Open SoundControl: A New Protocol for Communicating with Sound Synthesizers

Matthew Wright
Adrian Freed
Center for New Music and Audio Technologies, U.C. Berkeley
matt,adrian@cnmat.berkeley.edu, http://www.cnmat.berkeley.edu/People

Abstract

Open SoundControl is a new protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology. Entities within a system are addressed individually by an open-ended URL-style symbolic naming scheme that includes a powerful pattern matching language to specify multiple recipients of a single message. We provide high resolution time tags and a mechanism for specifying groups of messages whose effects are to occur simultaneously. There is also a mechanism for dynamically querying an Open SoundControl system to find out its capabilities and documentation of its features.

1 Introduction

A better integration of computers, controllers and sound synthesizers will lead to lower costs, increased reliability, greater user convenience, and more reactive musical control. The prevailing technologies to interconnect these elements are bus (motherboard or PCI), operating system interface (software synthesis), or serial LAN (Firewire, USB, ethernet, fast ethernet). It is easy to adapt MIDI streams to this new communication substrate [11]. To do so needlessly discards new potential and perpetuates MIDI's well-documented flaws. Instead we have designed a new protocol optimized for modern transport technologies.

Open SoundControl is an open, efficient, transport-independent, message-based protocol developed for communication among computers, sound synthesizers, and other multimedia devices. Open SoundControl is a machine and operating system neutral protocol and readily implementable on constrained, embedded systems.

We begin by examining various networking technologies suitable for carrying Open SoundControl data and discuss shared features of these technologies that impact the design of our protocol. Next we discuss the encoding and formatting of data. We describe our novel addressing scheme based on a URL-style symbolic syntax. In our last section we specify a number of query messages that request information from an Open SoundControl system.

2 Transport Layer Assumptions

Open SoundControl is a transport-independent protocol, meaning that it is a format for data that can be carried across a variety of networking technologies. Networking technologies currently becoming widely available and economical include high speed busses such as PCI [8] and medium speed serial LANs such as USB [10], IEEE-1394 (“Firewire”) [3], Ethernet, and Fast Ethernet [4]. Although Open SoundControl was not designed with a particular transport layer in mind, our design reflects features shared by modern networking technologies.

We assume that Open SoundControl will be transmitted on a system with a bandwidth in the 10+ megabit/sec range. MIDI’s bandwidth, by contrast, is only 31.25 kilobit/sec, roughly 300 times slower. Therefore, our design is not preoccupied with squeezing musical information into the minimum number of bytes. We encode numeric data in 32-bit or 64-bit quantities, provide symbolic addressing, time-tag messages, and in general are much more liberal about using bandwidth for important features.

We assume that data will be delivered in packets (a.k.a. "datagrams") rather than as a stream of data traveling along an established connection. (Although many LANs *transmit* data serially, they *deliver* the data as a single block.) This leads to a protocol that is as stateless as possible: rather than assuming that the receiver holds some state from previous communications (e.g., MIDI's running status), we send information in larger, self-contained chunks that include all the relevant data in one place. This packet-based delivery model provides a mechanism for *synchronicity*: messages in the same packet (e.g., messages to start each of the notes in a chord) can be specified to have their effects occur at the same time as each other. We assume that the network services will tell us the length of each packet that we receive.

All modern networking technologies have the notion of multiple devices connected together in a LAN, with each device able to send a packet to any other device. So we assume that any number of clients might send Open SoundControl messages to a particular device. We also assume that the transport layer provides a return address mechanism that allows a device to send a response back to the device that sends it a message.

3 Data Representation

All Open SoundControl data is aligned on 4-byte boundaries. Numeric data are encoded using native machine representations of 32-bit or 64-bit big-endian twos-complement integers and IEEE floating point numbers. Strings are represented as a sequence of non-null ASCII characters followed by a null, padded with enough extra null characters to make the total length be a multiple of 4 bytes. These representations facilitate real-time performance by eliminating the need to reformat received data. (Except for the unavoidable big-endian/small-endian conversion. We note that most small-endian machines provide special instructions for this conversion.)

The basic unit of Open SoundControl data is a *message*, which consists of the following:

• A symbolic address and message name (a string whose meaning is described below)

• Any amount of binary data up to the end of the message, which represent the arguments to the message.

An Open SoundControl packet can contain either a single message or a *bundle*. A bundle consists of the following:

• The special string "#bundle" (which is illegal as a message address)

• A 64 bit fixed point time tag

• Any number of messages or bundles, each preceded by a 4-byte integer byte count

Note that bundles are recursively defined; bundles can contain other bundles.

Messages in the same bundle are atomic; their effects should be implemented concurrently by the receiver. This provides the important service in multimedia applications of specifying values that have to be set simultaneously.

Time tags allow receiving synthesizers to eliminate jitter introduced during packet transport by resynchronizing with bounded latency. They specify the desired time at which the messages in the bundle should take effect, a powerful mechanism for precise specification of rhythm and scheduling of events in the future. To implement time tags properly, message recipients need a real-time scheduler like those described by Dannenberg [1].

Time tags are represented by a 64 bit fixed point number. The first 32 bits specify the number of seconds since midnight on January 1, 1900, and the last 32 bits specify fractional parts of a second to a precision of about 200 picoseconds. This is the representation used by Internet NTP timestamps [6]. The Open SoundControl protocol does not provide a mechanism for clock synchronization; we assume either that the underlying network will provide a synchronization service or that the two systems will take advantage of a protocol such as NTP or SNTP [7].

4 Addressing Scheme

The model for Open SoundControl message addressing is a hierarchical set of dynamic objects that include, for example, synthesis voices, output channels, filters, and a memory manager. Messages are addressed to a feature of a particular object or set of objects through a hierarchical namespace similar to URL notation, e.g., /voices/drone-b/resonators/3/set-Q. This protocol does not proscribe anything about the objects that should be in this hierarchy or how they should be organized; each system that can be controlled by Open SoundControl will define its own address hierarchy. This open-ended mechanism avoids the addressing limitations inherent in protocols such as MIDI and ZIPI [5] that rely on short fixed length bit fields.

To allow efficient addressing of multiple destination objects for parameter updates, we define a pattern-matching syntax similar to regular expressions. When a message's address is a pattern, the receiving device expands the pattern into a list of all the addresses in the current hierarchy that match the pattern, similar to the way UNIX shell "globbing" interprets special characters in filenames. Each address in the list then receives a message with the given arguments.

We reserve the following special characters for pattern matching and forbid their use in object addresses: ?, *, [,], {, and }. The # character is also forbidden in object address names to allow the bundle mechanism. Here are the rules for pattern matching:

• ? matches any single character except /

• * matches zero or more characters except /

• A string of characters in square brackets (e.g., [string]) matches any character in the string. Inside square brackets, the minus sign (-) and exclamation point (!) have special meanings: two characters separated by a minus sign indicate the range of characters between the given two in ASCII collating sequence. (A minus sign at the end of the string has no special meaning.) An exclamation point at the beginning of a bracketed string negates the sense of the list, meaning that the list matches any character *not* in the list. (An exclamation point anywhere besides the first character after the open bracket has no special meaning.)

• A comma-separated list of strings enclosed in curly braces (e.g., {foo,bar}) matches any of the strings in the list.

Our experience is that with modern transport technologies and careful programming, this addressing scheme incurs no significant performance penalty either in network bandwidth utilization or in message processing.

5 Requests for Information

Some Open SoundControl messages are requests for information; the receiving device (which we'll call the *Responder* in this section) constructs a response to the request and sends it back to the requesting device (which we'll call the *Questioner*). We assume that the underlying transport layer will provide a mechanism for sending a reply to the sender of a message.

Return messages are formatted according to this same Open SoundControl protocol. There is always enough information in a return message to unambiguously identify what question is being answered; this allows Questioners to send multiple queries without waiting for a response to each before sending the next. The time tag in a return message indicates the time that the Responder processed the message, i.e., the time at which the information in the response was true.

Exploring the Address Space

Any message address that ends with a trailing slash is a query asking for the list of addresses underneath the given node. These messages do not take arguments. This kind of query allows the user of an Open SoundControl system to map out the entire address space of possible messages as the system is running.

Message Type Signatures

Because Open SoundControl messages do not have any explicit type tags in their arguments, it is necessary for the sender of a message to know how to lay out the argument values so that they will be interpreted correctly.

An address that ends with the reserved name "/type-signature" is a query for the "type signature" of a message, i.e., the list of types of the arguments it expects. We have a simple syntax for representing type signatures as strings; see our WWW site for details.

Requests For Documentation

An address that ends with the reserved name "/documentation" is a request for human-readable documentation about the object or feature specified by the prefix of the address. The return message will have the same address as the request, and a single string argument. The argument will either be the URL of a WWW page documenting the given object or feature, or a human-readable string giving the documentation directly. This allows for "hot-plugging" extensible synthesis resources and internetworked multimedia applications.

Parameter Value Queries

An address that ends with the reserved name "/current-value" is a query of the current value of the parameter specified by the prefix of the address. Presumably this parameter could be set to its current value by a message with the appropriate arguments; these arguments are returned as the arguments to the message from the Responder. The address of the return message should be the same as the address of the request.

6 Conclusion

We have had successful early results transmitting this protocol over UDP and Ethernet to control real-time sound synthesis on SGI workstations from MAX [9] programs running on Macintosh computers. Composers appreciate being freed from the limited addressing model and limited numeric precision of MIDI, and find it much easier to work with symbolic names of objects than to remember an arbitrary mapping involving channel numbers, program change numbers, and controller numbers. This protocol affords satisfying reactive real-time performance.

7 References

[1] Dannenberg, R, 1989. "Real-Time Scheduling and Computer Accompaniment," in M. Mathews and J. Pierce, Editors, *Current Directions in Computer Music Research*, Cambridge, Massachusetts: MIT Press, pp. 225-261.

[2] Freed, A. 1996. "Firewires, LANs and Buses," from SIGGRAPH Course Notes *Creating and Manipulating Sound to Enhance Computer Graphics*, New Orleans, Louisiana, pp. 84-87.

[3] IEEE 1995. "1394-1995: IEEE Standard for a High Performance Serial Bus," New York: The Institute of Electronical and Electronic Engineers.

[4] IEEE 1995b. "802.3u-1995 IEEE Standards for Local and Metropolitan Area Networks: Supplement to Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications: Media Access Control (MAC) Parameters, Physical Layer, Medium Attachment Units, and Repeater for 100 Mb/s Operation," New York: The Institute of Electronical and Electronic Engineers.

[5] McMillen, K, D. Wessel, and M. Wright, 1994. "The ZIPI Music Parameter Description Language," Computer Music Journal, Volume 18, Number 4, pp. 52-73.

[6] Mills, D., 1992. "Network Time Protocol (Version 3) Specification, Implementation, and Analysis," Internet RFC 1305. (http://sunsite.auc.dk/RFC/rfc1305.html)

[7] Mills, D., 1996. "Simple Network Time Protocol (SNTP) Version 4 for Ipv4, Ipv6 and OSI," Internet RFC 2030. (http://sunsite.auc.dk/RFC/rfc2030.html)

[8] PCI 1993. *PCI Local Bus Specification, Revision 2*.0, Hillsboro, Oregon: PCI Special Interest Group.

[9] Puckette, M., 1991. "Combining Event and Signal Processing in the MAX Graphical Programming Environment," *Computer Music Journal*, Volume 15, Number 3, pp. 58-67.

[10] The USB web site is www.usb.org

[11] Yamaha 1996. "Preliminary Proposal for Audio and Music Protocol, Draft Version 0.32," Tokyo, Japan: Yamaha Corporation.