WebRTC 를 사용한 다자간 회의 지원을 위한 MCU 구조 설계

MCU Design For Multi-party Conference Support using WebRTC

정준호
소프트웨어학과
아주대학교 소프트웨어융합대학
대한민국경기도수원시
cak101435@ajou.ac.kr

임준혁
소프트웨어학과
아주대학교소프트웨어융합대학
대한민국경기도수원시
wnsgurl97@ajou.ac.kr

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

요 약

WebRTC 기술은 비대면 환경에서 웹 브라우저만으로 실시간 비디오, 오디오, 데이터 처리와 전송이 가능하여, 다중 사용자간 실시간 커뮤니케이션 구현에 매우 적합하다. 본 논문에서는 WebRTC 에서 MCU (Multipoint Communication Unit)를 오픈소스를 활용하여 효율적인 서버 로드 분산과 클라이언트의 부하 감소가 가능한 구조를 제안한다. 제안 방법은 성능이 제한된 스마트 기기들 (ex. 홀로렌즈, 스마트 워치 등)에서의 화상회의, 혹은 대규모 원격 협업응용에도 적용 가능하다.

키워드 : WebRTC, MCU, 다자간회의, 웹브라우저

1. 서론

최근 비대면 원격 회의 및 강의 수요가 급증함에 따라 다양한 화상회의 플랫폼들이 등장하면서, 이의 시장 규모도 약 5 배 증가하였다[1].

WebRTC 는 디바이스 환경의 제약을 받지않고 별도 프로그램 설치 없이 웹 브라우저만으로 실시간 화상회의를 가능하게 해준다[2]. WebRTC 는 오디오 및 비디오 데이터 캡처 및 처리, 시그널링, 코덱 협상 등 다양한 기능을 제공한다. WebRTC 는 기본적으로 P2P 방식으로 동작하므로, 참가자 수가 소수인 경우는 네트워크 트래픽을 최소화하는 장점을 갖지만, 참가자수가 증가할 수록 오디오 및 비디오 데이터 전송에 필요한 대역폭이 기하급수적으로 증가하는 문제가 있다. 이를 위하여, 대규모 사용자 환경에서도 적용가능한 MCU(Multipoint Communication Unit) 구조가 적용된다[3]. 그러나, 여전히 안정적 기능을 제공하는 MCU 미디어 서버는 오픈 소스로 제공되지 않고 있으며, 이는 MCU 차원의 성능문제와 네트워크 트래픽 문제, 방화벽 문제 등 다양한 문제가 발생할 가능성이 있다.

본 논문에서는 WebRTC 환경에 적용할수 있고, 오픈소스 프로그램을 통해 설계와 구현이 가능한 MCU 구조를 제안한다. 제안하는 구조는 서버 부하 분산을 통하여 효과적인 다자간 회의 환경을 지원할수 있다.

2. 관련연구

WebRTC 에서 기본적으로 메시지를 교환하는 방식은 P2P 와 그림 1 에 보인 바와 같은 SFU (Selective Forwarding Unit) 이다. SFU 에서는 각 참여자는 자신의 비디오 및 오디오 데이터를 SFU 서버로 보내고 다른 참여자들의 스트림들을 중앙 서버로부터 수신한다. SFU 는 P2P 에 비해 확장성이 늘어나긴 했지만, 참가자 수가 증가하면, 여전히 확장성의 문제를 갖는다.

그림 1. SFU 아키텍처개념도

MCU 는 그림 2 와 같이 중앙 미디어 서버에서 각 참여자의 미디어 스트림을 수신하여 디코딩한 다음 단일 스트림으로 혼합하여 각 참가자에게 송신하는 방식이 사용된다.

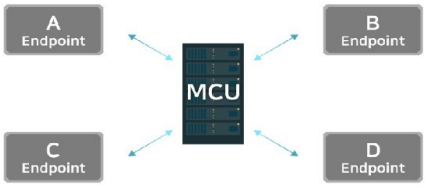

그림 2. MCU 아키텍처개념도

* 교신저자

3. 제안방법

3.1. 시스템 모델

그림 3 은 본 논문에서 제안하는 시스템 환경이며, Application Client(AC), 미디어 서버, Application Server (AS)로 구성되어 있다.

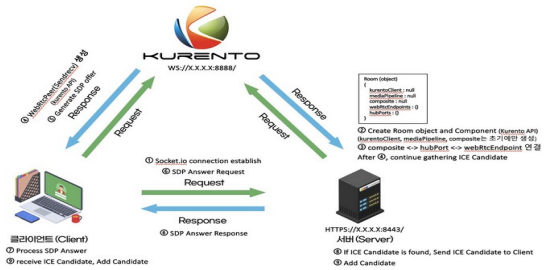

그림 3. AC-AS-KMS 통신구조

AC 는 사용자에 해당한다. 미디어 서버는 클라이언트 간의 통신을 중개하고, 미디어 스트림을 처리하여 각 클라이언트에게 중계하는 역할을 수행하며 Kurento Media Server(KMS) [4]를 고려한다. AS 는 WebRTC 핸드셰이크와 KMS 와의 통신을 담당한다.

AC 에서 화상회의 플랫폼에 접속하여 들어가고자 하는 세션을 입력하면 socket.io connection 이 만들어지고 [그림 3-①], Server 에게 join 메시지를 보낸다. AS 는 클라이언트가 들어가고자 하는 세션이 존재하는 지 확인하고, 세션이 있을 경우 해당하는 세션의 MediaPipeline 에 접속할 클라이언트의 WebRtcEndpoint 와 WebRtcEndpoint 가 연결될 Composite 의 hubPort 를 생성하여 할당해준다. [그림 3-②] 만약 들어가고자 하는 세션이 없을 경우, 새로운 room 객체를 생성하여 새로운 KurentoClient, MediaPipeline, Composite 를 KMS 에게 요청한다. 이후 마찬가지로 WebRtcEndpoint 와 HubPort 를 생성하여 접속한 클라이언트에게 할당해준다. 이후 할당된 WebRtcEndpoint 와 HubPort, Composite 를 연결하여 미디어 스트림이 WebRtcEndpoint 로부터 Compoiste 까지 흐를 수 있도록 한다. [그림 3-③] 이후 서버는 계속해서 클라이언트와 연결을 맺기 위한 네트워크 정보 후보인 ICE Candidate 를 수집한다. 그와 동시에 클라이언트 측은 KMS 로 WebRtcPeer 를 생성하는 요청을 보다. [그림 3-④] 이렇게 생성된 WebRtcPeer 에서 SDP offer 를 생성하고 서버로 보내면 [그림 3-⑤], 서버는 해당 소켓 ID 에 해당하는 WebRtcEndpoint 에서 SDP offer 를 처리하고, SDP answer 를 생성한 뒤 다시 클라이언트의 WebRtcPeer 에게 보내준다. [그림 3-⑥] WebRtcPeer 는 서버로부터 받은 SDP answer 를 처리하여 서버단의 WebRtcEndpoint 와 연결을 맺는다. [그림 3-⑦] 클라이언트와 서버 사이의 연결이 수립된 뒤 둘 사이의 적절한 네트워크 후보를 찾았다면 이를 교환하여 [그림 3-⑧] 서버와 클라이언트 모두 해당 후보를 등록하도록 한다. [그림 3-⑨] ICE 후보를 등록하는 과정까지 거치게 되면 참여자들은 웹 브라우저를 통해 자신의 미디어 스트림을 서버로 송수신할 수 있게 된다.

서버는 KMS 를 통해 MediaPipeline 내에서 WebRtcEndpoint 를 HubPort, Composite 등 MediaPipeline 미디어 노드를 통해 클라이언트의 미디어 스트림 흐름을 관리한다.

3.2. 제안 구조

본 논문에서 제안하고자 하는 MCU 구조는 다음과 같다. 기본적으로 Socket.io 와 Kurento API 를 활용하여 구현되며, WebRtcPeerSendrecv, WebRtcEndpoint, HubPort, Composite 의 미디어 노드를 사용한다. WebRtcPeerSendrecv 는 클라이언트 측에서 Kurento API 로 생성되며 클라이언트의 local stream 을 캡처하고, WebRTC 서버에 표준화된 방식으로 전송하며, 서버에서 전송된 remote stream 을 다시 출력한다. WebRtcEndpoint 는 서버측에서 클라이언트와 연결되는 KMS 의 Endpoint 이며, 클라이언트의 미디어 스트림을

받고 처리한다. 각 클라이언트는 각각 하나의 WebRtcEndpoint 객체와 연결된다. Composite 는 MCU 를 구현하는데 있어서 핵심적인 미디어 요소로, 여러 미디어 입력 스트림을 하나의 출력 스트림으로 혼합하는데 사용된다. Composite 내부에서 각 입력 스트림은 HubPort 를 통해 포워딩되며, 출력스트림 또한 HubPort 인스턴스를 통해 전송된다. 각각의 미디어 요소를 활용한 흐름도는 그림 4 와 같다.

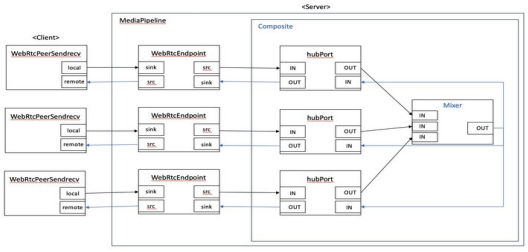

그림 4. Media Stream 흐름구조

사용자가 화상 회의에 참여하면, 클라이언트 측의 WebRtcPeerSendrecv 객체가 생성되어 카메라와 마이크에서 미디어 스트림을 캡처한다. 이후 클라이언트와 KMS 사이의 WebRTC 연결이 설정되고, WebRtcEndpoint 인스턴스가 서버 측에서 생성되며, 클라이언트와 연결된다. 새로운 참가자가 회의에 참여할 때마다, Composite 노드에서 HubPort 객체를 생성하고 참가자의 WebRtcEndpoint 객체에 연결된다. Composite 는 각 클라이언트와 연결된 HubPort 객체들로부터 미디어 입력 받고, 하나의 출력 미디어 스트림으로 혼합하여 다른 참가자들에게 전송한다.

4. 실험결과

4.1. MCU 구조 예시

본 논문에서 제안한 WebRTC 를 사용한 MCU 웹 화상회의는 디바이스의 환경이나 다른 플러그인에 구애받지 않고 WebRTC 를 지원하는 모든 웹 브라우저를 통해 진행할 수 있다.

클라이언트가 화상회의 웹사이트에 접속하면 서버와 소켓통신이 시작되고, 회의 참여자가 참여하고자 하는 세션과 자신의 이름을 입력하면 서버가 참여자의 소켓을 해당 세션에 등록하여 화상회의를 시작할 수 있다. 참여자는 왼쪽의 localVideo 을 통해 자신의 localStream 을 확인할 수 있다. 오른쪽의 remoteVideo 는 자신을 포함한 다른 참여자들의 localStream 들이 혼합된 하나의 미디어 스트림을 서버로부터 받아 재생한다. 각각의 Video 는 전체화면으로 재생할 수 있으며, audio 를 끄거나 킬 수 있다. 아래 그림 5 는 2 명의 컴퓨터 참가자, 1 명의 모바일 참가자가 MCU 회의을 하고있을 때의 예시이다.

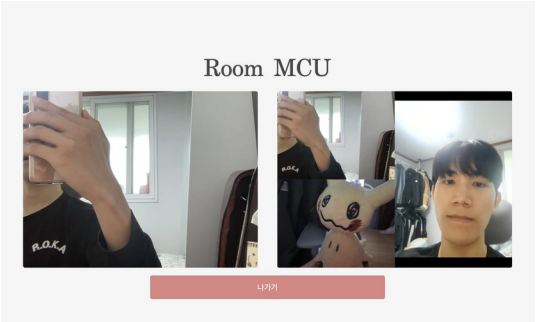

그림 5. local stream 과 혼합된 remote stream

이로써 모든 클라이언트는 단 2 개의 uplink 와 downlink 를 통해 모든 회의 참여자의 영상을 클라이언트의 부하 없이 수신할 수 있다.

4.2. SFU/MCU 성능 분석

아래 그림 6 은 SFU 구조 Kurento 기반 API, Openvidu 를 활용한 화상회의에 접속하여 총 9 명의 참가자의 화면이 모두 등장할 때까지 브라우저에서 측정한 트래픽이다.

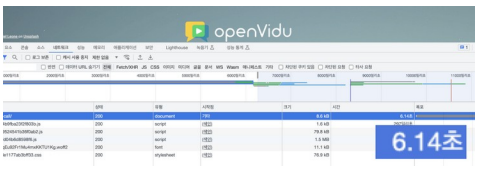

그림 6. SFU 9 인 원격 회의 트래픽 측정

그림 6 에서 마지막 9 번째 참가자의 화면이 등장할 때까지 총 6.14 초가 걸렸음을 확인할 수 있다. 아래 그림 7 은 본 연구에서 구현한 MCU 구조 화상회의에서 9 명의 참가자가 포함된 하나의 영상이 도착하기 까지 측정한 브라우저 환경의

트래픽이다.

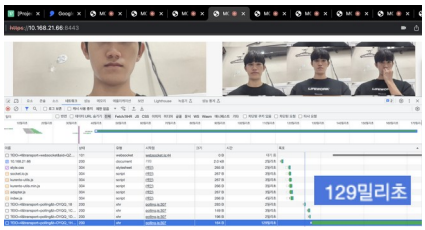

그림 7. MCU 9 인 원격 회의 트래픽 측정

첫 영상이 도착하기까지는 약 0.13 초가 걸렸음을 확인할 수 있다. 두 결과를 종합해보았을 때, MCU 구조의 화상회의가 클라이언트의 부하를 효과적으로 감소시킨 것을 확인할 수 있다.

5. 결론

본 연구를 통해, WebRTC 기반의 MCU 구조를 Kurento 와 socket.io 등 다양한 오픈소스 프로그램을 활용하여 성공적으로 구현하였다. 이러한 구조를 통해 대규모 트래픽의 화상회의를 원활하게 구현할 수 있으며, 성능 제한이 있는 차세대 스마트 기기(ex. 스마트 워치, 홀로렌즈 등)에서도 원격 회의를 효과적으로 진행할 수 있다.

최근 다양한 웹 기반 회의 및 채팅 플랫폼이 등장하였다. 많은 사용자들이 참가하는 환경에서 MCU 구조의 구현은 실시간 원격 회의에 있어 대규모 트래픽을 처리할 수 있는 핵심 솔루션이 될 수 있다. 본 연구에서는 이러한 MCU 구조의 기술적 측면을 최적화하고 실제 서비스 환경에 적용하여 대규모 원격 회의 및 차세대 기기 사이의 상호작용을 지원하고자 하였다. 또한 클라이언트 측의 부하가 줄어든다는 강점을 통해 차세대 스마트 기기와 원격 콜라보레이션을 고려한 웹 기반 회의 솔루션의 개발이 활발히 이루어질 것으로 예상되며, 본 논문에서 제시한 MCU 구조는 이러한 기기에서의 화상 회의를 더욱 쉽고 신속하게 진행할 수 있는 기반으로 활용될 수 있을 것이다. 마지막으로, 이렇게 구축된 WebRTC 기반 MCU 구조를 통한 대규모 트래픽 솔루션은 향후 비대면 환경 업무를 도입하는 산업 환경의 규모가 증가함에 따라 실시간 원격 회의와 비대면 산업 환경에 대응할 수 있는 신뢰성 있는 기술을 제공함으로써 사회와 산업 발전에 기여하게 될 것이다.

Acknowledgement

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

참 고 문 헌

[1] M. Sadler. (2021) 84 Current Video Conferencing Statistics for the 2021 Market. [Online]. Available: https://www.trustradius.com/vendor-blog/web-conferencingstatistics-trends

[2] WebRTC. (2023) WebRTC. [Online]. Available: https://webrtc.org/

[3] D. Samba. (2022) P2P, SFU and MCU-WebRTC Architectures Explained. [Online]. Available https://www.digitalsamba.com/blog/p2p-sfu-and-mcu-webrtcarchitectures-explained

[4] Kurento. (2023) Kurento Media Server. [Online]. Available: https://doc-kurento.readthedocs.io/en/latest/