Node and Message Management with the JunctionBox Interaction Toolkit

Lawrence Fyfe
InnoVis Group
University of Calgary
2500 University Drive NW
Calgary, AB T2N 1N4
Canada

Adam Tindale
Alberta College of
Art + Design
1407 14 Avenue NW
Calgary, AB T2N 4R3
Canada

Sheelagh Carpendale
InnoVis Group
University of Calgary
2500 University Drive NW
Calgary, AB T2N 1N4
Canada

ABSTRACT

Message mapping between control interfaces and sound engines is an important task that could benefit from tools that streamline development. A new Open Sound Control (OSC) namespace called Nexus Data Exchange Format (NDEF) streamlines message mapping by offering developers the ability to manage sound engines as network nodes and to query those nodes for the messages in their OSC address spaces. By using NDEF, developers will have an easier time managing nodes and their messages, especially for scenarios in which a single application or interface controls multiple sound engines. NDEF is currently implemented in the JunctionBox interaction toolkit but could easily be implemented in other toolkits.

Keywords

OSC, namespace, interaction, node

1. INTRODUCTION

The JunctionBox interaction toolkit [2], a library for the Processing [4] development environment, was created to enable interaction designers to easily build sound and music control interfaces for touch-enabled devices like vision-tracking tables and Android [3] tablets. The toolkit is designed to make mapping touch actions to sound controls easier while still enabling developers to build highly customized interfaces.

A mapping task that invariably occurs in systems that use Open Sound Control (OSC) [5] is managing OSC servers and the messages in their address spaces. To make this task easier, JunctionBox now offers functions for managing OSC servers and their messages. These functions are especially useful for scenarios in which a single interface (OSC client) needs to control multiple OSC servers. It is important to distinguish OSC servers in these scenarios, especially if they have distinct OSC address spaces.

To facilitate node management, the Nexus Data Exchange Format (NDEF), a new namespace for OSC, was created to handle both node management and message management. NDEF works by offering messages that are always identified by their source and by allowing for queries of OSC servers by OSC clients. The namespace is deliberately simple while leaving open the possibility of future enhancement. While NDEF is currently implemented in the JunctionBox toolkit, it has potential for use as a standard node data exchange format for other toolkits and applications.

NIME'12, May 21 – 23, 2012, University of Michigan, Ann Arbor.

2. NEXUS DATA EXCHANGE FORMAT

The Nexus Data Exchange Format (NDEF) is a new namespace created for the JunctionBox toolkit that is meant to make the task of mapping easier. NDEF is simply a defined namespace for OSC rather than a change to the specification, so applications wishing to use NDEF do not require new OSC libraries.

2.1 Connections

One of the central features of NDEF is node identification. All NDEF messages have the IP address and port of the message source (OSC server or client) as the first two arguments. The generalized form of NDEF messages is:

```
/ndef/[container]/[method] [IP address] [port]
```

This kind of identification is particularly useful for cases where one OSC client is in a one-to-many relationship with multiple OSC servers or where multiple OSC clients are in a many-to-many relationship with multiple OSC servers.

To begin an NDEF exchange, the OSC client sends out a request message. The type tag for the request is si where the IP address is the string and the port is the integer.

```
/ndef/connection/request,si
```

The NDEF exchange system is similar in some ways to the TCP handshake [1] in which both ends acknowledge the connection. However, NDEF is a two way exchange rather than a three-way handshake. When a connection request is received (and accepted), an accept message is sent to the OSC client.

```
/ndef/connection/accept,si
```

This exchange allow nodes to determine whether another node has NDEF capabilities and whether it is available for connections. Another important element of the exchange is that both nodes then can identify each other using IP address and port as unique identifiers. When multiple nodes may be involved in a network, this identification is essential.

2.2 Messages

Once a connection has been established between two or more nodes, each node can send a message request to the other nodes. A message request has a form similar to a connection request.

```
/ndef/message/request,si
```

The reply message has a slightly different form than the other messages in the namespace since, besides the IP address and port of the message source, it has the OSC message encoded as a string as the last argument.

```
/ndef/message/reply,sis
```

An example reply message containing the OSC message /foo:

```
/ndef/message/reply, "192.168.1.1" 7000 "/foo"
```

Any number of reply messages can be received by a requesting node once an initial request has been sent. This allows for some flexibility in the setup of OSC servers including the sending of new message replies as they are created on the server. NDEF does not provide a message format for setting ranges on OSC arguments. The reason for this is that JunctionBox values are always sent normalized from 0-1. This eliminates the need to set ranges since OSC servers will simply scale arguments to any range without the interface being concerned with the specific range. Since all numbers output by JunctionBox are normalized, all numbers are floats.

3. IMPLEMENTATION

In the JunctionBox toolkit, all NDEF message handling is done via the Dispatcher class. The following code will create a new Dispatcher, initialize NDEF listening, request a connection, and request messages.

```
Dispatcher d = new Dispatcher();
d.startListening();
d.requestConnection();
d.requestMessages();
```

Internally, JunctionBox has a Relay class that stores the IP address and port of the target node/OSC server and any messages mapped to that server. When an NDEF exchange occurs, the messages provided via NDEF are automatically added to a Relay for that node. For each connection that is accepted, a new Relay object is created. Messages can only be received for mapping if the sending node has previously been connected via the connection request/accept exchange. Figure 1 shows the relationship of Relays to the internal class structure of JunctionBox.

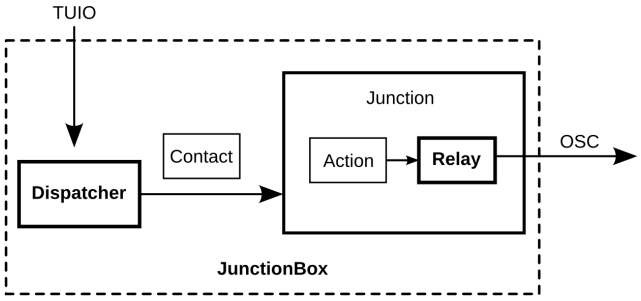

Figure 1: The use of the Relay in the JunctionBox toolkit.

Once the message request has been sent by the Dispatcher, message replies can be sent by the OSC server at any time and will still be added to the appropriate Relay for that OSC server since replies can be identified by the source IP address and port number.

Relay objects can be returned from the Dispatcher as an array with:

```
Relay[] relays = d.getRelays();
```

Messages can then be retrieved from each Relay as an array of strings. Since they are returned as strings, the messages can be displayed in any kind of interface or simply printed via the console for examination by the developer doing the mapping.

```
String[] messages = relay.getMessages();
```

That string array can then be used to map actions to messages via the JunctionBox mapping system. A Junction, representing a defined area of a touch interface that is active, can have it actions like translation, rotation and scaling mapped with the received messages. For example, the following snippet of code could be used to add the OSC messages from a node to an array of junctions for mapping to the Y translation action. The junctions could represent a series of vertical slider widgets.

```
for (int i = 0; i < message.length; i++) {
 junctions[i].addMessage(Action.TRANSLATE_Y,
 message[i]);
}
```

4. SUMMARY

This paper described the Nexus Data Exchange Format (NDEF), a new Open Sound Control (OSC) namespace for node management and message exchange. NDEF allows for the automatic exchange of messages from OSC servers to control interfaces. In addition to message mapping, NDEF also makes it easier to manage multiple OSC servers from a single interface by allowing for clear identification of all OSC servers. In this way, not only can multiple servers be identified but each server can have a completely separate OSC address space. NDEF is currently implemented in the JunctionBox interaction toolkit but could be implemented in a variety of other toolkits where node and message management are needed.

5. ACKNOWLEDGEMENTS

We would like to thank the Alberta College of Art + Design, the Canada Council for the Arts, the Natural Science and Engineering Research Council of Canada, SMART Technologies, the Canadian Foundation for Innovation, and the Alberta Association of Colleges and Technical Institutes for research support. We would also like to thank the members of the Interactions Lab at the University of Calgary.

6. REFERENCES

[1] V. Cerf, Y. Dalal, and C. Sunshine. Specification of internet transmission control program. `http://tools.ietf.org/html/rfc675`, 1974.

[2] L. Fyfe, A. Tindale, and S. Carpendale. Junctionbox: A toolkit for creating multi-touch sound control interfaces. In *Proceedings of the Conference on New Interfaces for Musical Expression*, pages 276–279, 2011.

[3] Google. Android developers. `http://developer.android.com/index.html`, 2012.

[4] C. Reas and B. Fry. Processing: programming for the media arts. *AI & Society*, 20(4):526–538, 2006.

[5] M. Wright. Open sound control: an enabling technology for musical networking. *Organised Sound*, 10(3):193–200, 2005.