Lua Workshop 2016

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**

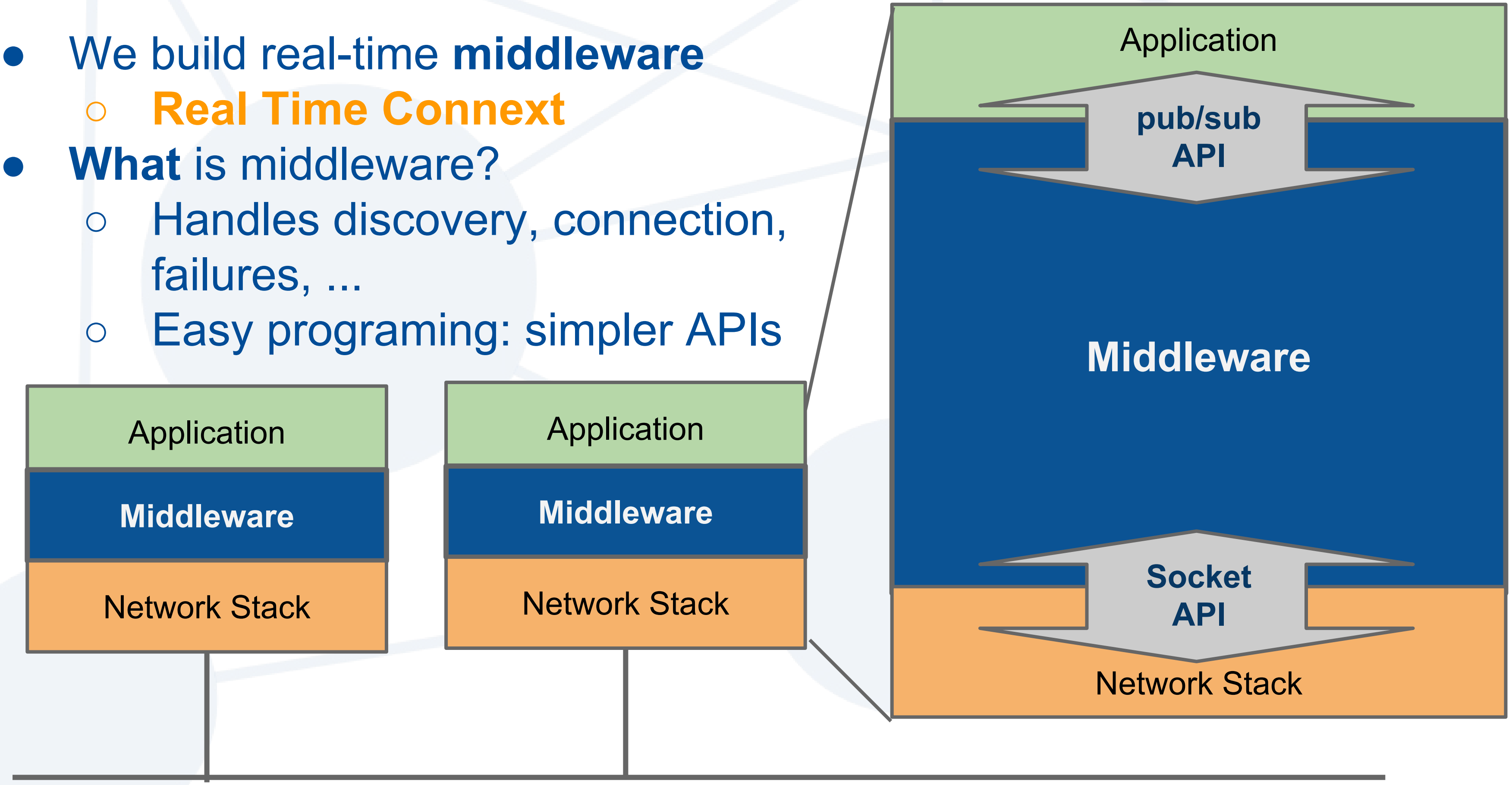
Who am I?

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 programing: simpler APIs



What is (RTI) DDS

https://en.wikipedia.org/wiki/Dentist#/media/File:US_Navy_030124-N-1328C-510_Navy_dentist_treats_patients_aboard_ship.jpg

DDS

Data Distribution Service

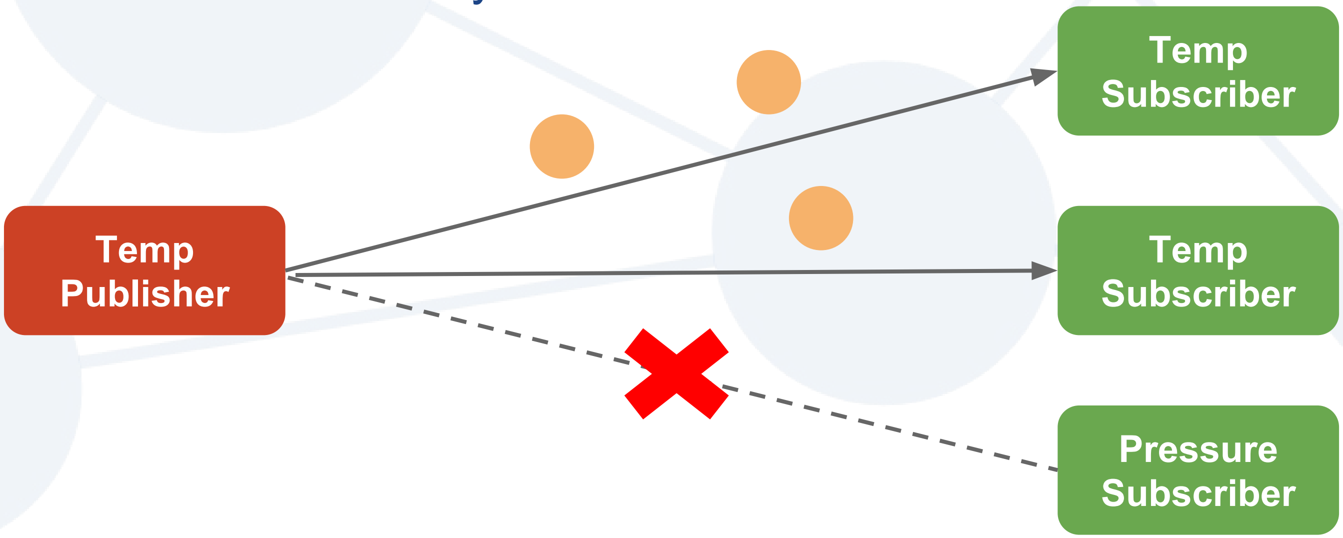
a **real time** communication technology
standard for the
Industrial Internet of Things

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

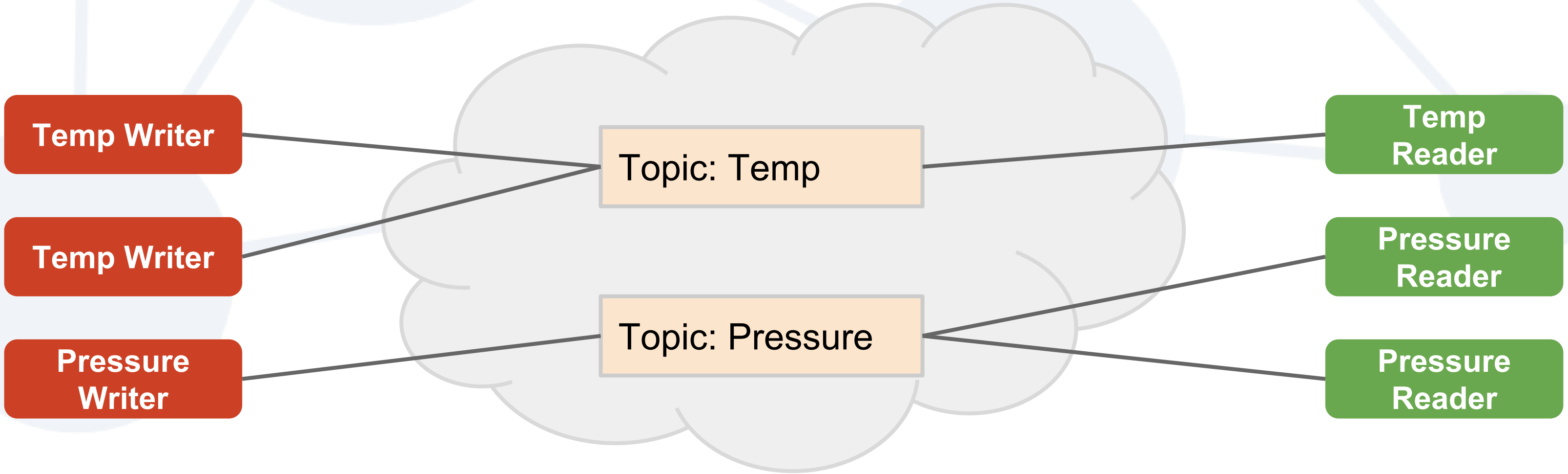
Publish/Subscribe

- 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:



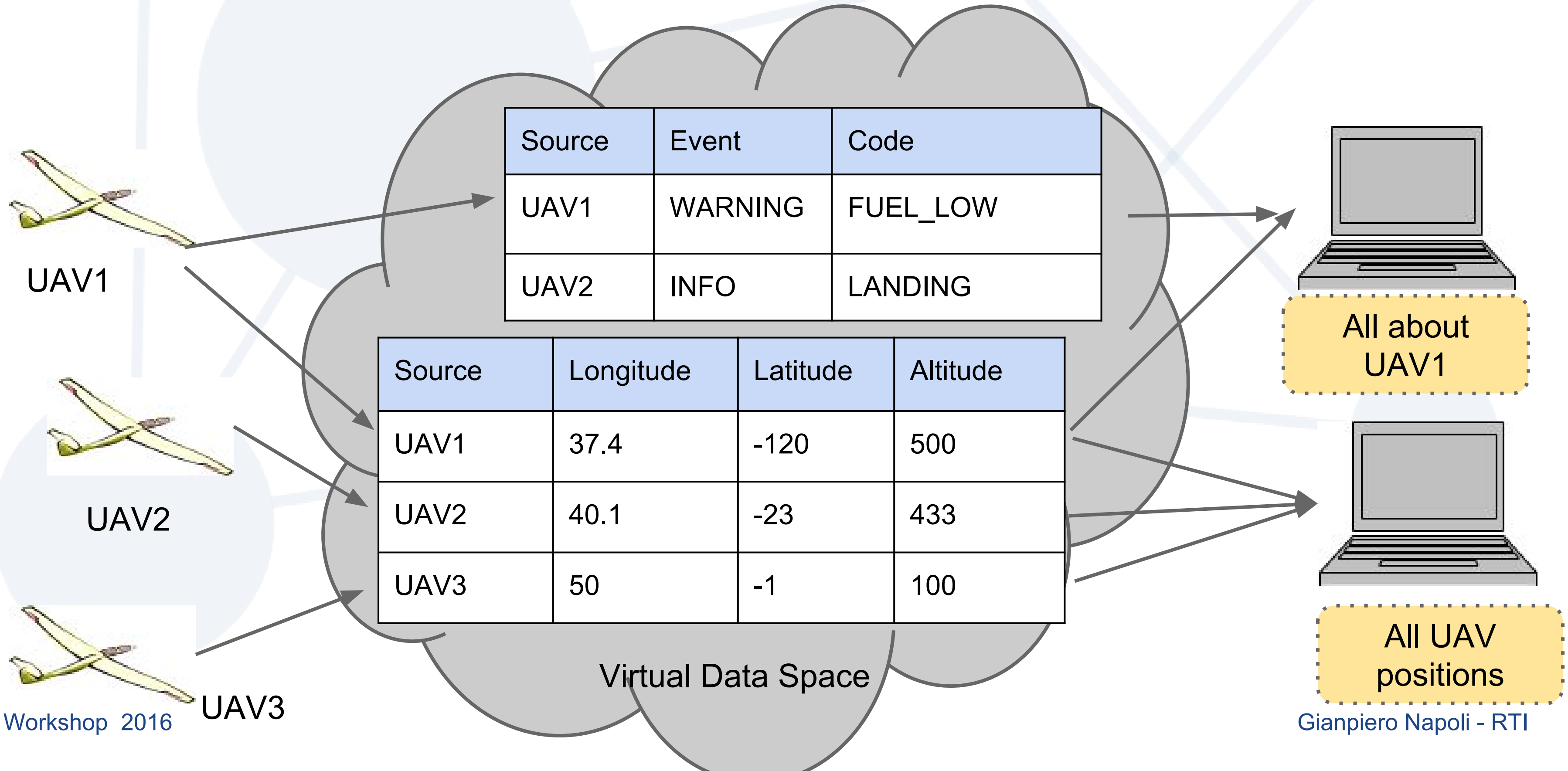
Data Centric Model

- 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

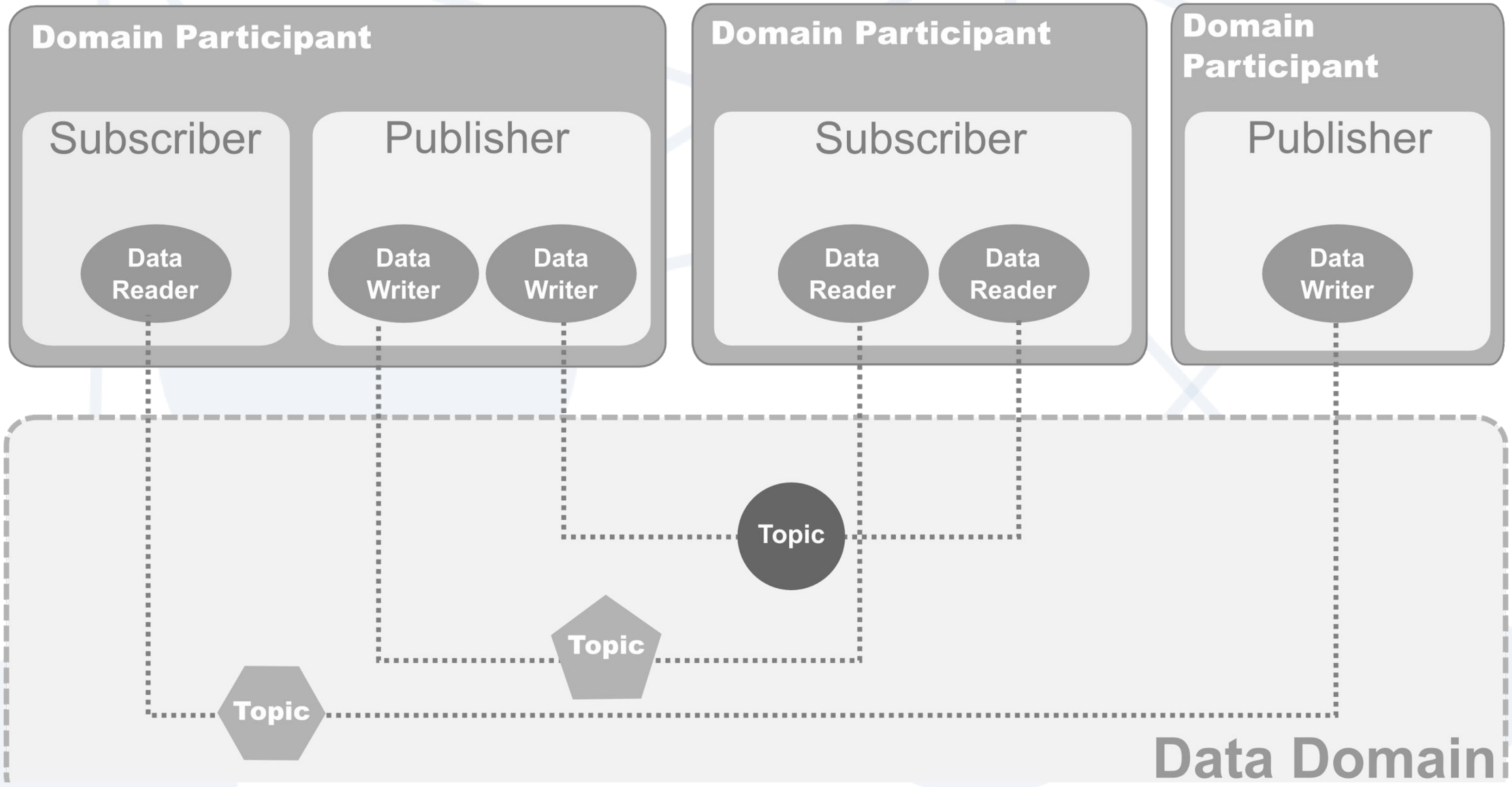
- Decentralized
- Acts as a distributed database/cache
- No servers involved



Source	Event	Code
UAV1	WARNING	FUEL_LOW
UAV2	INFO	LANDING

Source	Longitude	Latitude	Altitude
UAV1	37.4	-120	500
UAV2	40.1	-23	433
UAV3	50	-1	100

DDS Architecture



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)

	Quality of Service	Quality of service	
Volatility	DURABILITY	USER_DATA	User
	HISTORY	TOPIC_DATA	
	READER DATA LIFECYCLE	GROUP_DATA	
	WRITER DATA LIFECYCLE	PARTITION	Presentation
	LIFESPAN	PRESENTATION	
Infrastructure	ENTITY FACTORY	DESTINATION ORDER	
	RESOURCE LIMITS	OWNERSHIP	Redundancy
Delivery	RELIABILITY	OWNERSHIP STRENGTH	
	TIME BASED FILTER	LIVELINESS	
	DEADLINE	LATENCY BUDGET	Transport
	CONTENT FILTERS	TRANSPORT PRIORITY	

Example: Reliable Alarm/Events

Quality of Service	Quality of service
DURABILITY	USER_DATA
HISTORY	TOPIC_DATA
READER DATA LIFECYCLE	GROUP_DATA
WRITER DATA LIFECYCLE	PARTITION
LIFESPAN	PRESENTATION
ENTITY FACTORY	DESTINATION ORDER
RESOURCE LIMITS	OWNERSHIP
RELIABILITY	OWNERSHIP STRENGTH
TIME BASED FILTER	**LIVELINESS**
DEADLINE	LATENCY BUDGET
CONTENT FILTERS	TRANSPORT PRIORITY

Volatility

Infrastructure

Delivery

User

Presentation

Redundancy

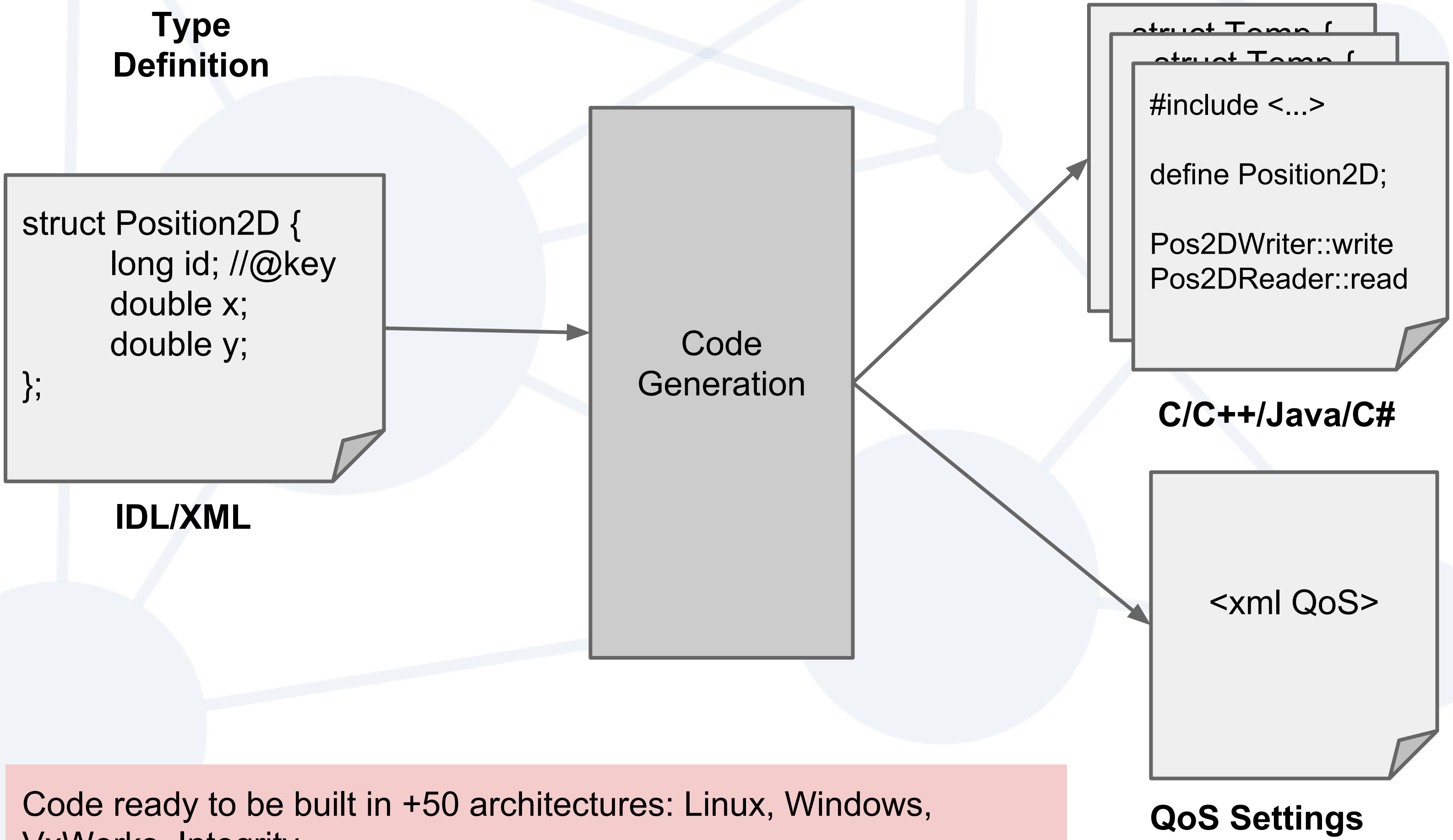
Transport

Example: Data Redundancy

	Quality of Service	Quality of service	
Volatility	DURABILITY	USER_DATA	User
	HISTORY	TOPIC_DATA	
	READER DATA LIFECYCLE	GROUP_DATA	
	WRITER DATA LIFECYCLE	PARTITION	Presentation
	LIFESPAN	PRESENTATION	
Infrastructure	ENTITY FACTORY	DESTINATION ORDER	
	RESOURCE LIMITS	**OWNERSHIP**	Redundancy
Delivery	**RELIABILITY**	**OWNERSHIP STRENGTH**	
	TIME BASED FILTER	**LIVELINESS**	
	DEADLINE	LATENCY BUDGET	Transport
	CONTENT FILTERS	TRANSPORT PRIORITY	

Ok.. but what about Lua?

Classical DDS Workflow



Type Definition

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

IDL/XML

Code Generation

```
struct Temp {
```

```
struct Temp {
```

```
#include <...>

define Position2D;

Pos2DWriter::write
Pos2DReader::read
```

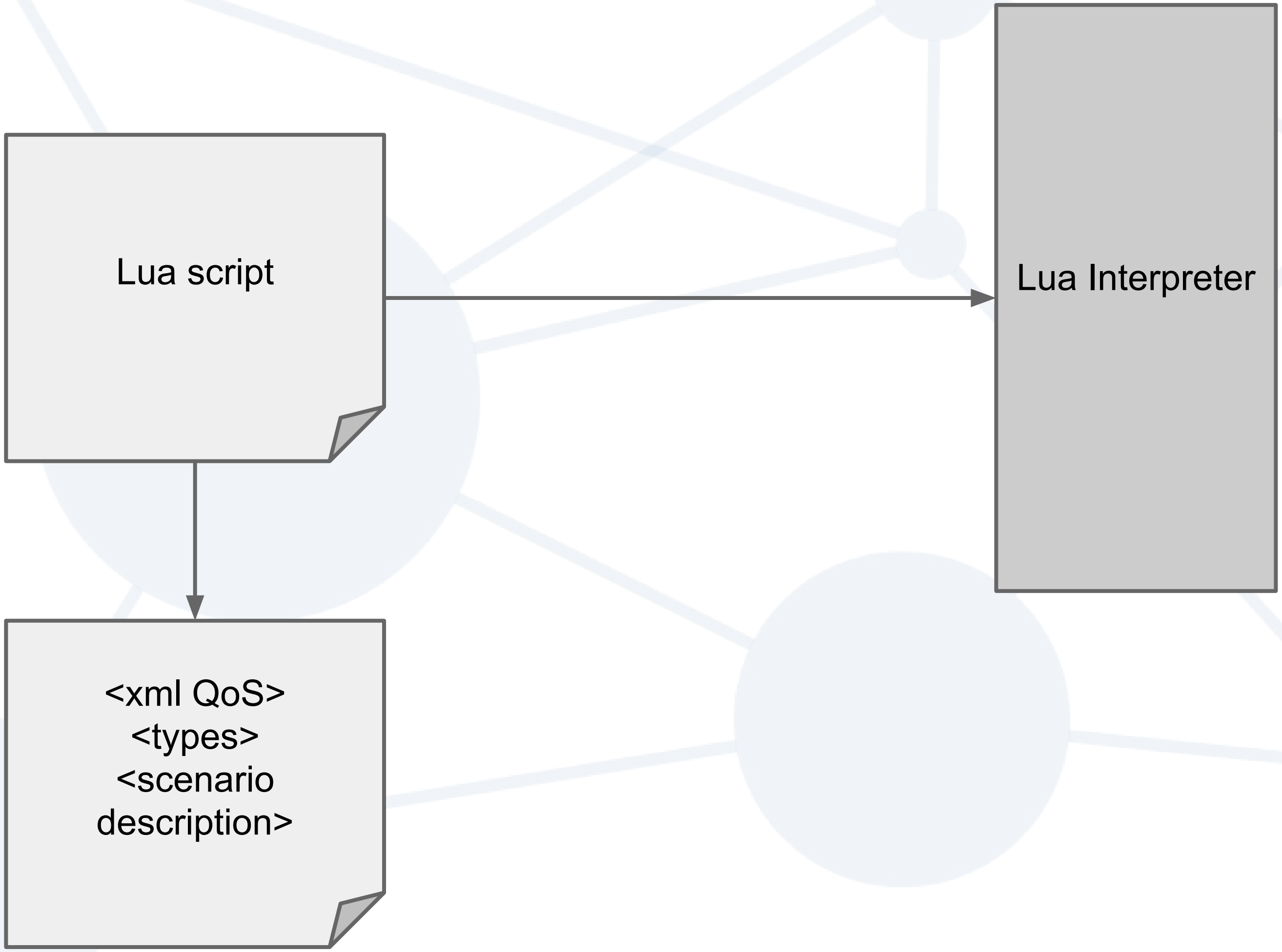
C/C++/Java/C#

```
<xml QoS>
```

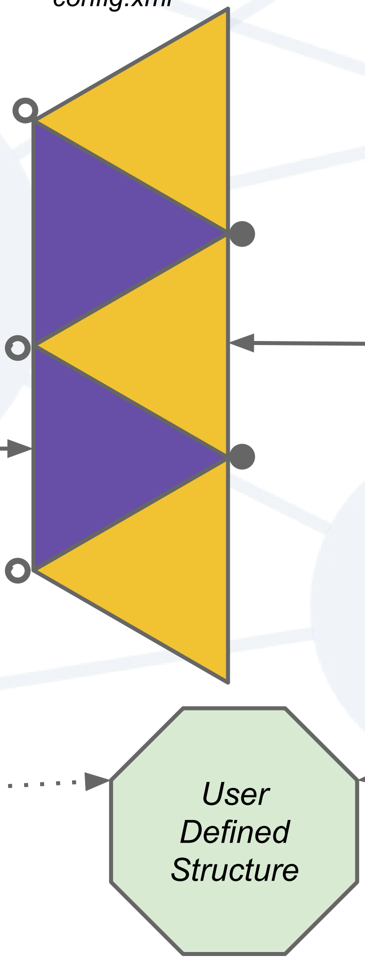
QoS Settings

Code ready to be built in +50 architectures: Linux, Windows, VxWorks, Integrity….
This process can be simplified even more: **RTI Prototyper/Lua**

Lua Connector Workflow



Lua Connector Input & Output ports



Connector (DomainParticipant+)

config.xml

Output
Data Port
(DataWriter+)

output()

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

instance

Input
Data Port
(DataReader+)

input()

wait() / on()

read() / take()

samples[]

XML Config

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

Hands On

Example: Basic pub/sub

- Objetive
  - In this example we show how to publish/subscribe to data
-

Example: History and Live changes

- Objetive
  - 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

- Objetive
  - 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:

- More info on RTI Prototyper With Lua here: https://community.rti.com/downloads/experimental/rti-prototyper-with-lua
- For any question contact me or write on our forum: https://community.rti.com/forums/technical-questions