Open APIs for Open Minds

Connecting to Robots

FIWARE Summit - Malaga - 15/Dec/2016

Jaime Martin Losa

eProsima CEO. I2ND Chapter – Advanced Middleware & Robotics

JaimeMartin@eProsima.com

FIWARE

Agenda

- FIWARE Advanced Middleware: When to use it
  - Fast RTPS
  - KIARA
- ROS2 (Robot Operating System)
- DDS/RTPS Quick Introduction
  - The Standard
  - Architecture
  - Shapes Demo
- Fast RTPS Hello World Example
- Connecting to ROS2 from FIWARE
  - Fast RTPS
  - FIROS2

FIWARE Advanced Middleware: When to use it?

FIWARE Advanced Middleware: When to use it

- Real Time Requirements
  - Latency measured in µSec
- High Throughput Requirements
  - Take advantage of Pub/Sub Architecture
- Low bandwidth, intermittent and unreliable datalinks
  - Radio networks
  - Wifi
- Many to Many communications
- Decoupled architectures
- Different QoS over different datalinks and performance requirements.
- Efficient Data Models

FIWARE Advanced Middleware: When to use it

- **eProsima Fast RTPS**
  - C++
  - Full RTPS (Real Time Publish Subscribe) implementation
  - RPC layer available through eProsima RPC over DDS
  - Robotics Adoption (ROS2)
  - Apache 2.0 License
- KIARA
  - Java
  - Complete RTPS implementation
    - No Support for large data (>64kb) yet
  - RPC included
  - LGPL License (Plans to migrate to Apache 2.0)
  - Interoperable with Fast RTPS

ROS (Robot Operating System)

ROS2: Robotics de facto Standard

- The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source. ROS has become a de facto standard for Robotic applications.
- OSRF Sponsors: **Bosh, DARPA, google, MathWorks, Nasa, Nissan, Qualcomm, rethink robotics, ROS-Industrial Consortium, Sandia National Laboratories, SICK, Willow Garage, Yujin Robot**

DDS/RTPS Quick Introduction

Introduction: Everything is distributed

- Enterprise Internet
- Internet of Things
- Cloud Computing
- Industry 4.0
- …

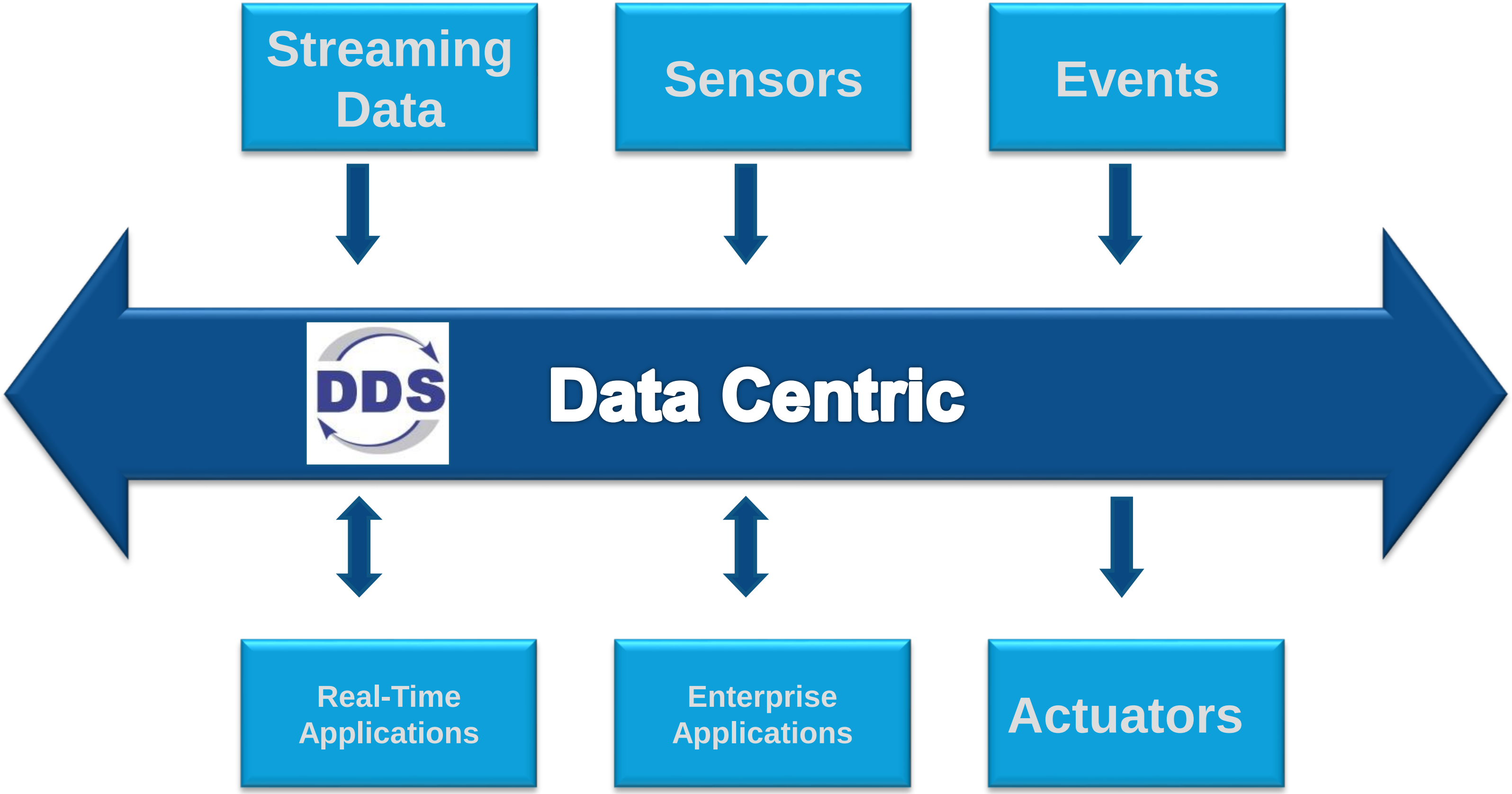
Participant
Participant
Network
Participant
Participant

- Next-generation systems needs:
  - Scalability
  - Integration & Evolution
  - Robustness & Availability
  - Performance
  - Security

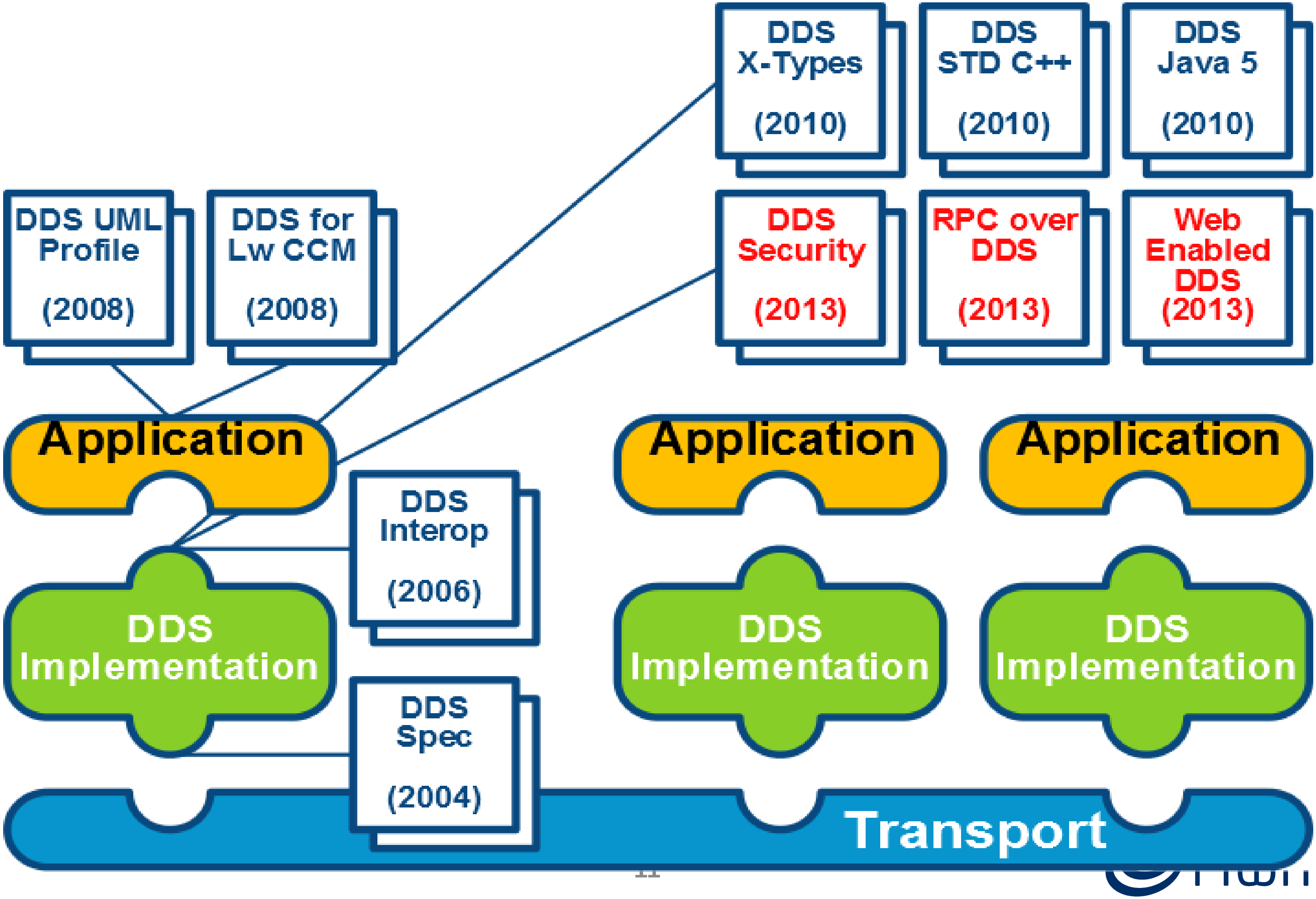
Challenge

- Everything is connected, and we should enable communication between the different nodes.
- And this means:
  - Common protocols
  - Common Data Types
  - Known interfaces
  - Different QoS over different datalinks and performance requirements.
  - Different comunications patterns.
  - Broad platform and programming language support.
  - Good Data Models!

DDS/RTPS: Standards-based Integration Infrastructure for Critical Applications



Family of Specifications



DDS X-Types (2010)

DDS STD C++ (2010)

DDS Java 5 (2010)

DDS UML Profile (2008)

DDS for Lw CCM (2008)

DDS Security (2013)

RPC over DDS (2013)

Web Enabled DDS (2013)

Application

Application

Application

DDS Interop (2006)

DDS Implementation

DDS Implementation

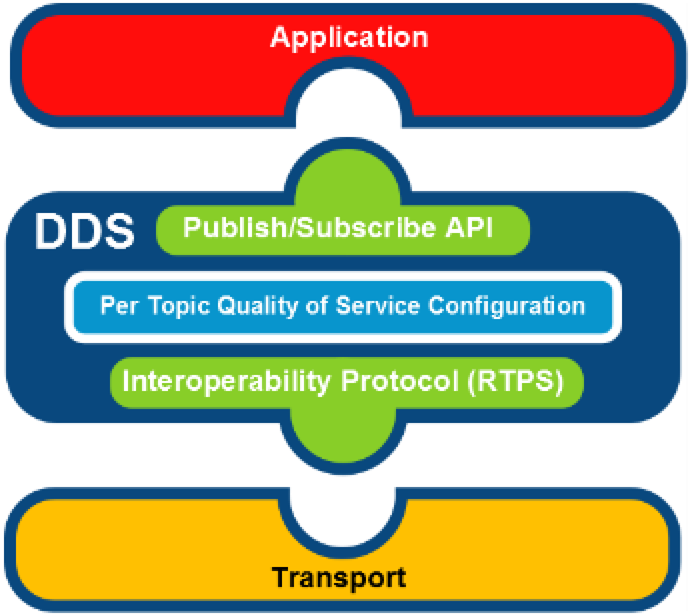
DDS Implementation

DDS Spec (2004)

Transport

Broad Adoption

- Vendor independent
  - API for **portability**
  - Wire protocol for **interoperability**
- Multiple implementations
  - 10 of API
  - 8 support RTPS
- Heterogeneous
  - C, C++, Java, .NET (C#, C++/CLI)
  - Linux, Windows, VxWorks, other embedded & real•time
- Loosely coupled


Application
DDS
Publish/Subscribe API
Per Topic Quality of Service Configuration
Interoperability Protocol (RTPS)
Transport

DDS adopted by key programs in Europe

- *European Air Traffic Control*
  - *DDS proposed for interoperate ATC centers*
- *Spanish Army*
  - *DDS is mandated for C2 Interoperability (ethernet, radio & satellite)*
- *UK Generic Vehicle Architecture*
  - *Mandates DDS for vehicle comm.*
  - *Mandates DDS-RTPS for interop.*

US-DoD mandates DDS for data-distribution

- DISR (formerly JTA)
  - DoD Information Technology Standards Registry
- US Navy Open Architecture
- Army, OSD
  - UCS, Unmanned Vehicle Control
- *SPAWAR NESI*
  - *Net-centric Enterprise Solutions for Interoperability*
  - *Mandates DDS for Pub-Sub SOA*

RTPS Adoption

- ROS (Robotic Operating System)
- FIWARE
  - EU R&D Software Platform
- Many Drone Companies
  - 3D Robotics
  - Magma UAVs
  - …

DDS Architecture

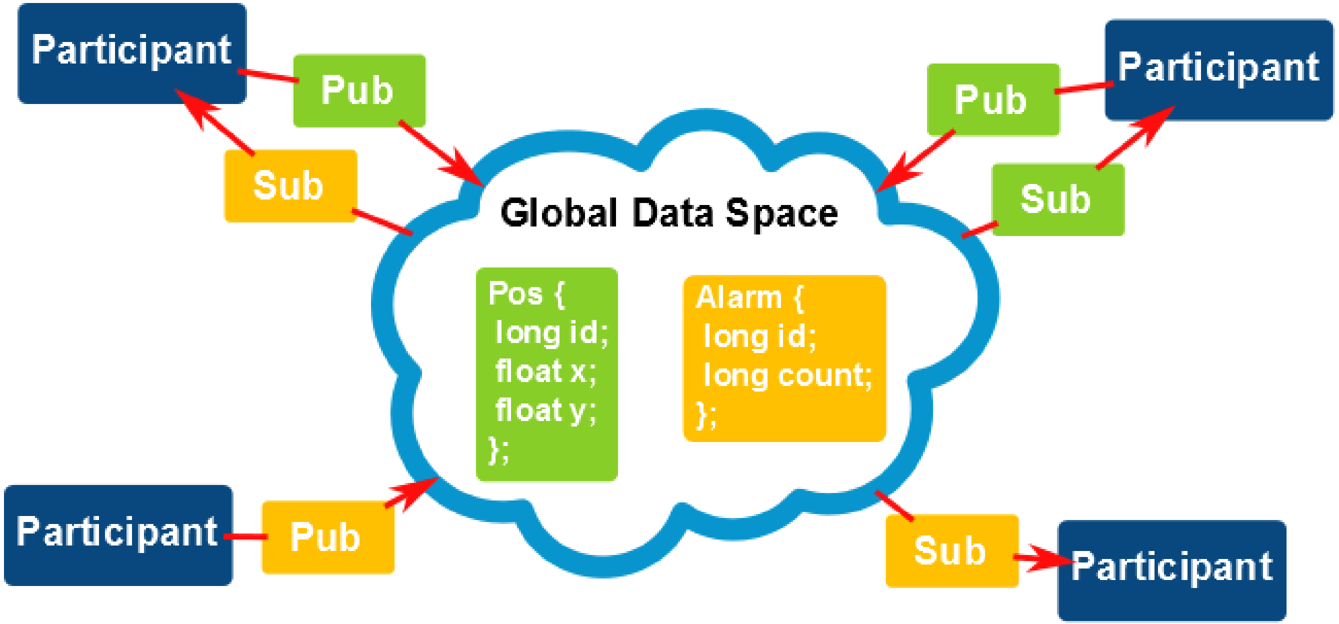
DDS

- DDS (Data Distribution Service for Real-Time Systems) is a OMG specification for a pub/sub data centric model (DCPS, Data Centric Publish/Subscribe) for Real-Time data comms in distributed systems.

- DDS is a networking middleware that:
  - Simplifies and Standardizes data flows in distributed real-time systems.
  - Provides robust comms (no single point of failure) and efficient (minimum latency)
  - Provides all kind of QoS to shape the data flows and deliver predictable results.

DDS

DDS uses the concept of **Global Data Space**. In this Space we define **topics** of data, and the **publishers** publish samples of these topics. DDS distributes these samples to all the **subscribers** of those topics. Any node can be a publisher or a subscriber.



Why DDS? Decoupled model

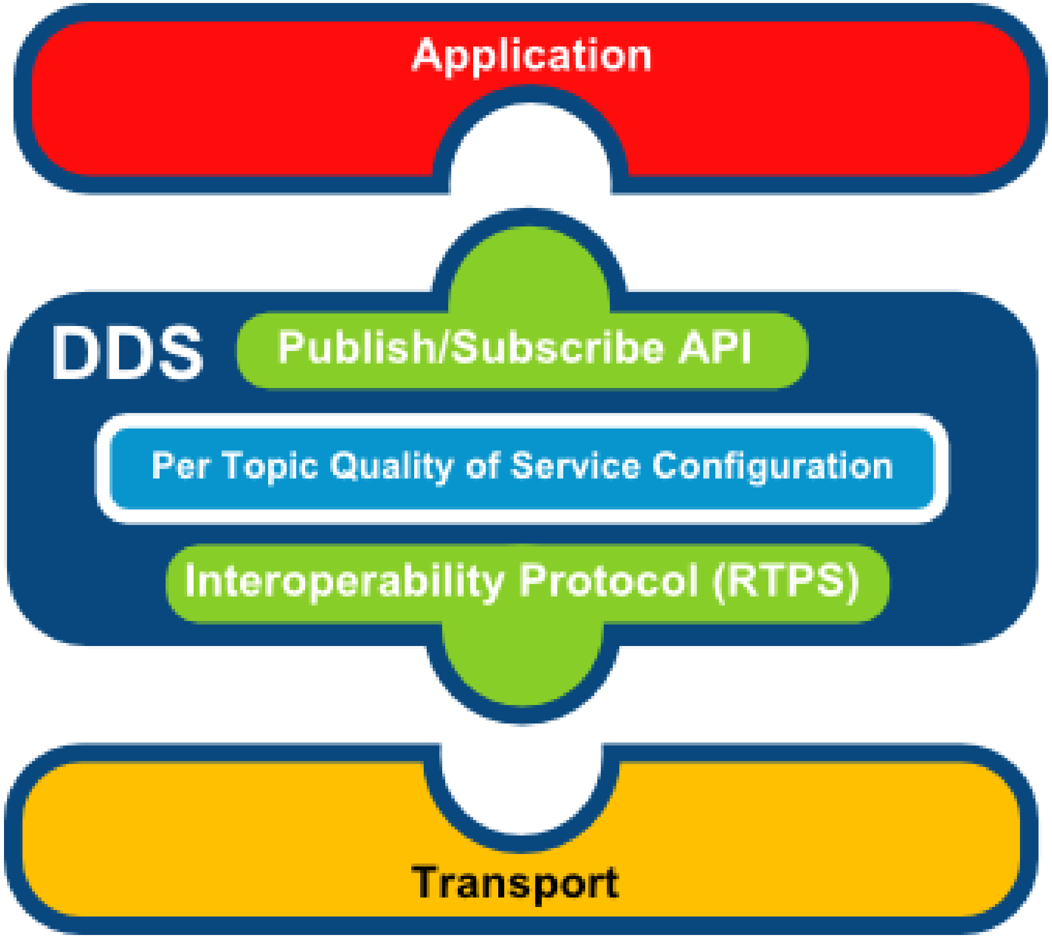
- **Space (location)**
  - Automatic Discovery ensures network topology independence
- **Redundancy:**
  - It is possible to configure redundant publishers and subscribers, primary/secundary and takeover schemas supported
- **Time:**
  - The reception of data does not need to be synchronous with the writing. A subscriber may, if so configured, receive data that was written even before the subscriber joined the network.
- **Platform:**
  - Applications do not have to worry about data representation, processor architecture, Operating System, or even programming language on the other side
- **Implementation:**
  - DDS Protocol is also an standard. Different implementations interoperate.

Why DDS? Fully configurable

	QoS Policy
Volatility	**DURABILITY**
	HISTORY
	READER DATA LIFECYCLE
	WRITER DATA LIFECYCLE
	LIFESPAN
Infrastructure	ENTITY FACTORY
	RESOURCE LIMITS
Delivery	**RELIABILITY**
	TIME BASED FILTER
	DEADLINE
	CONTENT FILTERS

	QoS Policy
User QoS	**USER DATA**
	TOPIC DATA
	GROUP DATA
Presentation	**PARTITION**
	PRESENTATION
	DESTINATION ORDER
Redundancy	**OWNERSHIP**
	OWNERSHIP STRENGTH
	LIVELINESS
Transport	LATENCY BUDGET
	TRANSPORT PRIORITY

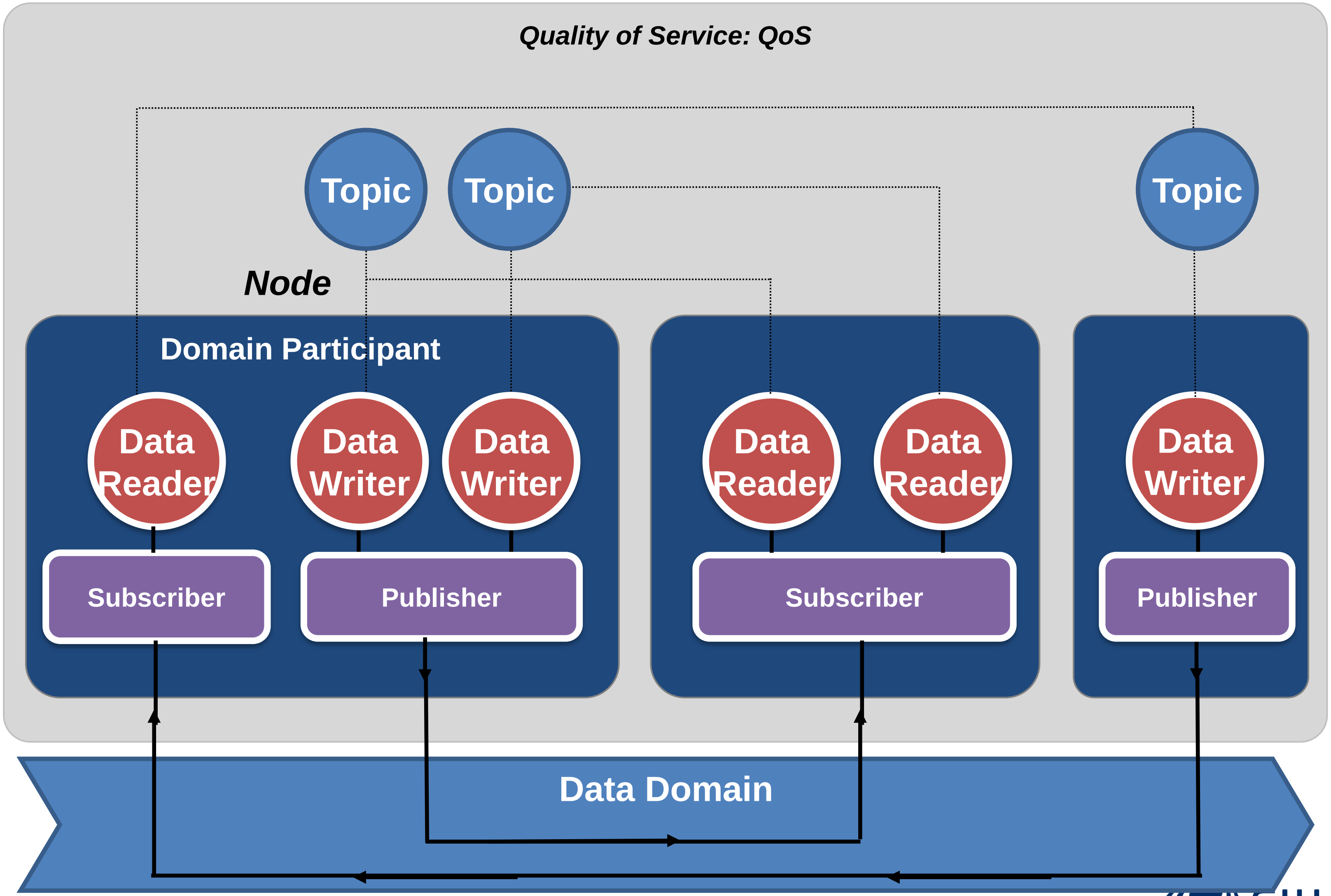
DDS Infrastructure


Application
DDS
Publish/Subscribe API
Per Topic Quality of Service Configuration
Interoperability Protocol (RTPS)
Transport

- Standard API for portability.
- RTPS can be implemented over any transport
- No central Broker/Service
- Different Comm channel per topic

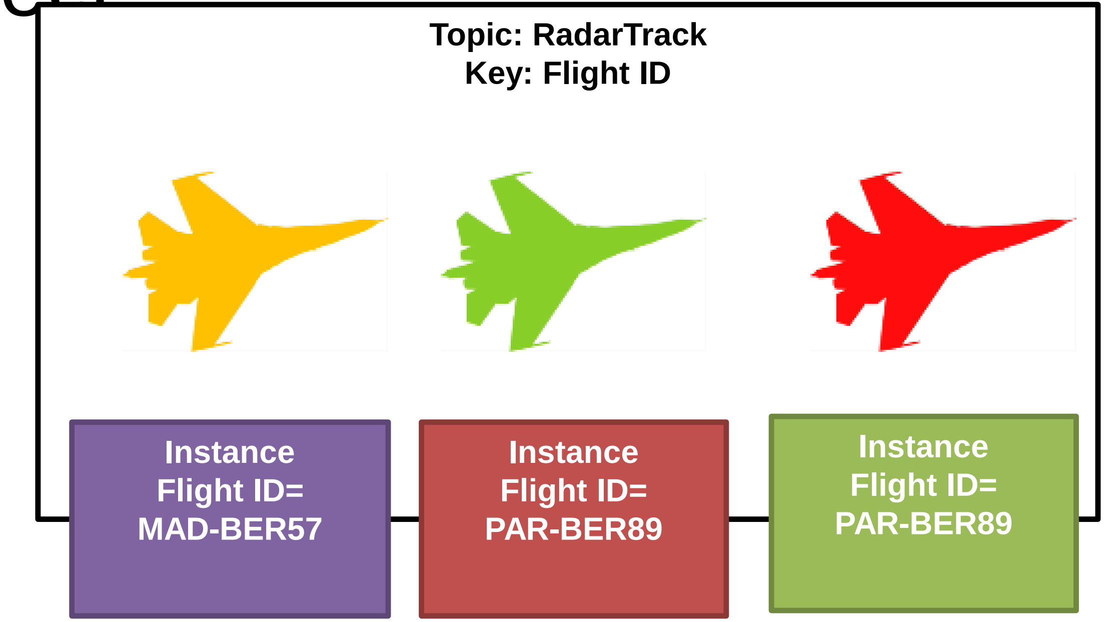
FIWARE

The DDS Model



Topics, Instances and Keys

- **Topic**: A set of similar objects, sharing a common Data Type
- **Instance**: A particular object of the set
- **Key**: Fields of the Data Type to identify an object



Topic: RadarTrack
Key: Flight ID

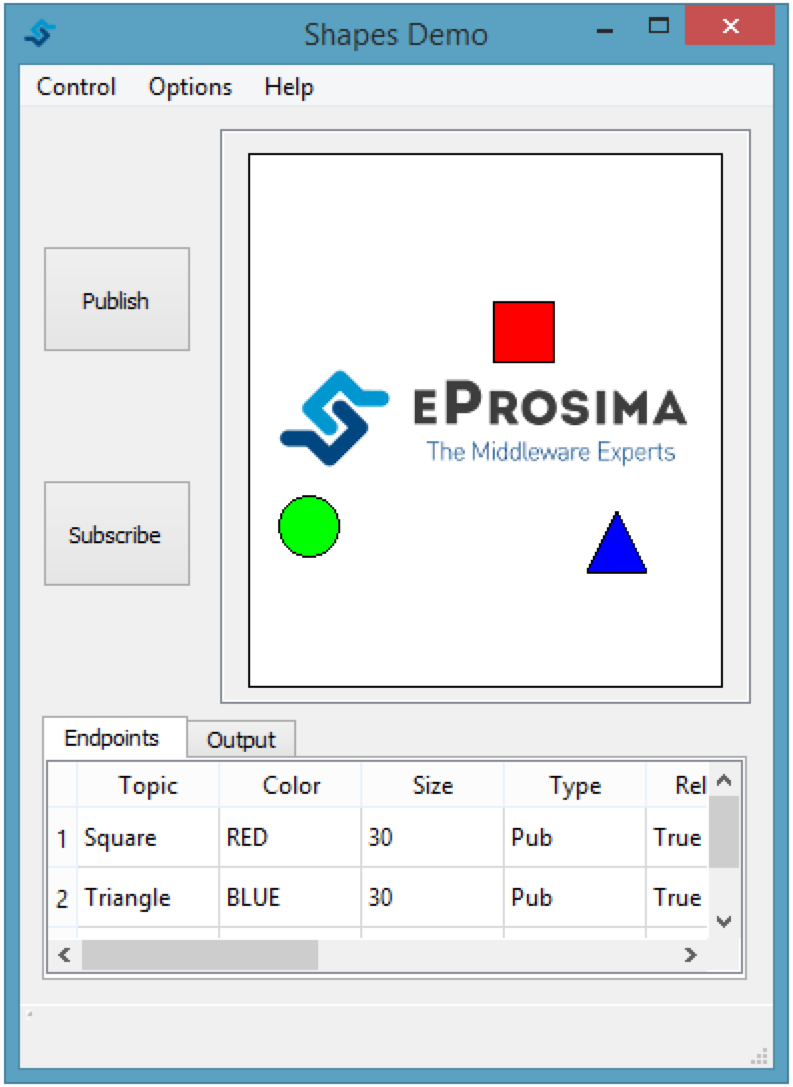
Instance
Flight ID=
MAD-BER57

Instance
Flight ID=
PAR-BER89

Instance
Flight ID=
PAR-BER89

Qos Applied by Instance.

Demo



	Topic	Color	Size	Type	Rel
1	Square	RED	30	Pub	True
2	Triangle	BLUE	30	Pub	True

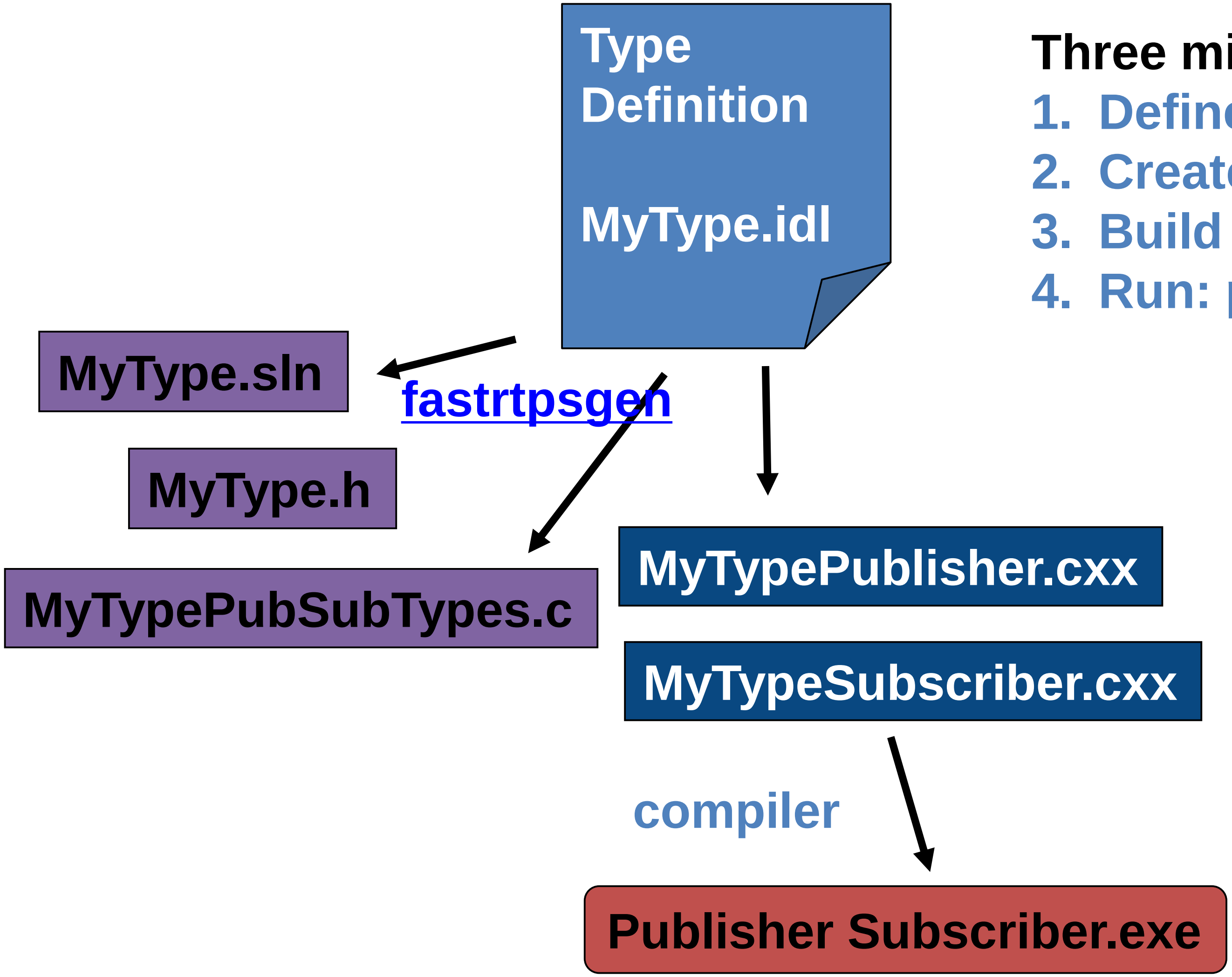
```
const long STR_LEN=24;
struct ShapeType {
  string<MSG_LEN> color; //@key
  long x;
  long y;
  long shapesize;
};
```

- **3 Topics:**
  - **Square, Circle, Triangle**
- **Color is the KEY**

Fast RTPS Hands On:

A Hello World

Hands-on Example (C++)



Type Definition

MyType.idl

fastrtpsgen

MyType.sln

MyType.h

MyTypePubSubTypes.c

MyTypePublisher.cxx

MyTypeSubscriber.cxx

compiler

Publisher Subscriber.exe

Three minutes to a running app!

1. Define your data
2. Create your project
3. Build
4. Run: publisher   subscriber

Example #1 - Hello World

We will use this data-type :

```
const long MSG_LEN=256;
struct HelloMsg {
 string<MSG_LEN> user; //@key
 string<MSG_LEN> msg;
};
```

Generate type support (for C++) [Windows]

```
fastrtpsgen HelloMsg.idl -example x64Win64VS2015\
                -replace -ppDisable
```

- Look at the directory you should see:
  - solution-x64Win64VS2015.sln
  - And Several other files…
- Open the Solution:
- Compile from visual studio

Execute the program [Windows]

- C++:
  - On one window run:
    - bin\x64Win64VS2015\HelloMsgPublisherSubscriberd.exe publisher
  - On another window run:
    - bin\x64Win64VS2015\HelloMsgPublisherSubscriberd.exe subscriber

- You should see the subscribers getting an empty string…

Writting some data

- Modify HelloMsgPublisher.cxx:

```
/* Main loop */
do
{
if(ch == 'y')
{
            st.msg() = std::string("Hello using cpp ") +
            std::to_string(msgsent);

            mp_publisher->write(&st);  ++msgsent;

            cout << "Sending sample, count=" << msgsent <<
                        ",send another sample?(y-yes,n-stop): ";
}
```

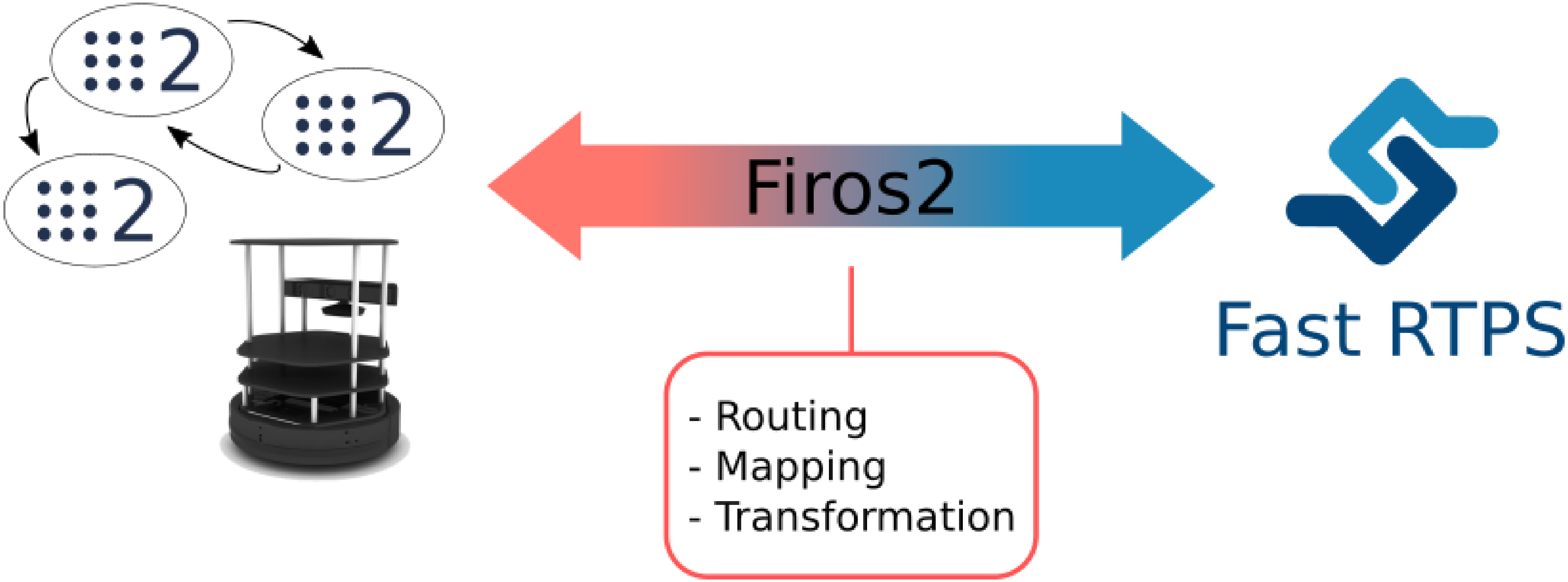
How to Get Data? (Listener-Based)

```
// Listener code
void HelloMsgSubscriber::SubListener::onNewDataMessage(Subscriber* sub)
{
        // Take data
        HelloMsg st;

        if(sub->takeNextData(&st, &m_info))
        {
                if(m_info.sampleKind == ALIVE)
                {
                        // Print your structure data here.
                        ++n_msg;
                        cout << "Sample received, count=" << n_msg << endl;
                        cout << " " << st.msg() << endl;
                }
        }
}
```

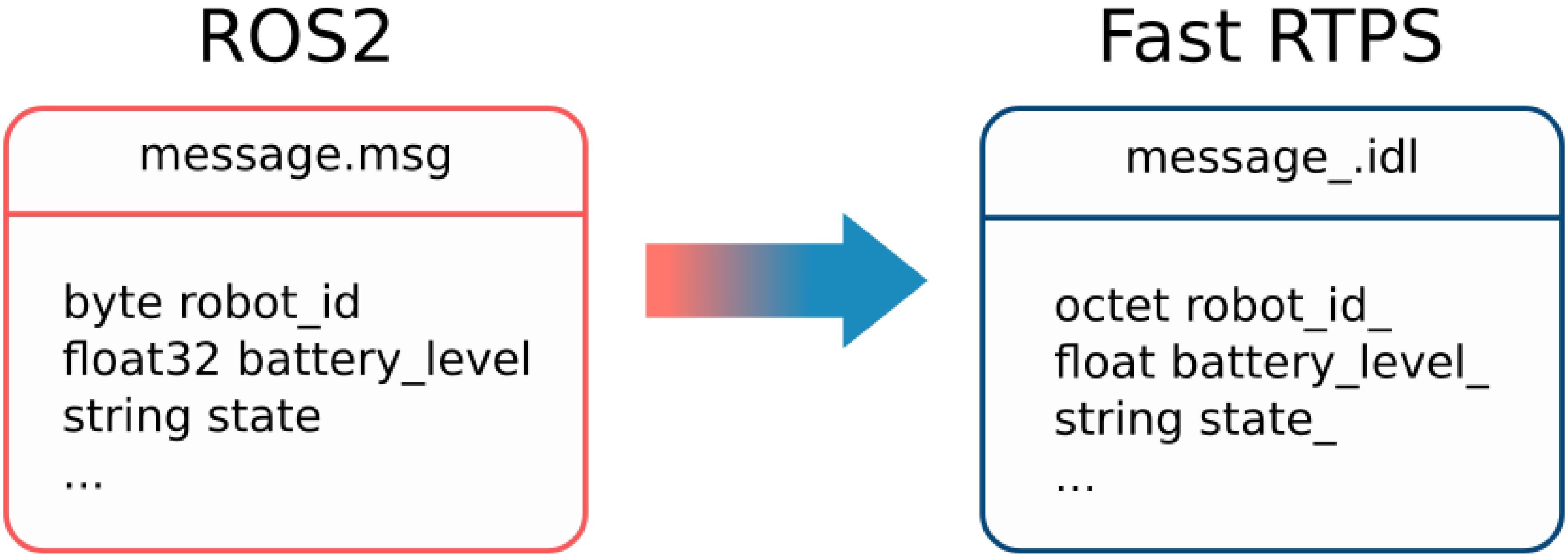
Connecting to ROS2 from FIWARE

FIROS2: ROS2 to Fast RTPS



- Routing
- Mapping
- Transformation

FIROS2: ROS2 to Fast RTPS



ROS2

message.msg

```
byte robot_id
float32 battery_level
string state
...
```

Fast RTPS

message_.idl

```
octet robot_id_
float battery_level_
string state_
...
```

FIROS2: Roadmap

- Bridge to Orion Context Broker

Want to know more?

- https://catalogue.fiware.org/enablers/fast-rtps
- https://catalogue.fiware.org/enablers/kiara-advanced-middleware

- www.eProsima.com
- Youtube: https://www.youtube.com/user/eprosima

- Mail: JaimeMartin@eProsima.com
- Phone: +34 607913745

- Twitter: @jaimemartinlosa
- http://es.slideshare.net/JaimeMartin-eProsima

Thank you!

http://fiware.org
Follow @FIWARE on Twitter

FIWARE