Synchronized Multi-Channel Audio for Multi-User Augmented Reality

Robert Fraher

University of Wisconsin-Stout

fraherr@uwstout.edu

INTRODUCTION

This paper contains a case study describing an augmented reality (AR) Web app. This Web app is a rich media composition that provides collaborative interaction to users distributed globally. The main focus of this paper is the design of the composition's multi-channel audio playback system. The key technical goals for the audio playback system were 1) to synchronize the playback of audio files across all devices running the app, 2) to maintain precise control of individual audio channels, and 3) to connect the sonic experience provided by the app to users' AR interaction. The app that serves as the context of this discussion is titled *Constant Shower* (http://constantshower.azurewebsites.net[1]), and was developed as an elaborate political cartoon.

This document proceeds by first providing some background info regarding the app, and then outlining its general structure. This is followed by a description of the solutions employed to accomplish the goals above.

BACKGROUND

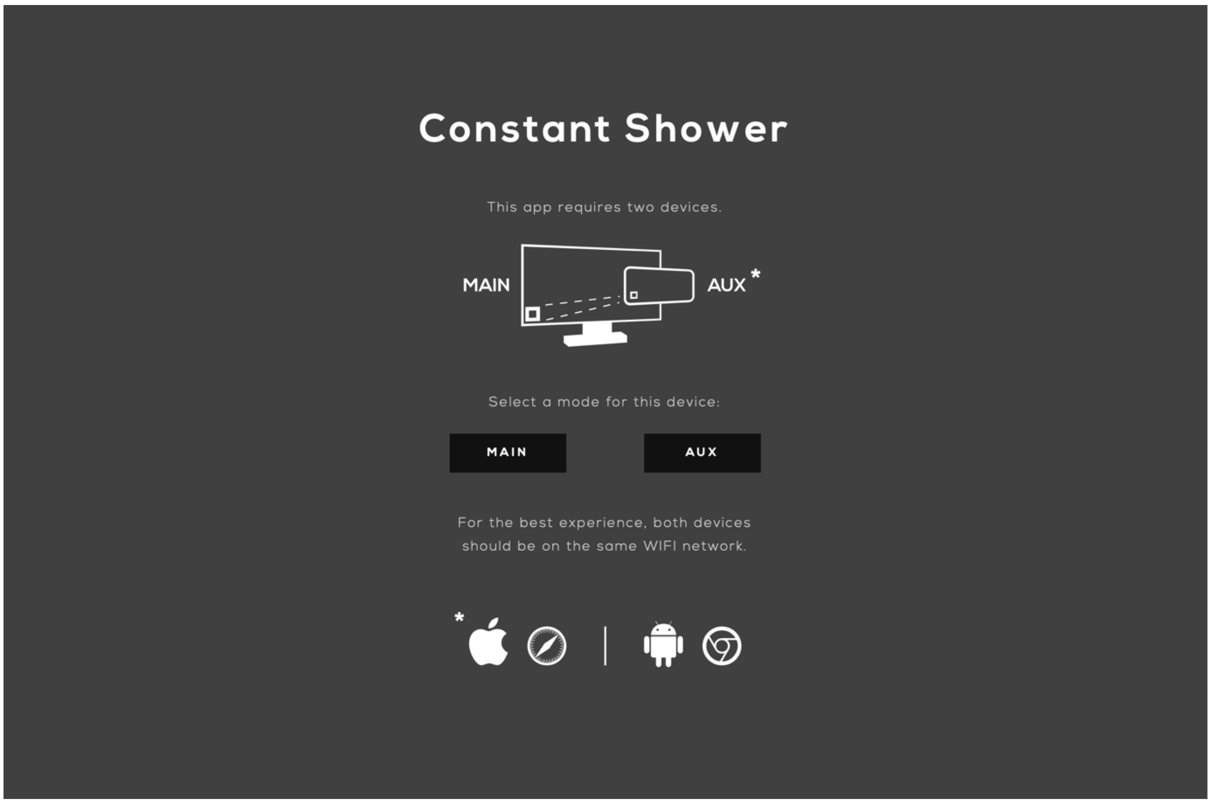
On Wednesday, March 6th, 2019, Jair Bolsonaro, a self-described homophobe (BBC, 2019), asked the world via Twitter, "What is a golden shower?" This app seeks to provide Jair with an answer to his question.

In fact, the app allows people to collaborate and maintain a constant shower for Jair. This collaboration is possible due to the app being Web-based, and therefore able to run in multiple locations simultaneously. The experience provided by each instance is connected to all other instances.

GENERAL STRUCTURE

The *Constant Shower* app consists of a Web server and a responsive browser-based client (Figures 1 and 2). The server component of the app was built with Node.js. The client interface is a one-page Web site created with HTML5, SCSS, and JavaScript. The client interface also relies on A-Frame for its AR functionality. This server-client architecture has emerged as a standard approach for networked audio synchronization (Schnell et al, 2017; Milde, 2017).

[1] When the composition is not being exhibited, the maximum number of server connections is limited to five to conserve financial resources. When the composition is being exhibited, the maximum number of server connections is increased based on the scope of the exhibition.



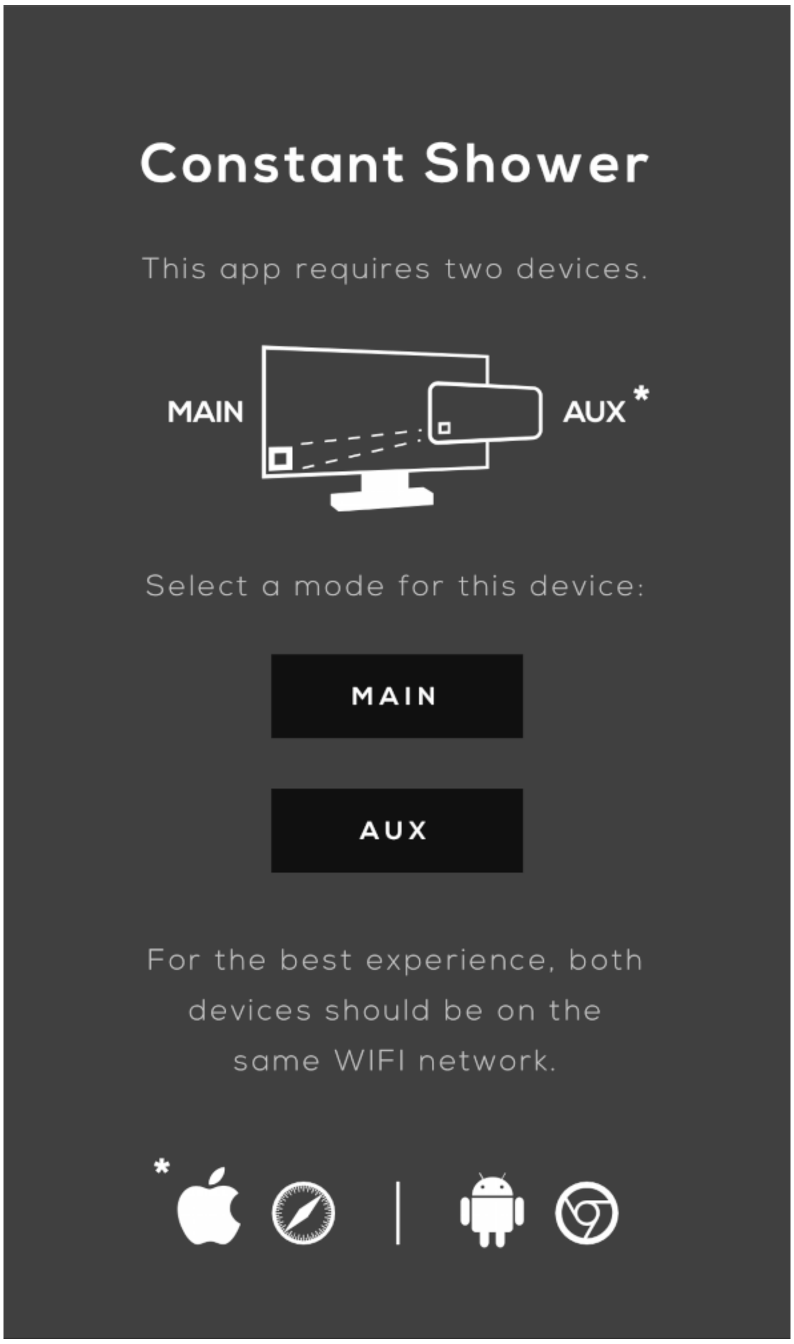
Constant Shower

This app requires two devices.

Select a mode for this device:

For the best experience, both devices should be on the same WIFI network.

Figure 1: Client interface on a desktop display



Constant Shower

This app requires two devices.

Select a mode for this device:

For the best experience, both devices should be on the same WIFI network.

Figure 2: Client interface on a desktop display

The client interface offers users a choice between two modes, MAIN and AUX. The MAIN mode can be understood as the display or broadcast mode. The AUX mode can be understood as the accessory or satellite mode.

When in the MAIN mode, the client interface is comprised of an animated character and simple context, a looping music track, and several interface controls. Amongst these controls is an AR marker in the bottom left corner of the interface (Figure 3).

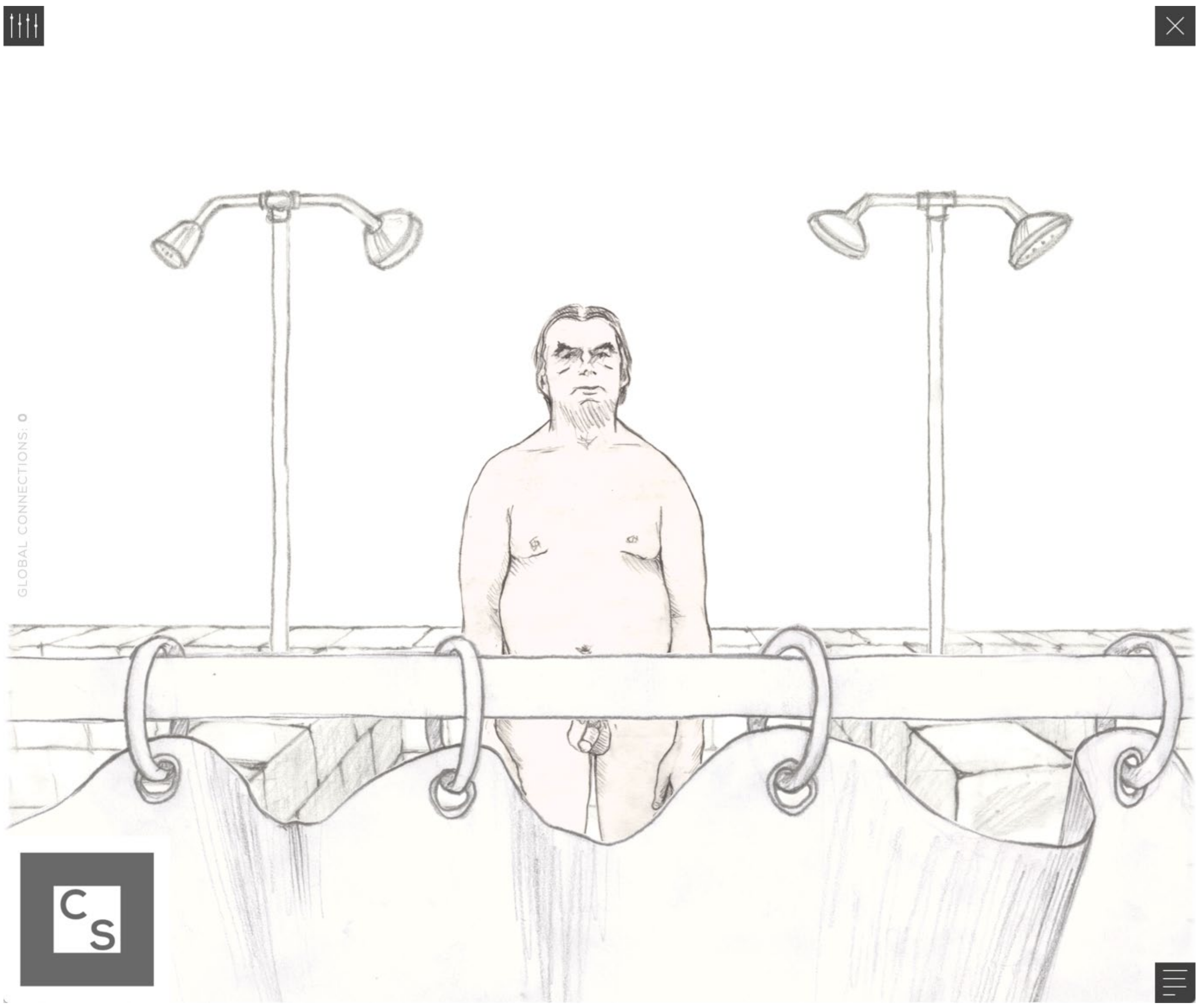


Figure 3: Client interface on a desktop display in the MAIN mode

When in the AUX mode, the client asks permission to access the device's camera. When granted, the input of the camera is sent to the client's interface and made full screen. Accompanying the video are a looping music track and several interface controls.

When the camera of a device in AUX mode is pointed at the display of another device in MAIN mode, the device in AUX mode registers the AR marker. This triggers a "markerFound" event and a message is sent to the server. When the server receives this message, it sends instructions to all devices running the app to make changes to their respective interfaces and soundscapes.[2] When the camera of a device in AUX mode is then turned away from the display of a device in MAIN mode, a "markerLost" event is triggered and another message is sent to the server, which in turn, sends instructions to undo the changes.[3]

TECHNICAL GOALS

The goal of synchronizing the playback of audio files across all devices running the app resulted in the use of a WebSocket connection between the server and all clients. The choice to use a WebSocket was based on two factors. First, testing revealed that a WebSocket connection is faster and more consistent than HTTP. This increased speed and consistency is due to the WebSocket protocol being a full-duplex system, providing

[2] This message is only sent by the server if these instructions have not already been sent by another client in AUX mode triggering the "markerFound" event. Due to the app being able to run in MAIN or AUX modes on multiple devices simultaneously, clients in AUX mode that register a "markerFound" event after another client has already done so are the only devices to experience interface and soundscape changes from the event.

[3] This message is only sent by the server if the client in AUX mode is the only device eliciting the AR experience. If other clients are also eliciting the AR experience, interface and soundscape changes are only experienced by the client that triggered the event.

persistent two-way communication with lower overhead than half-duplex protocols like HTTP. Second, the ability to establish a persistent two-way connection between the server and all clients allowed for real-time transfer of information in both directions. This two-way transfer of information turned out to be key in accomplishing the goal of connecting the sonic experience provided by the app to users' AR interaction.

With a WebSocket in place, synchronization is accomplished by requesting the time from the server (serverTime) and then using that time to calculate a duration from the Unix epoch (startTime) to the time at which playback is initiated (serverTime + trueElapsedTime). This duration (differenceFromStartTime) is divided by the duration of an audio file (trackLoopDuration) to produce a remainder (trackOffset) which serves as the start time of an audio track. In an effort to achieve the most precise synchronization, the duration of the server call is also measured and factored in (serverCallDuration). These calculations are outlined below.

```
currentTime = new Date().getTime();

serverCallDuration = timeAfterServerCall - timeBeforeServerCall;

elapsedTimeFromServerCallReturn = currentTime – timeAfterServerCall;

trueElapsedTime = elapsedTimeFromServerCallReturn + (serverCallDuration / 2);

startTime = new Date(0);

differenceFromStartTime = ((serverTime + trueElapsedTime) - startTime.getTime()) / 1000;

trackOffset = differenceFromStartTime % (trackLoopDuration / 1000);
```

The goal of maintaining precise control of individual audio channels resulted in the use of the Web Audio API. The decision to use the Web Audio API was based on two advantages over HTML5 audio elements. First, the ability to seamlessly loop audio playback through the Web Audio API allows for the use of shorter audio files and a more musical looping experience than HTML5 audio elements. Second, the facilitation of sample-accurate playback scheduling results in exact coordination of offset start times and volume fading across audio channels.

As mentioned above, the goal of connecting the sonic experience provided by the app to users' AR interaction was accomplished through use of a WebSocket. This protocol allowed the "markerFound" and "markerLost" events triggered by devices in AUX mode to be used by the server to keep track of the number of users, determine the initial state of the client interfaces when the app loads (based on whether other users have triggered the "markerFound event"), transition between interface states, and control the volume of the audio tracks. The design rationale supporting this goal is based on the premise that synthesizing visuals, audio, and interaction would promote increased user engagement and agency, and hopefully, a sense of immersion.

REFERENCES

BBC, (2019). *Bolsonaro: Brazil must not become 'gay tourism paradise'.* [online] Available at: https://www.bbc.com/news/world-latin-america-48062075 [Accessed 4August 2020].

Milde, J. (2017). 'Synchronized Mobile Devices Using Web Audio Technology on a Raspberry Pi', in *Proceedings of 3rd Web Audio Conference.* [online] London: Queen Mary University, pp. 84-87. Available at: https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/26154/84.pdf [Accessed 8 August 2020].

Schnell, N. et al. (2017). 'Collective Loops: Multimodal Interactions Through Co-located Mobile Devices and Synchronized Audiovisual Rendering Based on Web Standards', in *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction.* Yokohama: ACM, pp.217-224.