WebRTC 기반 메타버스 컨퍼런스 시스템 관한 연구

이현우O, 김보겸*, 김지민*, 노재윤*, 서민정*, 김병완*, 이병권*
O서원대학교 멀티미디어학과,
*서원대학교 멀티미디어학과
e-mail: hyeonw777@naver.com

A Study on WebRTC-based Metaverse Conference System

hyeon-woo LeeO, bo-kyeom Kim*, ,ji-min Kim*, jae-yoon Roh*, min-jeong Seo*,
byeong-wan Kim*, byung-kwon Lee*
ODept. of Multimedia, Seowon University,
*Dept. of Multimedia, Seowon University

● 요 약 ●

본 논문에서는 오프라인 컨퍼런스의 한계점인 접근성 문제와 온라인 컨퍼런스의 한계점인 참여자간 상호 소통과 네트워킹 문제점을 해결하기 위한 솔루션으로 WebRTC 기반 메타버스 컨퍼런스 시스템을 제안한다. 해당 솔루션은 WebRTC를 활용한 실시간 화상채팅 및 멀티접속을 구현하고 WebVR을 기반으로 하는 A-Frame 프레임워크를 활용하여 메타버스 컨퍼런스 시스템을 구현하였다.

키워드: WebRTC, A-Frame, WebVR, 메타버스(Metaverse)

I. Introduction

본 논문에서는 오프라인 컨퍼런스의 한계점인 접근성을 문제와 온라인의 컨퍼런스의 한계점인 참여자간 상호 소통과 네트워킹 문제점을 해결하기 위한 솔루션으로 WebRTC 기술을 활용한 실시간 화상채팅 구현과 WebVR을 기반으로 하는 A-Frame 프레임워크를 활용한 전시 및 컨퍼런스 시스템을 제안 및 개발한다. 본 논문에서는 2장에서 관련 연구 및 기술 동향을 제시하고 3장에서는 실제 구현 및 설계를 진행하고, 4장에서는 기술구현의 어려움과 연구에 대한 결론을 설명 후 향후 발전 방향을 제시한다.

II. Preliminaries

1. Related works

1.1 WebRTC

WebRTC(Web Real-Time Communication)은 웹 애플리케이션과 웹 사이트가 중간자 없이 브라우저 간에 영상 미디어나 오디오를 인식하고 스트림 할 뿐 아니라, 임의의 데이터도 교환 및 통신을 할 수 있도록 하는 기술이다. [1][2]. Fig. 1은 WebRTC 통신과정을 나타낸 그림이다. Signaling을 통해 peer간 네트워크 구성정보, 세션 제어 메시지, 미디어 가능등의 정보를 교환하고 peer와 연결을 맺은 후 단말에서 미디어를 가져와 교환한다.[2]

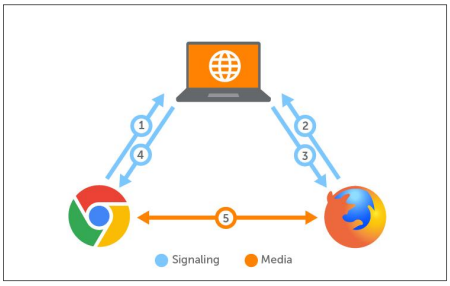

Fig. 1. How WebRTC works

1.2 A-Frame

A-Frame은 가상현실(VR) 경험을 구축하기 위한 웹 프레임워크이다. A-Frame은 three.js를 기반으로 확장한 프레임워크이며 대부분의 VR 헤드셋을 지원하며 증강현실에도 사용할 수 있어 몰입형 인터랙티브 VR 컨텐츠를 제작할 수 있다.[4]

III. The Proposed Scheme

WebRTC기반 메타버스 컨퍼런스 시스템 구현은 Fig. 2와 같이 ① Node js based server implementation ② Implementation of video chat using WebRTC ③ Implementation of A-Frame based metaverse space 순으로 진행한다.

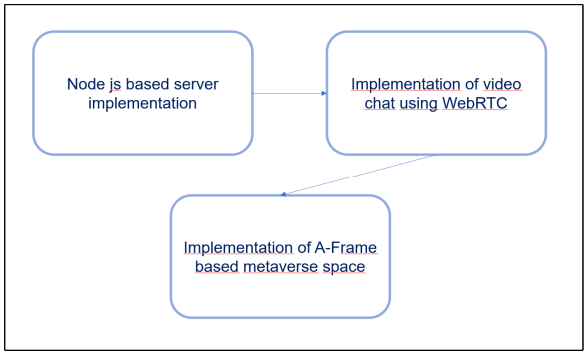

Fig. 2. Conference system implementation flowchart

① Node js based server implementation 단계에서는 Visual Studio Code를 사용하여 WebRTC 시그널링 서버를[1][Fig. 3]을 구현한다.

```
// Load required modules
const http = require("http");                 // http server core module
const path = require("path");
const express = require("express");           // web framework external module
const socketIo = require("socket.io");        // web socket external module
const easyrtc = require("open-easyrtc");      // EasyRTC external module

// Set process name
process.title = "networked-aframe-server";

// Get port or default to 8080
const port = process.env.PORT || 8080;

// Setup and configure Express http server.
const app = express();
app.use(express.static(path.resolve(__dirname, "..", "examples")));

// Serve the example and build the bundle in development.
if (process.env.NODE_ENV === "development") {
  const webpackMiddleware = require("webpack-dev-middleware");
  const webpack = require("webpack");
  const config = require("../webpack.config");

  app.use(
    webpackMiddleware(webpack(config), {
      publicPath: "/"
    })
  );
}

// Start Express http server
const webServer = http.createServer(app);

// Start Socket.io so it attaches itself to Express server
const socketServer = socketIo.listen(webServer, {"log level": 1});
const myIceServers = [
  {"urls":"stun:stun1.l.google.com:19302"},
  {"urls":"stun:stun2.l.google.com:19302"},
];
easyrtc.setOption("appIceServers", myIceServers);
easyrtc.setOption("logLevel", "debug");
easyrtc.setOption("demosEnable", false);
```

Fig. 3. Building signaling server program based Node Js

② Implementation of video chat using WebRTC 단계에서는 room에 입장시 이벤트처리와 socket 통신을 활용한 실질적인 connection을 맺는 부분을 구현한다. [Fig. 4]

```
// Overriding the default easyrtcAuth listener, only so we can directly access its callback
easyrtc.events.on("easyrtcAuth", (socket, easyrtcid, msg, socketCallback, callback) => {
    easyrtc.events.defaultListeners.easyrtcAuth(socket, easyrtcid, msg, socketCallback, (err, connectionObj) => {
        if (err || !msg.msgData || !msg.msgData.credential || !connectionObj) {
            callback(err, connectionObj);
            return;
        }

        connectionObj.setField("credential", msg.msgData.credential, {"isShared":false});

        console.log("["+easyrtcid+"] Credential saved!", connectionObj.getFieldValueSync("credential"));

        callback(err, connectionObj);
    });
});

// To test, lets print the credential to the console for every room join!
easyrtc.events.on("roomJoin", (connectionObj, roomName, roomParameter, callback) => {
    console.log("["+connectionObj.getEasyrtcid()+"] Credential retrieved!", connectionObj.getFieldValueSync("credential"));
    easyrtc.events.defaultListeners.roomJoin(connectionObj, roomName, roomParameter, callback);
});

// Start EasyRTC server
easyrtc.listen(app, socketServer, null, (err, rtcRef) => {
    console.log("Initiated");

    rtcRef.events.on("roomCreate", (appObj, creatorConnectionObj, roomName, roomOptions, callback) => {
        console.log("roomCreate fired! Trying to create: " + roomName);

        appObj.events.defaultListeners.roomCreate(appObj, creatorConnectionObj, roomName, roomOptions, callback);
    });
});

// Listen on port
webServer.listen(port, () => {
    console.log("listening on http://localhost:" + port);
});
```

Fig. 4. Code using webrtc API

③ Implementation of A-Frame based metaverse space 단계에서는 conference 가상공간(메타버스)을 A-Frame 기반으로 Fig. 5와 같이 구현한다.

Fig. 5. Implementation of A-Frame based metaverse space

IV. Conclusions

본 연구는 오프라인 컨퍼런스 및 행사의 한계점인 접근성을 문제와 온라인의 한계점인 참여자간 상호 소통과 네트워킹 문제점을 해결하기 위한 솔루션을 제안한다. 본 연구에서 제안하는 방법은 온라인 컨퍼런스의 한계점을 해결하고 물리적 행사, 전시에 참여할 수 없는 참여자들에게 접근성을 해결해 줄 수 있다. 향후 연구 방향은 webVR의 렌더링 최적화 문제를 해결하기 위한 연구를 진행할 것이다.

ACKNOWLEDGEMENT

본 논문은 과학기술정보통신부 정보통신창의인재양성사업의 지원을 통해 수행한 ICT멘토링 프로젝트 결과물입니다.

REFERENCES

[1] Cui Jian1, a, Zhuying Lin2, b, Research and Implementation of WebRTC Signaling via WebSocket-based for Real-time

Multimedia Communications
[2] Jennings, Cullen, Ted Hardie, and Magnus Westerlund. "Real-time communications for the web." Communications Magazine, IEEE 51.4 (2013): 20-26.
[3] https://developer.mozilla.org/ko/docs/Web/API/WebRTC_API/Protocols
[4] https://aframe.io/docs/1.3.0/introduction/