

ICAST: TRIALS AND TRIBULATIONS OF DEPLOYING LARGE SCALE COMPUTER-CONTROLLED SPEAKER ARRAYS

Stephen David Beck, Ph.D.

Louisiana State University
Center for Computation & Technol-
ogy
Laboratory for Creative Arts &
Technologies

Brian Willkie

Louisiana State University
Center for Computation & Technol-
ogy
Laboratory for Creative Arts &
Technologies

Joseph Patrick

Louisiana State University
Center for Computation & Technol-
ogy
Laboratory for Creative Arts &
Technologies

ABSTRACT

The Immersive Computer-controlled Audio Sound Theater (ICAST, pronounced “eye-cast”) is an on-going research project that addresses performance issues in electroacoustic music. By exploring a variety of differing performance modes through a computer-controlled loudspeaker environment that is malleable, adaptable and intuitive, we are able to explore new and novel metaphors for sound control using both conventional and unconventional interfaces. ICAST is a hardware and software solution, based on a client-server computer control system, commodity audio interface hardware, and high-quality audio amplifiers and loudspeakers.

Our system has been deployed for concerts and conferences throughout the past year. Through this initial deployment phase, we have identified critical issues with respect to our client/server systems design, our client software control, and our audio server. This paper identifies both the successes and the struggles we have had, and outlines the solutions we have taken to address these issues. What we have learned from this research has applications that go well beyond concert hall performances, and include virtual reality environments, cinema, and video games.

1. INTRODUCTION

Motivation for this project grew from our experiences hosting regional and national electroacoustic conferences (SEAMUS and Electric LaTex) as well as our own concert series, High Voltage, which presents 3 or 4 concerts throughout the academic year. Our response was the creation of the ICAST system, a portable 24-channel loudspeaker array, controlled by multiple computers in a client-server architecture. Our client software manages sound mixing and user interaction with the system, and is written using Max. Java code written to run within Max constructs OSC control messages that are sent over a wireless network to our server software, which is written in SuperCollider.

Having developed a sustainable and fairly robust system, we decided to deploy the system for our 2006-2007 concert season, which consisted of 3 major concerts and our regional LaTex conference (3 concerts). We were also invited to deploy ICAST at ICMC 2006 at Tulane University, where 5 afternoon concerts of fixed media and live interactive media works were presented.

In preparation for ICMC, we used the first two concerts as “warm-up” concerts, where we would anticipate the kinds of performance situations we would face in New Orleans, identify specific technical and structural problems, and quickly develop robust and transparent solutions for those problems. We would then use the robust code from ICMC as a base of development for new features we proposed in our previous work [Beck 2006].

2. ISSUES

The majority of problems faced in the development of ICAST were related to performance, and fell into three categories, client, server, and hardware. They were compounded by the lack of a dedicated rehearsal space in which to deploy the full system for testing. This led to feature additions that work on a small scale, but bump into processor limitations during full deployment for technical rehearsals and concerts resulting in audio dropouts and noise.

Noise was a serious problem that appeared in several guises. First was the noise from processor limitations that we did not anticipate in our original development. In tracking down the source of this noise, we discovered that the SuperCollider’s audio server, scserver, ran on only one of the machine’s two processors. Further research and correspondence with members of the SuperCollider community revealed that pre-compiled binaries behaved erratically on the dual-processor G5 and that compiling the source code for that specific architecture improved performance. But we have also discovered a problem with the high-pass filters native in SuperCollider. For many electroacoustically-generated sounds, these filters introduce a subtle but clearly audible distortion. At the time of this writing we are still seeking a clear diagnosis and solution to this problem.

Cycling ‘74’s Max, the environment in which we created our client software, was not originally designed to dynamically create objects, such as faders, number boxes, and other graphic controllers. While recent versions allow this functionality, it is not terribly efficient. One of our objectives was to dynamically create user interfaces based on the defined hardware architecture being used. This process creates a large number of graphical Max objects, and takes an inordinate amount of time that is particularly problematic at start-up time. Specifically, in order to support more than 24 channels, we create banks of fader strips, with each bank its own

window, increasing the number of objects to dynamically allocate upon start-up. To ameliorate this situation, we reduced the number of objects that Max had to create dynamically.

Beyond initialization, client performance problems were creating messaging bottlenecks. These bottlenecks resulted in long queues of messages between the various control elements and devices used in the system (i.e. TASCAM fader board). The increased communication latency created a most unusual situation where the TASCAM mixing controller appears self-animated. Very fast fader movements, not an uncommon situation in acousmatic sound diffusion, will crash the client, severing communication with the audio server. The server continues to process the audio at the levels set prior to the crash, but the user is not be able to adjust those levels until the client is restarted. On some occasions, these bottlenecks become so severe that we are forced to reinstall the TASCAM controller's firmware. We were able to improve client performance, stability and response by reducing the amount of messaging between Max and the controller through gating, and by using Java instead of JavaScript to construct control messages.

Our first, and most significant biggest problem we faced with the server was the playback of stereo files directly from within the audio server code. The files would simply stop for no apparent reason in the middle of playback at random points within the soundfile. Whether this behavior was connected to using pre-compiled binaries on a dual-processor G5, problems with our particular G5, some combination, or none of the above is unclear. Our solution thus far has been to play the audio files from a third computer and feed it to the physical inputs of the system. In the future we may attempt to stream the digital audio to the server in order to reduce the conversion to and from analog audio.

Finally, we addressed a number of performance issues with the server by significantly increasing the run-time values for some of the command line options such as the number of audio bus channels (-a, 2048 instead of 128), the max number of nodes (-n 2048 instead of 1024), and the real-time memory size (-m 1048576 instead of 8192).

2.1. The journey towards Java

Initially we found that Max does not provide sufficient support for text to construct OSC messages. Recent versions Max offered support for JavaScript and Java, and we initially began to use JavaScript to create the OSC messages. The code was very simple and much easier to write than if we attempted to program it in Max. Unfortunately JavaScript runs too slowly, contributing to the performance bottleneck mentioned earlier, thus we opted to port the JavaScript to Java.

One problem that arose after we began using Java occurred with initialization. For unknown reasons some objects were not correctly initialized and therefore created default synths on the server (the default synth in SuperCollider generates a loud sine tone, which proves

to be an effective indicator that something has gone wrong, if not an attractive addition to most electronic pieces). Our current solution is to use named attributes in Max to set the default values. This insures that the correct SynthDef will load, but does not insure that it will have the correct values, which may have to be reset by the user.

2.2. The journey towards distributed audio

As mentioned earlier, SuperCollider's server, *scsynth*, was not designed to run on in parallel on multi-processor systems. However recently a team of programmers has developed JackMP, a multi-processor version of the Jack Audio Connection Kit. JackMP permits the user route audio between two programs across separate processors thus allowing parallel audio processing. Currently we plan to distribute input channels (which incur the most overhead) across two instances of *scsynth* (*scsynthA* and *scsynthB*). All of the auxiliary busses and output busses will reside in a third instance of *scsynth* (*scsynthC*). Instances A and B will connect to all of our 60 physical inputs. Their outputs will connect to the loopback device provided by JackMP that allows the audio to be distributed while introducing no more than one predefined buffer of latency. The loopback device will connect to *scsynthC*, which in turn connects to the 60 physical outputs. The connections are made by *jack_connect* and can be automated with a shell script. At the time of this writing, performance results of this architecture are still pending.

In exploring this approach we uncovered a bug in the Mac OS X implementation of the JACK client. JACK expects each instance of the client to have a unique id. The Linux version of SuperCollider does so, however, the Mac version does not. To work around this we create multiple *scsynth* binaries, *scsynthA*, *scsynthB*, and *scsynthC*.

2.3. Client-Server Communication

OpenSoundControl (OSC) has made ICAST possible. Thus we rely heavily upon OSC support in Max and in SuperCollider. In 2006 Cycling '74 took steps to shore up its support for OSC with the introduction of the *udpsend* and *udpreceive* objects to replace the set of *udp* externals already in existence (*otudp* on Mac, *otdup-write* and *otudp-read* on Windows). While these objects offered some improvements, they introduced an unexpected problem. The Mac external *otudp* offered two modes, write and read. Additionally it required an IP address and a port number in write mode. Even when the user set its mode to "write," *otudp* provides an outlet where the user can access any messages sent to *otudp*'s port, whereas *udpsend* does not.

SuperCollider offers a method to get useful information from the server. To receive this information, one passes the OSC command/notify 1 to the server. The server then dutifully passes messages back to all IP/port

combinations from which it received a notify 1 command. Since `udp_send` does not provide read access to the write port, and `udp_receive` cannot simultaneously bind to the same port as `udp_send`, all of this useful information becomes inaccessible. Until Cycling '74 addresses this situation, we are forced to use `otudp`. This is a precarious situation since CNMAT has renounced further support of the `otudp` object.

3. REFINEMENT AND DEVELOPMENT THROUGH CONCERT EXPERIENCE

In June 2006, LSU-CCT was invited to present ICAST as the speaker array for use in concerts during ICMC 2006 at Tulane University. The system was to be used for the afternoon concerts held in McAllister Hall. Concerts held both before and after ICMC 2006 were used to debug and improve ICAST by introducing new features and exposing unplanned issues within the system.

3.1. September 2006

The ICAST system has grown in features with each concert. The first concert of the season was in September 2006 at the Shaw Center in Baton Rouge, with Elainie Lillios as the guest composer. This was the first concert produced on ICAST outside of the LSU School of Music, and showed the malleability of the system. The concert included 24 speakers plus a subwoofer. At this point, much of the scripting was done using JavaScript. This proved to be problematic for diffusion performance, as the JavaScript could not keep up with the movements of the diffusionist. This resulted in the TASCAM 2400 fader board crashing and needing the firmware to be reinstalled.

A lack of internal bussing caused issues when trying to implement new additions such as high pass filters. The model used in developing the routing within SuperCollider was a normal DAW. However, SuperCollider's tree-branch model of the nodes allows for a particular branch to input audio without affecting previous nodes. When adding any processing to an earlier node, later nodes are affected. This issue did not appear until technical rehearsals, as it was masked in small scale studio testing. After this concert, a new routing scheme was developed for use in future concerts, allowing for independent processing.

During the preproduction phase of the concert, it became apparent that file playback directly from SuperCollider was not working. This issue had arisen in previous concerts. The solution to date has been file playback using Logic Pro on a separate computer.

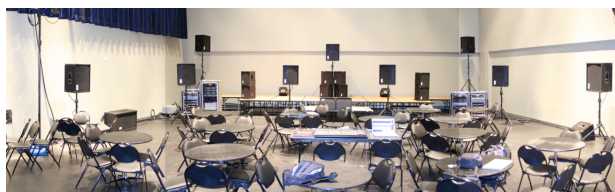


Figure 1: Photograph of Shaw Center deployment

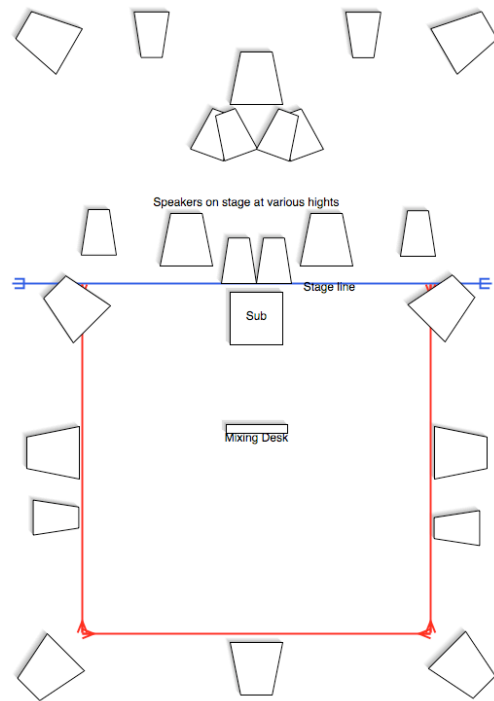


Figure 2. Speaker Configuration for the Shaw Center.

3.2. October 2006

The second concert of the season was in the LSU School of Music Recital Hall. For this concert, the JavaScript was ported to Java, causing a dramatic performance improvement. The synth nodes defined in SuperCollider were transformed from fixed numbers to dynamically assigned. This allowed for better management of the client-server communication, and was done with Java. Additionally, gates triggered by the touch sensors of the fader controller were added to prevent overloading. Four band parametric equalizers and simple delays were added to each channel strip. Tactile user control was increased with the addition of the TASCAM dials and the ability to select a control parameter from the fader board.

As with the September concert, the primary playback mechanism was Logic Pro. This concert employed a new feature of Macintosh OS 10.4, MIDI over Network. This allowed time-code from Logic to be transmitted for displayed within MAX, while also allowing for the User playback controls to operate Logic, creating a more user-friendly interface.

Due to hardware issues with the PowerMac G5 housing our server, the concert was run on an iMac (single 1.5GHz G5). By using another new Macintosh OS 10.4 feature, the aggregate audio device, SuperCollider was able to address multiple FireWire audio interfaces, and allowing the concert to use 24 speakers plus a subwoofer. The issues with the PowerMac arose with the full system in use, and did not appear in small scale testing, resulting in the replacement of the PowerMac. It showed the flexibility of the ICAST system.

3.3. ICMC 2006

The only major addition to the system was the ability to solo individual channels for live level setting. Our only technical issue relating to ICAST during ICMC 2006 in the McAllister venue was the computer used as a playback device failed on the last day of the conference. However, a backup computer for playback was available, and none of the concert delays were a direct result of ICAST.

At ICMC, the array was configured with 0 fixed speakers, four variable position speakers, and two stage monitors. Many composers did not take advantage of the speaker array for diffusion, in part because they were unfamiliar with either (a) the general concept of multi-channel diffusion, or (b) the specific layout of the diffusion array. In either case, a 15-30 minute rehearsal window was insufficient for many composers to develop a diffusion plan for their work. So they set a simple static mix that would suffice for the entire work.

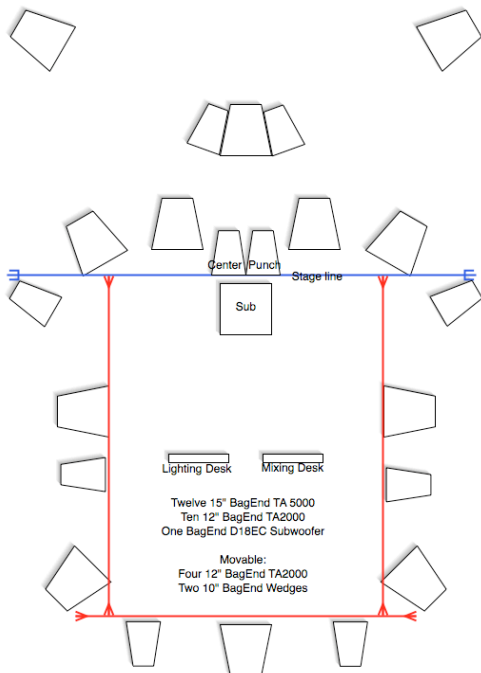


Figure 3. Speaker Configuration for ICMC 2006

3.4. Electric LaTeX 2006

Less than a week after ICMC was Electric LaTeX, a mini-festival of comprised of LSU, Tulane, Rice, UT-Austin and University of North Texas. This concert employed 26 speakers plus a subwoofer. To date, this is the largest array driven by ICAST. The limit on ICAST speaker control is 44 speakers (if external Digital to Analog converters are added, this number rises to 52 at 96kHz).

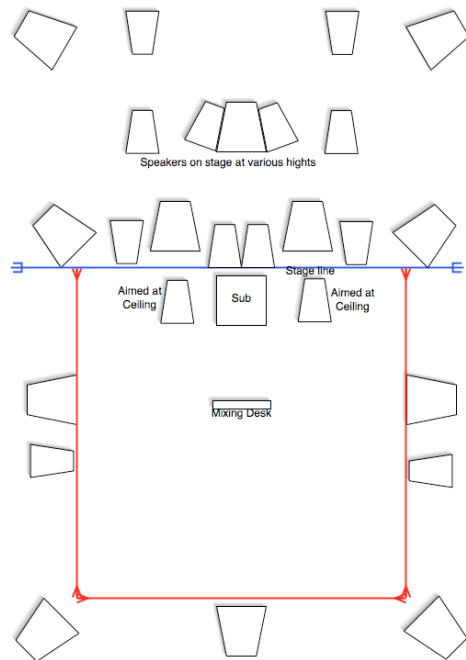


Figure 4. Speaker Configuration LSU Recital Hall.

3.5. March 2007

Jeff Stolet was the guest composer for the March 2007 held at the Shaw Center. This concert used 24 speakers with a subwoofer, and mimicked the September 2006 concert in speaker placement, with two exceptions; the front center speaker was turned to be rear-facing and the subwoofer was placed in a corner of the room rather than at the foot of the stage. More of the dynamic object creation code was ported from Max scripts to Java. The reduction in required Max objects being created by the Java code decreased the startup time of the client by 9.1%.

A new type of diffusion mode was introduced during this concert: vector fading. Vector fading is based on light board 'scenes,' where one fader can control multiple speakers at various preset levels, and each speaker can be controlled by any fader. Currently, a limit of 24 vector scenes can be programmed. The Vectors are programmed in Java.

Some issues arose with Vectors. In order to allow for multiple users to have individual Vector programs, a method of storage is needed. Within Max, the Java File objects were not working properly within our configuration, so text dump/read methods were developed to utilize the built in text objects of Max. The second issue is performance. Vectors are computationally expensive, and could not maintain realtime. During the concert, Max crashed as a result of the heavy Vector computation load. Because of the client/server architecture model of ICAST, the piece performed at the time of the crash played without issue from the server.

4. FUTURE DIRECTIONS

4.1. Alternative Controllers

In the quest to break from the fader-board paradigm, alternative controllers such as Lemuer are being developed. The client software currently uses the built-in MIDI capabilities of the TASCAM-2400 and Max. Expansion of these to other objects creates a new level of complexity for both the programmer and the end user. A technique to utilize multiple controller devices needs to be able to handle simultaneous controller commands without conflicts. Any performer using a non-standard controller needs to have time to practice and understand the response of the controller, making it difficult for guests to use ICAST alternatives. A user might have a device that he/she wishes to use, but without knowing the proper messaging commands, even standard MIDI devices would take time to reprogram for use with ICAST. To facilitate differing control devices, an API is being developed to allow performers to bring in a controller foreign to ICAST and connect with minimal difficulty.

4.2. Ambisonics

ICAST can accommodate pieces in many forms without changing the position of the speaker array. ICAST already has two-dimensional controllers in the form of X-Y joysticks, and while these have not been employed as of yet, they are available for use in Ambisonic performance. Code has been written but not yet fully tested for real-time encoding of mono and stereo inputs into first-order Ambisonic fields. Accompanying this is a first-order decoder for an arbitrary speaker array. This code is currently being tested for accuracy. Progressing to higher order encoder-decoder fields and being able to manipulate multiple fields in performance is a priority.

4.3. Simplification of Use

ICAST is still in the process of improving and upgrading. As a result, end user concerns have not been fully integrated within the current operational parameters of the system. ICAST currently requires Max/MSP (Client), Audio/MIDI Setup (playback synchronization and control with Logic), and Terminal (to control the server) to be running. Additionally, knowledge of specific command line options for SuperCollider are necessary to run scsynth properly on the server. The goal is to reduce the requirements to a few basic commands that can be applied via Max/MSP.

The operation of ICAST also needs optimization. The storage methods for unique users are cumbersome, and the assignment of operational controls is tedious. One solution is to create templates for the common venues where the system is deployed.

4.4. Intel Processors and Apple Computers

As Apple has moved all of its new computer lines to Intel processors, testing must be done to ensure continued operation on the different processor. Some of the issues with vectors and speed might be significantly improved on Intel machines, as the JavaVM was initially designed for Intel chips. The MacBook Pro line currently has multi-core processor, and a faster internal bus, which may alleviate some issues with Vectors and other client-side performance issues.

5. CONCLUSION

Sound diffusion is the performance practice of electroacoustic music, and ICAST is one of a growing collection of performance instruments in the United States. ICAST is an attempt to create a class of "diffusion instruments" that facilitate complex sound mixing through traditional and novel interfaces. As others begin to develop similar systems, we look forward to sharing our experiences and collaborating on developing similar approaches with the hope that a collective approach will spawn more and improved instruments.

We gratefully acknowledge the support of the Center for Computation & Technology at Louisiana State University and its Laboratory for Creative Arts & Technologies for supporting this project.

6. BIBLIOGRAPHY

Austin, L. (2001). "Sound diffusion in composition and performance practice II: An interview with Ambrose Field." *Computer Music Journal* 21(4), 21–30.

Beck, S.D., J. Patrick, K. Malveaux, B. Willkie (2006). "The Immersive Computer-controlled Audio Sound Theater: Experiments in multi-mode sound diffusion systems for electroacoustic music performance." *Proceedings of the 2006 International Computer Music Conference*, New Orleans, LA

Chadabe, J. (1997). *Electric sound : the past and promise of electronic music*. Prentice Hall.

Clozier, C. (2001). "The gmebaphone concept and the cybernephone instrument." *Computer Music Journal* 21(4), 81–90.

Gerzon, M. (1983). "Decoders for Feeding Irregular Loudspeaker Arrays." United States Patent 4,414,430.

Harrison, J. (1999, 9). "Diffusion: theories and practices, with particular reference to the beast system." *eContact* 2.4.

Malham, D. G. and A. Myatt (1995). "3D sound spatialization using ambisonic techniques." *Computer Music Journal* 19(4), 58–70.

McCartney, J. (1996). "SuperCollider: a new real time synthesis language." In *Proceedings of the International Computer Music Conference*.

Moore, F. R. (1989). "A general model for spatial processing of sounds." In C. Roads (Ed.), *The Music Machine*. MIT Press.

Roads, C. and J. Strawn (1985). *Foundations of Computer Music*. MIT Press.

Stampfl, P. and D. Dobler (2004). "Enhancing three-dimensional vision with three-dimensional sound." In *SIGGRAPH Proceedings*.

Ullmer, B., S. D. Beck, E. Seidel, and S. Iyengar (2005). "Development of "viz tangibles" and "viznet": Implementation for interactive visualization, simulation and collaboration." NSF MRI Award CNS-0521559.

Wright, M. and A. Freed (1997). "OpenSoundControl: A new protocol for communicating with sound synthesizers." In *Proceedings of the International Computer Music Conference*.

Zicarelli, D. (1997). *Max/MSP Software*. San Francisco.