지 않다. 대표적으로 호스팅 서비스의 확장성이다. 만약 급격하게 서버에 접근하는 클라이언트들의 수가 많아져 하드웨어 자원을 늘릴려고 하면 이는 매우 시간이 오래걸리며, 기존의 서버가 과부화를 일으켜 서비스가 정지될 수 있다. 이 문제는 클라우드 서비스를 이용하면 해결할 수 있다.

클라우드는 수 많은 하드웨어 자원을 묶어 놓고 고객이 원하는(원하는 자원, 원하는 금액 대비) 가상의 영역을 (인스턴스) 제공하는 개념이다. 고객이 원하는 특별한 자원을 제공하거나 원하는 서비스만 제공하는 차이에 따라 IaaS, PaaS, SaaS 등 다양한 서비스로 나누어진다. (더 이상 자세한 설명은 하지 않는다.)

WebRTC 기술을 사용하려면 HTTPS 프로토콜을 지원해야한다. 조사해 본 결과 일반 호스팅 서비스들은 HTTPS 프로토콜 지원에 별도의 요금이 부과되고, 클라우드에는 기본적으로 HTTPS를 지원한다. 대표적으로 Google App Engine과 Heroku가 있다. Google App Engine은 사용한 만큼 요금을 내야하며 Heroku는 월별 요금제이다.

우리 3조는 Heroku 클라우드 서비스를 이용하였으며, 무료 요금제를 사용한다. 한달에 550시간 서비스를 사용할 수 있으며, 자세한 하드웨어 스펙은 알 수 없으나, 500MB의 램 메모리를 제공한다. Node.js 엔진을 선택하여 RTCMultiConnection 시그널링 서버를 구축하였으며 단순히 소켓을 이용해서 시그널링 처리만 하기 때문에 큰 부담은 없을꺼라 예상한다.

Heroku 에 대한 자세한 내용은 https://www.heroku.com/에서 확인 할 수 있다.

3.5 Github page

깃허브에서 제공하는 서비스이다. 사용자명.github.io 라는 저장소를 생성 후 HTML이나 Markdown언어로 웹페이지를 제작한다면 바로 홈페이지를 만들 수 있다.

깃허브 계정안에 다양한 저장소가 존재하기 때문에 각각의 저장소의 페이지를 생성할 수 있다. 해당 저장소에 대해 gh-pages 라는 브랜치를 생성 후 해당 브랜치에 웹 페에지를 제작하면 특정 프로젝트에 대한 홈페이지를 만들 수 있다.

우리 3조는 HTML, CSS, JS 파일들을 시그널링 서버와 불리하여 깃허브 페이지에 업로드하였다. 서버의 트래픽을 줄이고, 여러 개발자들이 좀 더 쉽게 소스코드를 볼 수 있게 하기 위함이다.

github page에 대한 자세한 내용은 https://pages.github.com/에서 확인 할 수 있다.

3.6 Markdown

마크다운(markdown)은 일반 텍스트 문서의 양식을 편집하는 문법이다. README 파일이

나 온라인 문서, 혹은 일반 텍스트 편집기로 문서 양식을 편집할 때 쓰인다. 마크다운을 이용해 작성된 문서는 쉽게 HTML 등 다른 문서형태로 변환이 가능하다.

협업을 하는 도중 문서를 제작해야하는 경우가 생길 때를 생각해보자. 만약 hwp, doc, pdf 등을 이용해서 파일을 수정하고 업로드한다면 단순히 파일만 업로드 되기 때문에 깃허브에서 어느 부분을 수정, 추가, 삭제 되었는지 알 수가 없다. HTML를 사용하면 문서 작성에 필요한 부가적인 요소가 너무 많아지게 된다. 이러한 문제는 마크다운 언어를 사용해서 해결할 수 있다.

Markdown의 자세한 내용은 https://daringfireball.net/projects/markdown/에서 확인 할 수 있다.

4. 결과 소스 공개

모든 소스코드, 문서를 Github (https://github.com/rudebono/dguchat)에 모두 공유하였다.

이 과제에서 프로그램 구현 부분은 총 3가지로 나눌 수 있다. RTCMultiConnection를 이용한 시그널링 서버, RTCMultiConnection API를 이용한 Front-End, Qt를 이용한 하이브리드 어플리케이션이다.

시그널링 서버 (Back-End)에 대한 소스코드와 자세한 내용은
https://github.com/rudebono/dguchat/tree/master/server에서 확인할 수 있다.

화상 채팅 웹페에지 (Front-End)에 대한 소스코드와 제세한 내용은
https://github.com/rudebono/dguchat/tree/gh-pages에서 확인 할 수 있다.

Qt를 이용한 하이브리드 어플리케이션에 대한 소스코드와 자세한 내용은
https://github.com/rudebono/dguchat/tree/master/client에서 확인 할 수 있다.

과제 진행을 하면서 제출했던 모든 문서들은
https://github.com/rudebono/dguchat/tree/master/homework 에서 확인 가능하다.

5. 프로젝트 추진 과정

프로젝트 추진 과정은 팀 회의와 소스코드 개발 부분 (commit)으로 나누어 설명한다.

5.1 Meeting

2017. 05. 20 - 최종 제안서 제출, 1차 중간 발표
2017. 05. 31 - 2차 중간 발표
2017. 06. 07 - 3차 중간 발표
2017. 06. 14 - 결과 보고서 제출, 최종 발표

매주 화요일 실습시간, 보고서 제출 및 발표 3일 전 오프라인 회의,
카카오톡 및 Github를 이용하여 온라인 회의 진행하였다.

아르바이트 및 불가피한 상황 이외 모든 팀원들이 적극 참여하였다.

5.2 Commit

master 브랜치 - https://github.com/rudebono/dguchat/commits/master
gh-pages 브랜치 - https://github.com/rudebono/dguchat/commits/gh-pages

그 외, pulse, graphs 확인 하여 좀 더 자세한 내용들을 볼 수 있다.

pulse - https://github.com/rudebono/dguchat/pulse
graphs - https://github.com/rudebono/dguchat/graphs/contributors

6. 기대효과

Open Source, WebRTC, Qt, Markdown, Cloud 등 위에서 입이 마르도록 칭찬을 했다. 하지만, 이런 사회공헌, 기술적인 부분 말고 상업적으로 따졌을 때 3조의 프로젝트가 어떤 기대 효과가 있는지 설명한다.

국내에 WebRTC 기술을 한번 더 가공해서 (예: RTCMultiConnection의 서버와 API) 제공해 주는 업체가 있다. SK telecom 사의 PlayRTC이다.

기본 API 호출은 1,000회 이며 요금제에 따라 호출량이 달라진다. 또한 상용 서비스를 하기 위해서 SK 측에 비용을 지불 해야한다.

개발자가 대규모 서비스를 구축한다면, SK 회사가 제공하는 솔루션을 이용하는 것이 오히려 나을 수 도있다. 품질 관리나 기본적으로 고화질의 영상 및 데이터를 교환해도 문제가 없는 국내 STUN, TRUN 서버를 제공하기 때문이다. (사실 이 서버가 공개된 STUN, TURN 서버들의 성능과 어떠한 차이가 있는지 모른다. 예 : Google STUN/TURN Server)

만약, 소, 중 규모의 서비스를 개발해야 한다면 우리가 사용했던 기술들과 공개소프트웨어를 사용한다면 비용을 매우 저렴하게 들여 서비스를 구축할 수 있고 생각한다. 무료라고 기능이 적은 것도 아니다. PlayRTC API의 기능을 살펴본 결과 아직까진 RTCMultiConnection API의 기능과 별 다른 차이점이 없다.

Heroku 클라우드의 유료 서비스 중 hobby (한달 7$) 무제한 요금제를 선택해서 시그널링 서버를 구축하고, 원하는 어플리케이션을 제작한다면 스타트업 규모의 비즈니스는 cover가 가능하다고 기대한다.

playrtc에 대한 자세한 내용은 https://www.playrtc.com/ko/home-ko/ 에서 확인 가능하다.

7. 느낀점 및 팀원 평가

프로젝트를 진행하면서 교수님께서 항상 강조하신 내용들이 이제야 깨닫게 되었습니다. 이번 프로젝트의 느낌점을 한마디로 이야기하자면, "설계 과목 주니어 디자인 프로젝트가

왜 공개 소프트웨어 프로젝트로 이름이 바뀐것에 대해 이해하였다."입니다.

팀원들과 협업에서 초반에는 다들 프로젝트에 대해 어려움을 느껴 점점 포기를 하는 모습을 보았지만, 대화를 통해 해결을 하였고 끝까지 노력해서 모두 함께 프로젝트를 완수하였습니다.

누가 코드를 더 많이 짰고, 신 기술을 사용했고는 의미 없는 것 같습니다. 모두 정말 열심히 했었고 끝까지 믿고 따라준 팀원들에게 고맙다는 말을 전해주고 싶네요.

그리고,
한 학기 동안 좋은 수업 해주신 김동호 교수님과 조교님들께 감사의 인사 드립니다.