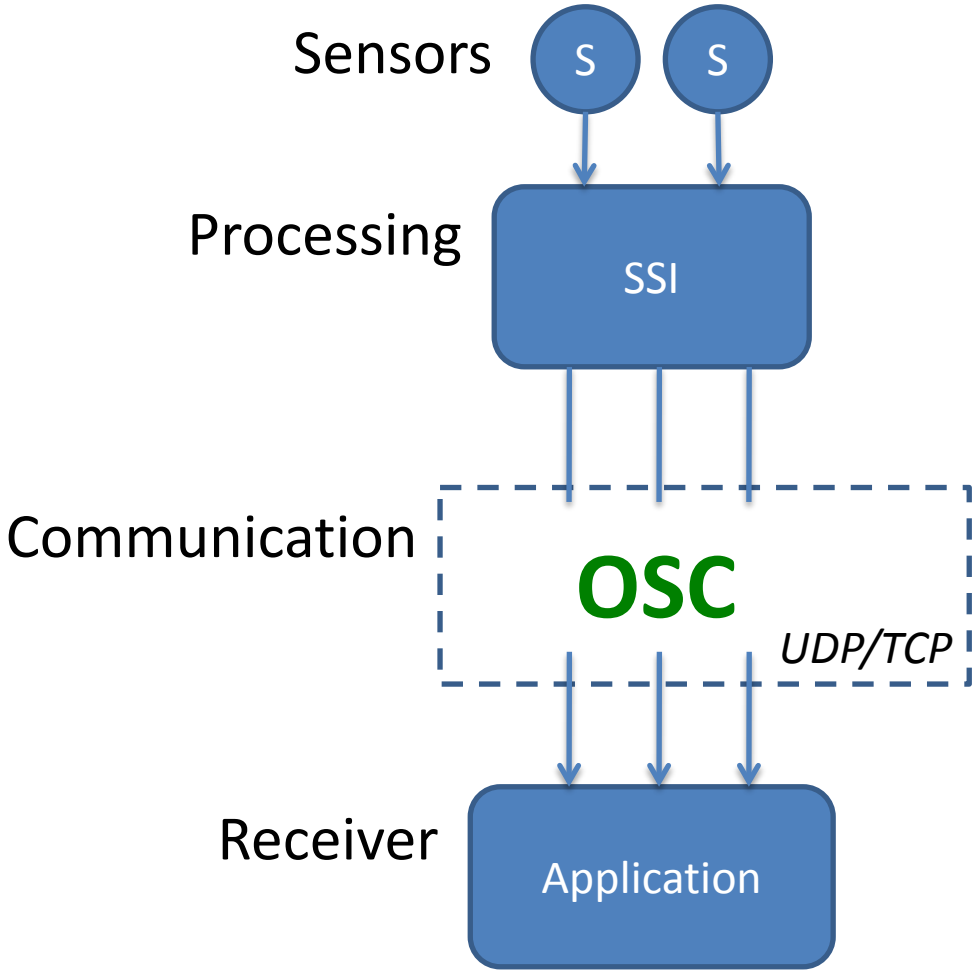
SSI & Open Sound Control

Johannes Wagner

05.02.2013

Architecture



Open Sound Control (OSC)

- Transport-independent and message-based light ware protocol to share data between multimedia devices
- Similar to MIDI but more powerful
- Free implementations available for all kind of systems and languages
- Website: http://opensoundcontrol.org/

OSC Messages

- Starts with an OSC Address Pattern followed by an OSC Type Tag String and a corresponding number of OSC Arguments
- **Int32:** 32-bit big-endian two's complement integer
- **Float32:** 32-bit big-endian IEEE 754 floating point number
- **OSC-string:** A sequence of non-null ASCII characters followed by a null
- **OSC-blob** An int32 size count, followed by that many 8-bit bytes of arbitrary binary data

Specification

- Based on UDP and OSC
- Three type of messages:

  `/strm` continuously data stream

  `/evnt` event-related data including tags

  `/text` text based control messages
- Each message has sender id and timestamp in order to synchronize messages

Stream

Pattern	Arguments	Type	
/strm	id	String	Identification
	time	Int32	Time in milliseconds
	sr	Float32	Sample rate in Hz
	num	Int32	Sample number
	dim	Int32	Sample dimension
	bytes	Int32	Size of a single sample value in bytes
	type	Int32	Sample type
	data	Blob	Byte array with data (interleaved)

Sample types:

SSI_UNDEF = 0,
SSI_CHAR = 1,
SSI_UCHAR = 2,
SSI_SHORT = 3,
SSI_USHORT = 4,
SSI_INT = 5,
SSI_UINT = 6,
SSI_LONG = 7,
SSI_ULONG = 8,
SSI_FLOAT = 9,
SSI_DOUBLE = 10,
SSI_LDOUBLE = 11,
SSI_STRUCT = 12,
SSI_IMAGE = 13,
SSI_BOOL = 14

Stream Example

- Sending head position as pair of integer values sampled at 30 fps, e.g.

```
sample    0   1   2   3   4
     X   20  20  14  11  10
     Y   10  10  11   9   9
```

`/strm “head” 0 30.0 5 4 5 <data>`

where `<data>` has a total size of 4x4x5 bytes, e.g.: `{80, [20 10 20 10 14 11 11 9 10 9]}`

Event and Text

Pattern	Arguments	Type	
/evnt	id	String	Identification
	time	Int32	Time in milliseconds
	duration	Int32	Duration in milliseconds
	state	Int32	Event state (0=completed,1=continued)
	num	Int32	Number of following events
	eventTag_1	String	Tag of first event
	eventValue_1	Float32	Value of first event
	…		
	eventTag_num	String	Tag of last event
	eventValue_num	Float32	Value of last event
/text	Id	String	Identification
	Time	Int32	Time in milliseconds
	Duration	Int32	Duration in milliseconds
	Message	String	Message content

Event and Text Example

- Sending key words and probabilities detected in interval [3.0 5.5]s (event is completed)

```
/evnt “words” 3000 2500 0 3 “I” 0.7
“feel” 0.4 “good” 0.9
```

- Sending key words as message:

```
/text “words” 3000 2500 “I feel good”
```

sockspy

- Tool for monitoring messages and data
- Plain messages:

```
sockspy --simple <port>
```

- OSC messages:

```
sockspy --osc <port>
```

```
- ascii : log stream
- console : output data on console
```