An Update on the Development of OpenMixer

Elliot KERMIT-CANFIELD and **Fernando LOPEZ-LEZCANO**
CCRMA (Center for Computer Research in Music and Acoustics),
Stanford University
{kermit|nando}@ccrma.stanford.edu

Abstract

This paper serves to update the community on the development of OpenMixer, an open source, multichannel routing platform designed for CCRMA. Serving as the replacement for a digital mixer, OpenMixer provides routing control, Ambisonics decoders, digital room correction, and level metering to the CCRMA Listening Room's 22.4 channel, full sphere speaker array.

Keywords

Multichannel Diffusion, Spatialization, Ambisonics, Open Source Audio Software, 3D Audio

1 Introduction

The Listening Room at the Center for Computer Research in Music and Acoustics (CCRMA) is a versatile studio space serving the CCRMA community. Featuring a 22.4 channel spherical speaker array and a 32 speaker wave field synthesis system, the Listening Room is an important facility for research, music composition, and 3D sound diffusion among other uses. As conventional mixers are limited in configurability, channel count, and usability, we have implemented a software routing system—called OpenMixer—to control the primary speaker system.

Although conceived in 2006, the implementation of OpenMixer began in 2009. Since then, the capabilities of the Listening Room developed—there are more speakers, faster computers, and more users. To meet the demands, OpenMixer has been under continuous development. This paper serves as an update to its current status and extends Lopez-Lezcano and Sadural's 2010 LAC paper [1].

1.1 Original Implementation

OpenMixer is our hardware and software solution to the problem of routing, mixing and controlling the diffusion of multiple multichannel sources over a large number of speakers, and was tailored to the Listening Room studio at CCRMA. The control hardware is comprised of standard PC computer components housed in a fan-less case and off-the-shelf RME audio sound cards, plus external Solid Stage Logic AlphaLink MADI AX and Aphex 141 boxes for A/D and D/A conversion. The control software runs on Gnu/Linux and is a collection of free, open source software programs coordinated from a master control program written in the SuperCollider programming language. Inputs to the system include 3 ADAT connections to and from the Listening Room Linux workstation (a regular CCRMA Linux computer that can be used for music or sound work and has direct access to all speakers), 2 additional ADAT I/Os for external digital sources, 16 line level analog inputs, 8 microphone preamplifier inputs, 8 analog inputs for a media player, and 4 ethernet jacks connected to a separate network interface (which can also provide Internet access) that enable computers which have NetJack easy access to all speakers and features of the system.

At the time the previous paper was written, the Listening Room was equipped with only 16 speakers in a 4 + 8 + 4 configuration, which could either be addressed independently or be fed the signals of an Ambisonics decoder (second order horizontal, first order vertical) integrated into the system. Two Behringer USB control surfaces (BCF200 and BCR200) provided a simple and easy-to-use user interface.

1.2 Changes

The following sections outline most of the changes to OpenMixer since 2009, both in hardware and in the underlying control software and functionality.

2 Hardware Upgrades

2.1 Speakers

The number of speakers in the Listening Room has grown over time. The last major upgrade happened in March 2011, when the number of main speakers was changed from 16 to 22 (23 if we count an additional "center channel" which is only used for more accurate 5.1 or 7.1 playback). The spatial configuration was changed from 4 + 8 + 4 to 1 + 6 + 8 + 6 + 1, which enabled us to cover the sphere more uniformly and thus decode full periphonic third order Ambisonics with high accuracy (see figs. 1 to 3). In addition, four subwoofers were added, which required adding an additional eight channel ADAT D/A converter to the equipment rack.

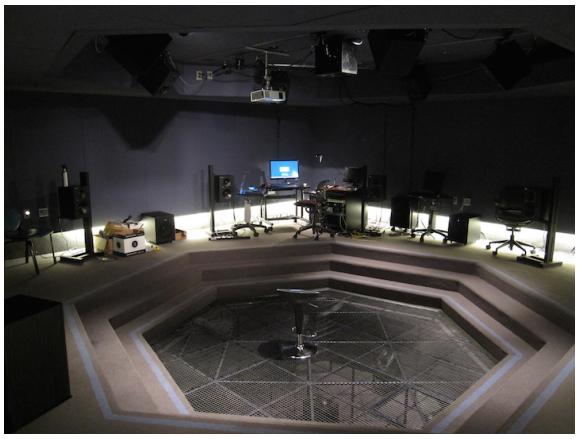

Figure 1: The Listening Room

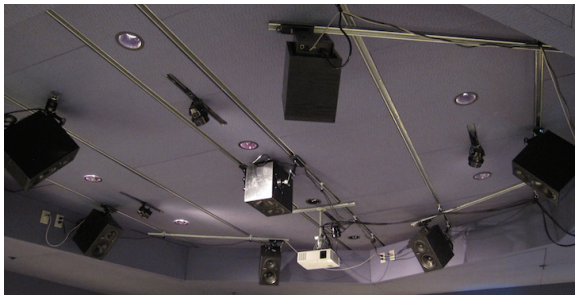

Figure 2: 7 elevated speakers

2.2 Computer and Peripherals

One of CCRMA's fan-less Linux machines is at the heart of the Listening Room—OpenMixer system. As we needed more processing power to support some of the planned expansions of the OpenMixer system, we migrated the hardware from an old quad core Q9550 CPU running at 2.83GHz, an EP45-DS3R Gigabyte motherboard, and 8G of RAM to a newer quad core i7-3770K CPU running at 3.50GHz and using a DZ77GA-70K Intel motherboard with 32G of RAM. At the same time we moved from the old Fedora 14 platform to Fedora 20 (new real time (RT) patched kernel and newer versions of all packages). An updated hardware signal-flow chart can be seen in fig. 4.

Figure 3: 7 speakers below the grid floor

The upgrade was anything but boring. Although much faster, the upgraded computer refused to work with the PCI audio cards used by the old hardware. We were using two RME cards in tandem—one RME Hammerfall DSP MADI PCI that interfaced with the main SSL A/D D/A box and an RME 9652 PCI card for additional ADAT ports. The upgraded motherboard included two legacy PCI slots for that purpose, but we were never able to have both cards working at the same time. The second card would fail to get interrupts delivered to it. This was difficult to debug and we spent countless hours switching cards around. We eventually had to move to newer PCI express cards with equivalent functionality (an RME HDSPe MADI and an RME RayDAT PCIe).

Even then, the technique of aggregating both cards into a composite ALSA device using *.asoundrc* did not work anymore. The newer kernel, ALSA drivers, and hardware could not get the card periods aligned, even though the sound cards are synced using word clock. JACK would immediately start generating xruns when started on the composite device—each card by itself would work fine. To get around this problem we tried running JACK on only one of the cards and connecting the second card to the system. We did this through either JACK's `audioadapter` internal client or `alsa_in` and `alsa_out` clients. In both cases, the DSP load was too high even for the upgraded hardware

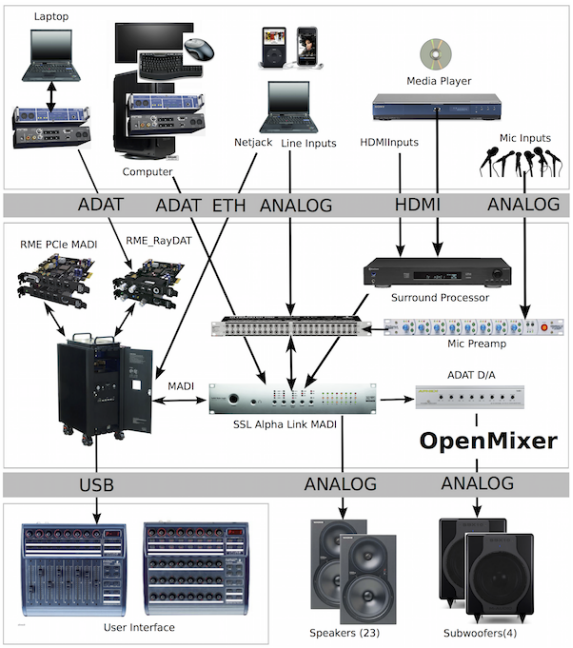

Figure 4: System hardware diagram

platform.

We finally used Fons Adriaensen's Zita-ajbridge to add the second card as a separate JACK client [2]. As in the previous solutions, this performs unnecessary sampling rate conversion of the second card inputs and outputs, but it was the only way to get the system to work reliably and the additional load was acceptable (between 9 and 10% of a core each for 64 input and output channels). Another twist to the story is that we could not get the MADI card to run as the master card when using the RayDAT as a secondary audio card. We had to run the RayDAT as the master, which increased the channel count to sample-rate conversion as well as the load. All these sound driver peculiarities need serious debugging, but we just did not have the time to do that.

For cases like this one it would be nice to have an option in Zita-ajbridge to not do sampling rate conversion at all, as the cards run in sync through word clock.

3 Software Upgrades

3.1 SuperCollider and Supernova

The majority of OpenMixer runs inside SuperCollider, a programming environment specifically designed for audio applications [3]. In addition to being well supported by a large developer/user community, SuperCollider is extendable through custom UGens and plugins and handles multichannel audio, MIDI, and OSC in a native and intuitive fashion.[1] SuperCollider itself is really two programs—an interpreted language (`sclang`) controlling a separate synthesis server (`scsynth`) running as a separate unix process.

In this new version of OpenMixer, we switched the synthesis server to `SuperNova`, an alternative server written by Tim Blechmann which supports automatic parallelization of the DSP load across multiple cores [4]. This simplified the software considerably, as before this, we were performing load balancing across cores by instantiating several instances of `scsynth` and controlling which server was chosen for particular tasks by hand.

When the OpenMixer computer boots, SuperCollider runs automatically as a boot service and starts all other ancillary software. The main task of SuperCollider is routing audio from multiple sources (Linux workstation, ADAT, analog, BluRay, mic preamps, and networks sources) to the speaker array, as well at to the Linux workstation and networks sources for recording. This represents a complex many-to-many routing relationship through the use of software buses (see fig. 5). In addition to routing straight from a channel to some number of speakers (each with independent gain control), OpenMixer also supports decoding of up to third order Ambisonics audio streams. Another important task that SuperCollider performs is as a task manager for subsidiary programs such as JACK and Jconvolver. If any of OpenMixer's auxiliary processes dies, the system restarts them automatically.[2]

3.2 System software

Together with the hardware changes, we upgraded the operating system to Fedora 20 and a new RT patched kernel [5]. To get the best real-time performance from the new system, we needed to perform a few new tweaks to its configuration.

We found that the Intel i7-3770K processor together with the new kernel (currently 3.14.x with the latest RT patch) used the new `intel_pstate` frequency scaling driver instead of the conventional

[1] We could only imagine the horrors of programming OpenMixer in C or C++!

[2] Except in very bad situations...

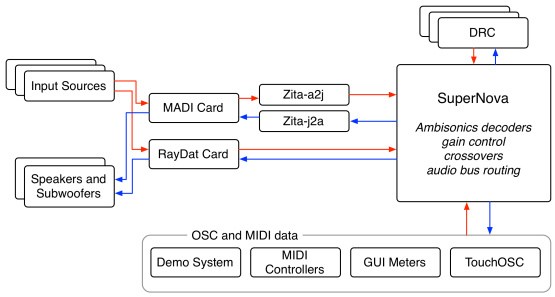

Figure 5: Simplified OpenMixer software routing diagram

governor-based scaling cpupower software. The `intel_pstate` driver can be controlled through variables in the `/sys` filesystem (`/sys/devices/system/cpu/intel_pstate`). The following incantation was used to tell the driver to try to run all cores at the maximum speed all the time:

```
echo 100 > /sys/devices/system/cpu/
  intel_pstate/min_perf_pct
```

We also disabled the `thermald` daemon which we normally run in all our Linux workstations, as it also tweaks the speed of the processor cores if the thermal load exceeds preset limits. We know what the load on the processor is, so there is no risk of thermal overloads. In both cases, we want the core speed to be pegged to the maximum available. It looks like the scheduler is not entirely aware of the current speed of each core and can migrate a computationally heavy task from a core running at full speed to one that is idling (and running at a very low clock frequency). The `intel_pstate` driver will of course adjust the speed of the slow core, but not fast enough to avoid an xrun in some cases.

We also had to disable hyper-threading (HT) in the BIOS. This disables the additional "fake" cores and brings down the total number from eight to four (but those four are real cores—we will use a cheaper Intel i5 processors which lacks hyper-threading for future deployments). Without these changes, we experienced occasional xruns in JACK. HT creates two logical cores out of each physical core by storing extra state information that enables both logical cores to appear to be independent processors. The potential improvement in performance is only realized by highly threaded applications and tops out at about 20% in ideal conditions. This comes at the cost of increased thread scheduling latency. We don't know how much latency HT actually adds, but it seems that it is enough in our current mix of tasks and threads to negate any advantage in raw processing power HT might offer [6–8].

The start-up scripts that boot OpenMixer have also changed. The boot activation of the software now happens through a static `systemd` service. The service executes a script that tweaks root level access configuration settings and then starts the main start-up script as the "openmixer" user. This in turn starts a private X server for future GUI extensions and transfers control to `sclang` (the SuperCollider language). The script also re-starts `sclang` if for some reason `sclang` dies.

3.3 User Interface

The user interface in the previous version of OpenMixer was limited to two Behringer control surfaces to access most common settings. This was a deliberate choice to keep the interface simple and mode-less, where every button, fader, and knob had only one clearly labeled function. Most of the interface has not changed but some new functions have been added (see fig. 6). For example, the master level control knobs also double as level meters for each bank of 8 channels (you can select the function of the knobs), there is a "DRC on/off" button that can be used to disable the correction filters on each speaker, we now have a monitor option (with optional pre-fader level) that routes any input back to the Linux workstation, we implemented a speaker test function to quickly make sure all speakers are operational and also a software reset that reinitializes OpenMixer.

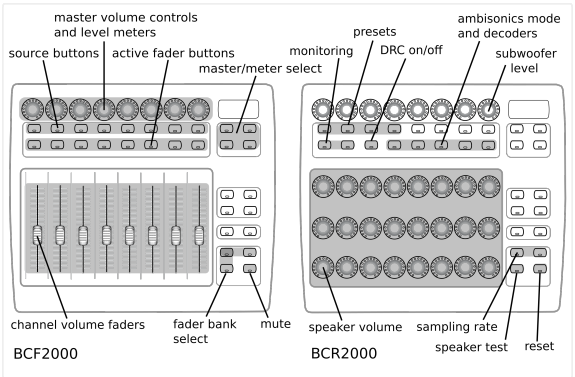

Figure 6: User interface

We have also employed SuperCollider's QT

GUI objects to build a graphic user interface so that a monitor connected to the control computer can display relevant information. For now, we have only implemented input and output VU level meters, but this change opens the door to future upgrades that should make OpenMixer even simpler to use, specially if we use a video screen with touch control.

OpenMixer is internally controlled through OSC messages. We have been therefore experimenting with applications like TouchOSC on tablets or smart phones to be able to control OpenMixer remotely through simple apps installed in user's equipment [9].

3.4 Subwoofer Processing

The addition of subwoofers to the Listening Room required us to implement proper crossovers in software. We used LinkwitzRiley 4^{th} order crossovers implemented as SuperCollider UGens that are part of the instruments that do software mixing and routing within OpenMixer.

3.5 Ambisonics Decoding with ADT and ATK

First through third order Ambisonics decoders are naively supported by OpenMixer. In the previous version of OpenMixer, Ambisonics decoders were provided by running `ambdec_cli` as a subprocess of the master SuperCollider program, with decoder matrices tuned to our speaker arrangement kindly supplied by Fons Adriansen. A stereo UHJ decoder was also provided by running Jconvolver as another subprocess with the supplied UHJ preset configuration.

In the new version of OpenMixer, we have switched to using decoders calculated by the Ambisonics Decoder Toolbox (ADT), written by Aaron Heller and Eric Benjamin [10–12]. ADT consists of Matlab/GNU Octave scripts that calculate Ambisonics decoders though a variety of techniques. In critical listening tests, the decoders have proved to perform very closely to the previous hand-tuned ones. In addition to generating Ambdec configuration files, ADT writes the decoders as Faust programs which can then be compiled to create native SuperCollider UGens [13]. For stereo UHJ decoding, we have switched to the Ambisonic Toolkit (ATK) UGens, written primarily by Joseph Anderson [14]. We have been able to fold Ambisonics decoding entirely into SuperCollider which minimizes the complexity of the code, the number of external programs used, JACK graph complexity, and context switches.

3.6 Digital Room Correction

From the start, OpenMixer calibrates all speakers for uniform sound pressure level and delay at the center of the room. While the new location of the speakers in the studio covers the sphere with better resolution, they are not really suspended in a free field condition. In particular, the speakers located under the grid floor (below the listener) have a different tonal quality due to the physical construction of the "pit" in which they are located. Furthermore, not all speakers are the same exact model (although they all share the same high frequency drivers). We have Mackie HR824s at ear level and smaller Mackie HR624s for the elevated and lowered speakers. The result of this led to incorrect and sometimes confusing rendering, especially when decoding full sphere Ambisonics (manifested through a tendency of sounds to be "pulled" towards the ceiling).

In the new version of OpenMixer, we have implemented digital room and speaker correction using Denis Sbragion's DRC software package, as described by Joern Nettingsmeier [15]. By recording the impulse responses of each speaker at the listening position, inverse FIR filters can be calculated to even out small differences in speaker impulse response due to the speakers themselves and their location in the room.[3] This, coupled with the very dry acoustics of the studio itself makes for an accurate and even reproduction of sound over the whole sphere around the listening position. The only trade off is an increase of approximately 10 milliseconds in the latency of the system. We have added an on/off switch to be able to bypass the DRC generated correction filters and access the speakers directly if necessary.

Jconvolver, written by Fons Adriaensen, is used as an external subprocess of the main SuperCollider program and provides an efficient, real-time convolution engine that performs the filtering [17].

3.7 Adding HDMI inputs

We recently had the need to allow users to connect HDMI audio based equipment (specifically game consoles—for research, of course—

[3] We used 20 second logarithmic chirps, recorded and analyzed with Fons Adriaensen's Aliki [16].

and also laptops). This adds another way to connect equipment to the system and can be used to play back 5.1 or 7.1 content (easier in those cases than using NetJack or analog interfaces). We bought an Outlaw Model 975 surround processor, a low cost unit that provides both HDMI switching and decoding of the most common compressed surround formats (Dolby Digital, DTS, Dolby TrueHD, and DTS-HD Master Audio, etc). The analog outputs of the unit were connected to the "player" analog interface, and the existing Oppo BluRay player was connected to one of the Outlaw's HDMI inputs.

The only gripe is that the unit cannot (yet) be controlled from OpenMixer itself. It would be nice to be able to control everything from the OpenMixer control surfaces so that the user does not need to know which HDMI input is which and how the surround processor is controlled. We could use an infrared remote control, or inquire if the RS232 interface included in the unit can be used for control in addition to firmware upgrades.

3.8 Demo Mode

OpenMixer makes the Listening Room easily configurable, however, it requires some effort between walking into the studio and hearing sound. In the past, demonstrations of the Listening Room's sound system—which are quite frequent—typically included hunting for saved Ardour sessions to recall and some amount of guesswork to find projects spread across CCRMA's filesystem. To simplify this process, we implemented a demo mode that highlights the capabilities of the system in an easy to use way. We used a Korg nanoKONTROL2 as a control surface that is patched directly to OpenMixer. This controller has eight channel strips (three buttons, a knob and a fader) and transport control buttons. We have pre-rendered an assortment of multi-channel works for the Listening Room's speaker configuration that can be easily triggered through the nanoKONTROL2.

With minimal effort, someone can listen to a curated set of mixes of great variety. We have included selections that show off different types of 3D diffusion (e.g., quad, 5.1, octophonic, third order periphonic Ambisonics, etc.) as well as mixing style (e.g., "concert hall-esque," "fully immersive," "academic electroacoustic," etc.). One such piece, a recording mixed through Ambisonics, is rendered both in full third order Ambisonics and stereo UHJ in a time-synced way so the "immersive-ness" of the decoders can be audified and evaluated.

4 Future Work

Although OpenMixer was originally written for the CCRMA Listening Room, we are working to parameterize the scripts so that it is more portable.[4] Once appropriately parameterized, we envision OpenMixer as a useful tool for other people controlling multichannel speaker arrays.[5] We are in the process of implementing OpenMixer in our small concert hall, the Stage, that is equipped with a 16.8 channel 3D speaker system. OpenMixer will make it possible to easily move pieces from the Listening Room environment to the concert hall. This is especially true in the Ambisonics domain, as new decoder technologies implemented in ADT make it relatively easy to design effective 3D decoders for dome speaker configurations such as the one we have on the Stage. CCRMA concerts frequently feature large speaker arrays (up to 24.8) which are controlled with similar systems. Making OpenMixer a general solution would simplify these large scale productions.

Much work remains to be done in the software itself, in particular, to use the new X-based GUI interface and to control the whole system through OSC from tablets and smart phones.

In the Listening Room itself, we have plans to add more subwoofers (for a total of 8) to increase both the fidelity and localization resolution of low frequencies.

As the technology gets faster and the number of speakers grow, OpenMixer will most likely remain a work-in-progress.

We originally intended to use JackTrip as an additional input source for remote performances, but the feature did not see much demand and it was impossible to make JackTrip work (without changes to the source code) as a reliable subprocess to the SuperCollider code. We plan to use the currently unimplemented JackTrip inputs to house Fons Adriansen's Zitanjbridge (network-JACK) JACK clients, thus

[4]Naturally, this is a challenging task as speaker configuration, sound cards, input sources, etc. are different for each system.

[5]More information about OpenMixer can be found at https://ccrma.stanford.edu/software/openmixer.

providing a way to connect to the Listening Room from remote locations [18].

5 Conclusions

OpenMixer is certainly not the only solution for controlling 3D speaker arrays at CCRMA, but it has fulfilled its purpose to make the Listening Room configurable for many different uses. The latest inclusion of DRC-based impulse response calibration has significantly improved the quality of sound diffusion in the Listening Room, and is being considered for other listening spaces at CCRMA.

6 Acknowledgments

The success of OpenMixer is largely due to the support of the CCRMA community. Many thanks to Chris Chafe and Jonathan Berger for embracing the Listening Room in their classes and projects. OpenMixer is built on free, open source software and would never have existed without all who contributed to those projects.

References

[1] F. Lopez-Lezcano and J. Sadural, "Openmixer: a routing mixer for multichannel studios," *Linux Audio Conference*, 2010.

[2] F. Adriaensen, "Zita-ajbridge, alsa-jack and jack-alsa resampler." `http://kokkinizita.linuxaudio.org/linuxaudio/`.

[3] J. McCartney, "Supercollider: real-time audio synthesis and algorithmic composition." `http://supercollider.sourceforge.net/`.

[4] T. Blechmann, "Supernova: a multiprocessor aware real-time audio synthesis engine for supercollider," Master's thesis, TU Wien, 2011.

[5] Various, "Real-time linux patches." `https://rt.wiki.kernel.org/index.php/Main_Page/`.

[6] A. Valles, "Performance insights to intel hyper-threading technology." `https://software.intel.com/en-us/articles/performance-insights-to-intel-hyper-threading-technology`.

[7] T. Leng, R. Ali, J. Hsieh, V. Mashayekhi, and R. Rooholamini, "An empirical study of hyper-threading in high performance computing clusters," *Linux Clusters Institute Conference*, 2002.

[8] Calomel.org, "Network tuning and performance: a simple guide to enhancing network speeds." `https://calomel.org/network_performance.html`.

[9] Hexler.net, "Touchosc: Modular osc and midi control surface for iphone/ipod touch/ipad." `http://hexler.net/software/touchosc`.

[10] A. Heller and E. Benjamin, "Ambisonic decoder toolkit." `http://www.ai.sri.com/ajh/ambisonics/`.

[11] A. Heller, E. Benjamin, and R. Lee, "A toolkit for the design of ambisonic decoders," *Proceedings of the Linux Audio Conference 2012*, 2012.

[12] A. Heller and E. Benjamin, "The ambisonics decoder toolbox: Extensions for partial-coverage loudspeaker arrays," *Proceedings of the Linux Audio Conference 2014*, 2014.

[13] Y. Orlarey, "Faust: the quick path from ideas to efficient dsp." `http://faust.grame.fr/`.

[14] J. Anderson, "The ambisonic toolkit: tools for soundfield-kernel composition." `http://www.ambisonictoolkit.net/`.

[15] D. Sbragion, "Drc: Digital room correction." `http://drc-fir.sourceforge.net/`.

[16] F. Adriaensen, "Aliki, impulse response measurement tool." `http://kokkinizita.linuxaudio.org/linuxaudio/`.

[17] F. Adriaensen, "Jconvolver, a convolution engine." `http://kokkinizita.linuxaudio.org/linuxaudio/`.

[18] F. Adriaensen, "Zita-njbridge, network-jack and jack-alsa resampler." `http://kokkinizita.linuxaudio.org/linuxaudio/`.