Peer to Peer publish/subscribe using Lua and DDS

Gianpiero Napoli
Senior Software Engineer - RTI

gianpiero@rti.com / @magopieri
Agenda

**Real-Time Pub Sub (for the IIoT) using Lua**

- Who am I
- What is (RTI) DDS
  - QoS
- How we used Lua
  - to simplify APIs and
  - to add scripting capabilities
- Demo
Who am I?
and what do I do
Who?
What we do

- We build real-time **middleware**
  - Real Time Connext
- **What** is middleware?
  - Handles discovery, connection, failures, ...
  - Easy programming: simpler APIs
What is (RTI) DDS
Data Distribution Service

a real time communication technology standard for the Industrial Internet of Things
The DDS Family

- **Object Management Group Standards**
- **Data Distribution Service (DDS)**
  - API
  - QoS
- **Real-Time Publish Subscribe (RTPS)**
  - Data encoding
  - Interaction Protocol
  - On the Wire Format
- **Extensions:**
  - XTypes
  - Security
Publish/Subscribe

- **Paradigm shift:**
  - From “give me your information” to “send me your data have when you have more”
- **Applications specify what can provide and what are they interested in**
  - Middleware handles sending, reception and conversion
  - E.g. “I offer temperature data”, “I’m interested in pressure data”
- **Applications are matched by interests:**
## Data Centric Model

- **Data drive the communication**
  - Data type and content define the interactions
  - e.g. Temperature data

- **Topic is the exchange unit**
  - Name + **Type**
  - Samples are univocally identified by keys (like in DB)
  - QoS per publication: matched vs. offered

- **Global Data Space**
  - Applications publish topics to a global data space
  - Global shared cache
Data Centric Model

- Decentralized
- Acts as a distributed database/cache
- No servers involved

<table>
<thead>
<tr>
<th>Source</th>
<th>Event</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAV1</td>
<td>WARNING</td>
<td>FUEL_LOW</td>
</tr>
<tr>
<td>UAV2</td>
<td>INFO</td>
<td>LANDING</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>Longitude</th>
<th>Latitude</th>
<th>Altitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAV1</td>
<td>37.4</td>
<td>-120</td>
<td>500</td>
</tr>
<tr>
<td>UAV2</td>
<td>40.1</td>
<td>-23</td>
<td>433</td>
</tr>
<tr>
<td>UAV3</td>
<td>50</td>
<td>-1</td>
<td>100</td>
</tr>
</tbody>
</table>
Quality of Service
choose your ingredients and you are ready to go...

https://commons.wikimedia.org/wiki/File:Salad_bar-02.jpg
# Quality of Service (QoS)

<table>
<thead>
<tr>
<th>Quality of Service</th>
<th>Quality of service</th>
</tr>
</thead>
<tbody>
<tr>
<td>DURABILITY</td>
<td>USER_DATA</td>
</tr>
<tr>
<td>HISTORY</td>
<td>TOPIC_DATA</td>
</tr>
<tr>
<td>READER DATA LIFECYCLE</td>
<td>GROUP_DATA</td>
</tr>
<tr>
<td>WRITER DATA LIFECYCLE</td>
<td>PARTITION</td>
</tr>
<tr>
<td>LIFESPAN</td>
<td>PRESENTATION</td>
</tr>
<tr>
<td>ENTITY FACTORY</td>
<td>DESTINATION ORDER</td>
</tr>
<tr>
<td>RESOURCE LIMITS</td>
<td>OWNERSHIP</td>
</tr>
<tr>
<td>RELIABILITY</td>
<td>OWNERSHIP STRENGTH</td>
</tr>
<tr>
<td>TIME BASED FILTER</td>
<td>LIVELINESS</td>
</tr>
<tr>
<td>DEADLINE</td>
<td>LATENCY BUDGET</td>
</tr>
<tr>
<td>CONTENT FILTERS</td>
<td>TRANSPORT PRIORITY</td>
</tr>
</tbody>
</table>

**Volatility**
- User
- Presentation
- Redundancy
- Transport

**Infrastructure**
- Delivery
- Quality of Service
- User Data Lifecycle
- Group Data Lifecycle
- Partition Data Lifecycle
- Presentation Data Lifecycle

**Delivery**
- Entity Factory
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Reliability**
- Volatility
- Infrastructure
- Delivery
- Entity Factory
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Quality of Service**
- User Data
- Topic Data
- Group Data
- Partition Data
- Presentation Data
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**User Data Lifecycle**
- Volatility
- Infrastructure
- Delivery
- Entity Factory
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Group Data Lifecycle**
- Volatility
- Infrastructure
- Delivery
- Entity Factory
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Partition Data Lifecycle**
- Volatility
- Infrastructure
- Delivery
- Entity Factory
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Presentation Data Lifecycle**
- Volatility
- Infrastructure
- Delivery
- Entity Factory
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Entity Factory**
- Volatility
- Infrastructure
- Delivery
- User Data Lifecycle
- Group Data Lifecycle
- Partition Data Lifecycle
- Presentation Data Lifecycle
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Destination Order**
- Volatility
- Infrastructure
- Delivery
- User Data Lifecycle
- Group Data Lifecycle
- Partition Data Lifecycle
- Presentation Data Lifecycle
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Ownership**
- Volatility
- Infrastructure
- Delivery
- User Data Lifecycle
- Group Data Lifecycle
- Partition Data Lifecycle
- Presentation Data Lifecycle
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Ownership Strength**
- Volatility
- Infrastructure
- Delivery
- User Data Lifecycle
- Group Data Lifecycle
- Partition Data Lifecycle
- Presentation Data Lifecycle
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Liveliness**
- Volatility
- Infrastructure
- Delivery
- User Data Lifecycle
- Group Data Lifecycle
- Partition Data Lifecycle
- Presentation Data Lifecycle
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Latency Budget**
- Volatility
- Infrastructure
- Delivery
- User Data Lifecycle
- Group Data Lifecycle
- Partition Data Lifecycle
- Presentation Data Lifecycle
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Transport Priority**
- Volatility
- Infrastructure
- Delivery
- User Data Lifecycle
- Group Data Lifecycle
- Partition Data Lifecycle
- Presentation Data Lifecycle
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority

**Resource Limits**
- Volatility
- Infrastructure
- Delivery
- User Data Lifecycle
- Group Data Lifecycle
- Partition Data Lifecycle
- Presentation Data Lifecycle
- Destination Order
- Ownership
- Ownership Strength
- Liveliness
- Latency Budget
- Transport Priority
### Example: Reliable Alarm/Events

#### Quality of Service

<table>
<thead>
<tr>
<th>Volatility</th>
<th>Infrastructure</th>
<th>Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>DURABILITY</td>
<td>ENTITY FACTORY</td>
<td>TIME BASED FILTER</td>
</tr>
<tr>
<td>HISTORY</td>
<td>RESOURCE LIMITS</td>
<td>DEADLINE</td>
</tr>
<tr>
<td>READER DATA LIFECYCLE</td>
<td>LIFESPAN</td>
<td>CONTENT FILTERS</td>
</tr>
<tr>
<td>WRITER DATA LIFECYCLE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quality of service</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_DATA</td>
</tr>
<tr>
<td>TOPIC_DATA</td>
</tr>
<tr>
<td>GROUP_DATA</td>
</tr>
<tr>
<td>PARTITION</td>
</tr>
<tr>
<td>PRESENTATION</td>
</tr>
<tr>
<td>DESTINATION ORDER</td>
</tr>
<tr>
<td>OWNERSHIP</td>
</tr>
<tr>
<td>OWNERSHIP STRENGTH</td>
</tr>
<tr>
<td>LIVELINESS</td>
</tr>
<tr>
<td>LATENCY BUDGET</td>
</tr>
<tr>
<td>TRANSPORT PRIORITY</td>
</tr>
</tbody>
</table>

#### Quality of Service

- **DURABILITY**
- **HISTORY**
- **READER DATA LIFECYCLE**
- **WRITER DATA LIFECYCLE**
- **LIFESPAN**
- **ENTITY FACTORY**
- **RESOURCE LIMITS**
- **RELIABILITY**
- **TIME BASED FILTER**
- **DEADLINE**
- **CONTENT FILTERS**

#### Quality of Service

- **User**
- **Presentation**
- **Redundancy**
- **Transport**
## Example: Data Redundancy

<table>
<thead>
<tr>
<th>Quality of Service</th>
<th>Quality of Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>DURABILITY</td>
<td>USER_DATA</td>
</tr>
<tr>
<td>HISTORY</td>
<td>TOPIC_DATA</td>
</tr>
<tr>
<td>READER DATA LIFECYCLE</td>
<td>GROUP_DATA</td>
</tr>
<tr>
<td>WRITER DATA LIFECYCLE</td>
<td>PARTITION</td>
</tr>
<tr>
<td>LIFESPAN</td>
<td>PRESENTATION</td>
</tr>
<tr>
<td>ENTITY FACTORY</td>
<td>DESTINATION ORDER</td>
</tr>
<tr>
<td>RESOURCE LIMITS</td>
<td>OWNERSHIP</td>
</tr>
<tr>
<td>RELIABILITY</td>
<td>OWNERSHIP STRENGTH</td>
</tr>
<tr>
<td>TIME BASED FILTER</td>
<td>LIVELINESS</td>
</tr>
<tr>
<td>DEADLINE</td>
<td>LATENCY BUDGET</td>
</tr>
<tr>
<td>CONTENT FILTERS</td>
<td>TRANSPORT PRIORITY</td>
</tr>
</tbody>
</table>

**Column: User**

- Presentation
- Redundancy
- Transport

**Row: Quality of Service**

- Volatility
- Infrastructure
- Delivery

**Table: Quality of Service**

- **Quality of Service**
  - DURABILITY
  - HISTORY
  - READER DATA LIFECYCLE
  - WRITER DATA LIFECYCLE
  - LIFESPAN
  - ENTITY FACTORY
  - RESOURCE LIMITS
  - RELIABILITY
  - TIME BASED FILTER
  - DEADLINE
  - CONTENT FILTERS

- **Quality of service**
  - USER_DATA
  - TOPIC_DATA
  - GROUP_DATA
  - PARTITION
  - PRESENTATION
  - DESTINATION ORDER
  - OWNERSHIP
  - OWNERSHIP STRENGTH
  - LIVELINESS
  - LATENCY BUDGET
  - TRANSPORT PRIORITY
Ok.. but what about Lua?
Classical DDS Workflow

struct Position2D {
    long id; //@key
    double x;
    double y;
};

struct Temp {
    long id; //@key
    double value;
};

#include <...>
#define Position2D

Pos2DWriter::write
Pos2DReader::read

C/C++/Java/C#

<xml QoS>

QoS Settings

This process can be simplified even more: RTI Prototyper/Lua

Code ready to be built in +50 architectures: Linux, Windows, VxWorks, Integrity….
Lua Connector Workflow

Lua script

<xml QoS>
  <types>
    <scenario description>
  </types>

Lua Interpreter
Lua Connector Input & Output ports

Connector (DomainParticipant+)

config.xml

Input Data Port (DataReader+)

Output Data Port (DataWriter+)

instance

output()

write() / dispose() / unregister() / clear_members()

User Defined Structure

XML Config

input()

wait() / on()

read() / take()

samples[]
Lua Connector - Why?

- Scripting!
- Runs on all the (75+) architectures we support
- Simplifies API for Data-Centric Publish/Subscribe
  - Reduce boilerplate code
  - Easy to implement tests and demo
Lua & DDS: two ‘flavors’

● “Embedded’ in RTI DDS Prototyper
  ○ Provides the main loop
  ○ Execute the script
    ■ On timer
    ■ On data available
    ■ On start
    ■ On stop

● ‘Extending’ as a stand alone
  ○ In a Lua interpreter
Anatomy of a Publisher in Lua Connector

```
local rti = require('rti_dds_connector')
1. local c0 = rti:new_connector("MyParticipantLibrary::Zero", ".\Simple.xml");
2. local writer = connector.WRITER['MyPublisher::MyWriter']
3. writer.instance['message'] = "Hello I am Paul!"
4. writer:write()
```

1. Create a connector
2. Get the datawriter
3. Set the instance values
4. Write the sample
Anatomy of a Subscriber in Lua Connector

local rti = require('rti.dds_connector')
1. local c1 = rti:new_connector("MyParticipantLibrary::One","./Simple.xml")
2. local reader = connector.READER['MySubscriber::MyReader']
3. reader:take()
4. print(reader.samples[1].message)

1. Create a connector
2. Get the datareader
3. Take the sample(s)
4. Print a field
Example: Basic pub/sub

- **Objective**
  - In this example we show how to publish/subscribe to data
Example: History and Live changes

- Objective
  - In this example we show how the history qos works
Example: Durability

- **Objective**
  - Learn how to provide recent history to late joiners

- **Description**
  - A console application will receive the recent history published before it was started
Example: Filtering

- **Objective**
  - Learn how to filter data per subscriber

- **Description**
  - The console application will only receive the data matching a certain criteria
Thanks for your attention!

Any questions?

gianpiero@rti.com / @magopieri

We are hiring!!! Visit http://www.rti.com/company/careers.html or talk to me

References:
- For any question contact me or write on our forum: https://community.rti.com/forums/technical-questions