Developer Summit

ROS2-powered PX4

Nuno Marques – 20/06/19

Nuno Marques

- Upstream Mavlink and PX4 contributor
- Co-maintainer of MAVROS

- Independent consultant / SW Engineer
- Dronecrew
- dronesolutions.io
- TSC21

Agenda

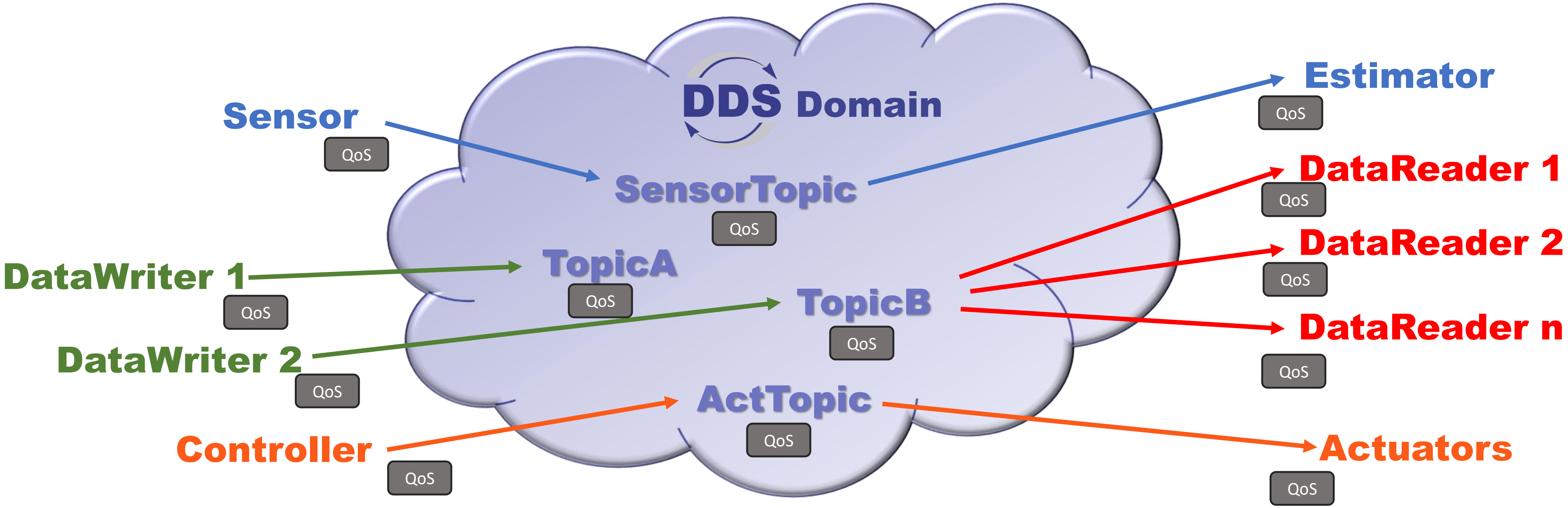
1. Introduction
   - DDS
   - Fast RTPS
   - ROS 2
2. Motivation
   - Mavlink vs RTPS/DDS
3. PX4 – Fast RTPS bridge
4. PX4 – ROS 2 bridge
5. *MAVROS* vs *px4_ros_com*
6. Future
7. Summary

- “Data Distribution Service (DDS™) is a middleware protocol and API standard for data-centric connectivity from the Object Management Group® (OMG®)”

www.dds-foundation.org

- Integration of all system components in a common infrastructure that provides:
  - High confidence and reliability
  - Data connectivity and low-latency
  - Scalable architecture
- Targets **mission-critical** applications

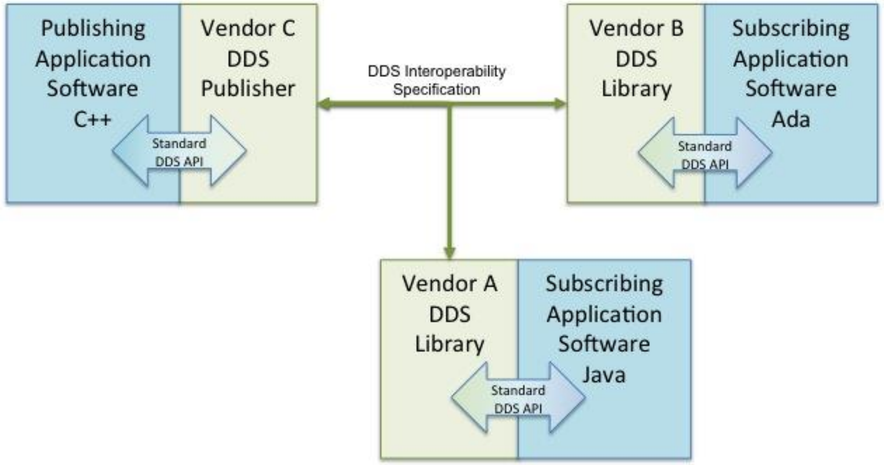
- DDS is **data-centric**:
  - All data exchanged on the middleware provides contextual information required by the different applications connected in the same domain
  - DDS handles the burden of implementing the complexity of message-centric middlewares



- DDS implements **QoS** mechanisms:
  - Specifications over **reliability**, **system health** and **security**
  - Flexible enough for one to manage all these specifications at once or just one or two in particular

- DDS provides **Dynamic Discovery**:
  - One application does not need to have its endpoints configured, as DDS provides a discovery mechanism to find publishers and subscribers under the same domain.

- **eProsima Fast RTPS** is a C++ implementation of the DDSI-RTPS (*Real Time Publish-Subscribe*) protocol, which provides a **decoupled communication middleware** with a model based on **publishers** and **subscribers**, over unreliable transport protocols such as UDP
- Implementation follows the RTPS wire-protocol defined for the Data Distribution Service (DDS) standard by the Object Management Group (OMG) consortium
  - Compliant with the OMG standard **RTPS 2.2**
  - Interoperable with other DDS implementations such as RTI Connext DDS, OpenSplice DDS, and Core DX.


Publishing Application Software C++
Vendor C DDS Publisher
Standard DDS API
DDS Interoperability Specification
Vendor B DDS Library
Subscribing Application Software Ada
Standard DDS API
Vendor A DDS Library
Subscribing Application Software Java
Standard DDS API

https://wiki2.org/en/Real-Time_Publish-Subscribe_(RTPS)_Protocol

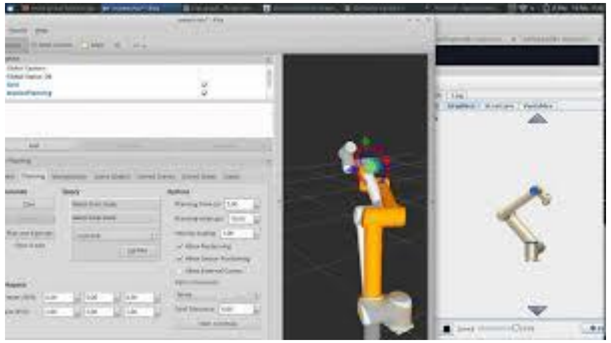
- Key features and benefits:
  - Lightweight;
  - C++ implementation;
  - Meets **high throughput** requirements of heavy data exchange systems;
  - Meets **real time requirements** of time-critical systems;
  - Well fitted to intermittent, unreliable and low bandwidth datalinks;
  - Provides **DDS Security** for access control, authentication and encryption;
  - Fully open-source;
  - Set as the **default middleware of ROS 2** and aligned with its development roadmap.

“The goal of the ROS 2.0 project is to leverage what is great about ROS 1.x and improve what isn’t.”

Dirk Thomas, in https://design.ros2.org/

2

- Key features and benefits:
  - Introspection tools (*ros2cli*)
  - Visualization tools (*rviz2*)
  - Discovery and Negotiation
  - Security (Authentication, Encryption, Access control)
  - Integration into larger multi-vehicle systems
  - Plug and play subsystems
    - Sensor drivers
    - Image processing pipelines



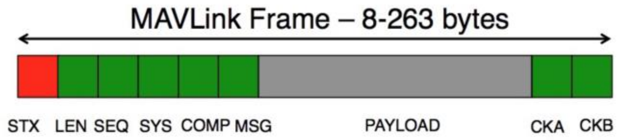
motivation

1. Take advantage of the **benefits of** using a **DDS middleware**;
2. Have a more clear, straightforward point-to-point (or point-to-multipoint) **data exchange** between the **PX4 internals** and **companion** (mission) computers;
3. Make a clear **separation** between **Mavlink telemetry streams** and **onboard communication** on a DDS domain
4. Facilitate the **integration** of PX4 with **ROS 2**, which default middleware is Fast RTPS
5. Guarantee near-optimal performant standard middleware for **mission-critical applications** which rely on external computational resources – obstacle avoidance, VIO, path planning, AI.

MAVLINK vs DDS

MICRO AIR VEHICLE COMMUNICATION PROTOCOL

- **Mavlink**:
  - **Messaging protocol** for communicating with drones;
  - Designed to support both telemetry links between drones and ground station and communication between onboard components;
  - Uses a **message-centric** implementation – the participants know about the system specificities and participants after they receive that information in a message (an *HEARTBEAT* message);
  - It’s designed to be lightweight and efficient, though, **doesn’t contain QoS mechanisms underline its implementation.**



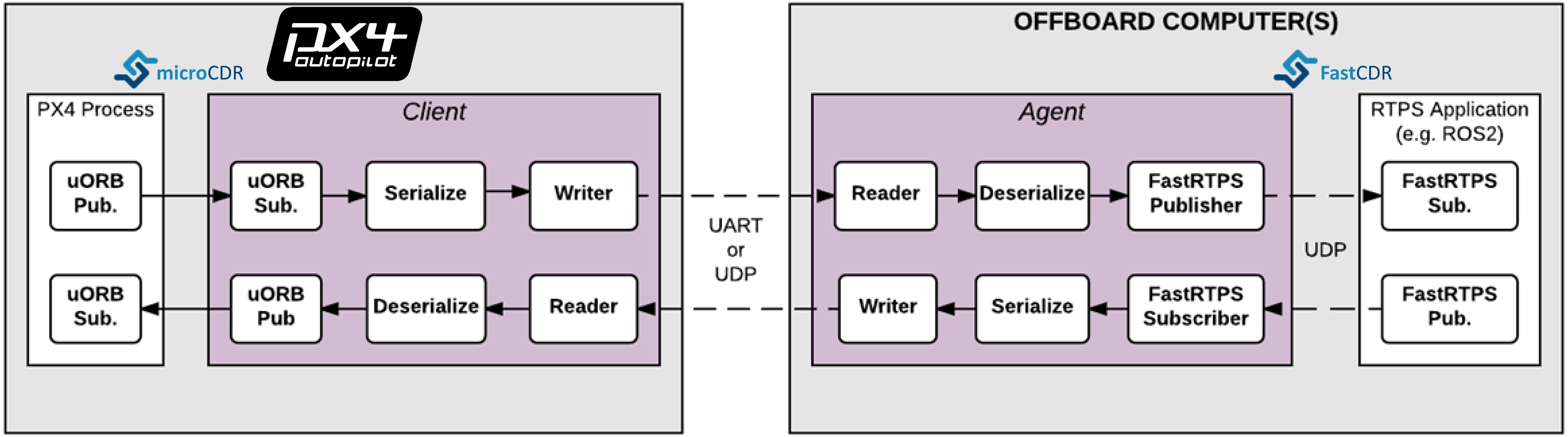
MAVLINK vs DDS

MICRO AIR VEHICLE COMMUNICATION PROTOCOL

- **DDS**:
  - **Data-centric middleware and API**;
  - Designed to provide **reliability**, **robustness**, **performance** and **scalability for IoT**;
  - The middleware is has embedded QoS mechanisms;
  - It’s a OMG industry standard well-proven in **mission-critical systems**;
  - Well suited for exchanging large scale critical data with low latency, high reliability and security.

PX4-Fast RTPS bridge

- aka PX4 **micro-RTPS bridge**
- First implementation in 2017
- Has received several updates since, specially in the code generators.

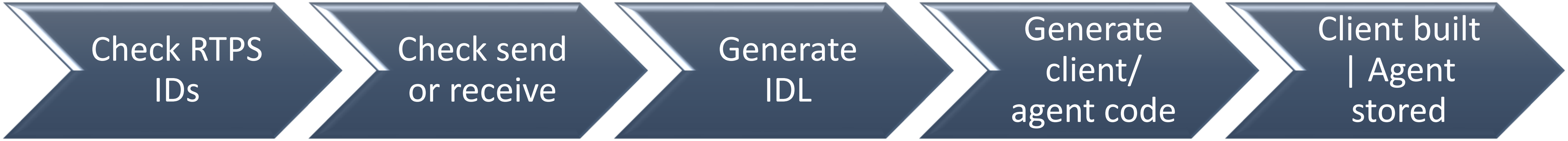


PX4-Fast RTPS – how?

- PX4 Devguide:
  - https://dev.px4.io/en/middleware/micrortps.html

PX4-Fast RTPS – how?

- PX4 build process: `make px4_<target>_rtps`



Check RTPS IDs

Check send or receive

Generate IDL

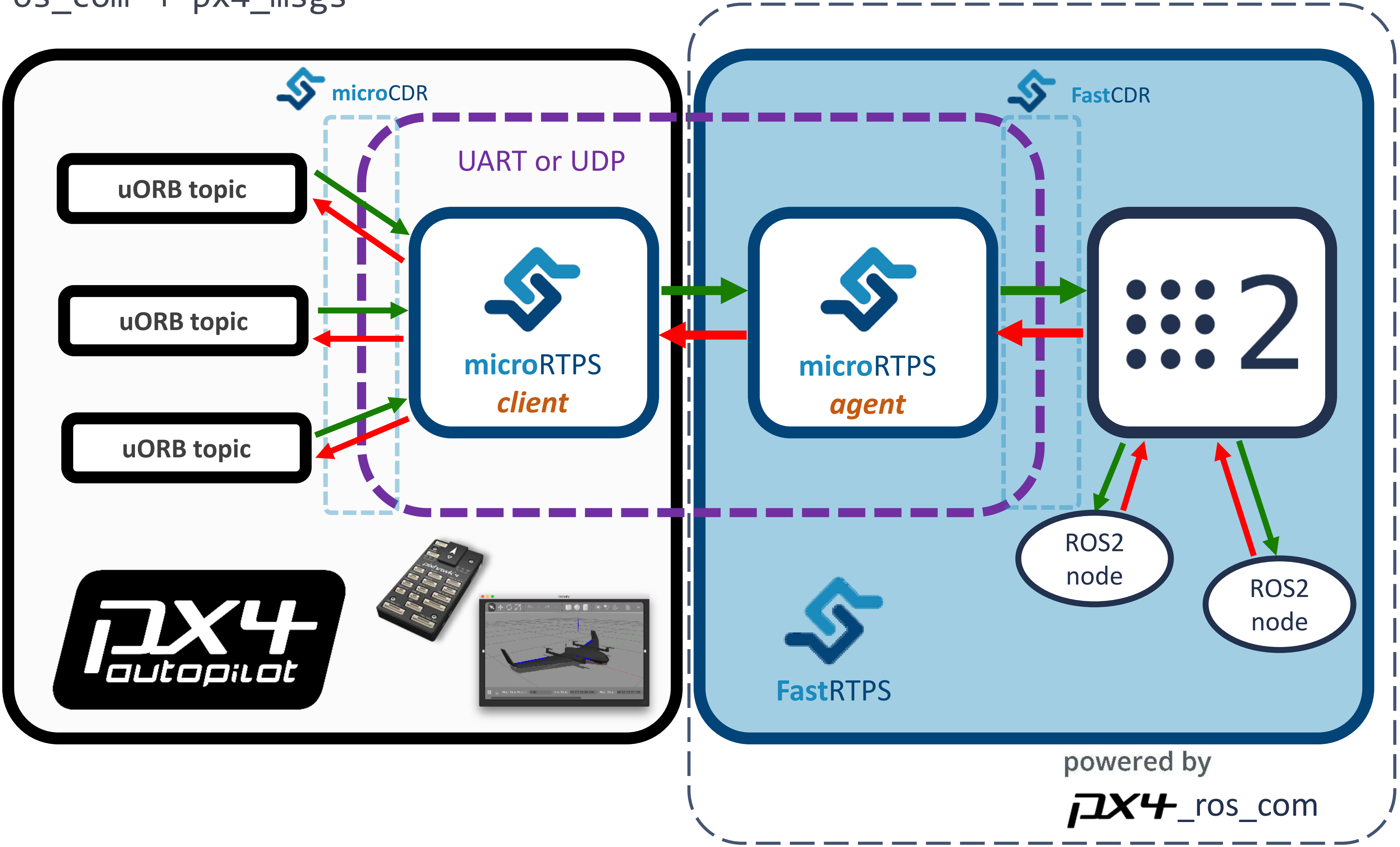
Generate client/ agent code

Client built | Agent stored

- Agent code build process – manually triggered:
  - Builds the agent application which **publishes and subscribes** to the ROS2/DDS topics.
- Listener application (optional) build:
  1. **`fastrtpsgen`** generates the required code to build an example for the specific onboard computer platform (`fastrtpsgen -example x64Linux2.6gcc <path_to_the_idl_file>`);
  2. 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

- px4_ros_com + px4_msgs



PX4-ROS 2 bridge – how?

- PX4 Devguide:
  - https://dev.px4.io/en/middleware/micrortps.html

PX4-ROS 2 bridge – how?

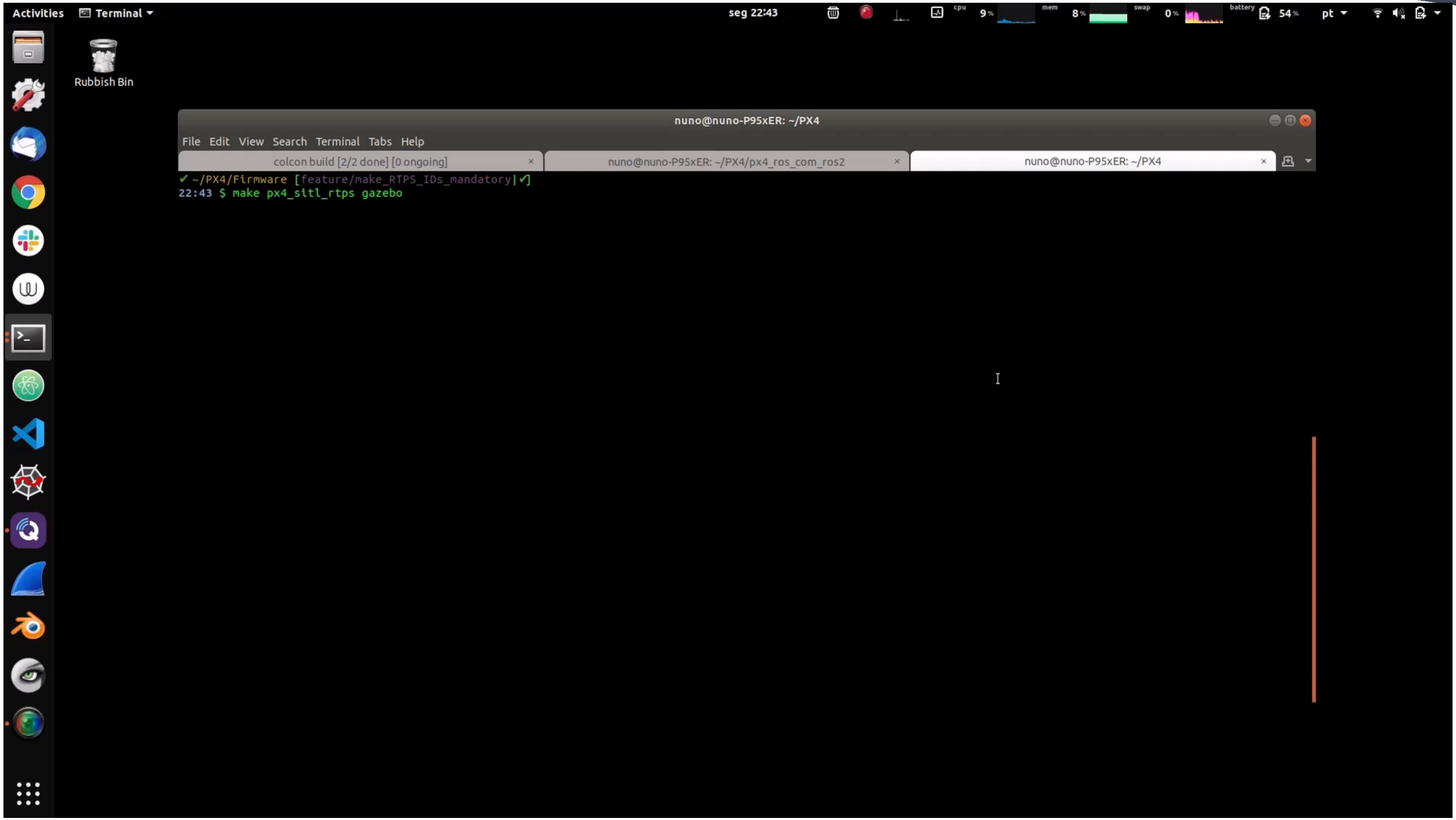
- `px4_ros_com`: https://github.com/PX4/px4_ros_com
  - Materializes the ROS2 side of PX4-FastRTPS bridge, establishing a bridge between the PX4 autopilot stack through a micro-RTPS bridge, Fast-RTPS and ROS2
  - Basically, **generates and allows building the agent side of the micro-RTPS bridge** to interface with Fast-RTPS – 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.

PX4-ROS 2 bridge – how?

- PX4 CI:
  - Automatically **deploy** all the **code generator scripts** to https://github.com/PX4/px4_ros_com every time they receive an update
  - Automatically **deploy ROS 2 message** counterparts of uORB messages to https://github.com/PX4/px4_msgs every time they suffer some change
- `px4_msgs` build:
  - Generates the **IDL** files required for the agent code
  - Generates the **typesupport** and **interface code** to be used by ROS 2 nodes
- `px4_ros_com` build:
  - Generates and builds the agent code
  - Builds **example nodes** that exist on the package by default.

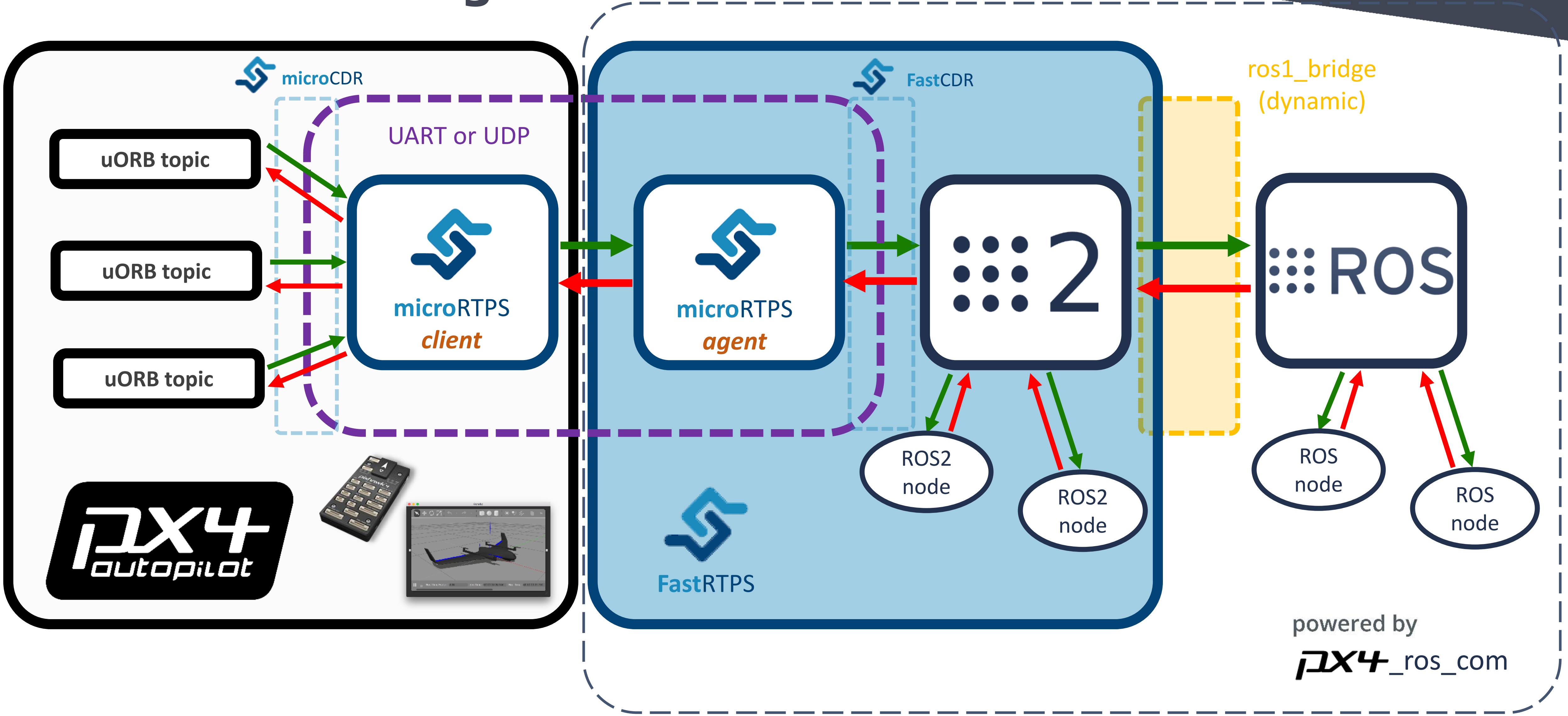
PX4-ROS 2 bridge – how?



Activities
Terminal
seg 22:43
Rubbish Bin
nuno@nuno-P95xER: ~/PX4
File Edit View Search Terminal Tabs Help
colcon build [2/2 done] [0 ongoing]
nuno@nuno-P95xER: ~/PX4/px4_ros_com_ros2
nuno@nuno-P95xER: ~/PX4

```
✔ ~/PX4/Firmware [feature/make_RTPS_IDs_mandatory|✔]
22:43 $ make px4_sitl_rtps gazebo
```

PX4-ROS 1 bridge?



microCDR

uORB topic

uORB topic

uORB topic

UART or UDP

microRTPS *client*

microRTPS *agent*

FastCDR

2

ros1_bridge (dynamic)

ROS

ROS2 node

ROS2 node

ROS node

ROS node

FastRTPS

PX4 autopilot

powered by PX4_ros_com

px4_ros_com vs MAVROS

px4_ros_com

✓

- Take advantage of all the benefits of DDS implementation and direct integration with ROS 2;
- (Theoretically) faster throughput and lower latency over the link;
- Direct and more tide relation between the PX4 internals and the offboard components;
- Can be tide to other RTPS (DDS) participants which are not registered over ROS – example: MAVSDK.

✗

- Obliges to have a one-to-one conversion between the uORB messages and the ROS messages, meaning it’s not parsed to standard ROS messages;
- For connecting to ROS (1), where most of the packages are still implemented, still requires a secondary bridge (`ros1_bridge`) so to be able to connect ROS (1) nodes with PX4.

px4_ros_com vs MAVROS

MAVROS

✓

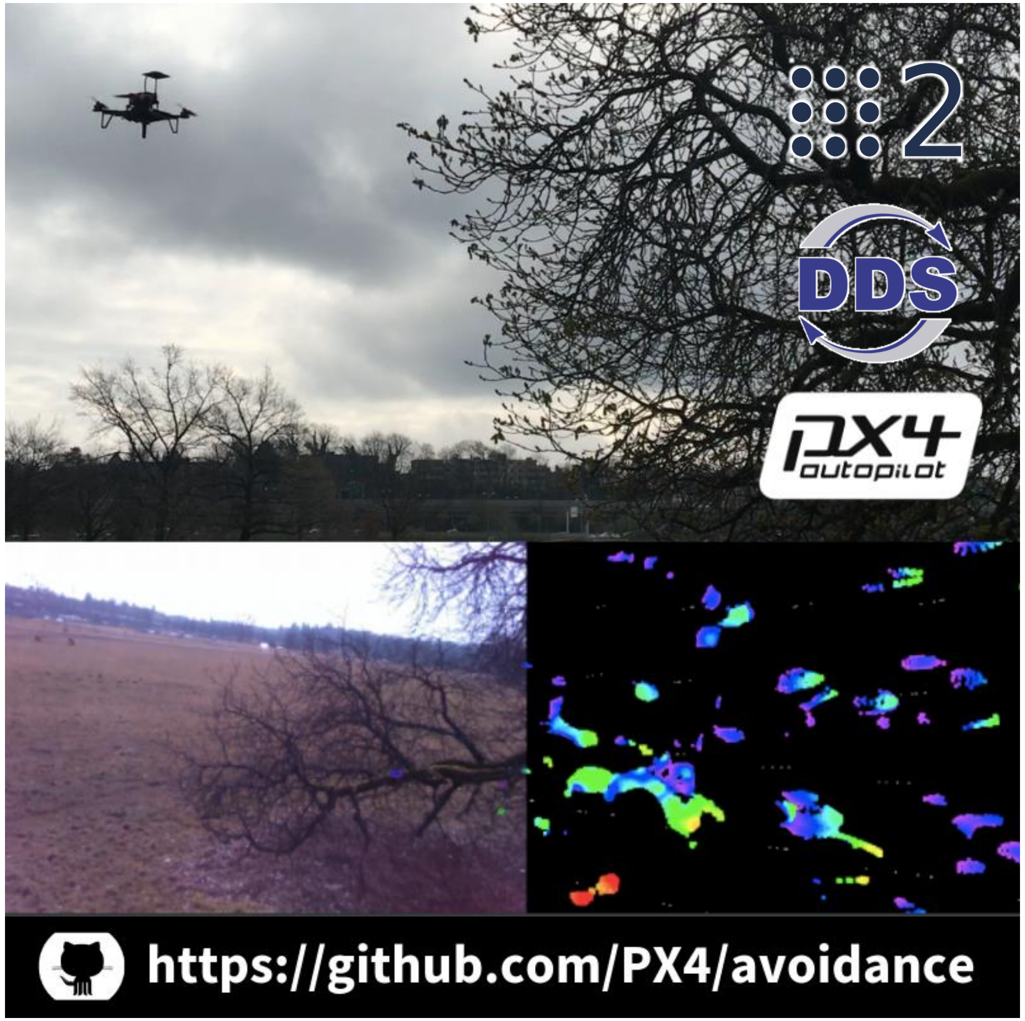
- Long-tested and industrial proven bridge between Mavlink and ROS;
- Allows parsing Mavlink messages to ROS standard messages;
- Allows network rebroadcast of the data to and from other hosts.

✗

- Not future proof – currently, no implementation going on to update the API and interface to support ROS 2.
- It's directly dependent on Mavlink and its interfaces, definitions and of course, limitations;
- Does not tide directly to the PX4 internals, though losing granularity for introspection and control.

Future

- Next steps:

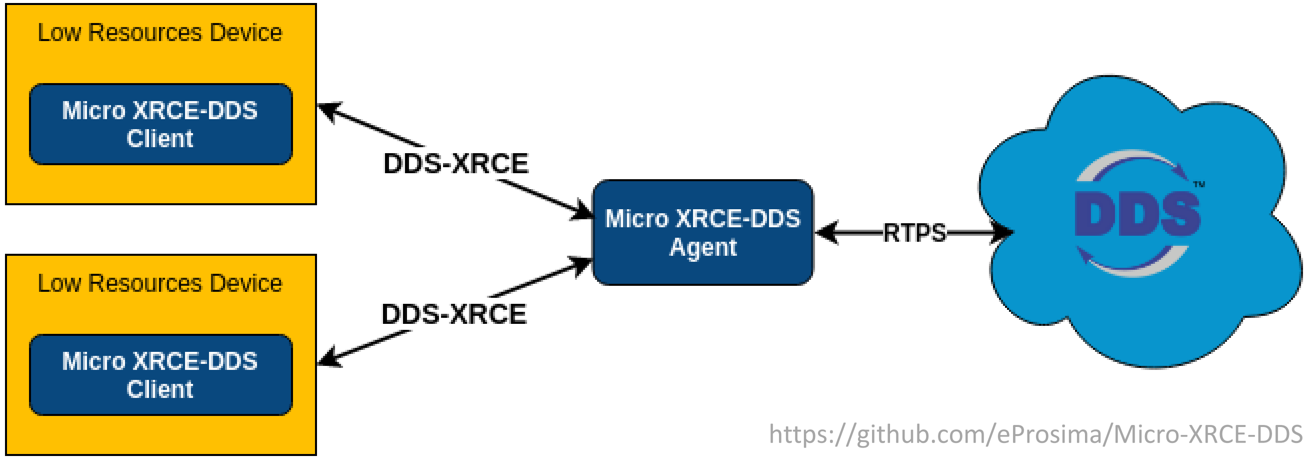


https://github.com/PX4/avoidance

- Assertion over performance
  - Throughput
  - Latency
  - Security

Future

- Mid-term:
  - Implement a **protocol splitter** on the agent side;
  - Improve micro-RTPS robustness and generalize implementation for **ROS2-over-serial**:
    - Create a generic approach to serial/network interfaces that can mimic a similar approach of **Micro XRCE-DDS** but using the ROS 2 API
    - Serial framing using COBS (*Consistent Overhead Byte Stuffing*) and checksum



https://github.com/eProsima/Micro-XRCE-DDS

Future

- Long-term:
  - Completely replace Mavlink with ROS2/DDS for data exchange between onboard components;
  - PX4 as a multi-node ROS 2 subsystem.

Summary

1. Introduction
   - DDS
   - Fast RTPS
   - ROS 2
2. Motivation
   - Mavlink vs RTPS/DDS
3. PX4 – Fast RTPS bridge
4. PX4 – ROS 2 bridge
5. *MAVROS* vs *px4_ros_com*
6. Future
7. Summary

Developer Summit

Thank you!

dronesolutions.io

✉ nuno.marques@dronesolutions.io

in https://www.linkedin.com/in/numarques