Tahakum: A Multi-Purpose Audio Control Framework

Zachary Seldess
King Abdullah University of Science and Technology
Visualization Lab
Thuwal, Saudi Arabia
zachary.seldess@kaust.edu.sa

Toshiro Yamada
University of California, San Diego
California Institute for Telecommunications and Information Technology
La Jolla, CA, USA
toyamada@ucsd.edu

ABSTRACT

We present "Tahakum", an open source, extensible collection of software tools designed to enhance workflow on multi-channel audio systems within complex multi-functional research and development environments. Tahakum aims to provide critical functionality required across a broad spectrum of audio systems usage scenarios, while at the same time remaining sufficiently open as to easily support modifications and extensions via 3rd party hardware and software. Features provided in the framework include software for custom mixing/routing and audio system preset automation, software for network message routing/redirection and protocol conversion, and software for dynamic audio asset management and control.

Keywords

Audio Control Systems, Audio for VR, Max/MSP, Spatial Audio

1. INTRODUCTION

Audio Systems within interdisciplinary and multi-media research facilities are often expected to fulfill a large variety of end-user and developer functions, ranging from simpler tasks such as live event sound reinforcement and fixed media playback, to more complex activities such as experimental real-time acoustics simulations, and new musical interface design. Successfully managing audio systems in multi-functional environments relies not only on quality hardware and software implementations, but also on the existence of an overarching audio control framework. Such a framework must tackle the unique challenge of achieving a balance between stability and end-user friendliness, and flexibility and low-level access and control, ensuring successful typical daily operations, while at the same time providing a fast development pipeline.

Audio interfaces break or get replaced with better alternatives, input and output channel counts and loudspeaker configurations change over time, as do notions of ideal software and hardware solutions for all manner of lab audio functions and research projects. A successful audio control framework, in addition to facilitating smooth workflow during stable periods of operation, must also attempt to make system changes as seamless as possible during times of growth and transition.

Ideally, within a research facility, mid and high-level control of an audio system must be readily available to staff for day-to-day activities such as project demonstrations, video conferencing, and fixed-media A/V playback. At the same time, audio systems developers and technicians need to be able to experiment, modify, and implement low-level hardware and software configurations with relative ease and fluency. In this paper we present "Tahakum[1]", a set of open source, extensible software tools, designed to enhance operations and development workflow in dynamic multi-media, multi-purpose spaces.

Organization: The paper is organized as follows: In section 2 we provide an overview of the audio control and development framework, including details on our general framework design philosophy. Sections 3, 4, and 5 provide more detailed functionality and design information for AudioSwitcher Server, Control Proxy, and Asset Manager, respectively. In Section 6, we provide concluding thoughts on the framework's current state, and discuss planned and potential future improvements to the system.

2. FRAMEWORK OVERVIEW

Tahakum, created using Max/MSP, is designed to allow audio developers and technicians to easily adapt customized control systems to a given room, and to minimize the downtime in hardware and software changes within a system, while providing a baseline control framework that is easily extensible and customizable to users' equipment, preferences, and needs. Much previous work has addressed specific workflow issues within multi-functional media environments, such as spatial audio post-production (e.g. [2], [3]), real-time spatial sound rendering and composition (e.g. [4], [6], [7], [9]), and interactive room acoustics simulation (e.g. [1], [5]). Other notable work, such as [8], provides a more comprehensive framework geared specifically towards the task of sound spatialization. And there are myriad sophisticated commercial audio show control tools available, such as Meyer Sound Laboratories' CueStation[2] and Figure 53's QLab[3]. Our intention in designing the Tahakum framework has been to provide, using Max/MSP, the software tools to enhance core operational and development workflows within complex multi-media spaces, while making a point of not hindering users' preferences towards enhancements and extensions to the system via 3rd party hardware and software, network and MIDI i/o control. Our framework provides this functionality using three software tools whose primary features breakdown as follows:

[1] Tahakum, or تحكم, means "control" in Arabic.
[2] www.meyersound.com/pdf/products/lcs_series/CSv4_20070919.pdf
[3] http://figure53.com/qlab/

- **AudioSwitcher Server** combines audio mixing, routing, and delays, network and MIDI i/o, and a graphical user interface, within a preset-based automation system, for storage and recall of complex audio system state changes and event sequences. Additionally, client control software enables multi-user simultaneous access to the server.

- **Asset Manager** provides dynamic loading, unloading and control over Max patches, easy network integration using Open Sound Control (OSC), and an abstraction layer that facilitates changes to a project's panning algorithms and software-to-hardware channel mappings.

- **Control Proxy** blends network message routing/redirection, network protocol conversion between incoming and outgoing messages, and a console interface for manually sending network messages to user-defined destinations.

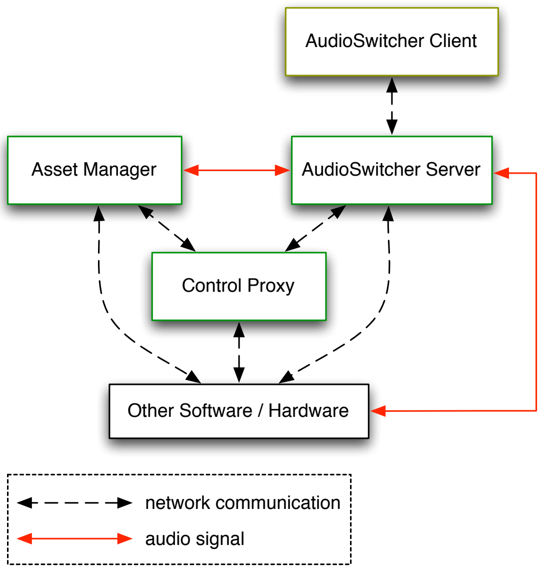


Figure 1: Typical Tahakum framework data and signal-flow

Each of the above applications is customized using plaintext configuration files with a simple syntax. By using a configuration file to control all aspects of the software's initialization and customization, the software can be updated, remotely or locally, while it is up and running. This has proven to be a significant workflow enhancement in managing our own audio systems, as well as providing an easy way to hand off new system updates to collaborators for review and discussion. Additionally, since customization is achieved via text files rather than manual Max re-patching, the software's core functionality remains the same whether running as a standalone application, with Max/MSP Runtime, or with an authorized full Max/MSP install.

All applications provide network i/o using standard protocols (including OSC) for two-way communication between each tool in the framework, as well as between various other 3rd party hardware and software products. Command syntax between applications is documented and consistent, making it possible for developers to extend the framework functionality with their own custom-designed software or hardware.

3. AUDIOSWITCHER SERVER

In this section, we give an overview of AudioSwitcher Server's functionality and briefly discuss some of the software's key controls and features.

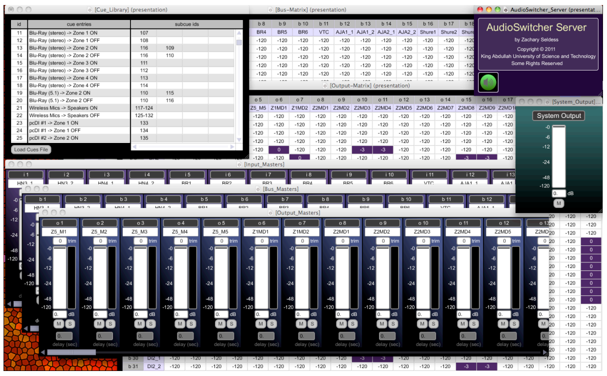


Figure 2: AudioSwitcher Server software

3.1 Overview

Let us assume we have a multi-purpose room with the following equipment:

- 1 Blu-Ray player (with 5.1 audio output)
- 4 wireless microphones
- 1 video conferencing unit (with 2 channels i/o)
- 1 computer for custom audio (with 8 channels i/o)
- 1 computer for graphics work (with 2 channels)
- 1 hardware audio mixer with 8 channels of i/o.
- 8 loudspeakers, 1 subwoofer

Ideally, all devices in the room need to be able to send audio out to any number of the nine available speakers. Additionally, some devices, such as the video conferencing unit and the custom audio computer, need to *receive* audio from various sources as well. In many situations, it is often desirable to pass all audio through one central hub, allowing easy control over the entire system without having to physically patch cable. Assuming you have a computer with enough digital and/or analog audio inputs and outputs, AudioSwitcher Server is designed to facilitate control of idiosyncratic configurations such as the one listed above. The software is built with scalability in mind and will therefore function in a variety of scenarios, ranging from very simple to complex i/o configurations.

Summary of Functionality:

- Custom hardware-to-software i/o channel mappings
- Preset-based automation system
- Configuration files for server setup and preset definitions
- Mute/solo/delay controls on input/bus/output channels
- Input-to-bus and bus-to-output sub-mixing
- Group fader assignments on input/bus/output channels
- Control of multiple servers via client control software
- User-defined network and MIDI i/o "bindings" of server controls
- Custom network message creation, storage, and delivery to remote destinations

3.2 Signal Control, Signal Flow

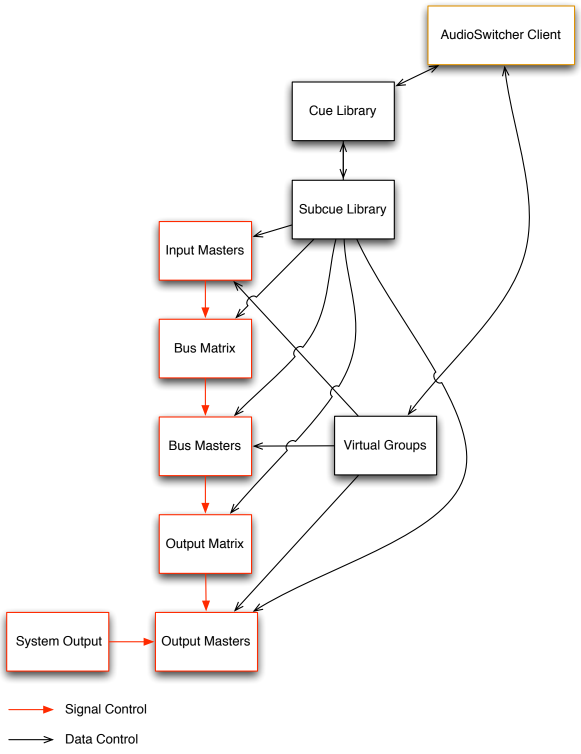


Figure 3: AudioSwitcher Server control and signal flow

3.2.1 Input/Bus/Output Masters, System Levels

Signal flow in AudioSwitcher Server resembles that of most DAW software tools, with input channels assigned to various bus channels, which are then sent on to output channels. Each input, bus, and output channel contains a post-fader signal-level meter, as well as controllable areas for its label, trim and fader levels, mute, solo, and delay states (Figure 4). Additionally, master control over all output channel level and mute states is provided in the System Output window. Level adjustments made to the system output are applied to all outputs at the pre-fader stage. All of the above controls can be manually adjusted or automated using AudioSwitcher Server's preset system (discussed in Section 3.3).

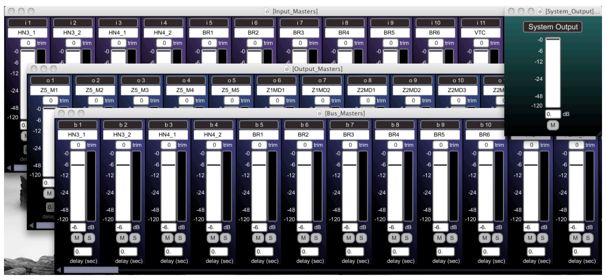


Figure 4: Input/Bus/Output Masters, and System Output

3.2.2 Bus and Output Matrices

The core of AudioSwitcher's signal flow centers around two variable-sized mixing matrices (Figure 5). By implementing mix matrices at two different stages in the signal flow, we provide a flexible vehicle for dealing with the complex mixing and routing scenarios encountered in multi-media spaces, effectively removing, for instance, the need for most output-to-input loopbacks (which the software also supports). All mix points in the matrices can be manually adjusted, or automated using the software's preset system (see Section 3.3).

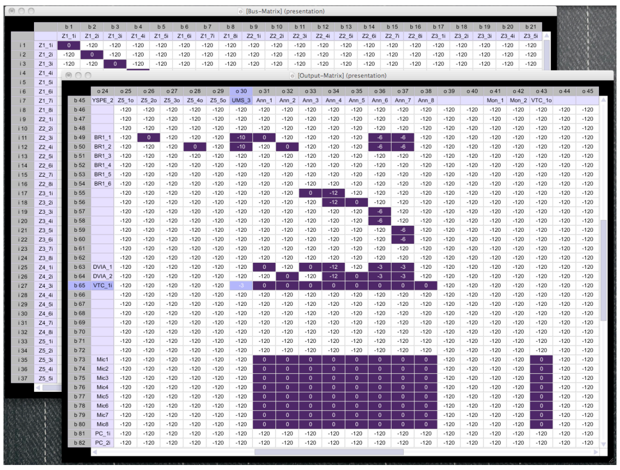


Figure 5: Bus Matrix and Output Matrix

3.3 Automation and Control

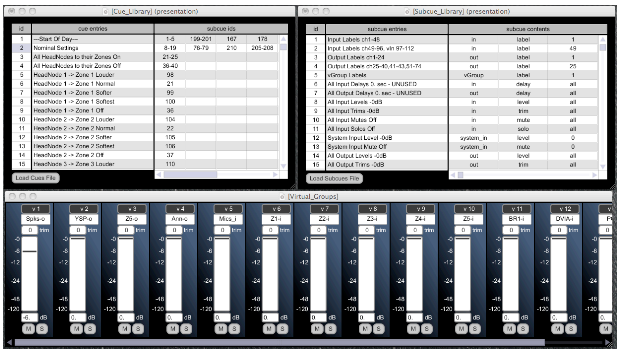
AudioSwitcher Server provides a control and automation layer for manual or remote event triggering and state changes over all facets of the software, such as Bus and Output Matrix assignments, Input/Bus/Output Masters labels, levels, mutes, solos, and delays, DAC on/off state, etc. Additionally, custom network messages can be created, stored and sent to remote destinations (triggering state changes in custom Max patches running within Asset Manager, for example). Once software and hardware configurations have been properly established, and signal control logic has been largely defined, three features within AudioSwitcher Server function as the primary vehicles for high level audio systems control: Cue Library, Subcue Library, and Virtual Groups.

3.3.1 Cue and Subcue Libraries, Virtual Groups

The Cue and Subcue Libraries provide display and control over user-defined presets. Cues exist solely as a means to store and recall one or more lower-level presets, called "subcues." Triggering a cue causes all subcues referenced by that cue to be sequentially recalled in a user-defined order (Figure 6). Whereas cues essentially act as subcue aggregators, subcues themselves apply automated control over virtually all aspects of AudioSwitcher Server's functionality; they do the actual work. Both cues and subcues can be manually triggered, or automated via calls from their control counterparts (i.e. cues referencing subcues, subcues referencing cues).

Virtual Groups provide high-level control over user-defined groups of input, bus, and output channels (Figure 6). In the Virtual Groups window, a user can manually set each group's label, trim, level, mute, and solo states, which in turn effect the corresponding states of all input/bus/output channels linked to that group. All virtual group controls can be manually adjusted, or automated via cues and subcues.

Establishing effective high-level control over complex audio systems is not unlike trying to hit a moving target, and as such it is important for a control system to support real-time user modification. Therefore, as mentioned earlier, cues, subcues, and virtual groups are all defined via text files that can be modified and reloaded while the server is running.



[Cue_Library] (presentation)

id	cue entries	subcue ids			
1	---Start Of Day---	1-5	199-201	167	178
2	Nominal Settings	8-19	76-79	210	205-208
3	All HeadNodes to their Zones On	21-25			
4	All HeadNodes to their Zones Off	36-40			
5	HeadNode 1 -> Zone 1 Louder	98			
6	HeadNode 1 -> Zone 1 Normal	21			
7	HeadNode 1 -> Zone 1 Softer	99			
8	HeadNode 1 -> Zone 1 Softest	100			
9	HeadNode 1 -> Zone 1 Off	36			
10	HeadNode 2 -> Zone 2 Louder	104			
11	HeadNode 2 -> Zone 2 Normal	22			
12	HeadNode 2 -> Zone 2 Softer	105			
13	HeadNode 2 -> Zone 2 Softest	106			
14	HeadNode 2 -> Zone 2 Off	37			
15	HeadNode 3 -> Zone 3 Louder	110			

Load Cues File

[Subcue_Library] (presentation)

id	subcue entries	subcue contents		
1	Input Labels ch1-48	in	label	1
2	Input Labels ch49-96, vln 97-112	in	label	49
3	Output Labels ch1-24	out	label	1
4	Output Labels ch25-40,41-43,51-74	out	label	25
5	vGroup Labels	vGroup	label	1
6	All Input Delays 0. sec - UNUSED	in	delay	all
7	All Output Delays 0. sec - UNUSED	out	delay	all
8	All Input Levels -0dB	in	level	all
9	All Input Trims -0dB	in	trim	all
10	All Input Mutes Off	in	mute	all
11	All Input Solos Off	in	solo	all
12	System Input Level -0dB	system_in	level	0
13	System Input Mute Off	system_in	mute	0
14	All Output Levels -0dB	out	level	all
15	All Output Trims -0dB	out	trim	all

Load Subcues File

Figure 6: Cue Library, Subcue Library, and Virtual Groups

3.4 Network and MIDI i/o

In order to provide users the ability to customize the way in which they interface with AudioSwitcher Server, most of the software's control features can be configured to communicate with 3rd party hardware and software over network and MIDI protocols. Via the software's primary configuration file, a wide range of control points (such as virtual group faders, output master mutes, cues, subcues, etc.) can be "bound" to one or more user-defined network/MIDI senders and/or receivers, thus allowing a user to easily set up one and two-way real-time connections with external software and hardware, exposing as few or as many server control points as is appropriate for the situation. Using this functionality it is possible, for example, to create a simple control interface on the iPhone that remotely triggers cues and adjusts output levels, or to use faders on a MIDI controller to both display and control all virtual group fader and mute states. The methods by which network and MIDI "bindings" are established in the configuration file are documented for all relevant controls in the software, and should therefore provide a vehicle for the majority of custom user extensions to the control system.

4. ASSET MANAGER

In this section, we present the overall functionalities and the new workflow introduced by Asset Manager's framework.

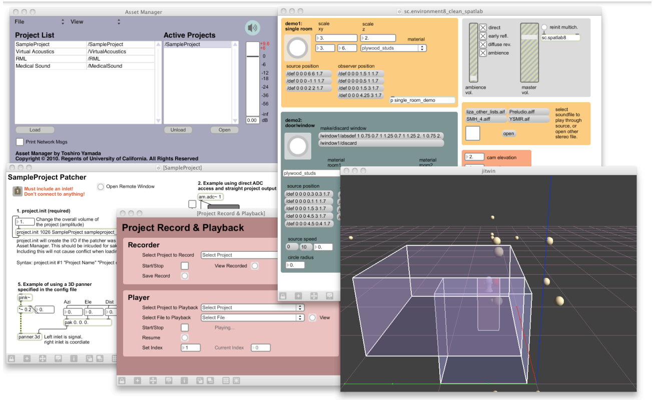


Figure 7: Asset Manager software with projects

4.1 Overview

Asset Manager is built to deal with multiple audio projects in an environment where projects need to be loaded on demand. We use Asset Manager extensively with virtual reality environments and other graphics engines, where each project requires a custom Max patch. Since each system has its own optimal spatialization setup, maintaining multiple versions of the same project implemented on different systems can quickly become cumbersome and time-exhaustive. Asset Manager addresses these difficulties by providing a framework that abstracts panning and signal flow, and helps optimize production workflow for complex sound systems.

Summary of Functionality:

- Configuration files for i/o setup, project definitions, signal paths and panning methods
- Versatile spatialization and i/o abstraction layers
- Network message specification in Open Sound Control protocol to control behavior of Asset Manager and projects
- Mixing control for each project and master outputs
- Recording and playback of network messages
- Built-in objects for spatialization signal processing, such as distance simulation, air absorption, Doppler effect, and source direction simulation

4.2 Signal Flow

4.2.1 *Project to System Outputs*

All audio signals from projects go through Asset Manager's system outputs layer, which serves as a final gain control stage before reaching the audio interface. Asset Manager provides a collection of Max patch abstractions that allow projects to utilize the software's signal paths. Furthermore, project volumes can be mixed independently from one another.

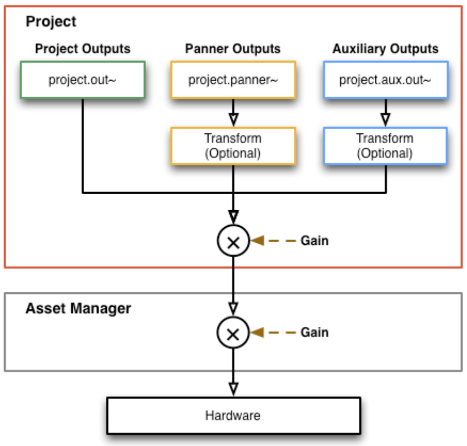


Figure 8: Project signal flow diagram

4.2.2 *Project Signal Paths*

Within a project, there are three paths a signal can take before reaching the system output. These paths bring logic separation and system abstraction that can be uniquely configured for different systems (Figure 8).

The main project outputs are accessed via the `project.out~` object. Signals passed to project outputs are routed directly to the main output without additional signal processing. In the project configuration file, channel IDs are mapped to hardware output channels. By using an indirect channel ID mapping (from project layer to hardware), the signal chain becomes independent from a specific audio system, and projects can be shared amongst different systems without having to modify the Max patch. This philosophy of abstraction is used throughout Asset Manager's signal flow design. Main project outputs can be used to route static audio sources, such as voice-overs, which are commonly routed to a single loudspeaker (e.g. to the center channel in a 5.1 surround sound setup).

Panner outputs are used to spatialize the sound – or "pan" the sound – using the `project.panner~` object. The panning method implemented in a project can be anything from stereo, 5.1 surround sound, Ambisonics, HRTF binaural, to custom implementations; the object is abstract and has no implementation on its own. The implementation is defined in the project configuration file where other project settings are also configured. Additionally, an optional transform function can be added after the panner signal path. A transform function is a black box that includes any operation that processes the signal from inputs to outputs. For example, it can be a simple matrix that routes five input channels and six output channels, where the sixth channel has the sum of all inputs that is routed to the subwoofer.

Auxiliary outputs, accessed with the `project.aux.out~` object, are used when the main project outputs and panner outputs do not fulfill a particular need. `project.aux.out~` is used similar to `project.out~` but also includes an optional transformation found in `project.panner~`. For example, auxiliary outputs are useful when a project contains pre-panned sources, e.g. 5.1 surround sound tracks, which require a transform function to match source outputs to system outputs. If the target system is headphones, a transform may be a 5.1 surround sound to HRTF binaural encoder.

These three signal paths are simple, yet powerful enough to support a variety of output requirements. Using these abstract objects, projects can be ported to work in Asset Manager's framework and take advantage of its workflow.

4.3 Workflow for Complex Sound Server Requirements

4.3.1 Abstraction of Panning Method

Much of the strength of Asset Manager comes from the ability to isolate the implementation of the panning method and rapidly adapt new panning methods in real-time. This abstract layer has saved many hours reconfiguring new panners, keeping multiple copies of different versions, and trying out various methods that may or may not work in a given system. By specifying the panner in a plaintext file, version control and project sharing becomes much easier. Moreover, once a well-behaved panner is chosen for a system, new projects can easily take advantage of it without altering the original Max patch. Reusing well-tested panners can also reduce the likelihood of using them improperly, thus diminishing time spent debugging.

4.3.2 Network Communication

Asset Manager uses Open Sound Control extensively for network communications and can be used with various 3rd party hardware and software. Via OSC (over TCP/IP or UDP sockets), remote applications can control and automate core functionalities, such as (un)load projects, change master and project volumes, (un)mute projects, dis-/en-able signal processing, and more. Asset Manager also includes rich tools for working with OSC messages.

4.3.3 Record and Playback Network Messages

OSC messages can be recorded and played for each project, keeping the exact timing as the messages arrive. Playback is done on a loopback socket to simulate real network messages. This is useful for archiving important events, generating reference materials, and demonstrating and debugging projects. The last point has been especially useful at our laboratories, where we have various complex graphics display systems that use Asset Manager for audio contents management and synchronous audio playback. Asset Manager runs on a dedicated audio server and communicates with the visual systems remotely. Operating these systems involves complex steps with many potential points of failure, and debugging these problems can be a tedious and time-consuming process. Using the network record and playback feature, we can test Asset Manager projects independent from other components of the systems and determine the point of failure faster.

5. CONTROL PROXY

Control Proxy is a simple software utility designed to facilitate network communication between various software and hardware components within an audio system. This software fulfills three primary functions:

1. Acts as a hub for network traffic, redirecting incoming messages from a given source to one or more IP and port destinations.
2. Applies network protocol translation between incoming and outgoing messages.
3. Provides a console interface for manually sending messages to user-defined destinations using the appropriate protocols.

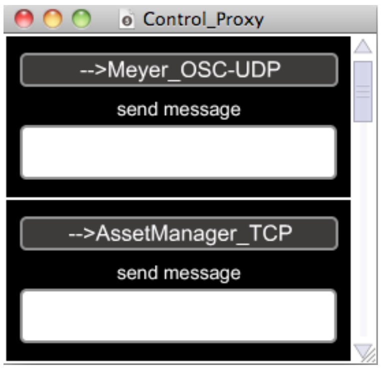


Figure 9: Control Proxy GUI

Logistically, this software provides developers with a *single* access point to a potential wide variety of destinations, without requiring them to know about destination-specific addresses, ports, and protocols. This allows audio staff to supply colleagues with a fixed set of ports at a single IP address that they can use when creating network links between the various non-audio software/hardware and audio systems. Text console network message windows serve as a convenient way to test interconnectivity, and to simulate commands coming from remote sources. Network protocol conversion serves to speed up development workflow when, for example, a software visualization tool sends messages only via UDP but needs to communicate with audio hardware that understands only TCP. All incoming to outgoing network redirection and conversion, as well labeling of each console window in the software's GUI, is defined in a simple plaintext configuration file similar in syntax to those used in AudioSwitcher Server and Asset Manager (Figure 9).

6. CONCLUSIONS

In this paper we have presented an open collection of software tools designed to enhance operations and development workflow on multi-functional audio systems within research facilities. The three software tools within our framework supply what we believe to be a core group of critical audio control capabilities – functionality required across a broad spectrum of audio systems scenarios, that if implemented well, have the potential for significantly streamlining audio systems operations and development pipelines. At the same time, the software remains sufficiently open and capable of supporting a wide variety of custom extensions.

In refining our notions of essential functionality, we have drawn upon our own experience operating on and developing for complex audio systems within dynamic and multi-media research environments. This functionality can be summarized as follows:

- Mixing/routing and system preset automation
- Dynamic audio asset management and control
- Network message routing/redirection and protocol conversion

Moving forward, we plan to improve upon a variety of features within the software, placing particular emphasis on ease of use. We will work towards a more complete integration of each software tools' configuration files into their respective GUI front-ends, allowing for easier real-time creation and editing without the need for script. We will also explore the benefits of replacing our own simple configuration file syntax with standardized file formats such as JSON, YAML, or XML.

Several functionality enhancements are planned for individual tools within the framework. In AudioSwitcher Server, we will implement "effects chain" functionality at the Input, Bus, and Output Masters stages, enabling users to dynamically load and modify a variable amount of custom Max patches or VST plug-ins at a particular stage in the signal flow. This will prove useful when, for example, you need to add a multi-tap delay or high-pass filter to a signal before sending it out to the loudspeakers. In Asset Manager, various spatial sound effects are in development, such as geometric acoustic simulations and multichannel reverberations. Furthermore, future releases will include a database backend for saving and restoring software states and accessing sound parameters and objects in real-time.

Finally, in an effort to improve documentation and discover overlooked core functionality, we hope to broaden the framework's user-base by releasing the tools open source to the community, and by continuing to work with research partners on implementations within their facilities. All software, as well as documentation, sample configuration files, and projects can be found at http://vis.kaust.edu.sa/tahakum.

7. ACKNOWLEDGMENTS

This project would not have been possible without the support of Steve Cutchin, Thomas A. DeFanti and all our colleagues at California Institute for Telecommunications and Information Technology and the KAUST Visualization Lab. Thanks to Paul Riker for his editorial assistance, and for helpful feedback on the software in its current state. Finally, we would like to thank Peter Otto of Calit2's Sonic Arts R&D group, for his essential insight and guidance throughout the design and implementation of the Tahakum framework.

8. REFERENCES

[1] S. Ellison, P. Otto, *Acoustics for reproducing sound at the visualization labs at the King Abdullah University of Science and Technology: A case study*. 159th Meeting of Acoustical Society of America: NOISE-CON 2010, Baltimore, USA, 2010 April 19-23.

[2] J. Fischer, F. Gropengiesser, S. Brix, *Cooperative Spatial Audio Authoring: Systems Approach and Analysis of Use Cases*. 126th AES Convention, Munich, Germany, 2009 May 7-10.

[3] F. Gropengiesser, K. Sattler, *An Extended Co-operative Transaction Model for XML*, Work-shop for Ph.D. Students in Information and Knowledge Management (PIKM'08), Napa Valley, USA, 2008 October 26–30.

[4] N. Humon et al. *Sound Traffic Control: An Interactive 3-D Audio System for Live Musical Performance*. Proceedings of the 1998 Conference on Auditory Displays, Glasgow, UK, 1998 November 1-4.

[5] F. Melchior, C. Sladeczek, A. Partzsch, S. Brix, *Design and Implementation of an Interactive Room Simulation for Wave Field Synthesis*. Proceedings of the AES 40th International Conference, Tokyo, Japan, 2010 October 8-10.

[6] D. Murphy and F. Rumsey, *A Scalable Spatial Sound Rendering System*. 110th AES Convention, Amsterdam, The Netherlands, 2001 May 12-15.

[7] T. Musil et al. *The CUBEmixer a performance, mixing and mastering tool.* Proceedings of the 2008 Linux Audio Conference, Cologne, Germany, 2008 Feb 28 - March 2.

[8] N. Peters et al. *A stratified approach for sound spatialization*. Proceedings of the 6th Sound and Music Computing Conference, Porto, Portugal, 2009 July 23-25.

[9] S. Wilson, J. Harrison. *Rethinking the BEAST: Recent developments in multichannel composition at Birmingham ElectroAcoustic Sound Theatre.* Organized Sound (2010) vol. 15 (03) pp. 239-250.