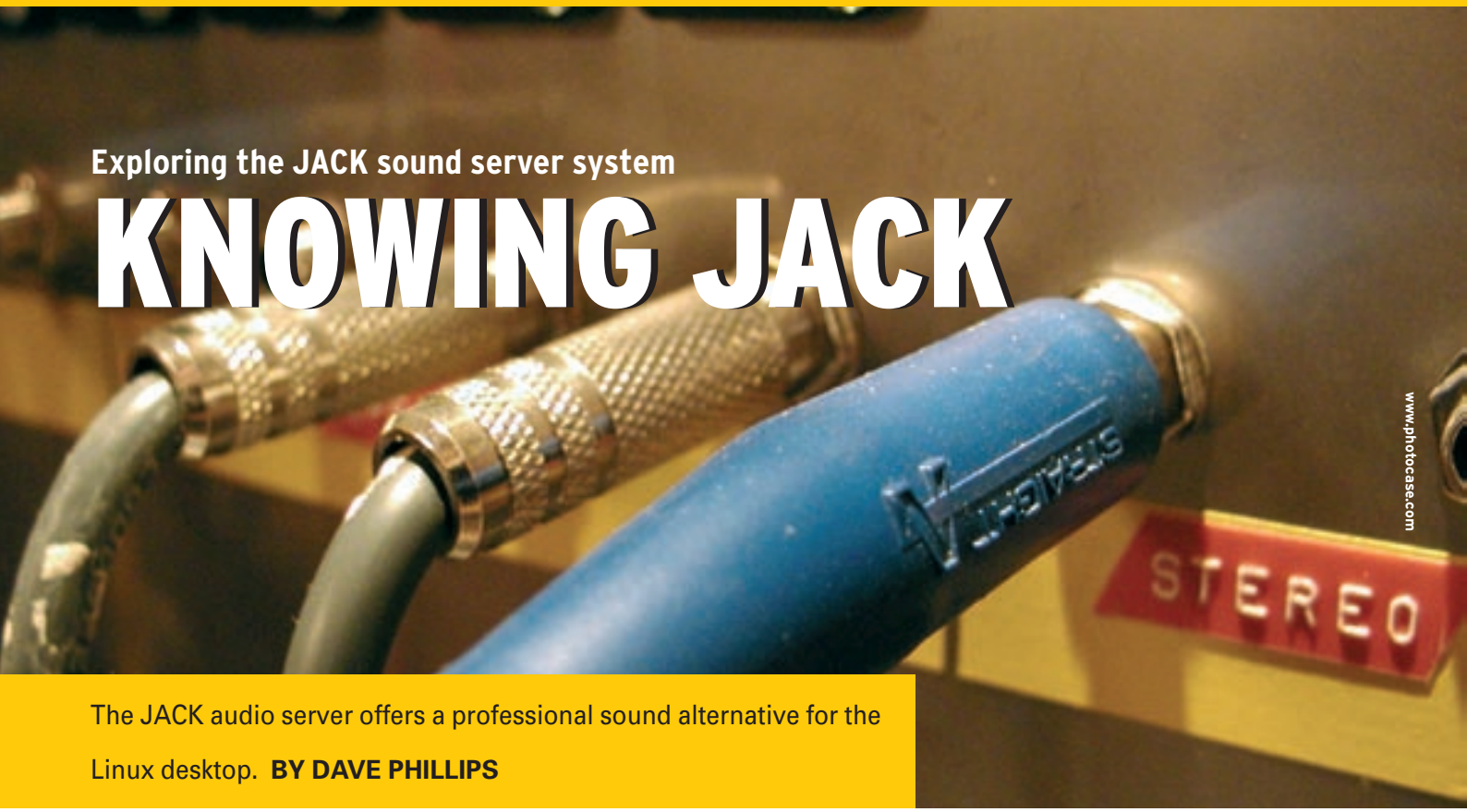

www.photocase.com

Exploring the JACK sound server system

KNOWING JACK

The JACK audio server offers a professional sound alternative for the Linux desktop. **BY DAVE PHILLIPS**

Most Linux users know that a *server* provides services, whether on one machine or on a network. The purpose of a server is to eliminate the need to directly access these services at the application level. For example, an X server manages access to and control of the video services of the computer's graphics chipsets, relieving user-level application developers of the burden of programming for those services directly.

An audio server manages access to the capabilities and services of the installed audio devices. These audio devices include soundcards, on-board audio chipsets, and any other audio hardware (such as telephony hardware, combined A/V cards, television, and radio boards).

The graphic Linux desktop depends upon X for its graphics and video services. Thus, your favorite KDE or GNOME applications include routines for accessing those services through the X application's programming interface. Programmers can code for the hardware via a generalized API instead of having to directly address the hardware. Alas, the sonic Linux desktop lacks a single standardized solution for serving audio resources system-wide. Instead, a variety of solutions have appeared, including the artsd, esd, NAS, and JACK systems.

For triggering system sounds, listening to CD and DVD audio, and simple recording, the demands on an audio server are relatively light. Managing several audio streams at this level requires no sample-accurate synchronization, nor is there a need for a highly flexible client routing system. Most users simply want to be able to play recorded audio without blocking other audio streams. Artsd and esd are sound servers designed to meet these requirements for the KDE and GNOME desktops. The Network Audio System (NAS) is an alternative network-friendly client/server audio delivery system intended to serve as an audio equivalent of the X server. Within their limits, Artsd, esd, and NAS all function effectively. However, none of these servers provides sample-accurate synchronized I/O of multiple audio data streams, nor were they designed for performance within low-latency systems. If your audio needs require these high-end capabilities, you have ventured into the domain of professional audio systems, and now you need to know JACK.

Introducing JACK

Developer Paul Davis has created one of the most remarkable pieces of open-source audio software, the JACK Audio Connection Kit, better known as just JACK. JACK is specifically designed for systems tuned for low-latency and high demand. Professional audio recording systems cannot afford audible delays and dropouts (known as xruns), and such systems are expected to support the synchronous operation of multiple clients in a low-latency environment.

JACK comes with features such as:

- support for any ALSA-compatible sound device
- support for a variety of audio system back-ends (ALSA, OSS/Linux, PortAudio, CoreAudio)
- free connectivity between clients – without delays or dropouts
- support for a master transport control system

JACK's primary task is the management of multiple audio data streams, coming from and going into a variety of applications with synchronized I/O. JACK requires an audio system – it is not a replacement for an audio system such as ALSA or OSS/Linux. JACK does not supply soundcard drivers, nor does it access hardware directly; instead it depends on a low-level audio layer to handle that communication, which in Linux is either ALSA or OSS. JACK doesn't care about the underlying hardware, it simply wants to manage the streams coming into and out of your devices.

Building and Installing JACK

JACK is available as a basic package in the AGNULA/Demudi and Planet CCRMA audio-optimized systems. A

source tarball of the latest public release is available from the JACK website. The site also provides instructions for building JACK from CVS sources for those who want to keep up with the latest development. No special build requirements are needed for JACK itself beyond Erik de Castro Lopo's libsndfile audio file I/O library.

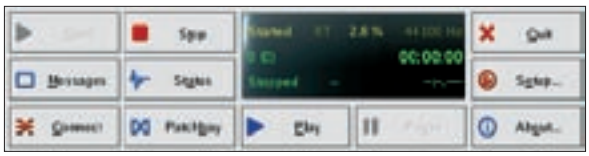

Figure 1: Controlling JACK from QjackCtl.

According to the JACK FAQ, you must have a recent Linux kernel (2.4 or higher) with the tmpfs file system turned on. Most modern distributions will have this turned on by default, but you can check for it by running *cat /proc/filesystems*. The FAQ also states that you must mount a shared memory filesystem on */dev/shm*, advising that the following line be added to */etc/fstab*:

```
shmfs    /dev/shm    shm    ➥
defaults        0       0
```

The FAQ further notes that you may have to create the */dev/shm* directory yourself.

After unpacking the sources, simply enter the new JACK directory, read the README for up-to-date instructions, then invoke *./configure --help* to see the available configuration options. JACK is built with the familiar autotools utilities, so for most users the compile process is as easy as running *./configure [your options here]; make; make install*.

Installing JACK from an RPM or other package also requires no special support. Follow the basic installation procedure for your system, and voila, you have a fresh JACK system ready for use.

Starting JACK

The JACK server is launched with either *jackd* or *jackstart*. The JACK manual page (*man jackd*) tells us that *jackd* invokes the JACK server daemon and that *jackstart* is used when using JACK's built-in support for realtime capabilities. All options are the same for either invocation. For most users working on systems with a patched 2.4 kernel, *jackstart* will be the preferred method of starting the server. Users working with 2.6 kernels should use *jackd*.

Here's a relatively simple beginning :

```
jackd -R -d alsa -d hw:0
```

In this example, JACK has been started with realtime capability, acknowledging the ALSA back-end and addressing the first hardware device in the audio system. The *-d hw:0* switch is actually unnecessary; the hardware selection always defaults to *hw:0* anyway. Obviously, you would use a different number for a different card or chipset in a system with multiple sound devices.

Here's a slightly more complex example for my SBLive soundcard :

```
jackstart -R -d alsa -d hw:1➥
 -p 512 -r 48000 -z s
```

Once again, we see the realtime and ALSA options. The device selector is numbered *hw:1* because the SBLive is the second card in that particular machine. I've added options for the buffer size (*-p*), for the JACK sample rate (*-r*), and for the audio dithering option (*-z*).

Note that the *-p* option sets the software buffer size. As Jack O'Quin points out, this is the buffer size seen by all JACK clients.

See the box titled "JACK Options" for more on command line options for JACK.

GUIs for JACK

We have already seen JACK in action at the command prompt. However, when working in an X environment it's nicer to have a GUI for JACK's setup and configuration, and thanks to developer Rui Nuno Capela, we have the wonderful QJackCtl (Figure 1). This most helpful utility provides an all-in-one graphic interface for configuring and controlling all of JACK's operations. In addition to the convenient setup dialog (Figure 2), QJC supplies an audio connections panel for JACK clients and a set of basic JACK transport controls (if you want to use QJC as the JACK transport master). QJC further supplies messaging and status display panels, controls to start and stop the server, and play/pause controls for JACK's transport control system.

QJC also includes a MIDI connections panel for ALSA sequencer clients, letting

Regarding Soundcards

The audio industry distinguishes between consumer and professional grade audio devices. Consumer-grade devices include PCI and USB audio interfaces, on-board chipsets for integrated desktop and laptop sound support, and more advanced hardware such as Creative's SB Live! and Audigy cards. These devices normally provide channels for a master volume control, PCM and CD audio output, and inputs for microphone and line-level signals. The master volume, CD, mic, and line channels are self-explanatory. The PCM channel is a general digital audio playback channel providing volume control for programs playing WAV, AIFF, OGG, MP3, and other sound-file types.

Depending on the audio chipset, these basic services may be expanded to include channels for on-board synthesizer output, digital audio connections, surround sound channels, and bass/treble tone controls. Software mixers such as alsamixer poll the audio hardware for its capabilities and configure the mixer to display the available channels and switches. Thus, my laptop's CS4232 audio chipset supplies little more than the basic services, while my desktop machine's SBLive provides a much larger array.

Professional-grade audio boards such as the RME Hammerfall or the M-Audio Delta cards are designed to satisfy different needs, providing higher-quality audio connectivity such as AES/EBU and balanced 1/4" plugs, a greater number of audio I/O channels, higher sampling rates, and hardware synchronization capabilities. These professional-quality cards may or may not include hardware MIDI connectivity, and they do not usually include consumer-grade amenities such as an on-board synthesizer or a connector to your CD drive's audio output.

The distinction between the grades is obscured by some more advanced devices intended for the consumer market, and it is certainly possible to achieve truly high-quality results from some of the more modern soundcards. However, for truly professional requirements you'll need professional-grade audio hardware.

users manage audio and MIDI connectivity from a single control interface. You can save and load your total connections graph as a Profile in QJC's Patchbay (Figure 3). The Patchbay's operation isn't quite automatic, but it is a real time-saver if your connections are many and complex.

QJC is my favorite standard tool for controlling JACK, but there are at least two other GUIs for managing JACK connectivity. Dave Robillard's Patchage is a patchbay for both JACK audio and ALSA MIDI connectivity via its unique visual interface (Figure 4). Matthias Nagorni's QJackConnect is a nice JACK-only QT-based patchbay, but it appears that project development is on hold.

Applications Using JACK

JACK support has become an expected feature in new Linux audio software. As a result, the list of implementations has become too lengthy to print here, but its

JACK Options

When you first meet JACK, you may be confused by some of its options. This brief summary will help you find your way around.

First the parameter settings:

- *-R, --realtime* – Starts JACK with realtime scheduler priority. Normally, you will want this option enabled, but be aware that it works only if you have root status or are running a kernel that grants such status to a normal user. Kernels from AGNULA/Demudi and Planet CCRMA are prepared for such status, but any kernel can be patched and modified for low-latency with root-user capabilities enabled. Jack O'Quin indicated to me that JACK needs root privileges only for realtime scheduling and memory locking. I asked members of the Linux Audio Users mail list whether there might be good reasons to *not* use the realtime option, and I learned that JACK is still useful on systems without realtime capability, hence the option. Additionally, you might want to turn off realtime capabilities in a testing or troubleshooting scenario.
- *-m, --no-mlock* – Signals JACK to keep memory unlocked. Paul Davis explained that this option could be useful when running JACK in realtime on a system whose physical RAM is being consumed by JACK and its clients.
- *-u, --unlock* – Unlocks memory claimed by graphics toolkits (GTK, QT, FLTK, WINE). Again, this option is useful for machines with low amounts of memory (physical RAM), but it is especially useful for users running VST/VSTi plugins and other WINE-dependent applications. In some cases, such applications may not run at all until this option is selected.
- *-s, --softmode* – Ignores xruns reported by an ALSA driver, making JACK less likely to disconnect unresponsive ports when run without realtime status. You might select this option to avoid too-copious error reports. This option might also be valuable for live performance.
- *-S, --shorts* – Forces JACK's I/O to 16 bits. As Lee Revell pointed out, JACK's internal processing is always carried out at 32 bits, and by default, it will attempt to set the bit resolution at its input and output stages to 32, 24, and 16, in that order, reporting success or failure with each attempt. Users with cards known to work optimally at 16 bits might want to use this option just to avoid the error reports.
- *-H, --hwmon* – Enables hardware monitoring of ALSA's capture ports, providing zero-latency monitoring of audio input. Requires hardware and device driver support. The jackd man page says when this option is enabled, "requests to monitor capture ports will be satisfied by creating a direct signal path between audio interface input and output connectors, with no processing by the host computer at all. This offers the lowest possible latency for the monitored signal."
- *-M, --hwmeter* Another ALSA-only option. Enables hardware metering if your soundcard supports it. Paul Davis notes that this option is used only rarely and that it is likely to be removed in future releases.
- *-z, --dither* – Dithering is a process that minimizes unwanted side-effects of reducing an audio file's bit-depth. Low-level noise is mixed into a signal to randomize digital audio quantization errors, turning audible and unpleasant digital distortion into something more closely resembling analog noise. According to Paul Davis, dithering is especially helpful when your soundcard's output is less than 24-bit resolution and you run JACK at the hardware's real sample rate.
- *-P, --realtime-priority* – Sets the realtime scheduler priority. Normally, you can leave this setting at its default value of 10. If your kernel includes realtime preemption, you might want to set this value to at least 70 to keep JACK running ahead of interrupt handlers.
- *-p, --port-max* – Sets the maximum number of JACK output ports. This option is especially valuable for people using a lot of tracks in Ardour. The default of 128 should be enough for most users. QjackCtl lets you select up to 512 ports, but more are available with sufficient memory.
- *-d, --driver* – Select hardware driver. In fact, you're selecting the audio system back-end with this option. Currently supported systems include ALSA, OSS/Linux, CoreAudio, PortAudio, and a dummy system (useful for testing). Most Linux users will want to choose either ALSA or OSS.
- *-r, --rate* – Sets JACK's sample rate. The default is 48000 Hz, but you may need to experiment to determine the best sample rate for your system. Lower-powered systems may find it necessary to bring down the sample rate to improve performance, but generally you want a rate of at least 44100 Hz for high-quality sound. Note too that some soundcards (e.g., the SB-Live) work well only at a single sample rate.
- *-p, --period* – Specifies the number of frames between JACK's *process()* function calls. The default value is 1024, but for low latency you should set *-p* as low as possible without producing xruns. Larger periods yield higher latency but also make xruns less likely, so you may have to experiment to find the optimal setting for your hardware. Incidentally, *man jackd* tells us that JACK's input latency (measured in seconds) is *--period* divided by *--rate*.
- *-i, --inchannels*; *-o, --outchannels* – These settings determine the number of audio I/O channels. The default is the maximum number supported by your hardware.
- *-n, --nperiods* – Specifies the number of periods in the hardware buffer. The default value is 2. The period size (*-p*) times *--nperiods* times four will equal the JACK buffer size in bytes.
- *-C, --capture*; *-P, --playback*; *-D, --duplex* – Set JACK to record-only, playback-only, or full duplex status (simultaneous play and record). This setting can be very important: Some cards will simply not perform well in duplex mode but work quite well in the simplex modes.

domains of implementation include hard-disk recording systems (Ardour, ecasound, Wired), drum machines/rhythm programmers (Hydrogen), software sound synthesis environments (Csound5, SuperCollider3), audio/MIDI sequencers (Rosegarden, MusE, seq24), soundfile editors (Snd, Audacity, mhWaveEdit, ReZound), and standalone softsynths (AMS, Om, ZynAddSubFX). Other significant JACK-savvy projects include the LinuxSampler and Specimen sampler projects and the various schemes for supporting VST/VSTi audio plugins under Linux (these schemes also require the WINE software). Linux media playback systems such as MPlayer, XMMS, and AlsaPlayer also provide JACK support.

Readers should note that these applications vary in the scope of their JACK support. Some use only its audio connectivity, some use only partial implementations of its transport control, and a few already take more complete advantage of JACK's features. Please consult the documentation for any JACK-aware application to determine the extent of its support.

The basic JACK package includes a number of useful command-line tools such as *jack_connect/jack_disconnect* (manages client connections), *jack_metro* (a configurable metronome), *jack_lsp* (lists JACK ports, their connections and properties), and *jack_transport* (manages JACK transport control status).

JACK has also inspired a wave of cool tools and utilities. Bob Ham's JACK-Rack is a very useful container for LADSPA plugins that lets you build a virtual rack of audio processing modules with MIDI control of plugin parameters. Steve Harris's JAMin is the result of a collective effort by Linux audio professionals to create a pro-quality stereo mastering interface based on LADSPA audio signal processing plugins. Timemachine is another treat from Steve Harris. It's essentially a recorder that always maintains a buffer of the last ten seconds of recorded material. When fully armed, Timemachine writes the buffer to disk and continues recording in real-time. Fons Adriaensen's JAAA (JACK and ALSA Audio Analyser) is a professional-grade signal generator and spectrum analyser designed for accurate audio measurement. And just to show that there's no absolute need for a fancy GUI, Florian Schmidt's jack_convolve is a JACK-based command-line convolution engine, very handy for creating high-quality reverb effects and other interesting sounds.

URLs for all these and other JACK applications are listed on the Linux Sound & MIDI Software site [9].

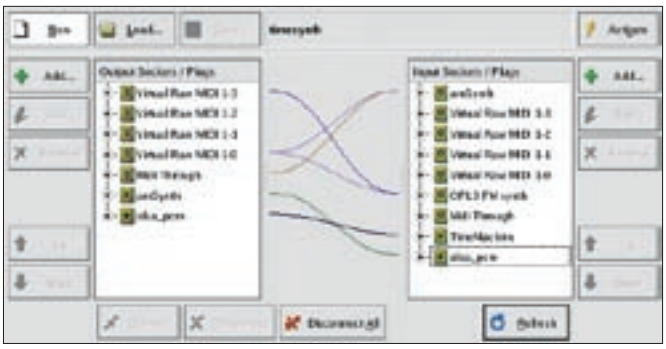

Figure 3: QJC's Patchbay helps you manage connections.

JACK In Action

Figures 5 and 6 show off JACK in two typical uses here at Studio Dave. Figure 6 illustrates the simpler use in an audio-plus-MIDI network combining the seq24 MIDI sequencer, the QSynth soundfont-based synthesizer, and the JACK-Rack, all operating on my PII 366 Omnibook and its humble Crystal Sound CS4232 chipset. Figure 5 demonstrates a more ambitious set of routing and connections with JACK managing I/O.

There's little more to say about using JACK in these scenarios. Once I've configured JACK, its performance is completely transparent. All I have to do is make my connections and make my music.

Programming With JACK

Programming with the JACK API is beyond the scope of this article. Interested readers can find excellent instructional material in the JACK source code (see simple_client.c in the example_clients directory) and on various Web sites. James Shuttleworth's tutorial at *http://www.dis-dot-dat.net/index.cgi?item=/jacktuts/starting/* is a well-written introduction to adding JACK to a simple audio application, Lewis Berman has contributed a PDF on writing a JACK audio recorder at *http://userpages.umbc.*

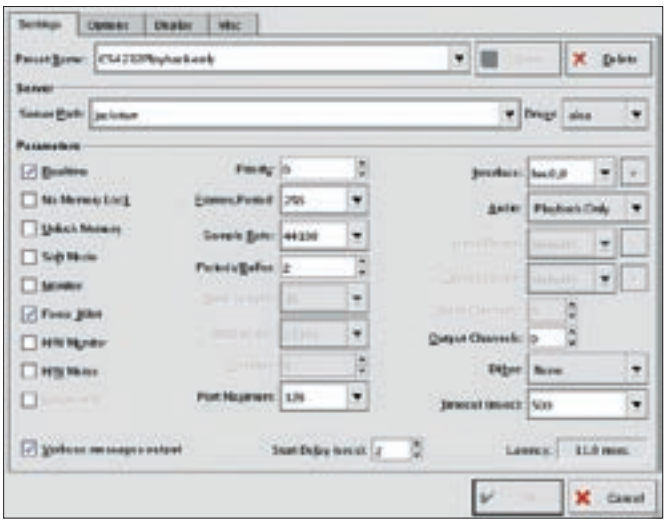

Figure 2: The QJackCtl setup dialog box.

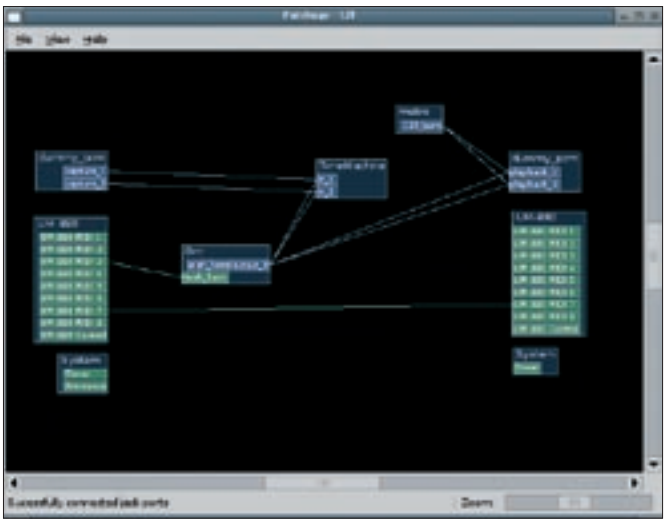

Figure 4: Patchage is a patchbay for JACK and ALSA.

edu/~berman3/, and of course, the JACK API can be read and studied in the well-commented *jack.h* header file.

If you build JACK yourself and you have the doxygen software installed, you can generate JACK's developer documentation. This documentation is also available on the JACK website, but it is out of date as of September 15, 2005.

JACK's Future

In 2004, JACK won a well-deserved Bronze award in the Merit Awards granted by the Open Source Initiative. At that point, JACK's development was at version 0.9x. As I write this article, JACK is now at version 0.100.0, heading steadily towards its 1.0 release, and the future is looking good for JACK.

Stephane Letz has successfully ported JACK to OSX. Support for OSX has become common in new Linux audio software. Incidentally, an implementation for Java has already appeared.

MIDI musicians are familiar with time code implementations not currently supported by JACK, and a coordination of synchronization capabilities would be most welcome. Some work in that direction has already begun, so it is likely that a blend of MIDI and JACK will evolve.

JACK's attractions may seem irresistable, but it may not be the best solution for common desktop audio services. Unlike ALSA, JACK is not planned for inclusion with the Linux kernel sources, so its presence in any Linux distribution results from a decision made by the distro's producer. Also, JACK is not so transparent to the user as the artsd and esd servers, and more configuration is required for obtaining best performance. Nevertheless, JACK is a very flexible system and may yet become the de facto audio server for the Linux desktop.

For the more professionally inclined JACK is a godsend. Its performance stability has already been tested and verified in high-demand real audio world applications, and its applicability can be seen in an expanding suite of increasingly powerful JACK-based programs. JACK's API has paved the way for a new wave of high-quality Linux audio applications. Whether you need rock-solid audio system performance for Ardour or you just want to have some fun routing the output from XMMS, you need to know JACK. ■

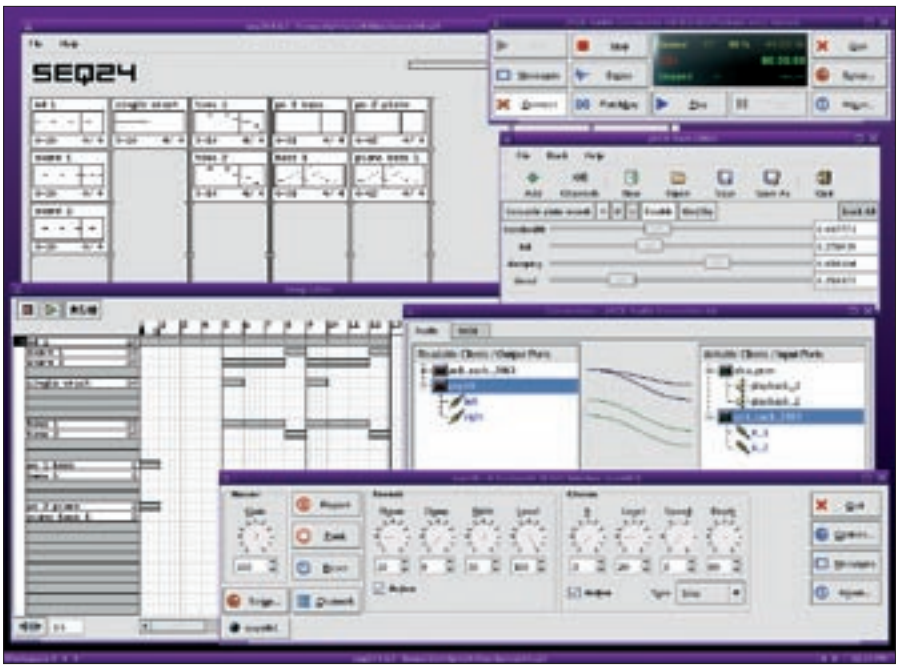

Figure 6: JACK with an audio-plus-MIDI network.

Figure 5: A more ambitious JACK configuration.

INFO

[1] The JACK Home Page: *http://jackit.sourceforge.net*

[2] ALSA Soundcard Matrix: *http://www.alsa-project.org/alsa-doc/*

[3] Patchage: *http://www.scs.carleton.ca/~drobilla/patchage/*

[4] QJackConnect: *http://www.suse.de/~mana/jack.html*

[5] QJackCtl: *http://qjackctl.sourceforge.net/*

[6] Interview with Paul Davis at Builder.com: *http://builder.com.com/5100-6375-5136755.html?tag=tt*

[7] The Low-latency Mini-HOWTO: *http://www.djcj.org/LAU/guide/Low_latency-Mini-HOWTO.php3*

[8] Florian Schmidt's notes on building a low-latency 2.6 kernel: *http://tapas.affenbande.org/?page_id=3*

[9] The JACK page at linux-sound.org: *http://linux-sound.org/jack.html*