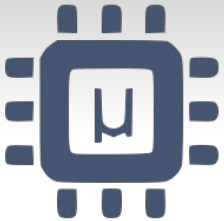
Bringing Micro XRCE-DDS & micro-ROS to PX4-based flying systems



PX4 Developer Summit Virtual 2020
07/07/2020

Jaime Martin Losa
eProsima - CEO
jaimemartin@eprosima.com

www.eprosima.com

Nuno Marques
Founder and Lead Developer
nuno.marques@dronesolutions.io

dronesolutions.io

Presentation Overview

- About eProsima
- Concepts
  - DDS
  - XRCE-DDS
  - Integration Service
  - Micro-ROS
- PX4-Fast RTPS bridge
- PX4-ROS 2 bridge
- Bridge new features and bug fixes
- PX4-DDS bridge
- PX4-ROS 2 bridge V2
  - Using the Integration Services or using micro-ROS
- Bridge migration

About eProsima

- Experts on middleware, focused on **DDS & ROS2**
- OMG Members – DDS Standard Contributors
- ROS2 TSC Members: Key ROS2 Contributors
- ROS Industrial Contributors

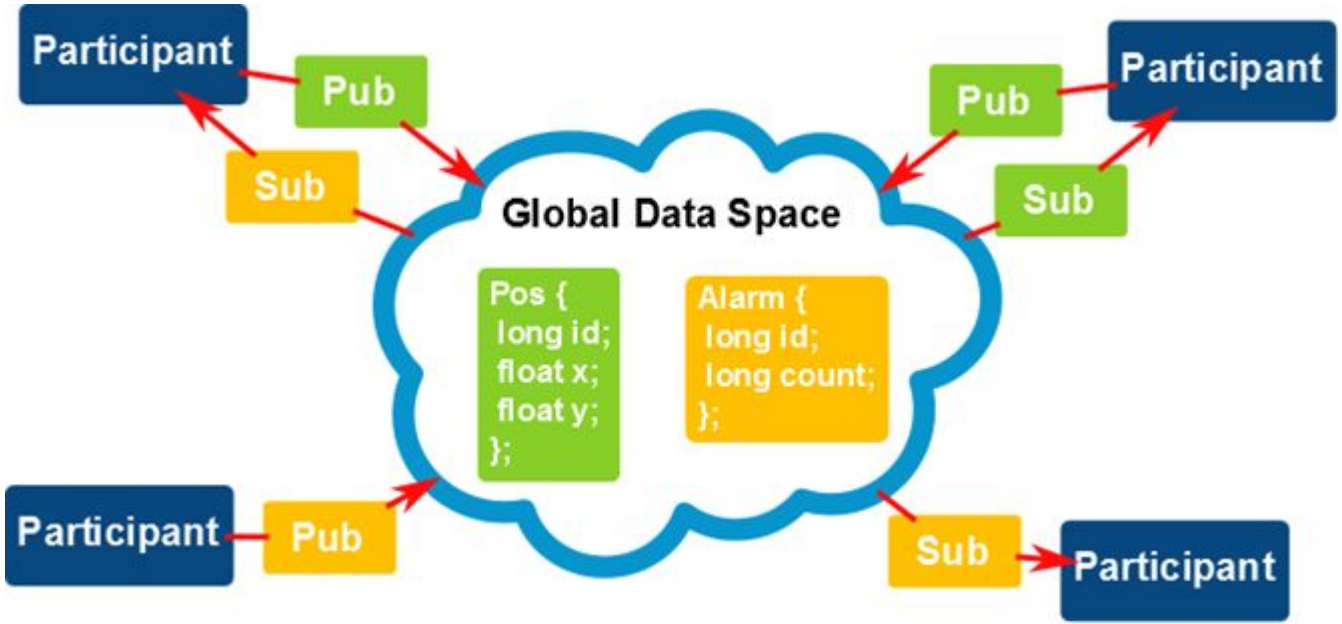
- **PX4 Contributors: Micro-RTPS Bridge**
  - uORB <-> DDS & ROS2

eProsima Products

- **eProsima Fast DDS:**
  - Data Distribution Service (DDS) implementation
  - **Adopted by ROS2**
- **eProsima Micro XRCE-DDS:**
  - DDS for eXtreme Resource Constrained Environments: Microcontrollers
  - **Base of Micro-ROS**
- **eProsima Integration Service:**
  - Connect DDS with other protocols, such as ROS1, ROS2, Web Sockets, Orion Context Broker, etc.
  - **Base of ROS Integration Service - SOSS**

DDS (& ROS2)

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.

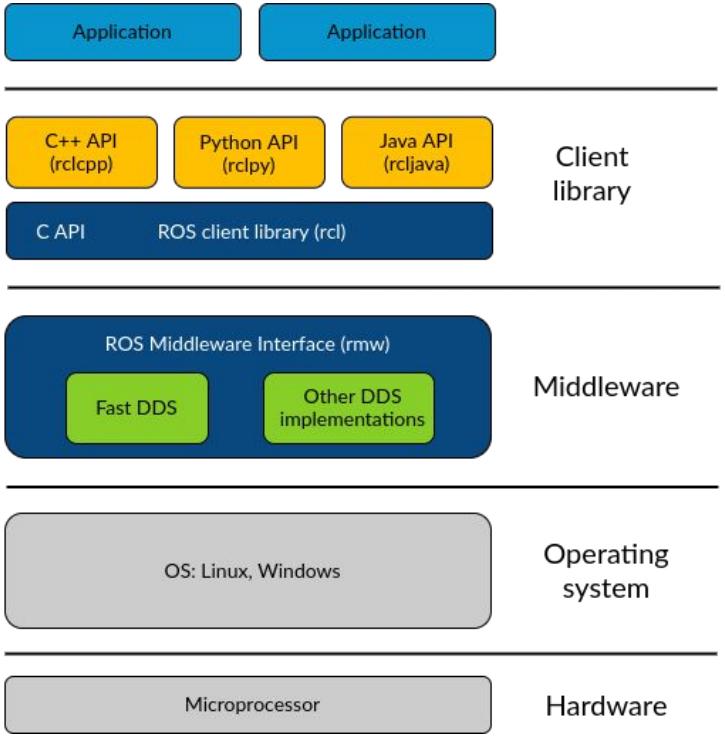


Fast DDS: Default Middleware for ROS2

DDS

Selected as Middleware implementation of the ROS Middleware Interface (rmw) layer

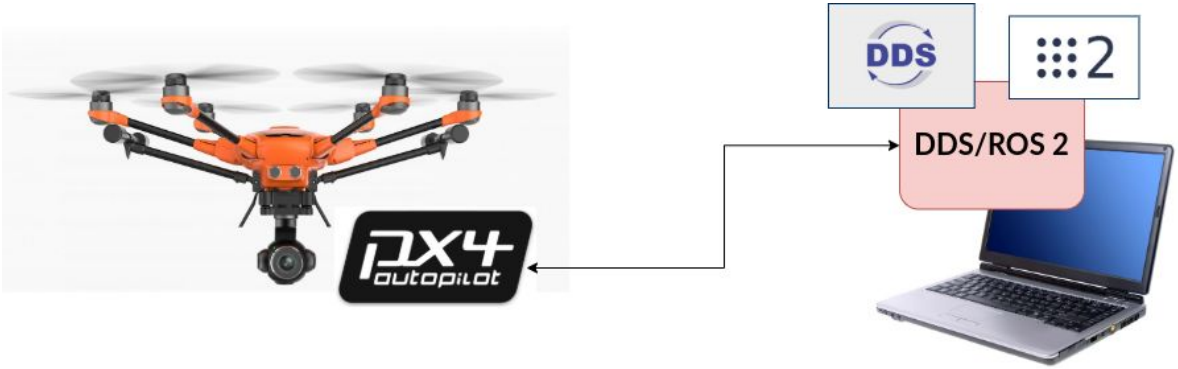
- 1-to-1 mapping between DDS and ROS 2 concepts
- Fast DDS: default implementation



Why bring DDS to PX4

Benefits of integrating PX4 with DDS

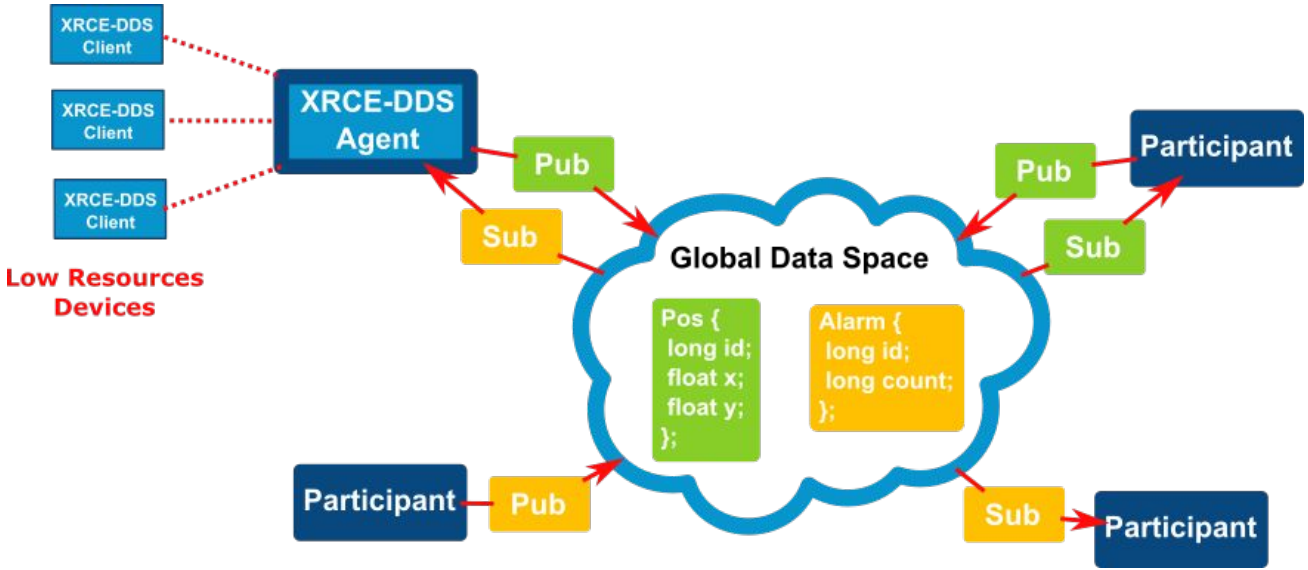
- High reliability, robustness, performance
  - Long success history in the defense and aerospace sector, including **many UAVs**
- Scalable architecture
- Eases integration of PX4 into ROS 2: straightforward many-to-many data exchange between PX4 internals using uORB and off-board components using DDS/ROS 2
- Optimally performing middleware for time-critical applications



XRCE-DDS: DDS for Microcontrollers

DDS-XRCE: Wire protocol for DDS on e**X**tremely **R**esource-**C**onstrained **E**nvironment.

Clients - XRCE entities on low-resource consumption devices.
Agent - XRCE entity connected with DDS global data space. Acts on behalf of Clients in the DDS world.



Benefits of Micro XRCE-DDS

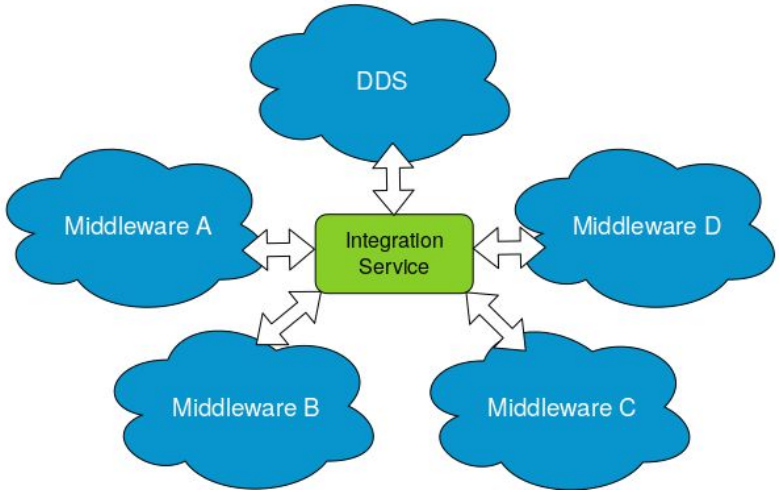
- **All the power of DDS available from the micro-controller**
  - **It is not a bridge, but a proxy.**
- **Compliant with DDS-XRCE standard**: general-purpose product in spite of dedicated bridge
- **Supports NuttX**: reference RTOS for the project
  - **FreeRTOS & Zephyr** too. Easy to port.
- External dependencies-free Client library: only depends on transport and a single POSIX time-related function
- Additional features:
  - Fragmentation: allows exchanging big-size messages
  - Services
  - IPv6
  - Best effort and reliable streams of communication
  - Time synchronization

eProsima Integration Service

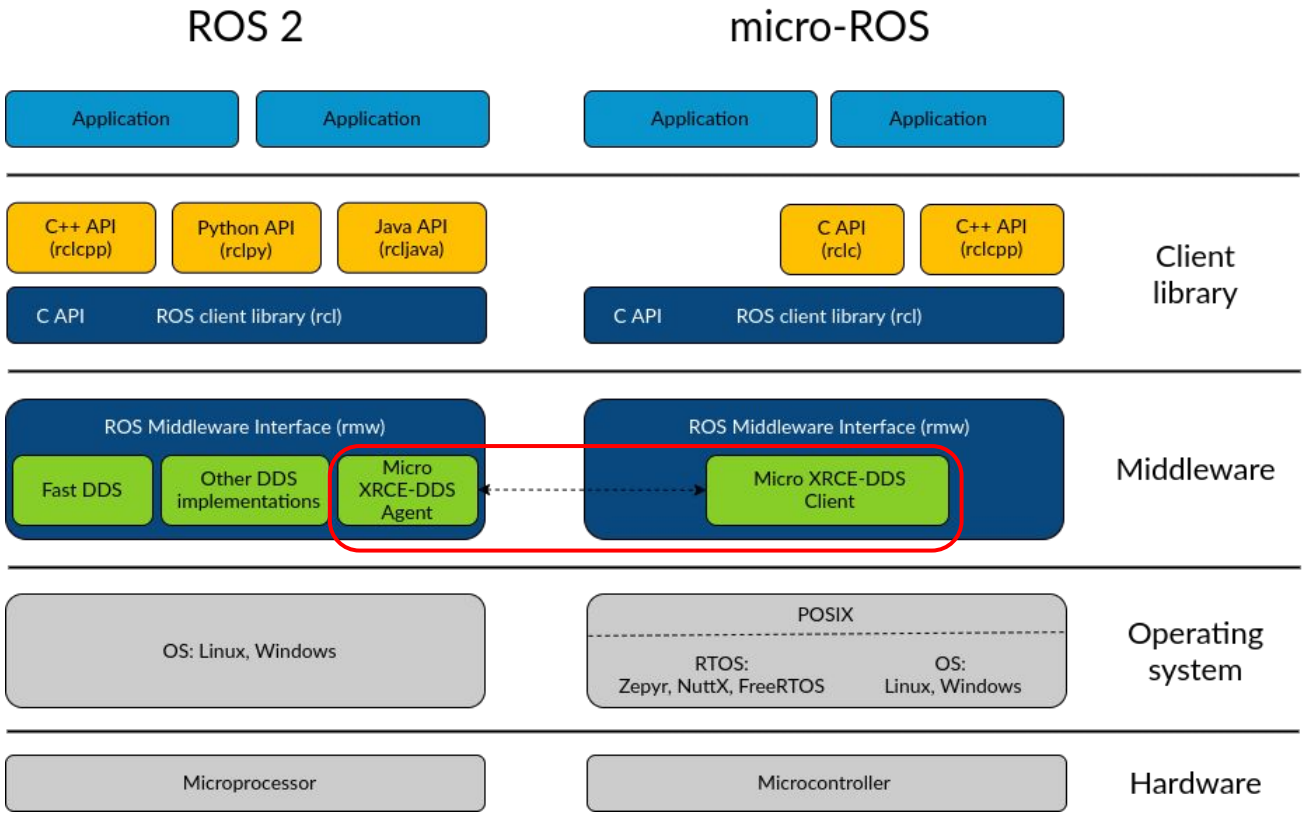
Enables communication between a DDS-based system and any other protocol.

Main Features:

- **Designed for DDS & ROS2**
- Supported by **eProsima & Open Robotics**
- Dynamic Data Representation
- Dedicated System Handles (SH) for external Middlewares
  - **WebSockets, ROS1, ROS2, Orion Context Broker**, etc.
  - User defined.
- WAN support (TCP Tunnels)

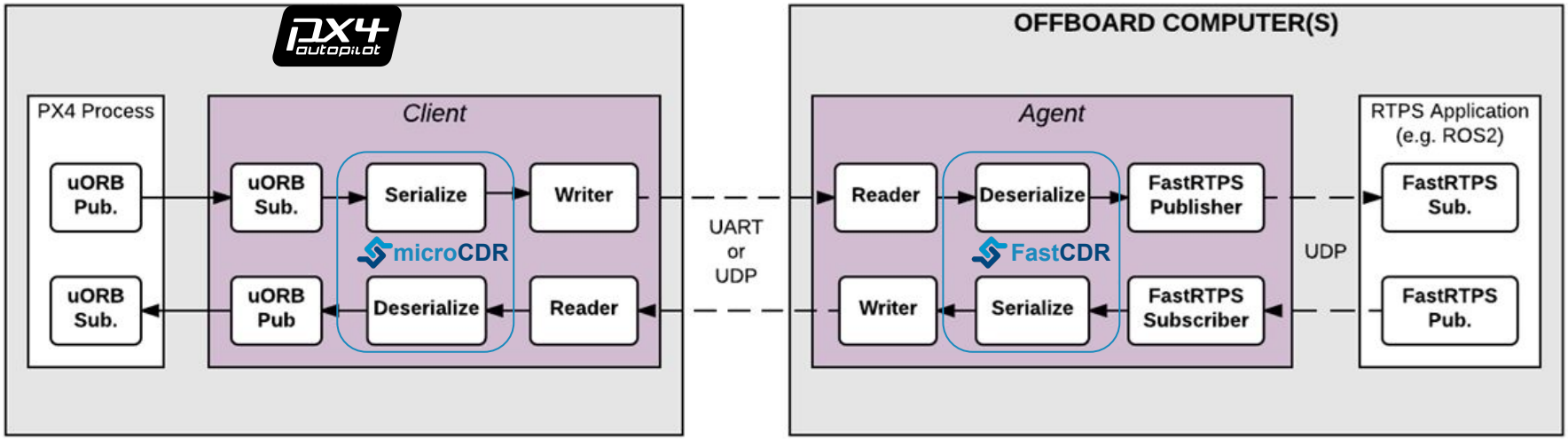


micro-ROS Architecture



PX4-Fast RTPS bridge

- aka PX4 **micro-RTPS bridge**
- First implementation in 2017



Jaime Martin Losa & Nuno Marques

PX4-Fast RTPS (DDS) bridge - how?

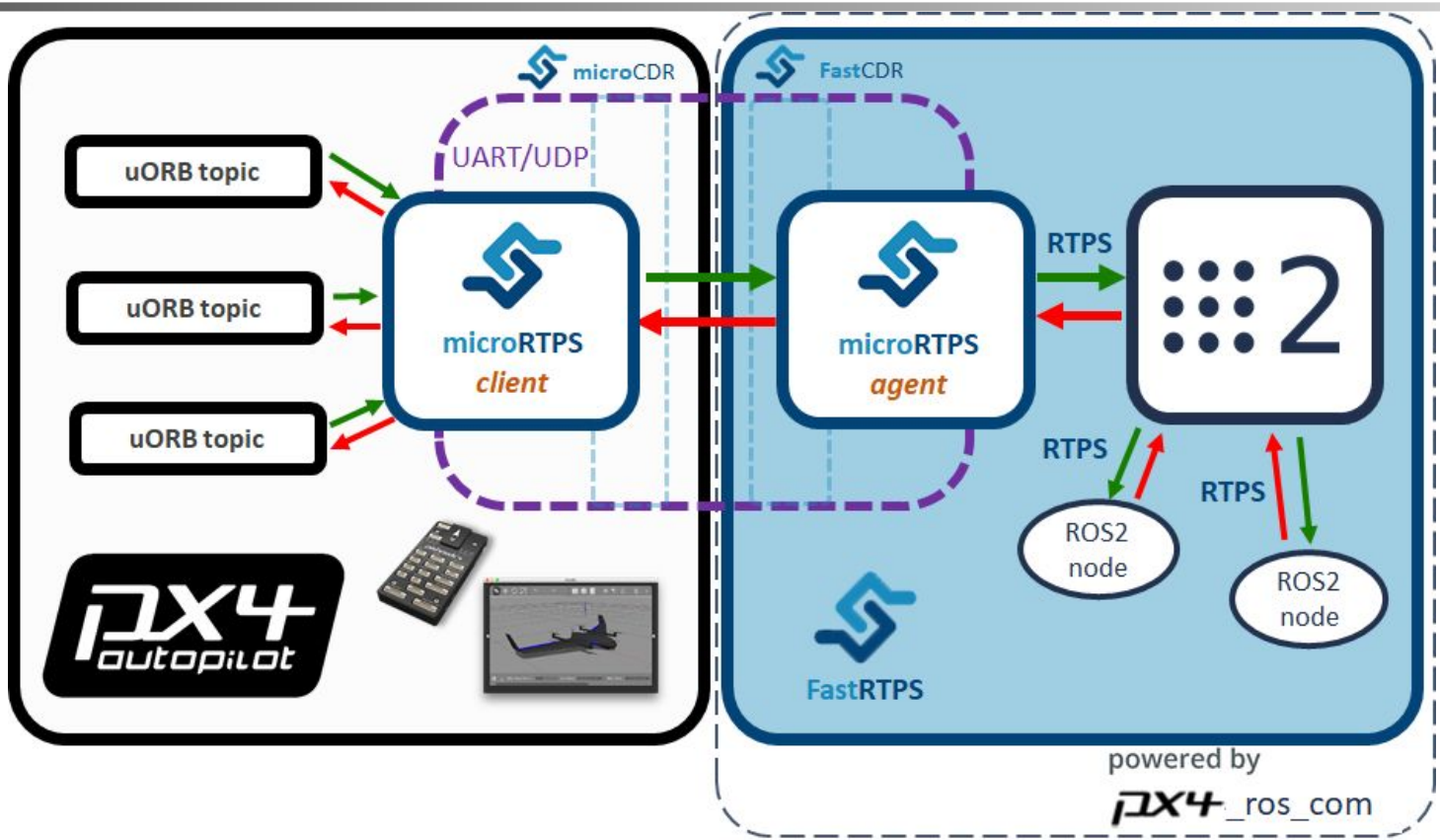
1. PX4 build process: `make px4_<target>_rtps`

2. Agent code build process – manually triggered
    a. Builds the agent application which **publishes and subscribes** to the ROS2/DDS topics
3. Listener application (optional) build:
    a. *Fast-RTPS-Gen* generates the required code to build an example for the specific onboard computer platform

    (`fastrtpsgen -example x64Linux2.6gcc <path_to_the_idl_file>`)

    b. Allows to launch an RTPS participant that subscribes to a specific a topic which type is set by the IDL file

PX4-ROS 2 bridge



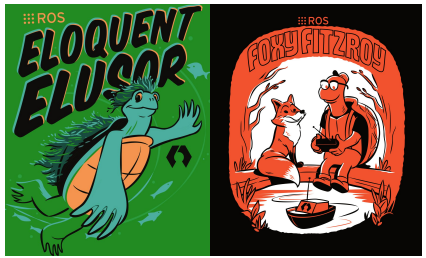
Jaime Martin Losa & Nuno Marques

PX4-ROS 2 bridge - how?

- `px4_ros_com`: https://github.com/PX4/px4_ros_com
  - Materializes the ROS2 side of PX4-Fast RTPS bridge, establishing a bridge between the PX4 autopilot stack through the micro-RTPS bridge and ROS 2;
  - With the aid of *Fast-RTPS-Gen*, **generates and allows building the agent side of the micro-RTPS bridge** to interface with Fast-RTPS (DDS) – and, by consequence, with ROS2
- `px4_msgs`: https://github.com/PX4/px4_msgs
  - Contains the **ROS2 message definitions** that represent the uORB counterparts in PX4
  - *PascalCased* naming with ROS specific types
  - Its build process generates:
    - the **IDL** files required for the **agent** code
    - the **typesupport** and **interface code** to be used by ROS 2 nodes

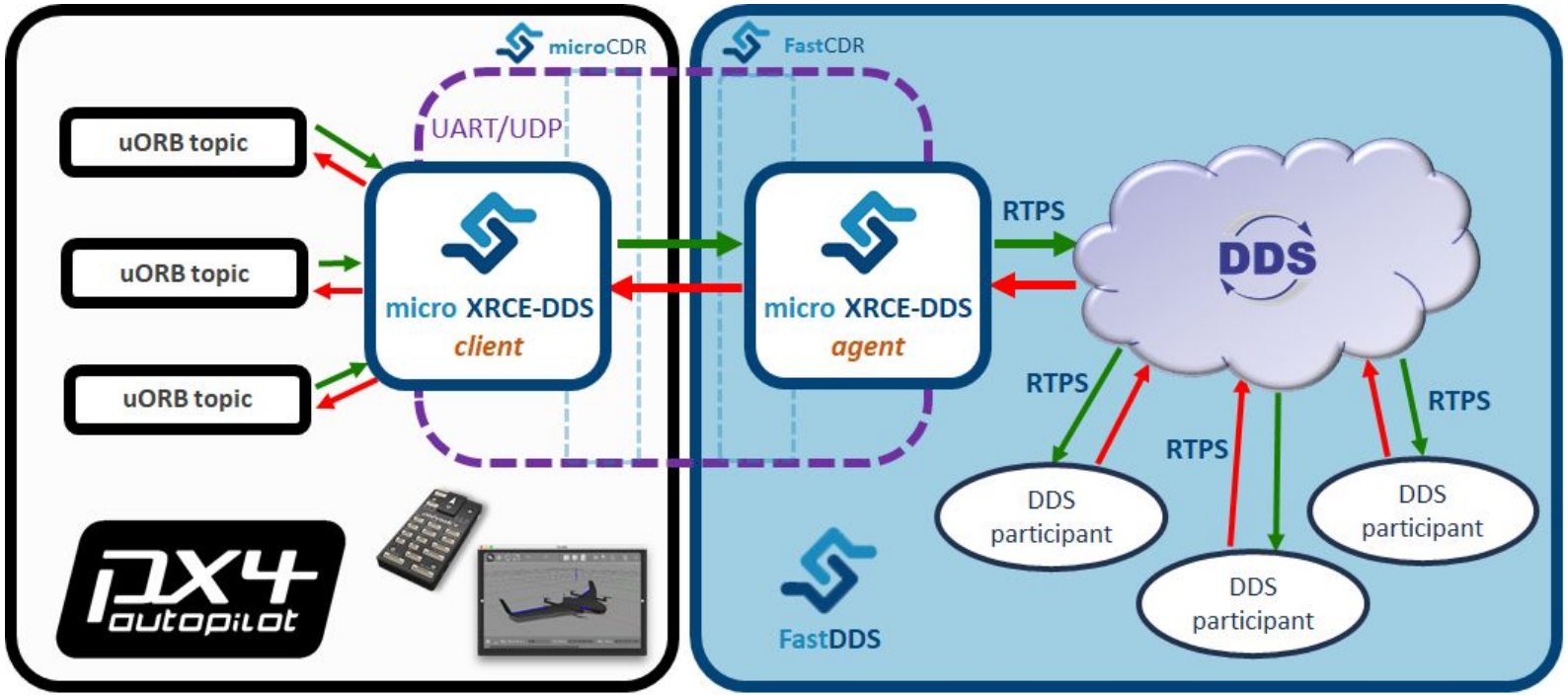
Bridge new features and bug fixes

- ROS2 typesupport for **Dashing**, **Eloquent** and **Foxy**
  - Not only on the templates, but also mirrored as a feature in *Fast-RTPS-Gen* (1.0.4)
- **Time synchronization**
  - Processed initialized by the agent which allows **synchronization of the messages timestamp** on both agent and client
- Participant and topic **filtering**
  - ex. avoid that a participant that is set to be publishing and subscribing the same type on the same topic doesn't get data that himself published
- Currently **supports *eProsima Fast-DDS*** (*Fast-RTPS* 2.0.0)
  - Keeps back-support to all versions since *Fast-RTPS* 1.6.0



PX4-DDS bridge

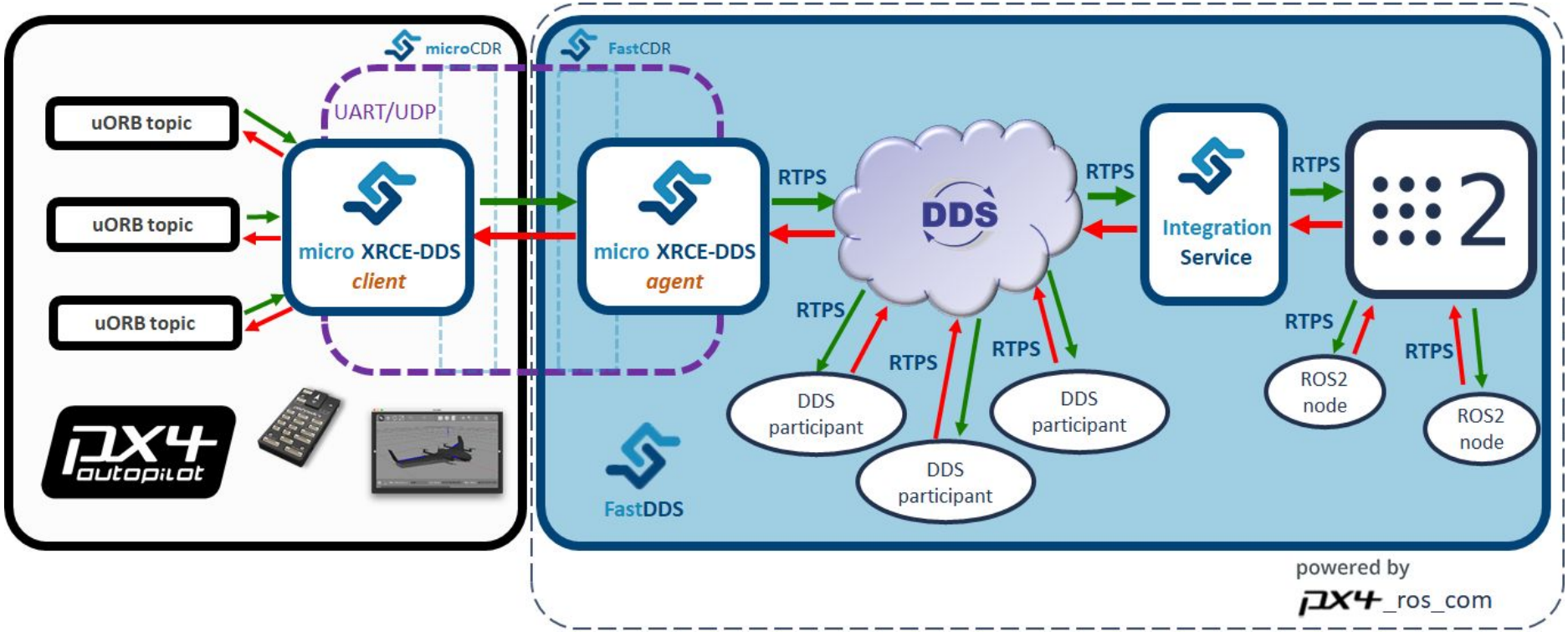
- aka PX4 **micro-DDS bridge**



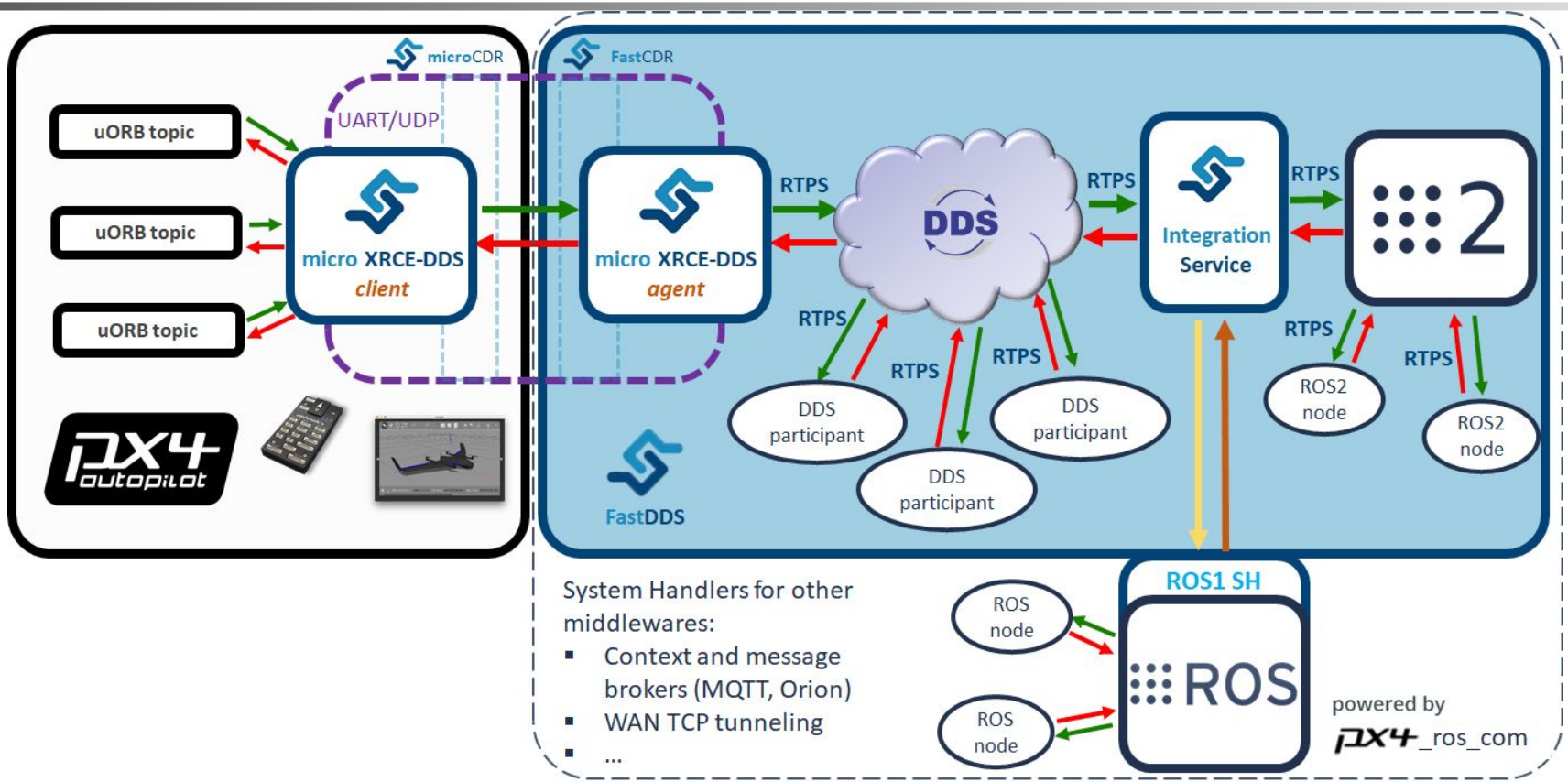
micro-RTPS to micro-DDS migration - why?

- Straightforward **many-to-many data exchange** between PX4 internals and DDS/ROS 2
- Takes full advantage of XRCE-DDS standard communication protocol from the OMG consortium
    - Brings **full DDS capabilities** to the the microcontroller
- The client library is **dynamic** and **static memory free**
- The client is built with a ***profiles*** concept
- Uses a generator tool specific to the client called *micro XRCE-DDS Gen* that simplifies the **generation of serialization and deserialization code** using micro-CDR
    - Uses as input **IDL files**
    - Removes the need for **custom templates**

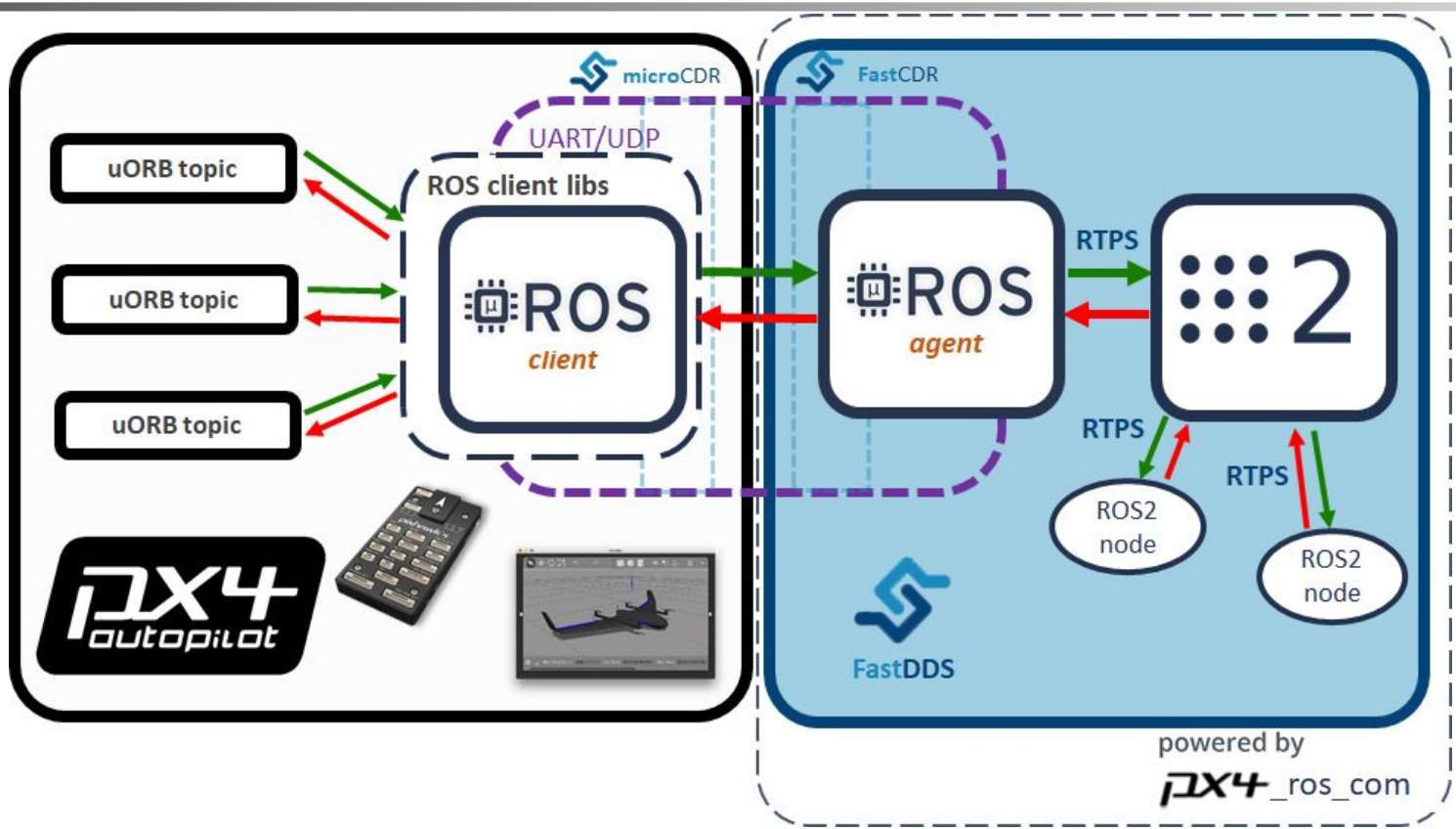
PX4-ROS 2 bridge V2



PX4-ROS 2 (and others) bridge V2



PX4-ROS 2 bridge V2 with micro-ROS

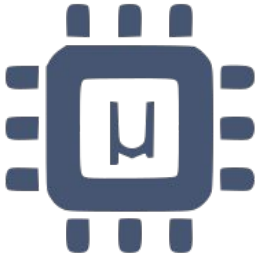


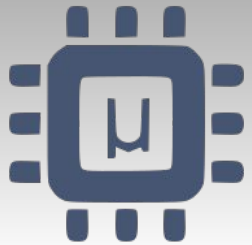
Bridge migration - how?

- Phase I - **micro-RTPS to micro-DDS migration**
  - Remove the current client templates
    - Use generated code from *micro XRCE-DDS Gen*
  - Code adjustments and integration with the ***micro XRCE-DDS*** client
    - uORB-to-IDL type conversions
    - Timesync and filtering
  - eProsima ***Integration Service*** configured and launched with *px4_ros_com*
  - Validation and documentation update
    - Unit and integration tests
    - Example applications (system monitoring, simple vehicle control, others)

Bridge migration - how?

- Phase II - **bring micro-ROS to PX4**
  - Integrate the ROS client libraries into the different target platforms
    - **STM32F7** to be the first to have it integrated
    - Others to follow after successful integration and validation
  - Code adjustments in the client side to allow both micro-DDS and micro-ROS client to live in the same code base, but built only when set
    - This will leverage the client libraries and allow the usage of the ROS API inside PX4
  - Validation and documentation update
    - Unit and integration tests
    - Example applications (system monitoring, simple vehicle control, others)





Thank you!

PX4 Developer Summit Virtual 2020
07/07/2020

Jaime Martin Losa
eProsima - CEO
jaimemartin@eprosima.com

www.eprosima.com

Nuno Marques
Founder and Lead Developer
nuno.marques@dronesolutions.io

dronesolutions.io