A Query System for OSC...
towards a common real-time command/control protocol

Andrew Schmeder  
(presenting)

and Matthew Wright

June 30 2004
A Query System for OSC...
towards a common real-time command/control protocol

Preamble

➔ History
  • “A Query System for OSC...” might be OSC/Command
    • OSC/Command : Control + Command
    • QUERY is a type of command

➔ Acknowledgements
  • Matthew Wright, OSC Specification 1.0 and the Query System for OSC
  • James McCartney, creator of SuperCollider
  • Amar Chaudhary, creator of Open Sound World

➔ Disclaimer
  • Work under construction...
  • Statements are provided “as is”
A Query System for OSC...
towards a common real-time command/control protocol

**Background : Open Sound Control**

- Open Sound Control Specification 1.0
  - Packetized
  - Real-time
    - Small, bounded resources
    - Lightweight
  - Stateless
  - Scalable
  - Translateable
    - Human-readable hierarchical namespace
  - Transport and Platform-Independent
- Draft Proposals for an OSC Query System (Wright, et. al)
  - */documentation, */type-signature, etc
A Query System for OSC...
towards a common real-time command/control protocol

Background: Open Sound World

- Implementation of ... an interface for OSW with OSC
  - Hierarchical address space
  - Every node corresponds to an Object, e.g.,
    - Container
    - Transform
    - State
    - Activation
  - Nodes are unordered sets, addressed by name
  - Flow of execution is determined by directed graph
- “Method oriented” command system --
  - /container/transform/activation-command [ args... ].
A Query System for OSC...
towards a common real-time command/control protocol

Background: SuperCollider

- SuperCollider Server Synth Engine Command Reference
  - Tree where a node is a Synth or Group
  - Nodes are in ordered lists, overall tree order determines execution of Synths.
- “Agent Oriented” command system
  - `/command [ node, args, [ node, args, [ ... ] ] ]`
  (A. Freed)
A Query System for OSC…
towards a common real-time command/control protocol

Control and Command…

➔ control, n., dictionary.com :
  • A controlling agent, device, or organization
  • A spirit presumed to speak or act through a medium
  • An instrument... used to operate, ... or guide a machine.
    i.e., Controls.

➔ command, n., dictionary.com :
  • (Computer Science) A signal that initiates an operation defined by an instruction
Motivation

- Complex or general-purpose applications
  - Open Sound World
  - SuperCollider
  - ... plugin automation, distributed/peer computing
- Can diverse applications share a common interface?
  - To some extent, yes – it has been done with OSC
  - For complex applications; as of yet there is no sharing
- Do we need this?
  - Yes.
  - Inter-application multi-media is a growing trend
  - Programmers should not need to invent their own command syntax...
A Query System for OSC... 
towards a common real-time command/control protocol

**Requirements...**

- **Transport**
  - Reliable delivery
  - Bi-directional; or multi-directional
  - Peer discovery

- **State**
  - **Client/Server relation assigns responsibilities**
    - **Client**
      - Track server namespace
      - Monitor important parameters
    - **Server**
      - Answer queries
      - Send notifications

- **Optimization Support**
  - Time/Space tradeoff
A Query System for OSC...
towards a common real-time command/control protocol

...Requirements

➔ Resource Management
  • Testing and control of platform limits

➔ Optimization
  • Some people think we need it...
  • Closing the efficiency gap

➔ Compatibility
  • Don't break existing OSC use patterns (control-patterns)

➔ OSC Extensions
  • More types (primitive and compound)
  • More powerful pattern matching language
  • Parsing rules for optimization

➔ Interoperability
  • Negotiation of capabilities between peers
  • Identification of common namespaces (schema)
Survey : Command Syntax

* * n i x  C L I

/command [ args... ] [ files... ]

* U R L

scheme://domain/path/file.ext#fragment?query

* X S L / X P a t h

<xsl:template match="/">
  <xsl:command select="*">
    <xsl:apply-templates>...</xsl:apply-templates>
  </xsl:command>
</xsl:template>

* X M L - R P C

<methodCall>
  <methodName>object.command</methodName>
  <params>
    <param><value><int>42</int></value>
  </params>
</methodCall>
A Query System for OSC...
towards a common real-time command/control protocol

Survey: Response Syntax

- *nix CLI
  - Standard Input, Output, Error
- HTTP
  - HTTP Status Code (200: OK, 404: Not Found...)
- XML-RPC
  - `<methodResponse>`
    - `<fault>` | `<params>` ...
- SuperCollider
  - `/done` | `/fail` | `/late` ...
- Open Sound World
  - `/input/address (int)return-code [ args... ]`
A Query System for OSC...
towards a common real-time command/control protocol

Survey: Asynchronous Notification Syntax

- **SuperCollider**
  
  /n_on | /n_off ...

- **Open Sound World**
  
  Set Parameter => Trigger Activation

- **GUIs**
  
  bind(activator, function)

- **POSIX**
  
  select(...) | poll(...) ...  
  signals and handlers

- **Max / PD + OSC**
  
  [ route /input ... ]
  
  => [ inlet X is hot ... ]
A Query System for OSC...

towards a common real-time command/control protocol

Proposal...

- Let OSC/Command addresses have two axes.
  - The Control Axis
  - The Command Axis
    - /control/axis/#command.axis
  - Keeps control and command layers separate
  - Conform with “typical OSC address” use pattern; i.e., address corresponds to objects/parameters.
  - **OSC => URI**
    - osc://domain:port/path#command?args...

- Types of Signals
  - Control (no command, back-compatible)
  - Prompt / Query
  - Response : Normal / Error / Fault
  - Notification
...Proposal

- Example:
  
  ```
  -> /container/node/#prompt
  <- #reply [ /container/node/#prompt ] [ answer ]
  ```

- Some basic queries (so far):
  
  ```
  */#documentation
  */#type-signature
  */#return-type-signature
  */#current-value
  */#osc-schema
  ```

- Some extra types:
  
  ```
  {} dictionary
  v* vector
  u  unicode string
  ```
A Query System for OSC...
towards a common real-time command/control protocol

**Query = Prompt + Response/Error**

- Response might be *asynchronous*
  - It must contain sufficient information to disambiguate the prompt; a copy of the input message (or at least its address) as the first argument is sufficient.

```
#reply [ copy of input message ... ] ...
```

- Error conditions
  - Same as #reply, but adds an error-code, e.g.,

```
#error.code [ copy of input ... ] ...
```

- Some error codes:
  - undefined
  - relocated
  - mismatch
  - infeasible
  - missing
  - failed
  - late
  - volatile
Query: Documentation

- Gets human-readable documentation
  
  \[ \text{\#documentation} \]
  \[ \text{\#reply} /\text{#documentation} \quad \text{“Open Sound World 1.0”} \]

- Returns a string, or a URL.
- Multi-lingual; specify language preferences in query.

\[ \text{\#documentation} 'fr', 'en' \]
\[ \text{\#reply [ documentation 'fr', 'en' ] [ language-matched ] [ string-or-url ]} \]

- Use unicode strings, as necessary.
A Query System for OSC...
towards a common real-time command/control protocol

Query : Type-Signature

- Find out what types are expected as input arguments at an address
  
  `/container/node/#type-signature
  #reply [ ...#type-signature ] 'fiis'

- Determine syntax for type-code pattern matching.
  - e.g., standard regex

- Extra information:
  - Parameter constraints
  - Class name
  
  `#reply [ /cycle~/freq/#type-signature ] 'f'
  { min => 0.0, max => 22050.0, ... }`
A Query System for OSC...
towards a common real-time command/control protocol

Query : Return Type Signature

- For addresses which represent callables
  - Describes their return format.
    
    `/container/method/#return-type-signature
    #reply [ ...#return-type-signature ] '...'`
Query : Current-Value

- Get the current-value of a parameter
  - Value should match type-signature
    
    ```
    /param/#current-value
    #reply /param/#current-value 42.0
    ```
  
  - Query should be used with a timestamped input; i.e., *when* do you want to know the current value?
A Query System for OSC... towards a common real-time command/control protocol

Query: List Namespace

- List sub-nodes at a location
  
  /container/
  
  #reply /container/ 'node1', 'node2', ...

- A namespace might be volatile
  
  /grain-cloud/atoms/
  
  #error.volatile [ /grain-cloud/atoms ]

- Volatility is determined by platform resource constraints...
- Incremental notification is more efficient way to monitor activity in a volatile space...
Query: OSC Schema

- Identify common namespace use patterns.
  - Schema = Namespace + Semantics
  - Similar to xml-ns
    - `/midi_port/#osc-schema`
    - `#reply /midi_port/#osc-schema`
    - `http://midi.opensoundcontrol.org`
  - Decentralized system
    - Authors create their own ideas for namespace
    - Identify namespace usage by URL
  - Centralized also... (M. Cantor)
    - Schema hosted on opensoundcontrol.org are the “official recommendations”.
A Query System for OSC...
towards a common real-time command/control protocol

Pattern Expansion

- How to handle use of OSC address patterns
  
  /\*/(#command
  
  - Expand
    - Globbing pattern is equivalent to sending \( n \) messages to each matching target... respond accordingly.
      
      \rightarrow /\*/(#documentation
      \leftarrow #reply /node1/#documentation 'Node #1'
      \leftarrow #reply /node2/#documentation 'Node #2'
      \ldots
  
  - Interpret
    - e.g., Container with Template
      
      \rightarrow /grain-cloud/atoms/*/#type-signature
      \leftarrow #reply /grain-cloud/atoms/*/#type-signature
      ' ' { class => 'Grain...' }
Container Patterns

- Set value of contained objects with relative addressing...

```plaintext
/container/  {
    node1  =>  42.0,  
    node2  =>  43.0,  
    subcontainer/  =>  {
        ...  
    }  
    ...  
}
```
A Query System for OSC...
towards a common real-time command/control protocol

Notifications

➔ Parameter state monitoring
  • on-set
  • on-change

➔ Namespace changes
  • on-move
  • on-destroy
  • on-create

➔ Binding Options
  • return-address
  • update-rate
  • timeout

➔ Syntax

/address/#notify.on-set { timeout, max-rate, .. }
#notice.on-set [ ... ]
A Query System for OSC...
towards a common real-time command/control protocol

Optimization

- String-Intern System
  - All OSC 'command strings' can be replaced by an integer ID.
    - Use negative numbers: (first-bit set to differentiate from printable strings...)
    - Apply to addresses and type-tag strings
  - Connects to notification system
    - As the server maps addresses => ints, a notification message to the client will make it aware of the location...
- Hash Codes instead of copy of input message (McCartney)
- Specifics of how this works...
  - TODO
A Query System for OSC...
towards a common real-time command/control protocol

Compliance Verification

- Server states its features / limitations
  - Validation service can test for proper implementation
  - Max packet size, etc...
  - TODO
Transport Reliability

- Parity / Journaling:
  
  #notice.parity [ parity-data ... ]
  
  - Client can detect if a packet was missed
  - Missing packets can be reconstructed from parity
  - Quantity of parity depends on QoS
  - TODO... (non trivial?)
  - J. Lazzarro can comment more on this topic...
A Query System for OSC...
towards a common real-time command/control protocol

Working Group

Interested?
- andy@a2hd.com
- osc-dev mailing list
Working Group: Bundle Handling

- How does the reply handle bundles when it needs to copy the input into the reply... (J. McCartney)
- Would be easy if you could reference by message ID
- Or... Expand them -- a bundle is equivalent to \( n \) bundles with the same timestamp.

\[
\text{\#bundle (t) } n \quad [ \\
\quad /\text{message1}, \\
\quad /\text{message2}, \ldots \\
\text{]}
\]

\[
\text{\#reply [ \#bundle (t) /\text{message1} ]}
\]

\[
\text{\#reply [ \#bundle (t) /\text{message2} ]}
\]
Leases are an effective way to allow the client to request resources on the server (i.e., the request expires if not renewed), without compromising server reliability. (A. Freed)

- e.g., connection streams with timeouts...
A Query System for OSC...
towards a common real-time command/control protocol

Working Group: String Registry

- Hash codes
- State
The end.