

References

- [Angebrannt91] Susan Angebrannt et al., “Integrating Audio and Telephony in a Distributed Workstation Environment,” *USENIX Proceedings*, Summer 1991.
- [Billman92] Richard Billman et al., “Workstation Audio in the X Environment,” *The X Resource*, Issue 4, O’Reilly and Associates, 1992.
- [Glauert93] Tim Glauert et al., *X Synchronization Extension*, X Consortium draft standard, 1993.
- [Levergood93] Thomas M. Levergood et al., *AudioFile: A Network-Transparent System for Distributed Audio Applications*, Digital Equipment Corporation Cambridge Research Laboratory, 1993.
- [Scheifler92] Robert W. Scheifler and James Gettys, *X Window System*, Digital Press, 1992.
- [Shelley93] Robert N. C. Shelley et al., *X Image Extension Protocol Reference Manual, Version 4.12*, X Consortium Draft Standard, Special Issue C, O’Reilly & Associates, Inc., January 1993.

Example Programs

The sample implementation of NetworkAudio includes a variety of example programs, including a sound file “chooser,” an audio data editor, a translator for converting among the various file and data formats, a telephone dial-tone generator and recognizer, and a simple *audiotool*-like utility for use with OpenWindows Deskset applications. A growing number of games and commercial software are being shipped with NetworkAudio support built in.

Future Directions

Although NetworkAudio is rapidly gaining wide-spread use for simple-to-moderate audio applications, it still needs additional support for tight synchronization with graphics for high-end video conferencing. Some mixture of the model used by AudioFile and the primitives provided by the proposed X Synchronization extension [Glauert93] will probably be added to NetworkAudio shortly.

The Network Audio System was designed to be extensible at several levels. New data formats and device types can be added to the server without requiring changes to applications; similarly, new file formats can be added to the library transparently. As audio becomes more pervasive, both of these elements will be tested by the addition of support for compressed data streams and MPEG. Finally, virtual devices implemented in software, such as the radios mentioned before, should provide interesting avenues of exploration.

Summary

NetworkAudio provides a powerful, yet simple, mechanism for transferring audio data between computers, converting sample rates and data formats as needed. The standard media used in the server was reduced to multiple times more can be ascertained directly, or attached

The actual generation of audio data is done by an interrupt handler that is called from either a hardware interrupt or an interval timer, depending upon the implementation. Multiple values are

Sending Audio Data Over The Network

Although NetworkAudio provides a number of input devices and built-in sound generators, its primary purpose is to allow applications to play and record audio data over the network.

Clients send data to a special type of input element in a flow. Data from this element is used in the flow just as if it were coming from any other input device. If a sound is to be used more than once, the application can store it in the server by creating a ***bucket*** object and directing the data to it instead of to an output device.

This process can be reversed to retrieve audio data from the server. Large amounts of data can be passed back to the application directly from the various input devices without requiring temporary storage for all of it in the server. Although the most common use of this mechanism is to record

All inputs and outputs may be shared among the applications. When two or more flows direct sound data to the same output, the server automatically merges the data streams together so that both sounds come out at the same time. If the various components in an flow contain data of

Playing and Recording

The instructions that a client provides for arranging various inputs and outputs are called **Flows** (see Figure 4), similar to the photoflow pipelines of the X Image Extension (see [Shelley93]). They are used in much the same way that musicians use audio mixers or patch-panels. They indicate how the components should be connected, how multiple sounds should be mixed, and what changes in volume should be made.

A flow is made up of a series of *elements* which describe sources of input data, operations to perform on data, and places to output the data. A flow is often represented graphically as a s 0 T -1

Data Formats

The sequence of numbers that are used to encode audio data vary from -1.0 to +1.0, representing the two extreme positions that a diaphragm can reach in a given device. The value 0.0 corresponds to the center of the device. However, since floating point numbers occupy a large amount of space and are sometimes difficult to manipulate, most data formats store the sample values by multiplying them by 128 or 1768 and rounding to the closest integer.

The most common data formats are:

- **Linear** – Each 8- or 16-bit value contains a scaled sample; the larger version can more accurately represent the original data. Some variants store the signed values directly; others add 128 or 1768 to make them be unsigned. The 16-bit versions also come in most-significant byte first and least-significant byte first varieties.
- **μLAW** – Each 8-bit value contains the sign and the *logarithm* of a scaled sample. This enables a larger range of values to be represented in 8 bits by trading accuracy at the extreme positions (which aren't used very often and can tolerate small errors) for precision at the positions nearer to zero (which are used most of the time).

NetworkAudio supports both of these data formats and their variants. In addition, since different devices and applications use different data formats, NetworkAudio automatically converts among the various data formats whenever needed.

~~The sequence of data inputs and outputs~~ is that it allows applications to “wire up” one or more inputs to one or more outputs. Servers provide a variety of inputs and outputs, based on the available physical hardware or software emulators:

- **Physical Devices** – Most platforms support standard input and output jacks for attaching microphones, CD players, speakers, tape recorders, etc.
- **Virtual Devices** – ~~When~~ audio server itself provides software that emulates hardware tone generators and radio transmitters and receivers (for data that is broadcast over a local area network).
 - **Client Data** – Applications can send data to and from the server over the network. By splitting the transmissions into pieces, applications can send or receive an unlimited amount of data.
 - **Buckets** – Audio data can be stored in the server, either as a result of being recorded from another device or having been sent over the network from the application. This allows the data to be played multiple times without having to be retransmitted or rerecorded.

Outputs can receive data from more than one input at once; the server automatically merges the data streams so that the sounds are heard together. Operations for mixing sounds, extracting individual tracks of data, and changing the volume are provided.

What Is Audio Data?

Physically, sounds are waves of compressed and expanded air. We are able to hear them because the bones of the inner ear vibrate when struck by the waves; these vibrations are interpreted by the brain as sound. Similarly, we speak by reversing the process, causing air to pass over our vocal cords which vibrate back and forth. Audio devices such as microphones and speakers work by this pushersion

The Network Audio System

Make Your Applications Sing (As Well As Dance)!

Jim Fulton[†]

Greg Renda[‡]

Abstract

Audio input and output is rapidly becoming a standard feature in desktop devices and an expected