Multimedia '89, Apr. 20-23 1989, Ottawa, Ontario
2nd IEEE Comsoc International Multimedia Communications Workshop

The VOX Audio Server

Barry Arons, Carl Binding, Keith Lantz, Chris Schmandt*
Olivetti Research Center†
Menlo Park, California

1 Abstract

In the recent past, user interfaces have vastly improved due to the widespread availability of high-resolution graphics displays. It is now a new challenge to incorporate other media into the user interface. We strongly believe that voice and audio will be a key component of next-generation interfaces. Towards this end, we are developing an *audio server* that supports the integration of voice and audio features into the user interface. Application areas in which we are particularly interested include telephone management, voice annotation, real-time teleconferencing, conversational answering machines, and, more generally, computer-based tools to support collaborative work.

The intent of the *VOX Audio Server* is to accomplish for audio what current state-of-the-art window servers (such as in the X Window System) have done for graphics. VOX provides integrated access to and control of audio, voice, and telephony capabilities. It provides for the dynamic configuration of audio devices and their sharing between multiple client applications. We also believe that the server architecture can be extended to support other media, particularly video.

2 Goals

To satisfy the perceived needs of our target applications, the server architecture emphasizes:

- *Sharing*: By default, the architecture supports the sharing of audio hardware by multiple applications. Multiple clients can express interest in a resource; well behaved clients request access to the resource for limited periods. Applications are also able to gain *exclusive* access to critical audio resources for a limited amount of time. For example, a telephone may only be used by one application during an actual telephone conversation.
- *Routing*: The architecture enables applications to create *dynamic* routings between audio components. For example, in a conversational answering machine application it is desirable to rapidly and conveniently switch from a speech recognizing configuration to a sound recording configuration since both activities are useful to gracefully handle an incoming call.
- *Real-time behavior*: The architecture must address the issues of real-time behavior in the handling of audio events. To that effect, VOX supports a queuing mechanism that minimizes the overhead of processing audio requests and events at time-critical moments, thereby permitting the implementation of the server on a time-shared operating system.

*Chris Schmandt consulted to Olivetti in the design of the VOX Audio Server. He is affiliated with the MIT Media Laboratory.

†The authors may be reached at: Olivetti Research Center, 2882 Sand Hill Road, Suite 210, Menlo Park CA, 94025. They can be contacted by electronic mail at: arons@orc.olivetti.com or at [sri-unix|oliveb]!orc!arons.

- *Device independence*: The architecture attempts to shield the clients from the idiosyncrasies of particular audio hardware.
- *Extensibility*: The architecture allows for unforeseen uses of audio, new types of audio or telephony devices, and the integration of video.

3 Basic Architecture

The basic architecture needed to achieve our goals is a network-transparent server-based configuration similar to many contemporary window systems (figure 1). Multiple client processes are connected to a server process that provides various audio operations such as recording and playback of sounds, speech recognition, and text-to-speech synthesis. Requests to gain access to audio resources are mediated by a *workstation manager* (WSM), analogous to the use of window managers in many window systems.

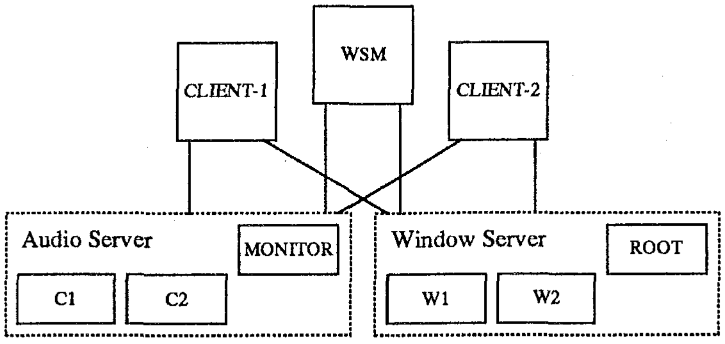

Figure 1: Audio and Window Server Architecture

4 Logical Audio Devices

To support dynamic routing between audio devices, we use the paradigm of assembling audio circuits in a manner similar to actual electronic hardware. That is, lower-level components are grouped into higher-level components of increased functionality. We call the lowest level building block a *LAUD* for Logical **AU**dio **D**evice (pronounced *loud*). LAUDs can be combined into a *composite* LAUD, called a *CLAUD*. Each LAUD has a set of *audio ports* that are connected ("soldered") together when composing a CLAUD, and each LAUD is associated with a *device* that implements the actual audio activity. Examples of LAUDs include abstractions for playback, recording, and mixing.

In essence, clients assemble LAUDs into CLAUDs to serve a specific purpose and CLAUDs are considered as a single logical entity. The primitive LAUD abstractions are provided by the VOX server and accessed and manipulated from clients distributed throughout a network. Figure 2 is an example of a simple telephone answering machine built according to our model. Three primitive LAUDs for playback, recording, and telephone control are grouped into one logical unit: the telephone answering CLAUD.

Each CLAUD specifies which audio resources shall be used at run-time for the execution of audio requests. In order to limit access to critical resources—namely hardware audio devices—the client must explicitly map a CLAUD onto the devices it needs to perform the desired audio activity. Resource sharing is thus achieved through the concept of mapping and unmapping CLAUDs to their devices. It is only while a CLAUD is mapped to its devices that it can actually execute audio activity. Ultimately, only a specialized client, namely the *workstation manager*, will be able to perform map and unmap calls, while regular clients may only express their desire to be mapped.

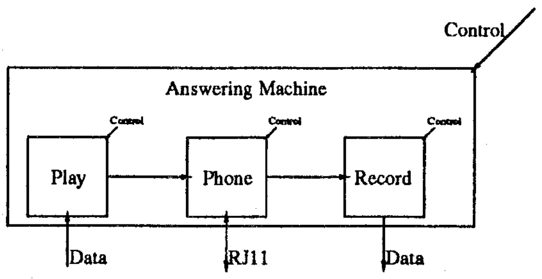

Figure 2: Answering Machine CLAUD

5 Input and Output

Input and output to CLAUDs is based on an event-driven scheme. The client submits *output requests* to a CLAUD and may receive *input events* from a CLAUD.[1] For example, the client may submit play and record requests, and receive tokens from a speech recognizer.

Most output requests can be *prepared* in advance. To that effect, a CLAUD embodies a *queuing* abstraction, which serves as a buffer for client requests. When enqueuing output requests, the server attempts to prepare the request as much as possible. This may involve the opening of a sound file, prefetching an already recorded sound, or establishing the state of a speech recognizer. All these activities can be executed *before* the actual servicing of the request takes place, thereby reducing the execution latency at time-critical moments.

6 Experimental Testbed

We are currently implementing a prototype of the VOX Audio Server on an Intel 80386-based Olivetti PC running Unix System V.3, but will be migrating the server to the Mach operating system[2] once it becomes available on our host platform. An experimental audio processing card is currently supported for play, record, and sound editing functions. The hardware supported by the server is being expanded and a wide range of voice and audio devices, as described in this section, should be supported by the end of 1989.

Some devices controlled by VOX will have internal switching capabilities and thus may directly support a small number of VOX "solder" calls. However, to handle the general case of device connection we will use an external 16 $\times$ 16 audio crossbar switch to implement the interconnection calls of the server. All inputs and outputs of the audio peripherals are routed through the crossbar switch, which provides an extremely flexible environment for the rapid prototyping of audio applications. Figure 3 represents a typical workstation hardware configuration with the crossbar switch acting as a general purpose "soldering" and switching mechanism.

Random access audio and a telephone interface are key components of many desired applications, so we will soon replace our experimental audio card with a more powerful commercially available one. Most such existing products for the AT-bus were originally designed to be used in an MS-DOS environment, not on a Unix system, but this trend seems to be changing. We would like the audio board to support two independent voice and telephone channels to allow for maximum flexibility in designing multimedia applications.

Each workstation will be equipped with a computer-controlled audio mixer.[3] The mixer will be used for simple local adjusting of levels and equalization as well as for the automated control and mixing of signals

[1] *Output* is data sent from the client to the server; *input* is sent from server to client, similar to I/O handling in window systems.

[2] Mach was developed at Carnegie-Mellon University and is based on Berkeley 4.3 Unix. The Mach operating system is well suited for distributed systems and applications.

[3] MIDI (Musical Instrument Digital Interface) controlled equipment was found to be suitable for our experimentation.

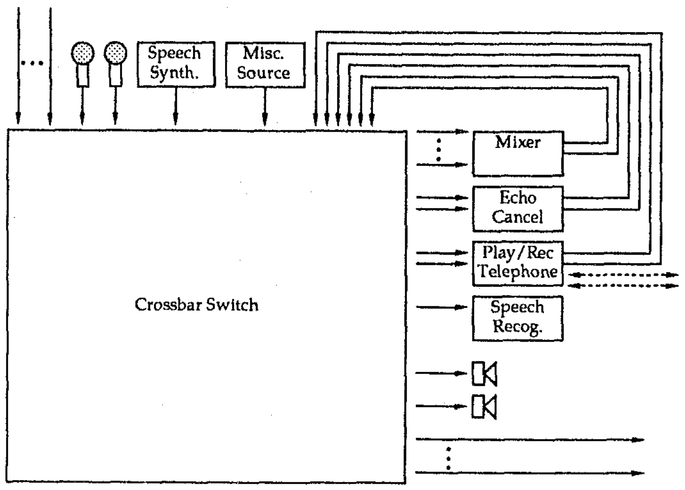

Figure 3: Typical workstation audio configuration.

based on requests by the workstation manager that oversees the sharing of audio resources. Other technologies supported by the server architecture include speech recognition and text-to-speech synthesis. A full duplex audio processor card (echo canceler) will be used to provide a high quality hands-free speakerphone using the microphone and speakers available on each desktop.

In addition to the local hardware on each workstation, there are several tie lines to a larger centralized crossbar switch. These facilities will provide a high-fidelity audio network for local audio conferencing between workstations. The application software will attempt to use the local audio system before using the lower bandwidth lines of the telephone network.

7 Future Hardware Platforms

Entirely digital audio devices with compatible digital input and output are becoming more prevalent, but are not yet readily available on a large scale. Note, however, that the majority of the components of our testbed workstation are digital devices *internally*, but appear as analog devices externally, and are interconnected via conventional audio cables. Similarly, almost all non-proprietary telephone interfacing is analog, but this will hopefully change as ISDN provides digital audio paths and signaling from the network. With these ideas in mind, we chose to use off-the-shelf computer-controllable analog audio components because of their relatively low cost and the wide variety of available hardware.

Virtually all of the functionality provided by the aggregation of hardware described in section 6 can be implemented digitally, possibly with a single next-generation digital signal processing (DSP) chip. There are currently no general-purpose DSP boards available for our hardware platform with appropriate software to support all of our applications. Most current DSP boards can provide a handful of the audio functions shown in figure 3, but cannot support all of these functions simultaneously. We envision that within a few years the bulk of the audio support for VOX server can be implemented on a single audio board. Therefore, we are currently focused on developing the audio server and prototype applications rather than developing audio hardware or DSP algorithms. In a completely digital audio environment there is still a need for a coherent software architecture for structure and control of audio resources as addressed by VOX.

8 Multimedia Applications

Other projects at the Olivetti Research Center are involved with software tools for computer-supported cooperative work (CSCW), in particular real-time computer-based teleconferencing [3]. One of our initial application areas will be to provide audio support to augment this shared window system. Geographically distributed participants will be automatically connected by the telephone or local audio network with real-time detection of voice used as a method of "floor control" in multi-person conferences.

A window system-based audio editor has been written on top of VOX to allow the easy creation and manipulation of sound files. A graphical tool such as this can be integrated into other applications to provide voice-annotation for a multimedia editor or for reviewing and editing messages from a conversational answering machine.

The overall architecture of the audio server is quite general and we believe that it can be extended to cover other classes of audio devices as well as other media. The basic server, for example, can be extended to control, share, and interconnect video equipment. It is still a somewhat open issue as to whether there should be separate media servers for audio and video or if there should be a single server controlling all aspects of the multimedia services.

9 Related Work

VOX attempts to integrate the functionality provided by systems built at IBM Research [5] and at MIT's Media Lab [7, 8, 9]. These include support for telephony as well as for text-to-speech synthesis and speech recognition. The work done at the Media Lab by two of the authors pioneered the notion of request preparation in order to have low latency execution of audio requests which VOX incorporates in its design. As extensions to these systems, however, VOX takes a more dynamic approach to audio routing, provides increased flexibility and operates in an environment supporting several, possibly distributed, clients at once.

In contrast with the Etherphone system developed at Xerox PARC [10, 11, 12], VOX does not rely on a centralized voice storage server. Instead, sounds are stored on the workstation on which the server is running and VOX provides features to migrate sounds over the network.

The influence of window systems and user interface software on VOX appears in several aspects. First, like the VGTS [4], NeWS [2], and X [6] window systems, VOX is network transparent. That is, a client can reside on any machine in a local area network and access the audio server. Secondly, VOX inherits the concepts of logical hierarchies from graphics systems such as VGTS and Eureka [1].

10 Summary

The "Desktop Audio" market is still in its infancy—where the computer graphics market was about 8 or 10 years ago. The VOX Audio Server tries to take the ideas developed by the graphics and window system community and successfully merge them with practical experience gained from building interactive audio and voice systems. We hope that the audio server architecture will skip a generation in the evolutionary development of audio services, providing the underlying software architecture for future audio applications.

The only way that voice will successfully be integrated into the workstation environment is if there is a standard architecture that supports voice and audio applications across all hardware and software platforms. Therefore, in the spirit of MIT's X Window System, the Olivetti Research Center is placing the VOX Audio Server specification and a prototype implementation in the public domain. We encourage other research and development efforts to use VOX and help make it a *de facto* audio server standard.

In summary, we believe that our architecture successfully deals with the routing and management of audio resources in a networked computing environment. We have integrated various voice and audio technologies into a coherent and novel architecture enabling applications to easily incorporate speech and sound into the man-machine interface. Extensions to video are supported by our basic architecture and allow for an even greater support of multi-media interactions and applications.

11 Acknowledgements

Pehong Chen has recently joined the VOX team and currently is writing the interface to the crossbar switch.

References

[1] Carl Binding. The Architecture of a User Interface Toolkit. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software*, pages 56–65, October 1988.

[2] James Gosling. SunDew - a Distributed and Extensible Window System. In Hopgood, F.R.A., et al. editor, *Methodology of Window Management*, pages 47–58. Springer Verlag, 1986.

[3] Keith A. Lantz. An Experiment in Integrated Multimedia Conferencing. In *CSCW86*, pages 267–275. MCC Software Technology Program. December 1986. Reprinted in I. Greif, editor, *Computer-Supported Cooperative Work: A Book of Readings*, pages 533-552. Morgan Kaufmann Publishers, 1988.

[4] Keith A. Lantz and William I. Nowicki. Structured Graphics for Distributed Systems. *ACM Transactions on Graphics*, 3(1):23–51, January 1984.

[5] Antonio Ruiz. Voice and Telephony Applications for the Office Workstation. In *Proceedings 1st International Conference on Computer Workstations*, pages 158–163. IEEE Computer Society, November 1985.

[6] Robert W. Scheifler and Jim Gettys. The X Window System. *ACM Transactions on Graphics*, 5(2):79–106, April 1986.

[7] Chris Schmandt and Barry Arons. A Conversational Telephone Messaging System. *IEEE Trans. on Consumer Electr.*, CE-30(3):xxi–xxiv, 1984.

[8] Chris Schmandt and Michael A. McKenna. An Audio and Telephone Server for Multi-Media Workstations. In *Proceedings 2nd IEEE Conference on Computer Workstations*, pages 150–160. IEEE Computer Society, March 1988.

[9] Chris Schmandt, Barry Arons, and Charles Simmons. Voice Interaction in an Integrated Office and Telecommunications Environment. In *Proceedings*. American Voice Input Output Society, 1985.

[10] Daniel C. Swinehart. Telephone Management in the Etherphone System. In *Proceedings of GlobeCom 87*. IEEE GlobeCom, November 1987.

[11] Daniel C. Swinehart, Larry C. Stewart, and Susan M. Ornstein. Adding Voice to an Office Computer Network. Technical Report CSL-83-8, Xerox Palo Alto Research Center, February 1984.

[12] Douglas B. Terry and Daniel C. Swinehart. Managing Stored Voice in the Etherphone System. In *11th ACM Symposium on Operating System Principles*, pages 48–61. ACM SIGOPS, November 1987.