Collab-Hub: A system for creating collaborative telematic music performances with web-based instruments and creative development platforms

Nick Hwang
Media Arts and Game Development
University of Wisconsin - Whitewater
hwangn@uww.edu

Anthony T. Marasco
School of Music
The University of Texas Rio Grande Valley
anthony.marasco@utrgv.edu

Eric Sheffield
Department of Music and Theater Arts
SUNY Broome
sheffielder@sunybroome.edu

ABSTRACT

Collab-Hub is a networking tool for sharing data across the internet for multimedia collaboration. Through Collab-Hub, web audio applications, custom hardware controllers/instruments, and digital instruments created for embedded and desktop computers can all be designed around a single framework, providing a unified approach to creating multi-modal networked experiences.

In this talk, we discuss the design of the Collab-Hub system and provide examples of using it to create interaction layouts between web-based instruments/interfaces and digital musical instruments designed in Max and Pure Data. We also showcase custom APIs for developing audiovisual software clients in Unity and hardware clients through Arduino. An analysis of pertinent historical precedents highlights the advantages Collab-Hub provides to artists who may not have experience developing collaborative projects between browsers and desktop software or may be entirely new to designing telematic and remote performance environments. A showcase of works created with Collab-Hub (featuring networked interactivity across the internet, digital instrument design through the aforementioned creative development platforms and the WebAudio API, and multimodal art technologies) demonstrates the wide variety of artistic endeavors made possible through the framework.

1. INTRODUCTION

Collab-Hub[1] was born out of a desire to create local telematic networked musical performances that promoted a platform-agnostic mentality towards the creation of client-side instruments and interfaces. Through the collection of academic research and creative works published at the Web Audio Conference, it is clear that the web browser alone serves as a more than capable framework for building legions of networked digital music instruments that can imbue their users with the ability to share and respond to control data. But working entirely within the boundaries of the browser and the Web Audio API—or any one development environment for that matter—can cut artists and audience participants off from any creative potential provided by complimentary tools (such as the graphical coding process provided by Max and Pure Data or the tangible interactivity of circuit design and hardware hacking).

Figure 1: The Collab-Hub logo

Using the internet and a client-server architecture as the keystone of our system, we set out to devise a framework that wasn't relegated solely to the browser, but one that could accommodate internet-enabled hardware and software created in environments frequently used by computer musicians, visual artists, hackers, and game designers. Mimicking the web browser's flexibility as a tool for facilitating collaborative creativity, we see Collab-Hub as a customizable and useful tool for laptop ensembles, mixed audiovisual ensembles, interactive installation artists, and those looking to create networked experiences with internet-enabled circuitry.

2. COMPARISON TO HISTORICAL EXAMPLES

Building on the works of networked musical ensembles such as the Hub and the League of Automatic Music Composers[6], a handful of web-centric systems for designing communication systems and interaction topologies between users, browser-based instruments, and creative software frameworks have been previously developed. Notable examples include NexusUI and NexusHub by Jesse Allison[1], Soundworks by Sébastien Robaszkiewicz and Nor-

[1]More information on the system along with tutorials and example interfaces can be found at https://www.collab-hub.io

Web Audio Conference WAC-2021, July 5–7, 2021, Barcelona, Spain.

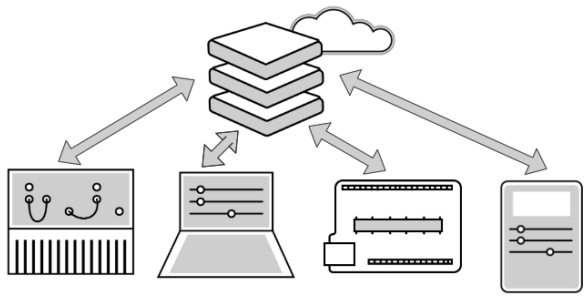

Figure 2: Collab-Hub brings together creative development environments like Pure Data, P5.js, internet-enabled hardware, and embedded devices.

bert Schnell[4], Nü Soundworks by David Poirier-Quinot et al.[3], and Rhizome by Sébastian Piqumel[5]. These frameworks place a majority of their developmental focus on the web browser, with only a handful of example templates and add-ons to their API centered on incorporating additional software environments for developing client interfaces. In comparison, we chose to center our efforts on bringing the core concepts of Distributed/Networked Musical Performance pieces to a wider audience of musicians, hardware designers, and visual artists who many not have experience with JavaScript but are eager to telematically collaborate with fellow electroacoustic performers. Users who develop digital musical instruments and signal processors within the Max and Pure Data environment can now easily control a fellow performer's patch and relinquish control of their own instrument's parameters through the addition of our Collab-Hub sub-modules. Furthermore, our interest in expanding the available hardware device set for works that utilize both the WebAudio API and custom or hacked hardware as complimentary sound engines[2] has led us to develop libraries to allow internet-enabled microcontrollers to connect to our server as independent clients.

An additional point of contrast lies in our remotely-hosted web server that serves to manage communication between the client scripts that run in the background of each unique client platform. Users do not need to write their own server code nor do they need to worry about designing routing topologies in Node.js. Users can customize exactly how data is shared and routed between connected clients dynamically in their own interfaces without having to rewrite any code or relaunch a locally-hosted server of their own. For those who do wish to run a version of our server locally, they may use our upcoming standalone application that allows for configuration of the server through a user-friendly GUI.

3. FEATURES OF COLLAB-HUB

3.1 Sharing Data in Performance

During the design phase, we focused on creating a system that would allow users to easily set-up and modify their preferred methods of sharing data and collaborating amongst each other in performance. If desired, the types of data and the routing topology can be easily altered on the fly with each environment client. When performing with a Collab-Hub-connected client, collaborators can send any number and types of data between Max patches and web instruments.

When sending data between collaborators, Collab-Hub uses three main data message labels: Control, Event, and Chat. Data sent with the Control label is meant to be used for continuous control of an instrument's parameters. An Event is an instantaneous occurrence, such as a button press, a toggled boolean state change, a signifier to start or stop a performance or playback action, or an indication of a section change in a scored piece. Data sent with the Chat label is designed for communicating directly with specific, groups, or all connected users through text messages.

3.2 Basic Data Routing

When sharing data, there are two components to data message routing: distribution and targets. The distribution and target component of data message can be different with each message.

Distribution can either be Publish or Push. Published messages are sent to connected clients but their accompanying data can only be grabbed by those clients who choose to opt-in and observe those specific data streams. Sub-modules within the client interface list available published controls and events. Clients can update their observation settings instantly and throughout a performance. With *publish* distribution, the *target* component determines whether or not that data is available to other clients. Targets can either be 'all' (data is available for all clients in a Namespace), a specific room name (data is available for all in that room), or a specific client/username.

Push distribution messages are only sent to specific clients based on their target component. Like Publish distribution, targets for Push are 'all', a specific room name, or a client/username. Push distribution messages do not require target to take actions in order to receive this distribution method.

3.3 Customizing Routing Topologies for Performances

Users are not limited to committing to single routing topology model in a performance and can design works that switch between each of these models at a whim, resulting in routing topology that changes as the piece progresses. Additionally, the individual distribution and target designation of a message can be changed at any time during performance.

When designing our routing topology methods, we felt that it was crucial to focus more on how we wanted clients to be able to interact with each other instead of simply adapting a traditional client-server communication scheme and sticking to that one model.

On a macro-level of organization, users can have any number of collaborators take part in a performance set inside of a segregated Namespace. Additionally, users within a Namespace can further be divided up into separate Rooms. These advanced layers of routing can be helpful for ensembles (relegated to their own unique Namespace on the server) who wish to design a series of different telematic works, each using specific instruments and a specific sub-set of performers (with each set of performers placed into specific Rooms).

4. COLLAB-HUB HARDWARE SUITE

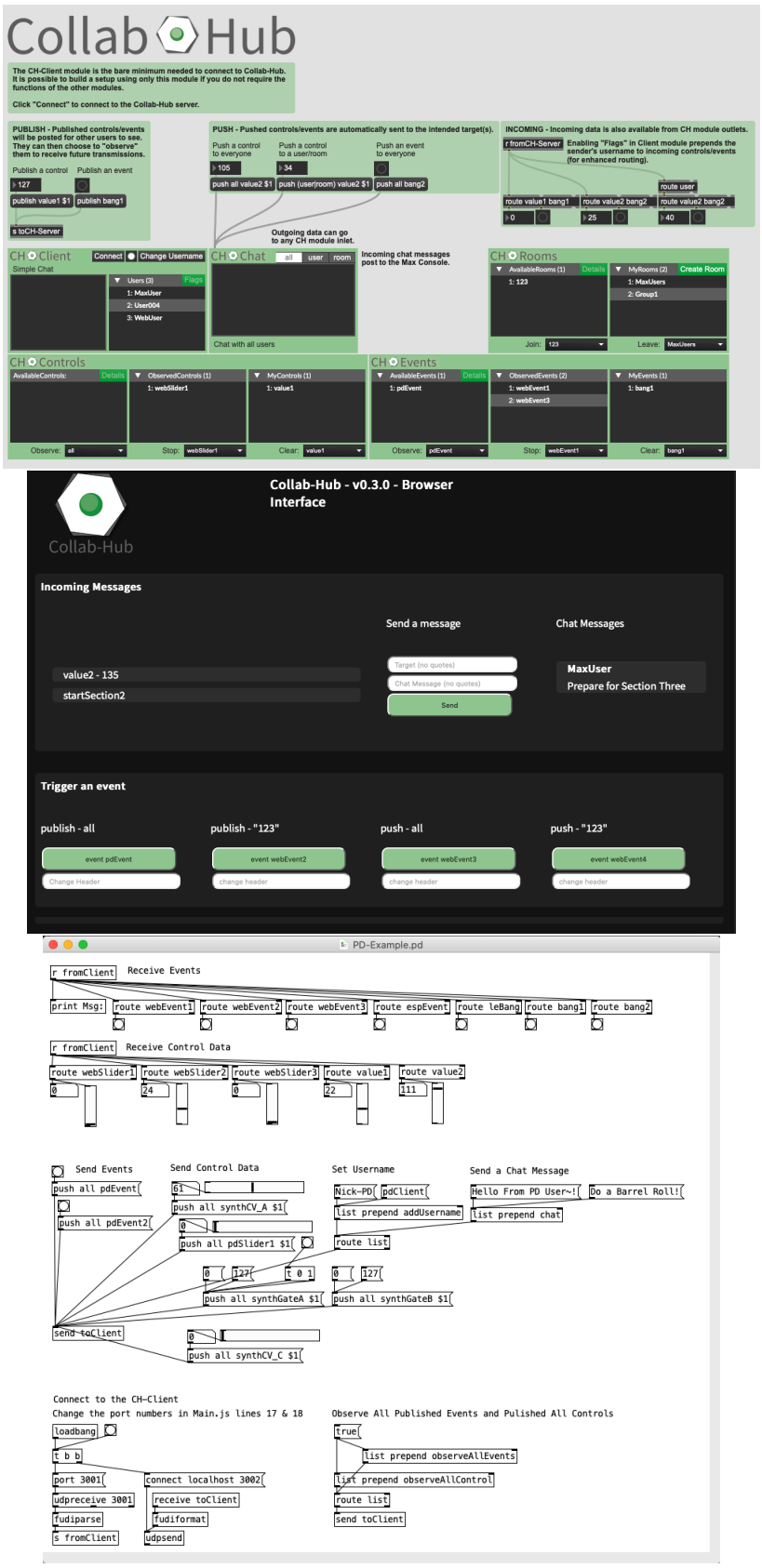

Figure 3: Collaborative interaction between three Collab-Hub clients developed in different frameworks: Max (top), the web browser (middle), and Pure Data (bottom).