MIDDLEWARE TRENDS AND MARKET LEADERS 2011

A. Dworak
P. Charrue, F. Ehm, W. Sliwinski, M. Sobczak
Controls Group, Beams Department
CERN

IGALEPCS 2011
14TH OCTOBER 2011, GRENOBLE
Middleware definition and types

**Middleware** – software that allows communication between other software components running on one or more machines.
CERN Controls System

GUI operator consoles

Controls Middleware

Front ends
“virtual devices”

Embedded Middleware

Hardware Devices
CERN Controls Middleware

- Linux/Windows
- C++/Java process
- Controls Middleware API
- CORBA
- TCP/IP
- CORBA
- Controls Middleware API
- C++ process
- RT Linux/LynxOS
CERN Controls Middleware

Problems:
- C++ and Java implementations differ 😞
- Heavy in memory usage 😞
- Complex error prone API 😞
- No direct support for pub/sub 😞
- Blocking issues (JacORB) 😞
- Shrinking community 😞
- Lack of new releases and bug fixes 😞

Long LHC shutdown at the end of 2012
A unique possibility!
Instead of patching let’s provide a new one!
CERN Middleware Requirements

- Desirable
- Mandatory
- Fundamental
CERN Middleware Requirements

Desirable

Mandatory

Fundamental

- C++/Java
- Linux/Windows
- Over TCP/IP LAN
CERN Middleware Requirements

Desirable

Mandatory
QoS
Asynchronous
Performance & Scalability
Stability, Maturity & Longevity
Open source, redistributable license

Fundamental
C++/Java
Linux/Windows
Over TCP/IP LAN
CERN Middleware Requirements

Desirable
- Lightweight
- Active community
- Friendly API, documentation
- Request/reply & pub/sub patterns

Mandatory
- QoS
- Asynchronous
- Performance & Scalability
- Stability, Maturity & Longevity
- Open source, redistributable license

Fundamental
- C++/Java
- Linux/Windows
- Over TCP/IP LAN
How did we evaluate → our criteria

**Appearance**
- **Creators**
  - specification
  - documentation
- **Users**
  - forums
  - bug reports
- **Internet**

**Simple usage**
- **Download**
  - licensing
- **Compile**
  - LynxOS & gcc 2.95
- **Run examples**

**Testing**
- **Communication patterns**
- **Performance**
- **QoS**
- **Exceptional situations**

**CRITERIA**
- API, look & feel, documentation
- resources, binary size, memory
- Community, maturity
- Communications patterns
- QoS
- performance
Surprising how many great new products!

- BSD sockets
- RPC
- CORBA

Timeline:
- 1969 - ARPANET
- 1974 - TCP
- 1980 - UDP
- 1981 - IP v4
- 1983 -
- 1991
- 1998
- 2011
- TODAY
Evaluated middleware products

All opinions are based only on our knowledge and evaluation. Each of the products, depending on the requirements, may constitute a good solution.

- CoreDX
- RTI DDS
- OpenSpliceDDS
- ZeroMQ
- Ice
- QPid
- RabbitMQ
- YAMI
- MQtt
- RSMB
- Mosquito
- JacORB
- omniORB
CORBA (omniORB, JacORB)

Object-oriented communication platform standardized by OMG. It is over 20 years old, and the standard is well established but...

- Complex, hard to learn and use ☹
- Differences between implementations ☹
- Big memory footprint ☹
- Shrinking community ☹
- Seldom updates and bug fixes ☹
Object-oriented middleware by ZeroC, people who left CORBA to fix its problems. Conceptually similar to CORBA but...

- Better design and implementation
- Modern easier to use API and IDL mapping
- Concise C++ and Java implementation
- Support for versioning
- Active community
Thrift

Middleware by Facebook (moved to Apache)

- Lightweight, free of unnecessary dependencies
- Modern, clean API
- Active community
- Ongoing development, still incomplete
- No documentation, empty tutorial
YAMI4

Message-oriented middleware developed by one of our former colleagues. Used by some CERN Controls services.

- Lightweight, free of unnecessary dependencies
- Well designed, modern API
- Asynchronous req/rep and pub/sub patterns
- Dynamic type specification (paid in performance)
- Small community
AMQP – OpenAMQ, Qpid, RabbitMQ

Middleware where messages are distributed by a broker. A few independent implementations of the broker and clients.

- Big community
  😊
- Designed to support pub/sub
  😊
- Protocol standard issues
  😞
- Broker – single point of failure
  😞
- Broker – additional hop, slower communication
  😞
- Broker – additional, non compliant monitoring tools
  😞
DDS – RTI, CoreDX, OpenSplice

Data-oriented middleware with p2p communication. Standard defined by OMG, with a few compatible implementations.

- Big industrial and military community
- Possible compatibility with CORBA
- Designed to support pub/sub
- Req/rep possible but needs 2 channels
- Steep learning curve
- Complex API
ZeroMQ

Message-oriented middleware by iMatix (they dropped OpenAMQ.)
API resembles BSD sockets but it is so much more.

- Simple yet powerful API
- Support for in-proc, inter-proc, TCP, PGM
- Lightweight, free of unnecessary dependencies
- Many communication patterns
- May be used as a concurrency framework
- Active community
Performance requirements reviewed

Instead of estimating

- We asked our users what they will need
- Gathered statistics from the current system

- Most demanding users identified
- Definition of a few performance tests
Performance tests, reliable req/rep

4000 msg/sec
Payload = 4B

5 msg/sec
Payload = 10MB

Many small messages

A few big messages
Performance tests, reliable req/rep

4000 msg/sec
Payload = 4B

5 msg/sec
Payload = 10MB

C++ server with client in:
Performance tests, reliable req/rep

4000 msg/sec
Payload = 4B

5 msg/sec
Payload = 10MB

C++ server with client in:
Performance tests, reliable req/rep

4000 msg/sec
Payload = 4B

5 msg/sec
Payload = 10MB

C++ server with client in:
Performance tests, reliable req/rep

4000 msg/sec
Payload = 4B

5 msg/sec
Payload = 10MB

C++ server
with client in:
Performance tests, reliable req/rep

4000 msg/sec Payload = 4B

5 msg/sec Payload = 10MB

C++ server with client in:
Performance/Scalability tests, reliable pub/sub

400 msg x 8 B
10 clients, <50ms

30 msg x 8 B
10 clients, <20ms

Throughput

Latency
Performance/Scalability tests, reliable pub/sub

400 msg x 8 B
10 clients, <50ms

30 msg x 8 B
10 clients, <20ms

![Graph showing performance comparison](image-url)
Performance/Scalability tests, reliable pub/sub

400 msg x 8 B
10 clients, <50ms

30 msg x 8 B
10 clients, <20ms
Performance/Scalability tests, reliable pub/sub

400 msg x 8 B
10 clients, <50ms

30 msg x 8 B
10 clients, <20ms
# Products comparison (according to the criteria)

<table>
<thead>
<tr>
<th></th>
<th>Sync, async &amp; msg patterns</th>
<th>QoS</th>
<th>Dependencies &amp; memory f-p</th>
<th>Performance</th>
<th>Look &amp; feel, API, docs</th>
<th>Community &amp; maturity</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZeroMQ</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>6</td>
</tr>
<tr>
<td>Ice</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>5</td>
</tr>
<tr>
<td>YAMI4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>4</td>
</tr>
<tr>
<td>RTI</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>3</td>
</tr>
<tr>
<td>Qpid</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
</tr>
<tr>
<td>CORBA</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>2</td>
</tr>
<tr>
<td>Thrift</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>2</td>
</tr>
</tbody>
</table>
# Products comparison (according to the criteria)

<table>
<thead>
<tr>
<th></th>
<th>Sync, async &amp; msg patterns</th>
<th>QoS</th>
<th>Dependencies &amp; memory f-p</th>
<th>Performance</th>
<th>Look &amp; feel, API, docs</th>
<th>Community &amp; maturity</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZeroMQ</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>6</td>
</tr>
<tr>
<td>Ice</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>5</td>
</tr>
<tr>
<td>YAMI4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>4</td>
</tr>
<tr>
<td>RTI</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>3</td>
</tr>
<tr>
<td>Qpid</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
</tr>
<tr>
<td>CORBA</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>2</td>
</tr>
<tr>
<td>Thrift</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>2</td>
</tr>
</tbody>
</table>
# Products comparison (according to the criteria)

<table>
<thead>
<tr>
<th></th>
<th>Sync, async &amp; msg patterns</th>
<th>QoS</th>
<th>Dependencies &amp; memory f-p</th>
<th>Performance</th>
<th>Look &amp; feel, API, docs</th>
<th>Community &amp; maturity</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZeroMQ</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>6</td>
</tr>
<tr>
<td>Ice</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>5</td>
</tr>
<tr>
<td>YAMI4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td>4</td>
</tr>
<tr>
<td>RTI</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>3</td>
</tr>
<tr>
<td>Qpid</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
</tr>
<tr>
<td>CORBA</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>2</td>
</tr>
<tr>
<td>Thrift</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>2</td>
</tr>
</tbody>
</table>
Conclusions

- Several good middleware solutions available.
- The choice is dictated by the most critical requirements for any given application. Not easy → QoS policies and performance matter, but also ease of use, community, ...

- Concerning CERN Controls Middleware...
  - Prototype with the most promising candidates:
    - Ice
    - ZeroMQ
    - YAMI

- Deploy the new middleware before the long accelerator shutdown at the end of 2012