Assignment 10

Due: Wed 22.01.2020; 18:00h (1 Week)

Goals

This assignment trains you in...

- WebSocket

Task 1: Native WebSocket APIs — Difficulty: Easy

In the tutorial you learned using a third-party library Socket.IO with a lot of encapsulation with native WebSocket APIs. In this task, you will be exploring the native WebSocket APIs.

Please read the document and answer the following questions: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications.

1. Answer the question with one line of code.
   a) How to create a secure WebSocket without third party library?
   b) How to send data to the server?
   c) How to receive data from the server?
   d) How to close a WebSocket connection?
2. Reimplement the MIMUC Twitch App in native WebSocket API

Hint: To use WebSocket in Node.js, you might checkout: https://github.com/websockets/ws

Task 2: Audio Streaming — Difficulty: Medium

Our MIMUC Twitch App haven't implement an audio streaming support yet. In this task, you will be adding the audio streaming support to the MIMUC Twitch.

Hint: Similar to the video example, you actually sending a series of images via WebSocket. To support audio streaming, you can send the audio data by a series of PCM buffers. Checkout AudioContext APIs for audio data processing: https://developer.mozilla.org/en-US/docs/Web/API/AudioContext.

Task 3: Protect Streaming Data — Difficulty: Hard

Our MIMUC Twitch App steams anchor's raw media. This is actually dangerous to protect the copyrights of the content and hard to guarantee the safety of the content, e.g. an cnchor starts streaming a copyrighted movie to the public. We have several approaches to prevent this:

1. Adding watermarks to the streamed data
2. Delayed broadcasting with content verification

Your tasks are:

1. Write a MIMUC OMM watermark to the streamed image on the server side: http://www.medien.ifi.lmu.de/lehre/veranstaltungssketches/mmn.png
   Hint: You can use a Node.js library Jimp (https://github.com/oliver-moran/jimp) to processing streamed images.
2. Let our server caching the streamed data and delayed broadcasting to all viewer after 30 seconds.

Submission

Please turn in your solution as ZIP file via Uni2Work. You can form groups of up to three people.

We encourage you to sign up for Slack! All you need is a CIP account and an email address that ends in "@cip.ifi.lmu.de". Ask us if you don't know how to get them.

If you have questions or comments before the submission, please contact one of the tutors. They are on Slack @Aleksa and @Andre, remember that they also want to enjoy their weekends ☺

It also makes sense to ask the question in our #omm-ws1920 channel. Maybe fellow students can help or benefit from the answers, too!