다중 스트림 믹싱을 지원하는 WebRTC 클라이언트 구조 설계

Architecture for WebRTC Client Supporting Multi-Stream Mixing

임준혁
소프트웨어학과
아주대학교 소프트웨어융합대학
대한민국경기도수원시
wnsgurl97@ajou.ac.kr

정준호
소프트웨어학과
아주대학교소프트웨어융합대학
대한민국경기도수원시
cak101435@ajou.ac.kr

노병희*
소프트웨어학과
아주대학교소프트웨어융합대학
대한민국경기도수원시
bhroh@ajou.ac.k

* 교신저자

요 약

WebRTC 를 사용한 다자간 화상회의 시스템 구축 시 사용할 MCU 서버와 단일 미디어 스트림을 주고 받기 위한 다중 스트림 믹싱을 지원하는 WebRTC 클라이언트를 구현한다. 다중 스트림 믹싱이 필요한 이유는 해당 클라이언트가 여러 미디어 디바이스들로부터 미디어 스트림을 제공받기 때문이다. 클라이언트는 제공받은 1 개 이상의 미디어 스트림을 하나의 단일 미디어 스트림으로 가공하여 MCU 서버에게 제공해야 하며, 해당 논문은 이 방식에 대한 해결책을 발견하는 과정을 담고 있다.

키워드 : WebRTC, Client Mixing, 다자간회의, 웹브라우저

1. 서론

WebRTC(Web Real-Time Communication)는 실시간 통신을 위한 오픈소스 프로젝트로, 웹 브라우저와 모바일 애플리케이션 간의 직접적인 통신을 가능하게 한다. 이 기술은 오디오, 비디오, 그리고 고급형 데이터 공유를 위한 API 와 프로토콜을 제공하며, 이를 통해 사용자는 플러그인이나 별도의 소프트웨어 없이도 실시간 통신이 가능한 웹 애플리케이션을 이용할 수 있다. 이와 같이 WebRTC 는 사용자 경험을 크게 향상시키는 측면에서 중요한 도구로 인식되고 있다[1].

그러나, 이러한 혁신적인 기술에도 불구하고, WebRTC 를 지원하지 않는 장치들이 여전히 존재하고 이로 인한 통신의 제약은 여전히 존재한다. 특히 낮은 가격대의 장치들, 일부 레거시 시스템, 또는 특정 국가에서 널리 사용되는 브라우저들은 여전히 WebRTC 를 지원하지 않는 경우가 많다[2]. 이러한 장치들이 제외되는 것은 실시간 웹 통신의 전역적 표준화를 저해하고, 특히 저소득 국가의 사용자들에게 불평등을 초래할 수 있다[3].

과거에는 이 문제를 해결하기 위해 다양한 방법이 제안되었지만, 대부분의 경우 서버 기반의 솔루션을 사용했다. 이는 서버를 중간에 두고 통신을 전달하는 방식으로, 추가적인 네트워크 지연을 초래하며, 개인 정보 보호와 보안 문제를 야기할 수 있다[4]. 이러한 문제를 피하기 위해, WebRTC 를 지원하지 않는 장치를 효과적으로 수용하는 새로운 방법을 연구하고 개발하는 것이 중요하다.

본 논문에서는 이러한 다양한 디바이스로부터 추출한 미디어 스트림을 WebRTC 를 지원하는 WebRTC Client 에게 전달하고, WebRTC Client 는 해당 미디어 스트림들을 단일 미디어 스트림으로 믹싱하여 MCU 서버와 단일 uplink, downlink 로 주고 받기 위한 방법에 대한 연구를 제안한다.

2. WebRTC 개요

WebRTC(Web Real-Time Communication)는 고유한 특징들을 지닌 웹 기반의 실시간 통신 기술이다. WebRTC 의 첫 번째 특징은 별도의 플러그인이나 추가 소프트웨어 없이 웹 브라우저 간에 오디오, 비디오, 데이터를 주고받을 수 있다는 것이다. 이로 인해 사용자는 어플리케이션의 설치 없이 웹에서 직접 통신할 수 있게 된다.

WebRTC 는 실시간 통신이 가능하다는 점이 있다. 이를 통해 화상 회의, 라이브 스트리밍 등의 실시간 서비스를 웹에서 바로 제공할 수 있게 되어, 사용자 경험을 대폭 향상시킬 수 있다.

WebRTC 는 Peer-to-Peer(P2P) 통신을

지원합니다. 이는 사용자 간에 직접적인 연결을 가능하게 함으로써, 통신의 효율성과 속도를 높이며 서버 부하를 줄일 수 있습니다.

WebRTC 는 모든 통신에 대한 암호화를 제공한다. 이를 통해 통신의 보안성이 크게 향상되며, 데이터의 무단 사용이나 유출을 방지할 수 있다.

WebRTC 는 다양한 플랫폼에서의 활용이 가능하다는 장점이 있다. 웹 브라우저뿐만 아니라 모바일 애플리케이션에서도 WebRTC 를 활용할 수 있어, 다양한 환경에서의 실시간 통신이 가능하다. 이와 같은 특징들로 WebRTC 는 다양한 분야에서의 활용이 가능하다.

3. WebRTC 클라이언트 구조 설계

3.1. 시스템 모델

WebRTC Server 와 WebRTC Client, Media Device 간의 연결 구조는 그림 1 과 같다.

- Media Device 들에서 영상 및 음성 Media Stream 데이터를 추출한다. Media Device 에는 CCTV, 무인 드론, 홀로렌즈와 같이 영상 및 음성 데이터를 추출할 수 있는 다양한 Device 가 해당된다.
- Media Device 들은 WebRTC Client 와의 TCP connection 을 통해 Media Stream 을 WebRTC Client 에게 직접 전달한다.
- WebRTC Client 는 전달받은 Media Stream 을 단일 Stream 으로 믹싱하여 WebRTC Server 에게 전달하게 된다.

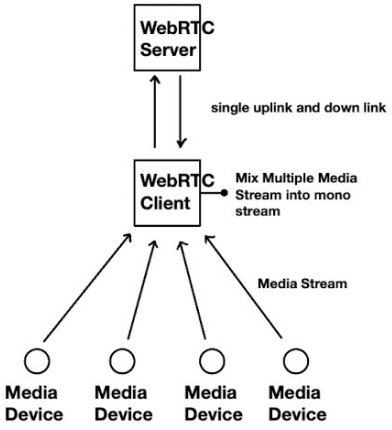

그림 1. WebRTC Server, WebRTC Client, Media Device 연결구조

3.2. 제안 시스템 구조

MCU Client 에서 Media Device 들을 연결하는 방법을 그림 2 에 나타내었다.

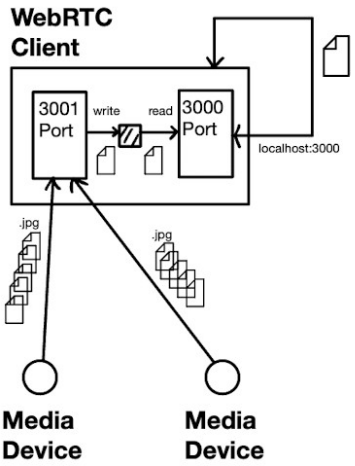

그림 2. WebRTC client - Media Device 간 연결

구현을 위해 고려되어야 하는 가장 중요한 사항은 두가지이다.

- WebRTC 를 지원하지 않는 디바이스로부터 미디어 스트림을 전달받을 수 있어야 한다.
- 전달받은 미디어 스트림을 하나로 묶을 수 있어야 한다. 이 두가지 조건에 부합하는 클라이언트 시스템을 만들기 위해서는, WebRTC 클라이언트가 Media Device 들의 서버 역할을 수행해야 한다. 무엇보다 웹 어플리케이션 상에서는 브라우저 정책으로 인해

Media Stream 을 믹싱하는 것이 불가능하다.

따라서, WebRTC Client 의 특정 포트에서 동작하는 서버 어플리케이션을 통해 여러 Media Device 들로부터 Media Stream (영상 데이터의 이미지 프레임)을 전달받아야 한다. 이후, 해당 서버 어플리케이션은 받은 Media Stream 들을 하나로 묶어 가공하여 WebRTC Server 로 보내게 된다.

Media Device 들에서 동작하는 응용 프로그램들은 TCP Socket 통신을 활용하여 WebRTC Client 로 Media Stream 을 보내게 된다.

이 방식을 통해 WebRTC 프로토콜을 이용할 수 없는 여러 Media Device 들이 WebRTC Client 를 통해 간접적으로 WebRTC 프로토콜을 이용하는 효과를 얻게 된다.

4. 구현 전략

WebRTC Client 에서 동작하는 Media Device 기준 서버 프로그램과, Media Device 에서 동작하는 프로그램을 만들어야 한다.

WebRTC Client 에서 동작하는 서버 프로그램은 두가지 핵심 역할을 수행해야 한다.

- 한 개이상의 Media Device 와 TCP 기반 Socket 연결을 맺은 뒤, 각 Device 들로부터 받아온 여러 개의 Media Stream 을 하나로 mixing 하는 역할을 담당해야 한다.
- 해당 서버 프로그램으로 브라우저가 웹 페이지를 요청할 경우, 서버 프로그램은 현재 전달되고 있는 미디어 스트림을 사용자에게 보여줄 수 있는 웹 페이지를 브라우저에게 전달해주어야 한다.

Media Device 에서 동작하는 프로그램은 WebRTC Client 에서 동작하고 있는 서버 프로그램의 IP 주소와 포트 번호로 자신이 생성하고 있는 Media Stream 을 전송할 수 있어야 한다.

5. 결론 및 향후 연구

그림 2 에서 제시한 구조의 시스템 일부를 현재 구현한 상태이다. 즉, 음성을 제외한 영상 데이터의 Media Device 에서 WebRTC Client 까지의 전달과 믹싱 과정을 구현한 상태이다.

Media Device 에서 동작해야 하는 응용 프로그램은 임베디드 시스템에서의 동작에 대한 염두 하에 C++로 작성하였다. WebRTC 에서 동작하는 서버 프로그램은 Node.js 를 기반으로 작성하였다.

먼저 Node.js 기반의 WebRTC Client 서버 프로그램을 실행시키면, 3000 번 포트와 3001 번 포트에서 각각 다른 역할을 담당하는 서버 프로세스가 실행된다. 먼저 3001 번 포트에서는 TCP 기반 socket 연결을 바탕으로 각 Media Device 들로부터 입력 영상의 프레임 이미지 데이터를 받아오는 프로세스가 실행된다. 해당 프로세스는 Media Device 들로부터 받아온 이미지 데이터들을 /videos 디렉터리에 write 하는 작업을 수행한다. 3000 번 포트에서는 HTTP 서버가 동작한다. 해당 HTTP 서버는 Pug Template Engine 으로 사용자에게 Media Device 들로부터 수신 받은 영상을 확인할 수 있는 웹 페이지 기반의 인터페이스를 생성, 제공한다. 또한, /videos 디렉터리에 지속적으로 write 되는 영상 프레임 데이터를 하나로 믹싱하는 작업도 수행한다.

실제 시스템을 동작 시킨 결과 WebRTC Client 에서 확인해볼 수 있는 사용자 인터페이스는 그림 3, 그림 4 와 같다.

그림 3. 연결된 Media Device 가 없는 상태

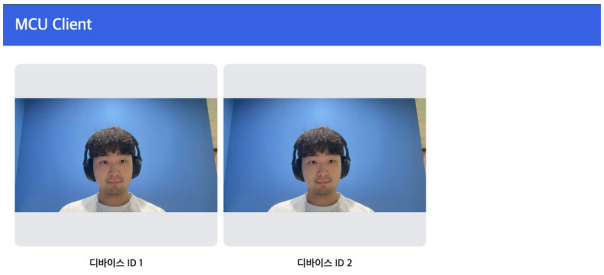

그림 4. 연결된 Media Device 가 있는 상태

/videos 디렉터리에 시시각각 저장되는 영상 프레임 데이터들을 0.1 초에 한번씩 가져와 <img> 태그에 갈아 끼워 넣음으로써 영상처럼 보이도록 하였다. 또한, 1 초에 한번씩 연결된 Media Device 의 목록 정보를 갱신하도록 하여 인터페이스에 실시간성을 부여하였다.

그림 5 는 이미지가 믹싱된 결과이다.

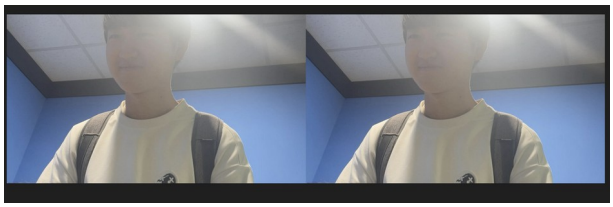

그림 5. 믹싱된 여러 영상 프레임

이러한 결과를 바탕으로, 향후에는 영상 데이터 뿐만 아니라 음성 데이터까지 Media Device 로부터 WebRTC Client 로 전달하는 방법을 강구할 계획이다. 또한, 영상 프레임 수신 및 /videos 디렉터리에의 저장 후 믹싱 과정에서 딜레이가 심각하게 발생하는 것을 확인하였다. 이를 해결하기 위해 영상 프레임의 저장 및 믹싱 과정에 멀티 스레딩을 적용하는 등의 다른 대안을 모색하여 문제를 해결할 예정이다.

Acknowledgement

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터육성 지원사업의 연구결과로 수행되었음" (IITP-2023-2018-0-01431)

참 고 문 헌

[1] J.Uberti. (2012) Real-time communication in the web: Webrtc. [Online]. Available: https://datatracker.ietf.org/meeting/83/materials/slides-83-rtcweb-7.pdf

[2] E. S. G. Almeshekah and M. Atallah.(2015) Improving security using deception techniques. [Online]. Available: https://www.cerias.purdue.edu/assets/pdf/bibtexarchive/2015−13.pd

[3] J. O. M. Bagnulo and S. Perreault.(2010) Nat revelations in peer-to-peer communication. [Online]. Available: https://ieeexplore.ieee.org/document/5447491

[4] H. W. S. Hu and Q. Li. (2016) A survey on internet performance measurement platforms and related standardization efforts. [Online]. Available: https://ieeexplore.ieee.org/document/7502158