Live Streaming

Alazar Alemayehu Abebaw (******)
Axel Ilmari Neergaard (******)

December 13, 2020

Source code: https://version.aalto.fi/gitlab/neergaa1/multimedia-assignment-3

For this project we were required to setup a WebRTC streaming server capable of streaming live from one system to another. Accompanying this WebRTC server we were also instructed to create a webpage as an interface for the streaming setup. Next, we had to setup an HLS server capable of consuming a live stream and splitting it into segments and a manifest file. Finally we had to enable forwarding between the WebRTC server to the HLS server. The assignment description recommended to use the Janus WebRTC server and the NGINX web server with its rtmp module enabled. As both of these servers are meant to be run on dedicated hardware, and the configuration of both *can* be quite arduous, we decided to run them on Docker containers.

0.1 Janus Configuration

With the Docker container Janus works almost out of the box for our purposes. As per the recommendation of the assignment description, we used the Video Room plugin for Janus, which comes pre-installed onto the server. We stripped down the configuration of a custom Video Room to only allow for one publisher at a time.

The Video Room plugin also comes with a demo webpage, with JavaScript for controls, which we copied and split into its own components. This page was also configured to automatically forward an RTP stream once the `webrtcup` event was received, for reasons found, e.g., here.

0.2 NGINX Configuration

The NGINX Docker container was also working almost out of the box for us. The only tweak that we performed was to disable the FFmpeg stream encoding for higher bitrates, and only allowed for the lowest one. This was simply to assure that our local machines did not get too overworked from running everything. With this setup we were able to stream straight to the HLS server endpoint `rtmp://localhost:1935/stream/<key>` and watch the stream from the endpoint `http://localhost:8080/live/<key>.m3u8`.

0.3 RTP Forwarding

As mentioned the streaming page was configured with RTP forwarding. To consume this RTP stream we had to setup a FFmpeg consumer on the NGINX Docker instance. This consumer used a SDP file to match the RTP forwarding configuration, and then sent this to the NGINX consuming endpoint `rtmp://localhost:1935/hls/janus`. This created a consumable stream at `http://localhost:8080/live/janus.m3u8`.

0.4 Stream Setup & Latency

For streaming purposes we used OBS studio to stream a local timewatch video. OBS enabled us to setup a virtual web camera as well as streaming straight to the HLS server. With the stream running we measured the difference between the time on the streamer video and the streamed video. The measurements were done on the same computer as the containers were running on, and only one subscriber was present (for each type)—it is also noteworthy that we only cared about relative latencies, and thus no larger network was tested. As the measurements were done "by hand" there are systematic errors, where the reaction time for humans are a bit under 200 milliseconds, and for the tired authors, who are looking at two video simultaneously, probably higher. The following measurements, with they corresponding observed latencies, were:

1. Streaming only through WebRTC – sub-second, about 200 milliseconds
2. Streaming only through HLS – multiple seconds, just above 8 seconds
3. Streaming through both WebRTC and HLS – multiple seconds, just below 12 milliseconds