Gesture Information
Storage and Retrieval
with OSC Stream DB

SpatDIF/GDIF Meeting, IRCAM May 20-21 2010

Andrew W. Schmeder
andy@cnmat.berkeley.edu

CNMAT / UC Berkeley
http://cnmat.berkeley.edu/
“Musicians’ physical motion is practically meaningless without knowing the context. Musicians constantly adapt their gestures based on auditory, haptic, and visual feedback”

Matt Wright, ICMC 2007 Gesture Formats Panel
Information Interaction Design

Nathan Shedroff, “A Unified Field Theory of Design”, 1999
Information Interaction Design

Nathan Shedroff, “A Unified Field Theory of Design”, 1999
Nathan Shedroff, “A Unified Field Theory of Design”, 1999
Information Interaction Design

Knowledge  

Interaction Design  

Control  

Information Design  

Content  

Sensorial Design  

Experience  

Nathan Shedroff, “A Unified Field Theory of Design”, 1999
Open Sound Control
Open Sound Control

- A “bring your own semantics” content format
Open Sound Control

- A “bring your own semantics” content format
- Separation of syntax and semantics (a la XML)
Open Sound Control

- A “bring your own semantics” content format
  - Separation of syntax and semantics (a la XML)
- WHEN is as important as WHAT
Open Sound Control

- A “bring your own semantics” content format
  - Separation of syntax and semantics (a la XML)
  - WHEN is as important as WHAT
  - Timetag everything (using OSC Bundles)
Open Sound Control

- A “bring your own semantics” content format
  - Separation of syntax and semantics (a la XML)
  - WHEN is as important as WHAT
  - Timetag everything (using OSC Bundles)
  - No temporal assumptions (isochrony, frequency)
Open Sound Control

- A “bring your own semantics” content format

  - Separation of syntax and semantics (a la XML)

- WHEN is as important as WHAT

  - Timetag everything (using OSC Bundles)

  - No temporal assumptions (isochrony, frequency)

- Optimize for low implementation complexity
Open Sound Control

- A “bring your own semantics” content format
  - Separation of syntax and semantics (a la XML)
- WHEN is as important as WHAT
  - Timetag everything (using OSC Bundles)
- No temporal assumptions (isochrony, frequency)
- Optimize for low implementation complexity
  - Tradeoff: higher bandwidth
OSC Practice
## OSC Practice

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>OSI Layer #</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>7</td>
<td>Control Semantics, Choreography</td>
</tr>
<tr>
<td>Presentation</td>
<td>6</td>
<td>OSC Structure</td>
</tr>
<tr>
<td>Session</td>
<td>5</td>
<td>Discovery, Enumeration, Authentication</td>
</tr>
<tr>
<td>Transport</td>
<td>4</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>Network</td>
<td>3</td>
<td>Bandwidth Reservation</td>
</tr>
<tr>
<td>Frame</td>
<td>2</td>
<td>Clock Synchronization</td>
</tr>
<tr>
<td>Hardware</td>
<td>1</td>
<td>Cabling, Wireless, Power</td>
</tr>
</tbody>
</table>
# OSC Practice

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>OSI Layer #</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>7</td>
<td>Control Semantics, Choreography</td>
</tr>
</tbody>
</table>
## OSC Practice

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>OSI Layer #</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>7</td>
<td>Control Semantics, Choreography</td>
</tr>
<tr>
<td>Presentation</td>
<td>6</td>
<td>OSC Structure</td>
</tr>
</tbody>
</table>

Friday, May 21, 2010
## OSC Practice

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>OSI Layer #</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>7</td>
<td>Control Semantics, Choreography</td>
</tr>
<tr>
<td>Presentation</td>
<td>6</td>
<td>OSC Structure</td>
</tr>
<tr>
<td>Session</td>
<td>5</td>
<td>Discovery, Enumeration, Authentication</td>
</tr>
</tbody>
</table>
## OSC Practice

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>OSI Layer #</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>7</td>
<td>Control Semantics, Choreography</td>
</tr>
<tr>
<td>Presentation</td>
<td>6</td>
<td>OSC Structure</td>
</tr>
<tr>
<td>Session</td>
<td>5</td>
<td>Discovery, Enumeration, Authentication</td>
</tr>
<tr>
<td>Transport</td>
<td>4</td>
<td>Quality of Service</td>
</tr>
</tbody>
</table>
## OSC Practice

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>OSI Layer #</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>7</td>
<td>Control Semantics, Choreography</td>
</tr>
<tr>
<td>Presentation</td>
<td>6</td>
<td>OSC Structure</td>
</tr>
<tr>
<td>Session</td>
<td>5</td>
<td>Discovery, Enumeration, Authentication</td>
</tr>
<tr>
<td>Transport</td>
<td>4</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>Network</td>
<td>3</td>
<td>Bandwidth Reservation</td>
</tr>
<tr>
<td>OSI Layer</td>
<td>OSI Layer #</td>
<td>Topic</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>Application</td>
<td>7</td>
<td>Control Semantics, Choreography</td>
</tr>
<tr>
<td>Presentation</td>
<td>6</td>
<td>OSC Structure</td>
</tr>
<tr>
<td>Session</td>
<td>5</td>
<td>Discovery, Enumeration, Authentication</td>
</tr>
<tr>
<td>Transport</td>
<td>4</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>Network</td>
<td>3</td>
<td>Bandwidth Reservation</td>
</tr>
<tr>
<td>Frame</td>
<td>2</td>
<td>Clock Synchronization</td>
</tr>
</tbody>
</table>
## OSC Practice

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>OSI Layer #</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>7</td>
<td>Control Semantics, Choreography</td>
</tr>
<tr>
<td>Presentation</td>
<td>6</td>
<td>OSC Structure</td>
</tr>
<tr>
<td>Session</td>
<td>5</td>
<td>Discovery, Enumeration, Authentication</td>
</tr>
<tr>
<td>Transport</td>
<td>4</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>Network</td>
<td>3</td>
<td>Bandwidth Reservation</td>
</tr>
<tr>
<td>Frame</td>
<td>2</td>
<td>Clock Synchronization</td>
</tr>
<tr>
<td>Hardware</td>
<td>1</td>
<td>Cabling, Wireless, Power</td>
</tr>
</tbody>
</table>
OSC File Format...
OSC File Format...

- Files are just another type of transport
OSC File Format...

- Files are just another type of transport
  - Serial transport, block-based delay distribution
OSC File Format...

- Files are just another type of transport
  - Serial transport, block-based delay distribution
- osc-dump -slip -port 5000 > myrecording.osc
OSC File Format...

- Files are just another type of transport
  
  - Serial transport, block-based delay distribution
  
- osc-dump -slip -port 5000 > myrecording.osc

- Header:
  MIME 1.0
  Content-Type: application/osc; version=1.0; framing=slip
OSC File Format...

- Files are just another type of transport
- Serial transport, block-based delay distribution
- osc-dump -slip -port 5000 > myrecording.osc

- Header:
  MIME 1.0
  Content-Type: application/osc; version=1.0; framing=slip

- Framing:
  SLIP RFC1055: A simple encoding that is robust to stream interruption, truncation, etc. (about 20 lines of C code to implement)
The Hard Part...
The Hard Part...

- Reconstruction of temporal order, synchronization
The Hard Part...

- Reconstruction of temporal order, synchronization

- Use OSC Bundles everywhere
The Hard Part...

- Reconstruction of temporal order, synchronization
  - Use OSC Bundles everywhere
  - Use absolute, synchronized clocks
The Hard Part...

- Reconstruction of temporal order, synchronization
  - Use OSC Bundles everywhere
  - Use absolute, synchronized clocks
  - Ethernet AVB will solve this to sample-accuracy
The Hard Part...

- Reconstruction of temporal order, synchronization
  - Use OSC Bundles everywhere
  - Use absolute, synchronized clocks
    - Ethernet AVB will solve this to sample-accuracy
- Efficient queries
The Hard Part...

- Reconstruction of temporal order, synchronization
  - Use OSC Bundles everywhere
  - Use absolute, synchronized clocks
    - Ethernet AVB will solve this to sample-accuracy
- Efficient queries
  - Selection of data in arbitrary time-interval
The Hard Part...

- Reconstruction of temporal order, synchronization
  
  - Use OSC Bundles everywhere
  
  - Use absolute, synchronized clocks
    
    - Ethernet AVB will solve this to sample-accuracy
  
  - Efficient queries
    
    - Selection of data in arbitrary time-interval
    
    - Selection of sub-streams by address
The Hard Part...

- Reconstruction of temporal order, synchronization
  - Use OSC Bundles everywhere
  - Use absolute, synchronized clocks
    - Ethernet AVB will solve this to sample-accuracy
  - Efficient queries
    - Selection of data in arbitrary time-interval
    - Selection of sub-streams by address
    - Selection of data by spatial value
OSC Stream DB

- Lossless storage of any OSC stream content
OSC Stream DB

- Lossless storage of any OSC stream content
- Efficient queries: $O(\log n)$ seek, $O(1)$ next
OSC Stream DB

- Lossless storage of any OSC stream content
- Efficient queries: $O(\log n)$ seek, $O(1)$ next
- Time indexed (BTREE)
OSD Stream DB

- Lossless storage of any OSC stream content
- Efficient queries: O(log n) seek, O(1) next
  - Time indexed (BTREE)
  - Data-index (RTREE)
OSC Stream DB

- Lossless storage of any OSC stream content
- Efficient queries: $O(\log n)$ seek, $O(1)$ next
  - Time indexed (BTREE)
  - Data-index (RTREE)
  - Address-index (RDTREE)
OSC Stream DB

- Lossless storage of any OSC stream content
- Efficient queries: $O(\log n)$ seek, $O(1)$ next
  - Time indexed (BTREE)
  - Data-index (RTREE)
  - Address-index (RDTREE)
- Real-time (blockwise) or offline retrieval
OSC Stream DB

- Lossless storage of any OSC stream content
- Efficient queries: $O(\log n)$ seek, $O(1)$ next
  - Time indexed (BTREE)
  - Data-index (RTREE)
  - Address-index (RDTREE)
- Real-time (blockwise) or offline retrieval
- Scalable to 10M+ rows, network distributed service, high-reliability
OSC Stream

Bundle
Message...

Bundle

(...)

OSC Bundle

Bundle Identifier
#bundle

NTP Timestamp
Seconds
Seconds Fraction

Encapsulated Message(s)
Message #1
Length
OSC Message

(...)

OSC Message

Address
/foo/bar

Data
Typetags
,ifs
Arguments
1, 3.14, "baz"
Implementation
Implementation

- SQL Backend: PostgreSQL
Implementation

- SQL Backend: PostgreSQL
- OSC Stream DB
Implementation

- SQL Backend: PostgreSQL

- OSC Stream DB
  - Command-line C application, (~ 2000 lines)
Implementation

- SQL Backend: PostgreSQL
- OSC Stream DB
  - Command-line C application, (~ 2000 lines)
  - 3 OSC ports on 3 threads with 3 DB links
Implementation

- SQL Backend: PostgreSQL
- OSC Stream DB
  - Command-line C application, (~ 2000 lines)
  - 3 OSC ports on 3 threads with 3 DB links
  - Binary interface to PG
Implementation

- SQL Backend: PostgreSQL
- OSC Stream DB
  - Command-line C application, (~ 2000 lines)
- 3 OSC ports on 3 threads with 3 DB links
- Binary interface to PG
- Client side (so far): MaxMSP objects,
Implementation

- SQL Backend: PostgreSQL

- OSC Stream DB
  - Command-line C application, (~ 2000 lines)
  - 3 OSC ports on 3 threads with 3 DB links
  - Binary interface to PG

- Client side (so far): MaxMSP objects,
  - osc.schedule, osc.timetag, osc.route (events)
Implementation

- SQL Backend: PostgreSQL
- OSC Stream DB
  - Command-line C application, (~ 2000 lines)
  - 3 OSC ports on 3 threads with 3 DB links
  - Binary interface to PG
- Client side (so far): MaxMSP objects,
  - osc.schedule, osc.timetag, osc.route (events)
  - realtime.sig~, realtime.sample~ (signals)
Variable-rate real-time gesture retrieval (scrubbing) with piecewise linear blocks
SLAB32 (David Wessel)
Touch Event with Annotation

- /touch:
- /pressure:
- /pressure/max:
- /release:
/filter/stream “recording1”

/filter/address “/pad/07/touch”
t1 <- /seek/next

/filter/address “/pad/07/release”
t2 <- /seek/next

/read (t1, t2, rate=1.0)
...these queries are also O(log n) efficient

(SQLite cannot do this, need Postgres GiST indexes to support this)
10 messages per bundle, 2.7 Mbit/sec = 1000 bundles/sec
Benchmark Conclusion
Benchmark Conclusion

- Peak throughput:
  ~3Mbit/sec write, 50Mbit/sec read
Benchmark Conclusion

- Peak throughput:
  ~3Mbit/sec write, 50Mbit/sec read

- Latency:
  <10 msec roundtrip
Benchmark Conclusion

- Peak throughput:
  ~3Mbit/sec write, 50Mbit/sec read

- Latency:
  <10 msec roundtrip

- Write operation is CPU bound, then I/O bound
Benchmark Conclusion

- Peak throughput:
  ~3Mbit/sec write, 50Mbit/sec read

- Latency:
  <10 msec roundtrip

- Write operation is CPU bound, then I/O bound

- Read operation is I/O bound
Benchmark Conclusion

- Peak throughput:
  ~3Mbit/sec write, 50Mbit/sec read

- Latency:
  <10 msec roundtrip

- Write operation is CPU bound, then I/O bound

- Read operation is I/O bound

- Parallel scaling should work well
Benchmark Conclusion

- Peak throughput:
  ~3Mbit/sec write, 50Mbit/sec read

- Latency:
  <10 msec roundtrip

- Write operation is CPU bound, then I/O bound

- Read operation is I/O bound

- Parallel scaling should work well

- Queries and random seeks are very fast...
  until sizeof(DB) > sizeof(RAM)
Problems
Problems

- At recommended sampling rates (2000hz) we can only record SLAB32 for about 1 hour on existing hardware
Problems

- At recommended sampling rates (2000hz) we can only record SLAB32 for about 1 hour on existing hardware

- Would like to scale to hundreds of hours...
Problems

- At recommended sampling rates (2000hz) we can only record SLAB32 for about 1 hour on existing hardware

- Would like to scale to hundreds of hours...

- Indexes on spatial data (RTREE) are also very expensive to maintain
Music Gesture Data...
Music Gesture Data...

- Very high frequency
Music Gesture Data...

- Very high frequency
- 10msec temporal accuracy, 1msec temporal precision
Music Gesture Data...

- Very high frequency

  - 10msec temporal accuracy, 1msec temporal precision

  - 1000FS -> 2000Hz sample rate (that's a lot!)
Music Gesture Data...

- Very high frequency
  - 10msec temporal accuracy, 1msec temporal precision
  - 1000FS -> 2000Hz sample rate (that's a lot!)
- Is also sparse
Music Gesture Data...

- Very high frequency
  - 10msec temporal accuracy, 1msec temporal precision
  - 1000FS -> 2000Hz sample rate (that's a lot!)
- Is also sparse
  - Rhythm
Music Gesture Data...

- Very high frequency
  - 10msec temporal accuracy, 1msec temporal precision
  - 1000FS -> 2000Hz sample rate (that's a lot!)
- Is also sparse
  - Rhythm
  - Discrete events
Music Gesture Data...

- Very high frequency
  - 10msec temporal accuracy, 1msec temporal precision
  - 1000FS -> 2000Hz sample rate (that's a lot!)
- Is also sparse
  - Rhythm
  - Discrete events
  - Locally linear (energy bounded)
Gesture Information Rate

\[ I = \log_2 \left( 1 + \frac{\rho}{\sigma} \right) \text{ bits} \]
Gesture Information Rate

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

- Essentially, number of bits that change per second.
Gesture Information Rate

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

- Essentially, number of bits that change per second.
- Is the fundamental determinant in Fitt's Law
Gesture Information Rate

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

- Essentially, number of bits that change per second.
- Is the fundamental determinant in Fitt’s Law
- Sliding real-time estimate from moving mean / moving standard deviation.
Gesture Information Rate

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

- Essentially, number of bits that change per second.
Gesture Information Rate

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

- Essentially, number of bits that change per second.
- Is the fundamental determinant in Fitt’s Law
Gesture Information Rate

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

- Essentially, number of bits that change per second.
- Is the fundamental determinant in Fitt’s Law
- Sliding real-time estimate from moving mean / moving standard deviation.
Example: Touch pad
Example: Recording

Moving estimate of rate of information transfer in (x,y)
Compression Summary
Compression Summary

- Raw data: 2000hz * 32 => 64000 bits/sec
Compression Summary

- Raw data: 2000hz * 32 => 64000 bits/sec
- Estimated information rate: ~150 bits/sec
Compression Summary

- Raw data: 2000hz * 32 => 64000 bits/sec
- Estimated information rate: ~150 bits/sec
- Compression ratio: > 400:1
Compression Summary

- Raw data: $2000 \text{hz} \times 32 \Rightarrow 64000 \text{ bits/sec}$

- Estimated information rate: $\sim 150 \text{ bits/sec}$

- Compression ratio: $> 400:1$

- Simple strategy:
Compression Summary

- Raw data: 2000hz * 32 => 64000 bits/sec
- Estimated information rate: ~150 bits/sec
- Compression ratio: > 400:1
- Simple strategy:
  - Adaptive sampling and linear interpolation
Compression Summary

- Raw data: 2000hz * 32 => 64000 bits/sec
- Estimated information rate: ~150 bits/sec
- Compression ratio: > 400:1
- Simple strategy:
  - Adaptive sampling and linear interpolation
  - Stream format is same (OSC)
Compression Summary

- Raw data: \(2000\text{hz} \times 32 \Rightarrow 64000\ \text{bits/sec}\)

- Estimated information rate: \(\sim 150\ \text{bits/sec}\)

- Compression ratio: \(> 400:1\)

- Simple strategy:
  - Adaptive sampling and linear interpolation
  - Stream format is same (OSC)
  - Needs timestamp support for reconstruction.

Friday, May 21, 2010
SpatDIF/GDIF... comments
SpatDIF/GDIF... comments

- Interchange file formats don’t need to be efficient data structures, leave the hard stuff to a database.
SpatDIF/GDIF... comments

- Interchange file formats don’t need to be efficient data structures, leave the hard stuff to a database.

- Do we still need to investing the effort to maintain our own standards (SDIF, OSC)?
SpatDIF/GDIF... comments

- Interchange file formats don’t need to be efficient data structures, leave the hard stuff to a database.

- Do we still need to investing the effort to maintain our own standards (SDIF, OSC)?

- Extensible formats are everywhere now...
SpatDIF/GDIF... comments

- Interchange file formats don’t need to be efficient data structures, leave the hard stuff to a database.

- Do we still need to investing the effort to maintain our own standards (SDIF, OSC)?

- Extensible formats are everywhere now...

  - XML
SpatDIF/GDIF... comments

- Interchange file formats don’t need to be efficient data structures, leave the hard stuff to a database.

- Do we still need to investing the effort to maintain our own standards (SDIF, OSC)?

- Extensible formats are everywhere now...
  - XML
  - Google Protocol Buffers
SpatDIF/GDIF... comments

- Interchange file formats don’t need to be efficient data structures, leave the hard stuff to a database.

- Do we still need to investing the effort to maintain our own standards (SDIF, OSC)?

- Extensible formats are everywhere now...
  
  - XML
  
  - Google Protocol Buffers
  
  - HTML5, SMIL, WebM, VP8, Ogg,
And, Real-Time Sucks
And, Real-Time Sucks

- Real-time clocks and good synchronization is still a big pain..... except now we are getting.....
And, Real-Time Sucks

- Real-time clocks and good synchronization is still a big pain..... except now we are getting.....

- Ethernet AVB
And, Real-Time Sucks

- Real-time clocks and good synchronization is still a big pain..... except now we are getting.....

- Ethernet AVB
  - Open standard from IEEE
And, Real-Time Sucks

- Real-time clocks and good synchronization is still a big pain..... except now we are getting.....

- Ethernet AVB
  
  - Open standard from IEEE
  
  - 2msec latency and 500 nsec sync (5% of FS at 96kHz) over 7 hop network
And, Real-Time Sucks

- Real-time clocks and good synchronization is still a big pain..... except now we are getting.....

- Ethernet AVB
  - Open standard from IEEE
  - 2msec latency and 500 nsec sync (5% of FS at 96khz) over 7 hop network
  - Hundreds of channels of audio, video
And, Real-Time Sucks

- Real-time clocks and good synchronization is still a big pain..... except now we are getting.....

- Ethernet AVB
  - Open standard from IEEE
  - 2msec latency and 500 nsec sync (5% of FS at 96khz) over 7 hop network
  - Hundreds of channels of audio, video
  - Real-time message streaming over a MIME-typed channel

Best Practices for Open Sound Control, Schmeder, Andrew; Freed, Adrian; Wessel, David, Linux Audio Conference, Utrecht, NL, (2010) Abstract | PDF

Efficient Gesture Storage and Retrieval for Multiple Applications Using a Relational Data Model of Open Sound Control, Schmeder, Andrew, ICMC 2009, (2009) Abstract | PDF

Features and Future of Open Sound Control version 1.1 for NIME, Freed, Adrian; Schmeder, Andy, NIME, 04/06/2009, (2009) Abstract | PDF

The end!

andy@cnmat.berkeley.edu