Best Practices for Open Sound Control

Andrew Schmeder and **Adrian Freed** and **David Wessel**
Center for New Music and Audio Technologies (CNMAT), UC Berkeley
1750 Arch Street
Berkeley CA, 94720
USA,
{andy,adrian,wessel}@cnmat.berkeley.edu.edu

Abstract

The structure of the Open Sound Control (OSC) content format is introduced with historical context. The needs for temporal synchronization and dynamic range of audio control data are described in terms of accuracy, precision, bit-depth, bit-rate, and sampling frequency. Specific details are given for the case of instrumental gesture control, spatial audio control and synthesis algorithm control. The consideration of various transport mechanisms used with OSC is discussed for datagram, serial and isochronous modes. A summary of design approaches for describing audio control data is shown, and the case is argued that multi-layered information-rich representations that support multiple strategies for describing semantic structure are necessary.

Keywords

audio control data, signal quality assurance, best practices, open sound control

1 Introduction

1.1 Definition

Open Sound Control (OSC) is a digital media content format for streams of real-time audio control messages. By audio control we mean any time-based information related to an audio stream other than the audio component itself. This definition also separates control data from stream meta-data that is essentially not time-dependent (e.g. [Wright et al., 2000]). Naturally such a format has application outside of audio technology, and OSC has found use in domains such as show control and robotics.

1.2 What is Open

It should be noted that OSC is not a standard as it does not provide any test for certification of conformance. The openness of Open Sound Control is that it has no license requirements, does not require patented algorithms or protected intellectual property, and makes no strong assertions about how the format is to be used in applications.

1.3 Format Structure

OSC streams are sequences of frames defined with respect to a point in time called a timetag. The frames are called bundles. Inside a bundle are some number of messages, each of which represent the state of a sub-stream at the enclosing reference timetag. The sub-streams are labelled with a human-readable character string called an address. In a message, the address is associated with a vector of primitive data types that include common 32-bit binary encodings for integers, real numbers and text (Figure 1).

Unlike audio data, which is sampled regularly on a fixed temporal grid, bundles may contain mixtures of sub-streams that are sampled at different or variable rates. Therefore, the number of messages appearing in a bundle may vary depending on what data is being sampled in that moment.

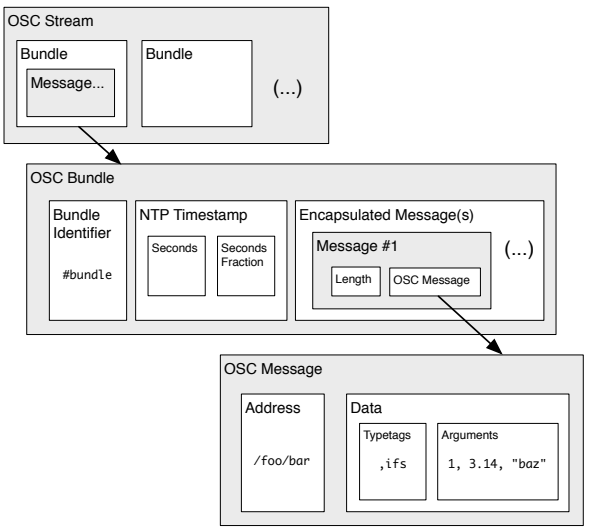

Figure 1: Structure of the OSC content format

1.4 History

OSC was invented in 1997 by Adrian Freed and Matt Wright at The Center for New Music and

Audio Technologies (CNMAT), where its first use was to control sound synthesis algorithms in the CAST system using network messaging (The CNMAT Additive Synthesizer Toolkit) [Adrian Freed, 1997]. CAST was implemented for the SGI Irix platform, which was one of the first general purpose operating systems to provide reliable realtime performance to user-space applications [Freed, 1996] [Cortesi and Thomas, 2001]. This capability was a key influence in the design of the OSC format in particular with respect to the inclusion of timestamped bundles that enable high quality time synchronization of discrete events distributed over a network. Following the success of the CAST messaging system, the protocol was refined and published online as the OSC 1.0 Specification in 2002 [Wright, 2002].

1.5 Best Practices

Considering OSC as a content format alone does not account for how it can and should be used in applications. A larger picture exists around the needs and requirements of sound control in general with respect to the signal quality and description of control data. In the past these needs have been underestimated by hardware and software designs, leading to less than ideal results. Research into the needs of audio control data are summarized in this paper along with recommendations for how to best apply OSC features so that the requirements are satisfied.

1.6 Systems Integration

In the context of the Open System Interconnection Basic Reference Model (OSI Model), OSC is classified as a Layer 6 or Presentation Layer entity.

However in the larger scope of how OSC is used, other layers are considered as part of the practice. Related topics and their associated layer are listed in Table 1.

OSI Layer	Topic in OSC Practice
7 Application	Semantics, Choreography
6 Presentation	OSC Format
5 Session	Enumeration, Discovery
4 Transport	Latency, Reliability
3 Network	Stream Routing
2 Frame	Hardware Clocks, Timing
1 Bit	Cabling, Wireless, Power

Table 1: OSC related topics in the context of associated OSI Model layers

2 Temporal Audio Control

2.1 Instrumental Gestures

Musical instrumental gestures are actuations of a musical instrument by direct human control (typically by kinetic neuro-muscular actuation). The transduction of a physical gesture into a digital representation requires measurement of the temporal trajectory of all relevant dimensions of the physical system such as spatial position, applied force, damping and friction.

An assessment of the performable dynamic range with respect to each physical dimension is beyond the scope of this document, however, in a somewhat general way it is possible to estimate the quantity of temporal information contained in an isolated sub-stream of a musical performance.

2.1.1 Temporal Information Rate

It is estimated that the smallest controllable temporal precision by human kinetic actuation is 1 millisecond (msec), based on an example of an instrumental gesture called the flam that is known to have a very fine temporal structure [Wessel and Wright, 2002]. This limit of 1msec coincides with the threshold for just noticable difference in onset time between two auditory events.

The flam technique in drumming is a method of striking the surface of a drum with two sticks so that the relative arrival time of each stick modulates the timbre (spectral quality) of the resulting sound. Because of the very close temporal proximity of the events, a human listener perceives them to be a single event where the spectral centroid of the timbre is correlated to the temporal fine structure. This inter-onset time can be reliably controlled by a trained performer between 1-10 msec.

The flam is an example of an instrumental gesture with temporal *precision* of 1 msec. However the temporal *accuracy* of instrumental gestures is at least an order of magnitude larger. The tolerable latency for performance of music requiring very tight rhythmic synchronization between two players has been measured to be 10 msec [Chafe et al., 2004]. Ignoring the complications of fixed versus variable delays, it seems reasonable to assume 10 msec as an estimate of temporal accuracy in an instrumental gesture event. And finally, it is also reasonable to suppose that trained musicians can perform event rates up to 10 events per second in a single sub-stream (polyphonic streams are not

considered here).

The temporal information present in musical events can be estimated from these numbers. The information in bits, also called the index of difficulty in Fitt's Law, is calculated from the ratio between effective target distance ρ and standard error of the effective target width σ:

$$I = \log_2 \left(1 + \frac{\rho}{\sigma}\right) \text{bits} \quad (1)$$

Suppose a musician performs the double-strike flam at a rate of 10 hz. This is equivalent to two tasks: 1) placement of events with an average separation of 100 msec and standard error of 10 msec, 2) placement of two sub-events with a separation of 10 msec and error of 1 msec. Assuming the dual-task is repeated at 10 hz then the total information rate is,

$$I' = \log_2 \left(1 + \frac{100}{10}\right) + \log_2 \left(1 + \frac{10}{1}\right) \text{bits}, \quad (2)$$

$$I' \times \frac{10}{\text{sec}} = 68 \frac{\text{bits}}{\text{sec}}. \quad (3)$$

This estimate informs us that if the gesture as described was transformed to a digital representation without loss of information, the temporal dimension alone would require 68 bits/sec to encode.

For sake of comparison the highest reported information transfer rates for target-selection with a mouse in the ISO-1941-9 test are around 3 bits/sec [MacKenzie et al., 2001].

A distinguishing feature of the musical context of gesture is that the human performer uses a combination of extensive training, anticipations of musical structure, and sensory feedback to continuously adjust and refine the gesture. Therefore, musical gestures cannot be directly compared to reaction-time studies or task-based assessments as they are used in the study of human-machine ergonomics. However it is clear from this example that musical gestures contain a far greater density of temporal information than is typical for human-computer interactions in other contexts.

2.2 Spatial Audio Control

Spatial audio effects such as early reflections and reverberation are broad-band temporal-spectral transformations, however the maximum useful rate at which a spatial audio effect can be controlled is limited to the sub-audible frequency band between 0-50 hz. This is due to the simple fact that if a spatial parameter is modulated with a high frequency, perceptual fusion takes place yielding a transformation of the source in some way other than the intended outcome. For example a virtual source with rotating dipole directivity pattern ceases to be perceived as rotating for frequencies above 10 hz [Schmeder, 2009a]. Similarly if the location of a virtual source alternates between two positions at a high rate, the observer perceives a stationary source with a wider apparent source width.

However, spatial audio control data requires very high temporal precision to avoid phase artifacts in multi-loudspeaker arrays. AES11-2003 recommends a between-channel synchronization error of +/- 5% per sample frame [Audio Engineering Society, 2003] [Bouillot and et al, 2009]. An audio signal stream at 96khz requires that the temporal synchronization error does not exceed 0.5 microseconds.

The effect of synchronization error in the control stream for a spatial audio rendering engine may have an impact on the final reproduction quality. For example in a phase-mode beamforming array ([Rafaely, 2005b]), synchronization inaccuracy is similar to a positioning error and synchronization jitter is functionally similar to transducer noise [Rafaely, 2005a].

2.3 Sound Synthesis

In a pure signal processing context audio control data can be considered as the component of the signal that is non-stationary. The representation of control information is a topic specific to the design of any given algorithm, and so it is impossible to state a universal set of requirements for audio control. It is worth noting that some audio synthesis algorithms can have significant bandwidth requirements, especially the data driven methods such as sinusoidal additive synthesis and concatenative synthesis.

3 Temporal Quality Assurance

3.1 Event Synchronization

Assuming clock synchronization is available, the timetag can be used to schedule events with fixed delays that account for the network transport delay in communication between devices. The delays must be known and bounded. This is called forward synchronization and can be efficiently implemented with the priority queue data structure [Schmeder and Freed,

2008] [Brandt and Dannenberg, 1998] (Figure 2).

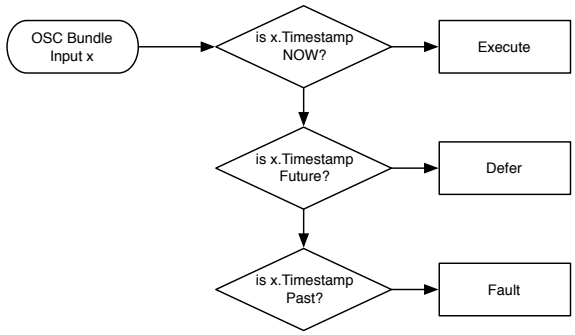

Figure 2: Forward synchronization scheduling for presentation of messages

Applications using OSC timestamps for synchronization should make clear in their documentation what type of clock synchronization is to be used, if any, as well as limits on tolerable network delay.

3.2 Effect of Jitter

Jitter is randomness in time. This may be found in the transport delay, or in the clock synchronization error. Unless it is removed, the effect of jitter on a signal is to corrupt it with noise. The noise is temporal so its magnitude depends on the rate of change of the signal, or its frequency content. Even for relatively low-rate gesture signals, jitter noise can play a significant role. In Figure 3 we see that a 2 msec jitter causes a significant reduction in the channel headroom. The fact that temporal jitter has a strong influence on signal quality is well known in the audio engineering community where a typical sampling converter operating at 96khz requires a clock with jitter measured in the picosecond range.

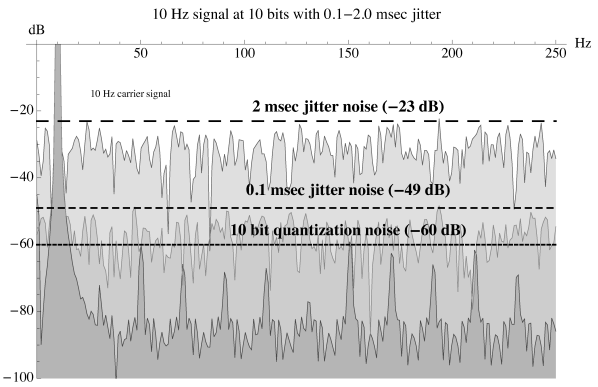

Figure 3: Effective channel headroom after jitter induced noise on a 10hz carrier signal

In Figure 4 we see what combinations of jitter and carrier frequency will degrade a gesture stream with 8-bit dynamic range.

	0.01 msec	0.1 msec	1. msec	2. msec	4. msec
0.5 Hz	100.806	80.942	60.5853	54.4588	48.2834
1. Hz	89.4672	69.2973	49.5129	**42.7719**	**37.1899**
2 Hz	83.5256	64.1865	**44.4936**	**37.811**	**32.166**
4 Hz	77.8606	58.3905	**38.2024**	**32.4498**	**25.4497**
8 Hz	72.3401	52.0053	**31.2989**	**25.7653**	**20.1786**
16 Hz	66.1133	**45.8497**	**25.8291**	**19.7408**	**14.3312**
32 Hz	60.2471	**39.6844**	**19.7202**	**13.546**	**8.26448**
64 Hz	53.9285	**33.8882**	**13.9203**	**7.90135**	**1.7457**

Figure 4: Signal headroom as a function of carrier frequency and standard deviation of delay error. **BOLD** where effective headroom is less than 8-bits dynamic range (8-bits = 48db).

3.3 Jitter Attenuation

From a simple inspection of the table shown, it is apparent that in order to transmit without loss of information an instrumental gesture data stream with frequency content up to 10 hz and 8-bit dynamic range the jitter must be less than 1/10th of a millisecond. Greater dynamic range requires proportionally less jitter where an error reduction by 50% improves dynamic range by 6 dB or 1-bit. To transmit a 10 hz signal with 16-bit dynamic range requires jitter to be less than .5 microseconds.

On contemporary consumer operating systems typical random delays of 1-10 milliseconds between hardware and software interrupts are unacceptably large [Wright et al., 2004], and this source of temporal noise ultimately inhibits the information transmission-rates for real-time control streams. If the data is isochronously sampled it is possible to use filters to smooth jitter [Adriensen and Space, 2005]. However this is not a typical expectation in audio control so something else must be done. If the clock synchronization error is smaller than the transport jitter (which is often the case) and the data stream uses timestamps, then it is possible to use forward synchronization to remove jitter from a control signal (shown in Figure 5).

This operation trades lower jitter for a longer fixed delay. Provided that total delays after rescheduling are less than 10 msec, a satisfactory music performance experience is possible.

3.4 Atomicity

Within each bundle exists a point-in-time sample of a collection of sub-stream messages. The scope over which the data is valid is defined both

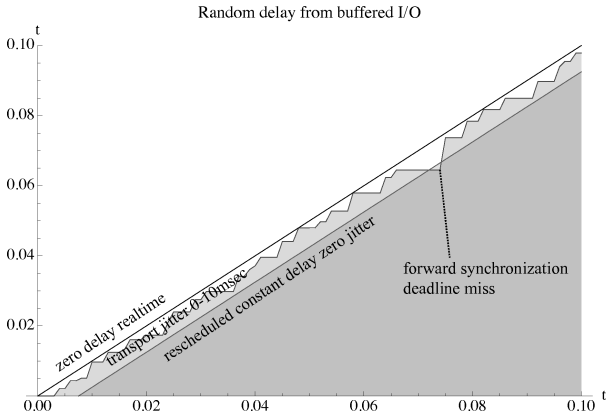

Figure 5: Typical transport jitter of 1-5 msec and its recovery by forward synchronization

with respect to the message addresses as well as some temporal window at the reference timetag.

In implementation practice for application design, message data needs to double-buffered or queued in a FIFO that is updated according to the associated timetag. This prevents unintended temporal skew between sub-streams.

4 Transport Considerations

A common transport protocol used with the OSC format is UDP/IP, but OSC can be encapsulated in any digital communication protocol. The specific features of each transport can affect the quality and availability of stream data at the application layer.

4.1 Datagram Transports

A datagram transport (UDP being a canonical example) is a non-assured transport. Each packet is either delivered in its entirety or not delivered at all. Packets may be out of order, in which case OSC bundle timestamps can be used to recover the correct order. Datagram transports provide a natural encapsulation boundary for each packet. In the case of UDP if the packet exceeds the maximum transmission unit (MTU) the packet may be fragmented over multiple pieces. The fragmentation can introduce extra delay as the UDP/IP stack must then reassemble the pieces before delivering the packet to an application.

4.2 Serial Stream Transport

Serial transports (TCP/IP being an example) provide a continuous data stream between endpoints. The OSC content format does not define a means for representing the beginning and end of a packet. In particular the OSC bundle can contain any number of encapsulated messages and there is no way for a parser to determine the total number until the end of packet is reached. Therefore the major need for sending OSC on a serial transport is that the packets must be encoded with some extra data to indicate where the packet boundaries are, called a framing protocol.

Two options have been proposed for packet framing on serial links. The idea proposed in the OSC 1.0 specification is an integer length-count prefixed on the start of each packet that indicates how many bytes to expect. This encoding requires a totally assured transport such as TCP/IP or USB-Serial. A serial transport with possible errors (such as RS232) will be broken if there is any error in the encoded length.

The SLIP method for framing packets (RFC 1055 [Romkey, 1988]) is an alternative that is robust to transmission errors and stream interruption. In general it is preferred over the former for its simple error recovery.

Assured transports must be used for any mission critical application of OSC and generally this means TCP/IP or something with a similar feature set is needed.

4.3 Isochronous Stream Transport

Isochronous protocols have guaranteed bandwidth, in-order delivery of data, but are not assured (no retries are made on failure). They may or may not provide natural packet boundaries.

Ethernet AVB and the isochronous modes of USB and Firewire are examples of this transport type. Ethernet AVB has additional features in that it also provides a network clock for synchronization and its own timestamp for synchronization of events with total latency as low as 2 msec and synchronization error of less than 0.5 microseconds. Class A streams in the AVB framework are unique among all the transports discussed here in that they are guaranteed to meet or exceed all synchronization and latency requirements needed for audio control data as described in this paper [Marner, 2009].

4.4 File Streams and Databases

For the archival recording and recall of audio control data streams, file systems and databases can be treated as serial stream transports with high block-based jitter in the retrieval phase. By considering a file to be a type of serial stream, OSC can use the same framing protocol

for serial stream encoding as a file format. Again the SLIP method is recommended, and it error recovery features enable robustness against file corruption and truncation.

When a recorded stream of OSC messages is replayed, the original timestamps are rewritten according to a simple linear transformation, and can then be reconstructed temporally using the forward synchronization scheduler. The rewriting of timestamps does not require relative time encodings. Since relative time can always be extracted from absolute time but not the converse, recording of OSC streams for archival purposes should always use absolute time values.

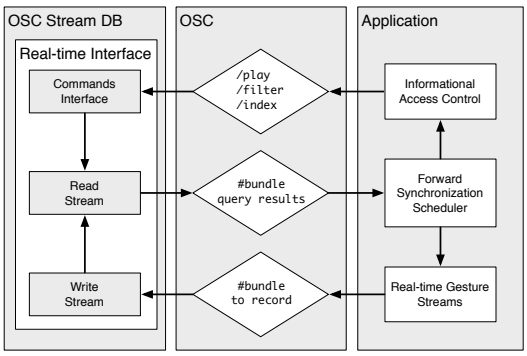

Figure 6: Multi-stream recording and playback interface to a database

A multi-stream approach for interfacing OSC streams to a database and efficient queries over the recorded data is demonstrated in the OSC-StreamDB project [Schmeder, 2009b] (Figure 6).

4.5 Bandwidth Constrained Transports

When bandwidth constrained transports are required such as wireless radios, OSC may be modified with some effort to enable lower bitrates. Depending on the nature of the data stream, adaptive sampling dependent on the information-rate can be used to reduce the number of messages on the network. Interpolation between frames can be used to recover a smooth control signal in reconstruction. This is likely to work well for instrumental gesture data as the actual effective number of new bits of information per second is relatively low.

Another major source of bit-sparsity in OSC streams is the message address field. Because it is typically a human-readable string, and English text has a bit rate of 1-1.5 bits per character, about 80% of the bits are redundant. A dictionary-type compression scheme could be used to compress the address strings if necessary.

4.6 Network Topology and Routing

The OSC 1.0 Specification included some language regarding OSC client and server endpoints. In fact this distinction is not necessary and OSC may be used on unidirectional transports, and more complex network topologies including multicast, broadcast and peer-to-peer. The OSCgroups project provides a method for achieving multicast routing and automatic NAT traversal on the internet [Bencina, 2009].

A shortcoming of many current OSC implementations using UDP/IP is missing support for bidirectional messaging. As a best practice implementations should try to leverage as much information as the transport layer can provide, as this makes more simple the task of configuration of the endpoint addresses in applications as well as stateful inspection at lower layers.

5 Describing Control Data

The address field in an OSC message is where descriptive information is placed to identify the semantics of the message data. The set of all possible addresses within an application is called an address space.

5.1 Descriptive Styles

Existing OSC practice includes a wide variety of strategies for structuring address spaces. Here we intend to clarify the differences between the styles rather than to promote any particular method as preferred. Four common styles have emerged over decades of software engineering practice: RPC, REST, OOP and RDF. Examples of OSC messages in each style accomplishing the same task (setting the gain on a channel) are given here.

5.1.1 RPC

The RPC (Remote Procedure Call) style employes functional enumeration, and lends itself to small address spaces of fixed functions:

```
/setgain (channel number = 3) (gain value = x)
```

5.1.2 REST

The REST (Representational State Transfer) style encourages state-machine free representations by transmitting the entire state of an entity in each transaction. Web application programmers are familiar with this style wherein every time a page is loaded, the application has two phases: setup (recreating the entire application state from the transferred representation)

and teardown (throwing it all away). The state-free property is what enables web-browsers to always pages outside the context of a browsing session by recalling a bookmark.

OSC address spaces using the REST style of resource enumeration have a familiar appearance since it is the most common style used in the construction of hyperlink addresses on the web.

```
/channel/3/gain (x)
```

5.1.3 OOP

The OOP (Object Oriented Programming) style is based on an intuitive concept of objects that are self-contained entities containing both attributes and specialized functions called methods that are procedures transforming their own attributes.

```
/channel/3@gain (x)
/channel/3/setgain (x)
```

OOP enables abstraction and layering in large systems. It may also need notations beyond the basic '/' delimiter used in path-style addresses since the OOP structure requires differentiation of the object, attribute and method entity types. In the above example we have used '@' following the XPath notation to indicate an attribute. In Jamoma a ':' character is used to similar effect [Place et al., 2008].

5.1.4 RDF

The RDF (Resource Description Framework) style employs ontological tagging to describe data with arbitrarily complex grammars. This style of control is the most powerful of the alternatives shown here, however its use also requires greater verbosity since it makes no semantic assumptions about the data structure.

The interpretation of the delimiter '/' in OSC as a hierarchical containment operator as it implies in the REST and OOP paradigms is not used in this style. Instead it is interpreted as an unordered delimiter between tags, and each tag is a comma-separated triple of subject, predicate, and object entities.

```
/channel,num,3
  /op,is,set
  /lvalue,is,gain
  /rvalue,units,dB (x)
```

5.2 Leveraging Pattern Matching

In OSC there a type of query operator called address pattern matching. Similar to the use of wildcard operators in command-line file system operations, patterns enable one-to-many mappings between patterns and groups of messages. However this technique is only useful if the target messages have a structure that enables the provided pattern syntax to make useful groupings. This structure is usually present when the address space follows the REST design paradigm. Designers of address spaces for applications should consider how the resulting addresses might make use of grouped-control by patterns.

```
/channel/*/gain (common gain value x)
```

Some effort is needed to retain efficiency of query operators in very large address spaces. This is possible using database structures such as the RDTree [Schmeder, 2009b].

5.3 Problems with Stateful Encodings

A stateful encoding of a control data stream is one where the meaning of a message has some dependence on a previously transmitted message. The interpretation of the message by the receiver requires some memory of previously received messages in addition to the necessary logic to correctly fuse the information. This logic is typically a finite state machine, although in general it can be more complex.

Suppose that there is a switch, called `/button`, with two possible states, off or on, represented by the numbers 0 and 1 respectively. A designer wishing to conserve network bandwidth decides only to transmit a message when the switch changes from one state to the other and so sends the number +1 to indicate a transition from 0 to 1 and the number -1 to indicate a transition from 1 to 0.

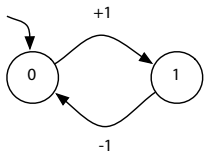

Figure 7: Finite state machine for parsing the transitions between a two states

The following is a valid sequence of messages that can be verified at by a receiver using the finite state machine shown in Figure 7.

```
/button +1
/button -1
/button +1
/button -1
```

A potential problem with this representation is that it is not robust to any errors in the

transmission of the data. Suppose that a message is lost due to the use of a non-assured transport such as UDP/IP, the sequence is then:

```
/button +1
/button +1
/button -1
```

In this example it is possible to make a more complex state machine that is capable of of recovering from the missing data, but we can immediately see that the complexity of the program has doubled (Figure 8).

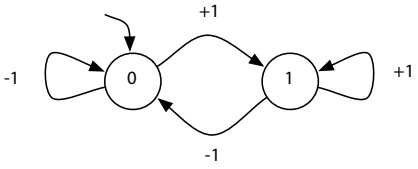

Figure 8: Finite state machine for parsing the transitions between a two states with error recovery logic

An alternative solution eliminates the need for a parsing engine entirely, by simply transferring the entire state of the switch in every message.

```
/button 0
/button 0
/button 0
/button 1
/button 1
/button 0
...
```

Furthermore, if these messages are continuously transmitted even when the state does not change, then the receiver needs to make no special effort to recover from a missed message. While this example is contrived to the point of being trivial, it does demonstrate that stateful encodings require more complex programs especially in the case of error handling. On modern network transports with typical bandwidth capacity of 1Gbits/sec, the simplicity of state-free representations is often more valuable than saving network bandwidth.

5.4 Layering Control Data

Between the source user action performed on a human input device to the high level application control stream, there are several intermediate layers of representation [Follmer et al., 2009] (Figure 9). The OSC format can be used at every layer where a digital representation of the data stream is present. Even though such streams may ultimately be not used in high level abstractions it is useful to retain the OSC address labels at each step.

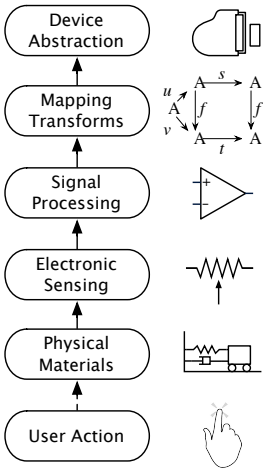

Figure 9: Intermediate layers of representation between user interface controller and an application

5.4.1 Complications of Mapping

The use of transformational mapping of gesture data is an important aspect in the design of interactive musical systems [Hunt et al., 2003]. In many cases useful mapping transformations carry out some type of information fusion that is a non-linear transformation of the data. However non-linear transformations require extra attention because they transform uniform noise into a non-uniform noise with a complicated spatial structure. It is possible to design adaptive filters that are optimal for a non-linear transformation, however this requires a more complex processing graph than what is shown in Figure 9.

Consider the non-linear transformation function,

$$f(x,y) = \frac{x-y}{x+y}. \tag{4}$$

If x and y are corrupted with any noise (which is inevitable) then the transformed variable $f(x,y)$ will greatly amplify that noise when x and y approach zero. This is evident from the fact that its derivative is unbounded as $(x,y) \to (0,0)$.

$$\partial_{x,y} f(x,y) = -\frac{x-y}{(x+y)^2} \pm \frac{1}{x+y} \tag{5}$$

Therefore thresholding (outlier rejection) and noise filtering must take place *after* the mapping transform, even though they are ostensibly signal processing layer operations (see Figure 10). In other words, the layer model of Figure 9 is not entirely correct as the layers are not strictly ordered. To enable out-of-order process-

ing across layers, designers should retain versions of data streams before and after mapping transformations under different address labels.

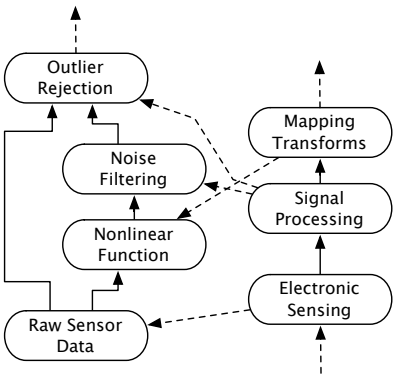

Figure 10: Cross-layer dependencies exist in the processing chain for input control data.

6 Conclusion

Here we present a brief summary of each point made in the document.

6.1 Transport and Synchronization

6.1.1 Instrumental Gesture Control

For data streams from measurement of human kinetic gestures, the temporal synchronization error should be not more than .1 milliseconds to ensure lossless transmission of periodic signals up to 10 hz with 8 bit dynamic range, with each extra bit of dynamic range requiring half the temporal error (50 usec for 9 bits, 25 usec for 10 bits, etc.). To resolve events with 1 msec relative temporal precision the sampling frequency of measurement should be 2000 hz.

6.1.2 Spatial Audio Control

The transport should be capable of updating the spatial audio parameters at rates of 100hz in order to resolve the full range of perceivable spatial effects that may extend up to 50 hz.

For spatial audio control data used in beamforming or wavefield synthesis rendering, following the AES recommended limits the synchronization error should be less than 5% of a sample frame for the highest controlled frequency. For example control over coefficients in a phase-mode beamforming array operating up to 10 khz requires a synchronization accuracy of 5 microseconds.

6.1.3 Digital Audio Synthesis Control

For audio synthesis algorithms in general the needs of control data are dependent on the nature of the algorithm and may vary widely depending on the level of detail and control bandwidth. Except for perhaps the most esoteric applications, a 0.5 microsecond temporal precision is sufficient for control of any audio synthesis algorithm.

6.1.4 Atomicity

In all cases careful use of double-buffering, lock-free queues and local memory-barrier operations should be used to ensure a best-effort is made for minimizing the synchronization skew between bundle-encapsulated sub-stream messages.

6.1.5 Latency and Jitter

The latency and jitter of secondary software interrupts typical of for human-input device streams are detrimental to the quality of control data. Bounded latency and minimal jitter should be ensured for audio control data.

6.2 Control Meta-data

6.2.1 Interface Design Patterns

Many styles of meta-data description are possible including procedural (RPC), resource-oriented (REST), object-oriented (OOP) and ontology-oriented (RDF). Application designers should feel free to choose the most appropriate style, however designers creating generic tools for OSC processing should support as many styles as possible.

6.2.2 Stateful Representation

When stateful representations of control data streams are used (with care), then assured transports should also be employed to reduce software errors that may be triggered by transmission errors.

6.2.3 Multi-Layered Representation

Retaining data stream representations at multiple levels of abstraction is useful as some operations on control data streams cannot be performed in a strictly sequential order. The general process structure for transformation of control data is a directed graph.

7 Acknowledgements

We are grateful to Meyer Sound Laboratories Inc. of Berkeley CA for financial support of this work. The anonymous reviewers provided helpful suggestions to improve this document. The users of OSC in the community including computer music application designers and researchers have played an important role in bringing to light important issues related to audio control.

References

Matthew Wright Adrian Freed. 1997. CAST: CNMAT Additive Synthesis Tools. `http://archive.cnmat.berkeley.edu/CAST/`.

Fons Adriensen and A Space. 2005. Using a DLL to Filter Time. In *Proceedings of the Linux Audio Conference*.

Audio Engineering Society. 2003. AES Recommended Practice for Digital Audio Engineering - Synchronization of Digital Audio Equipment in Studio Operations. Technical Report 11, AES.

Ross Bencina. 2009. OSCgroups. `http://www.audiomulch.com/~rossb/code/oscgroups/`.

Nicolas Bouillot and et al. 2009. *AES White Paper: Best Practices in Network Audio*, volume 57 of *JAES*. AES.

Eli Brandt and Roger Dannenberg. 1998. Time in Distributed Real-Time Systems. In *Proceedings of the ICMC*, pages 523–526, San Francisco, CA.

Chris Chafe, Michael Gurevich, Grace Leslie, and Sean Tyan. 2004. Effect of time delay on ensemble accuracy. In *In Proceedings of the International Symposium on Musical Acoustics*.

David Cortesi and Susan Thomas. 2001. *REACT™ Real-Time Programmer's Guide*. Number Document Number 007-2499-011. Silicon Graphics, Inc.

Sean Follmer, Björn Hartmann, and Pat Hanrahan. 2009. Input Devices are like Onions: A Layered Framework for Guiding Device Designers. In *Workshop of CHI*.

Adrian Freed. 1996. Audio I/O Programming on SGI Irix. `http://cnmat.berkeley.edu/node/8775`.

Andy Hunt, Marcelo M. Wanderley, and Matthew Paradis. 2003. The Importance of Parameter Mapping in Electronic Instrument Design. *Journal of New Music Research*, 32(4):429–440, December.

I. Scott MacKenzie, Tatu Kauppinen, and Miika Silfverberg. 2001. Accuracy measures for evaluating computer pointing devices. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 9–16, New York, NY, USA. ACM.

Geoffrey M Marner. 2009. Time Stamp Accuracy needed by IEEE 802.1AS. Technical report, IEEE 802.1 AVB TG.

Timothy Place, Trond Lossius, Alexander Jensenius, Nils Peters, and Pascal Baltazar. 2008. Addressing Classes by Differentiating Values and Properties in OSC. In *NIME*.

B. Rafaely. 2005a. Analysis and design of spherical microphone arrays. *Speech and Audio Processing, IEEE Transactions on*, 13(1):135–143, Jan.

B. Rafaely. 2005b. Phase-mode versus delay-and-sum spherical microphone array processing. *Signal Processing Letters, IEEE*, 12(10):713–716, Oct.

J. Romkey. 1988. RFC1055 - Nonstandard for transmission of IP datagrams over seria lines: SLIP. Technical report, IETF.

Andy Schmeder and Adrian Freed. 2008. Implementation and Applications of Open Sound Control Timestamps. In *Proceedings of the ICMC*, pages 655–658, Belfast, UK. ICMA.

Andrew Schmeder. 2009a. An Exploration of Design Parameters for Human-Interactive Systems with Compact Spherical Loudspeaker Arrays. In *Ambisonics Symposium*.

Andrew Schmeder. 2009b. Efficient Gesture Storage and Retrieval for Multiple Applications using a Relational Data Model of Open Sound Control. In *Proceedings of the ICMC*.

David Wessel and Matthew Wright. 2002. Problems and Prospects for Intimate Musical Control of Computers. *Computer Music Journal*, 26:11–22.

Matthew Wright, Amar Chaudhary, Adrian Freed, Sami Khoury, David Wessel, and Ali Momeni. 2000. An xml-based sdif stream relationships language. In *International Computer Music Conference*, pages 186–189, Berlin, Germany. International Computer Music Association.

Matthew Wright, Ryan Cassidy, and Michael Zbyszynski. 2004. Audio and Gesture Latency Measurements on Linux and OSX. In *Proceedings of the ICMC*, pages 423–429.

Matthew Wright. 2002. Open Sound Control 1.0 Specification. `http://opensoundcontrol.org/spec-1_0`.