Workshop:

“Deeply embedded software”

EUROPEAN ROBOTICS FORUM

Program

*Francesca Finocchiaro (eProsima) - **“micro-ROS: bringing ROS 2 to microcontrollers”***

*Jan Staschulat (Bosch Research) - **“micro-ROS: API and executor”***

*Anaëlle Sarazin (WYCA Robotics) - **“Elodie and micro-ROS”***

*Tomasz Rokosz (Hydra System) - **“micro-ROS enabled GNSS receiver”***

***Panel discussion** - Invited panelists: Pablo Garrido Sanchez (eProsima), Alexandre Malki (PIAP)*

EUROPEAN
ROBOTICS FORUM

micro-ROS: bringing ROS 2 to MCUs

Francesca Finocchiaro - eProsima

April 13th, 2021

Overview

Who are we?

Open-source project, now benefiting from a huge participation from a growing community!

https://micro-ros.github.io/
https://www.eprosima.com/
francescafinocchiaro@eprosima.com

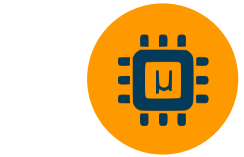
funded by

ROS 2

What is ROS 2?

- 2nd generation of **R**obotic **O**perating **S**ystem, provides infrastructure for robotics developers
- Open-source software with large community underneath
- [Long story short] Communication based on pub/sub paradigm:
  - Sensors publish to topics & actuators subscribe to topics
  - Based on DDS (Data Distribution System middleware)
  - Each node runs its own independent process
- Modular
- Benefits of QoS
- Has its own build/launch system
- CLI tools for deployment, monitoring and debugging
- Runs on Linux, Windows

micro-ROS



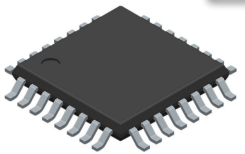
micro-ROS: puts ROS 2 onto microcontrollers!

A solution for creating ROS 2 nodes into embedded devices

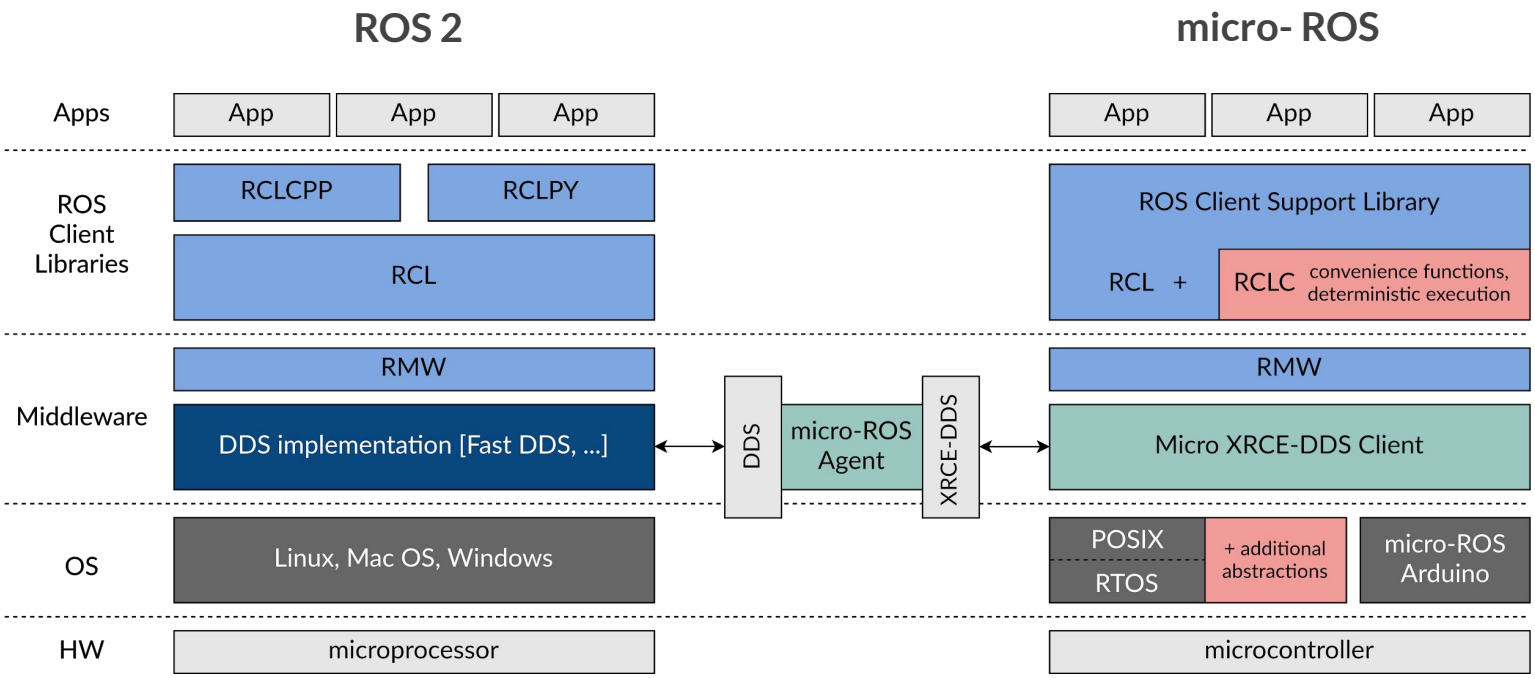
Why micro-ROS

Highlights

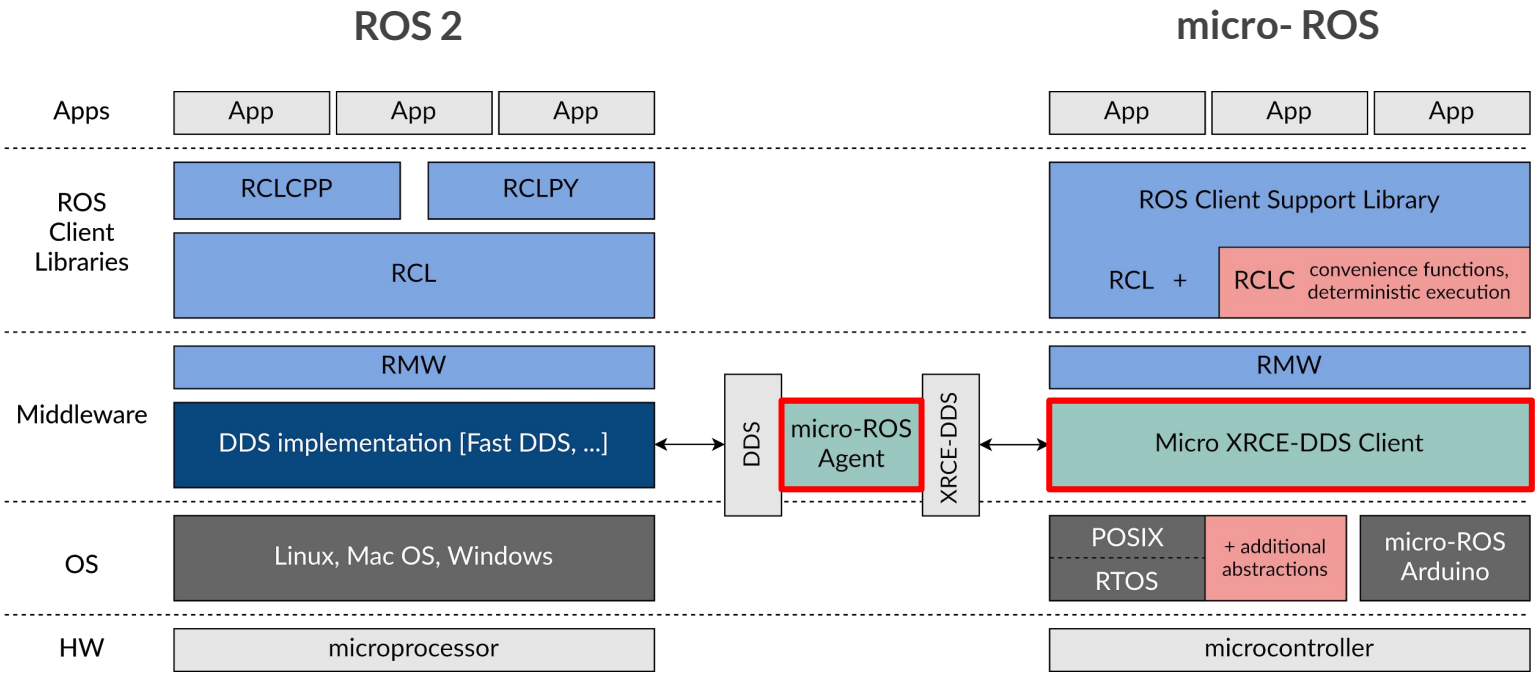
- Layer-compatible with ROS 2
- Integrated into ROS 2 ecosystem
- Allows to create a ROS 2 node with ~ all functionalities
- *Client/server* logics
- Fully customizable transports
- Runs on a variety of different platforms thanks to platform-versatile cross-compilation tools
- Supporting ROS 2 ***Foxy, Rolling*** and soon ***Galactic***!
- A growing community!



micro-ROS architecture



micro-ROS architecture



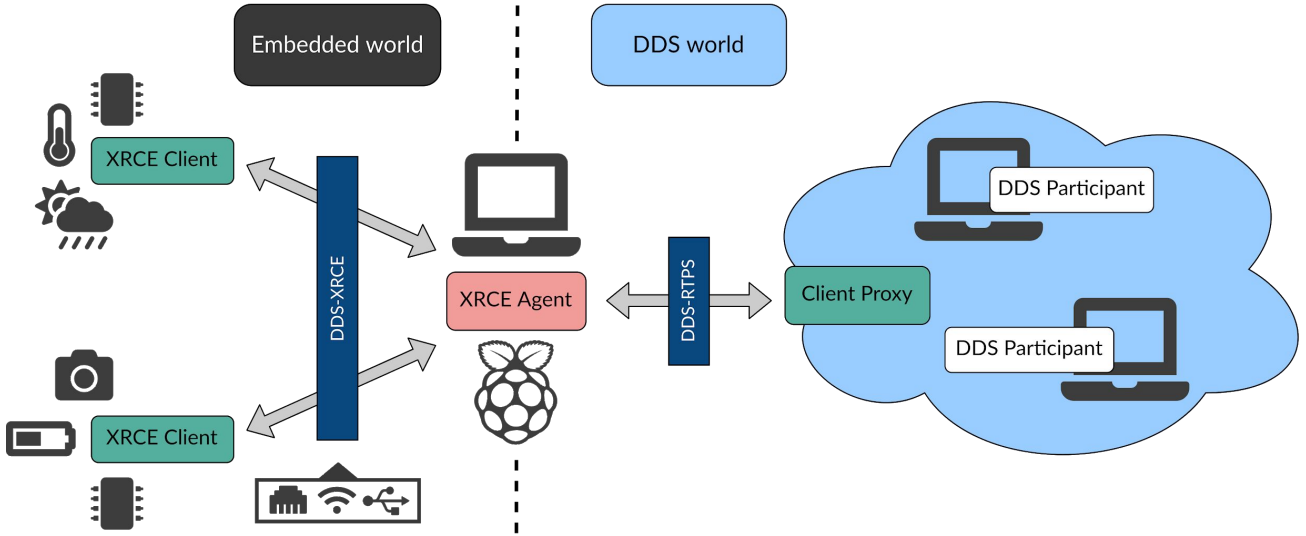
Middleware architecture

Micro XRCE-DDS: DDS for e**X**tremely **R**esource-**C**onstrained **E**nvironments.

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.

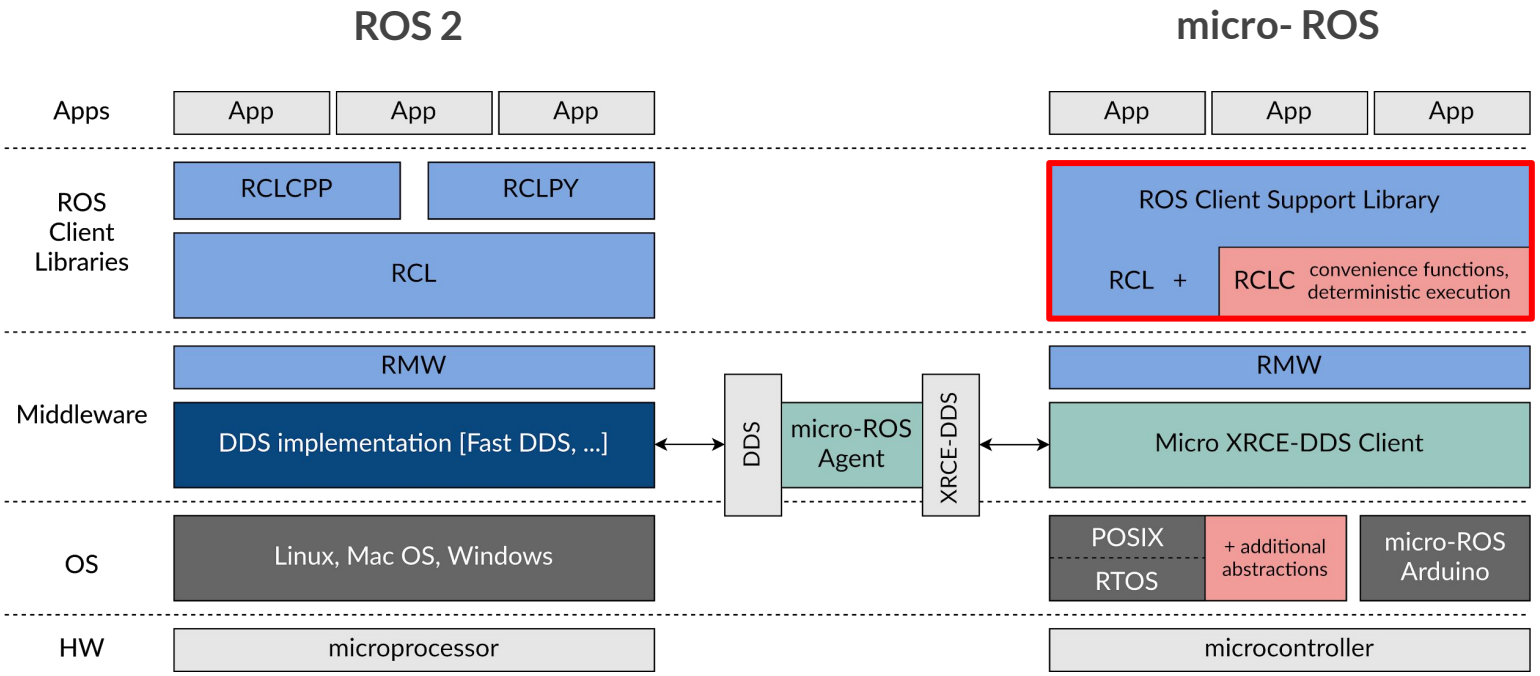
Main features:

- *Client-server architecture*
- *Request-response pattern*
- *Connection oriented*



eProsima
The Middleware Experts

micro-ROS architecture



ROS Client Support Libraries

App	App
RCLCPP, RCLPY	RCLC
RCL, RCUtils, rosidl_typesupport	RCL, RCUtils, rosidl_typesupport

C99 library:
provides utility functions for creating ***nodes***, ***publishers***, ***subscribers*** & ***redesigned executor*** [deterministic and LET semantics, dynamic memory allocation only at startup, domain-specific scheduling]

Same as in ROS 2
(many functionalities not used)

Supported platforms

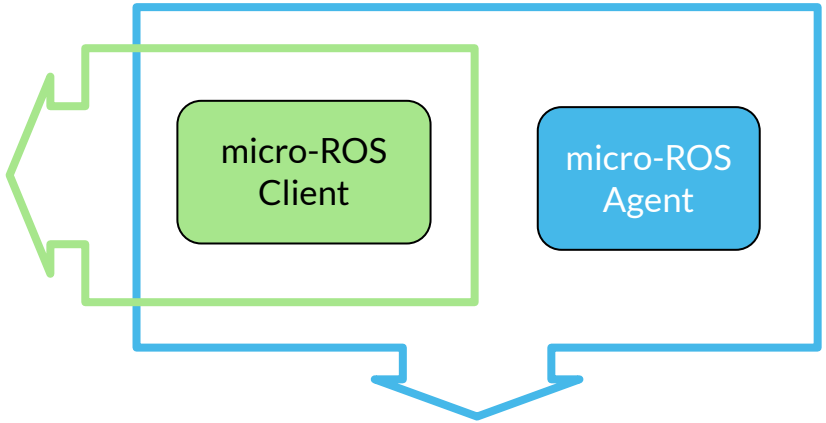
Supported RTOSes

NuttX

Zephyr

FreeRTOS

Mbed



Linux

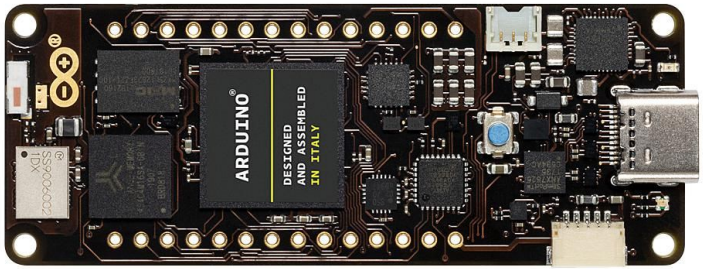
Windows

Supported HW

Target: mid-range microcontrollers.

Currently supported:

- ARM-M3/M4/M7 MCUs
- Xtensa MCUs
- RISC-V

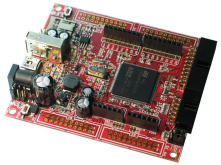


Typical features:

- ~ 1 MB of flash memory
- ~ 100 kB of RAM memory
- < 500 mA consumption
- General purpose input/output pins (GPIO)
- Communication peripherals: USB, Ethernet, SPI, UART, I2C, CAN, etc.

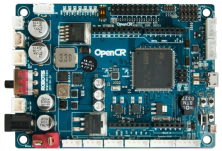
Supported HW

Olimex LTD
STM32-E407

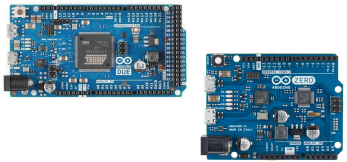


Crazyflie 2.1 drone

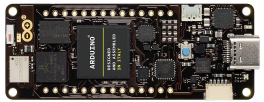
OpenCR 1.0



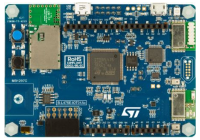
Arduino Due/Zero



Arduino Portenta

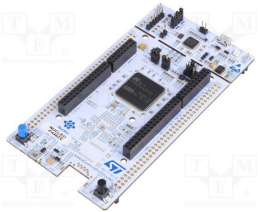


STM32L4
Discovery kit IoT

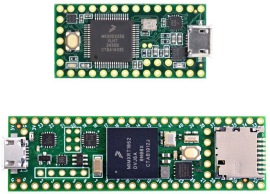


ESP32 family

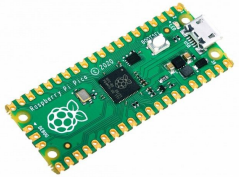
ST Nucleo family



Teensy 3.2/4.1



Raspberry Pi Pico



Integration into external tools

micro-ROS ESP-IDF component

ESPRESSIF

freeRTOS

micro-ROS Zephyr module

Zephyr™

micro-ROS Arduino

ARDUINO™

micro-ROS STMCube

STM32 Cube

Recent developments and features

Versatile API for custom transports

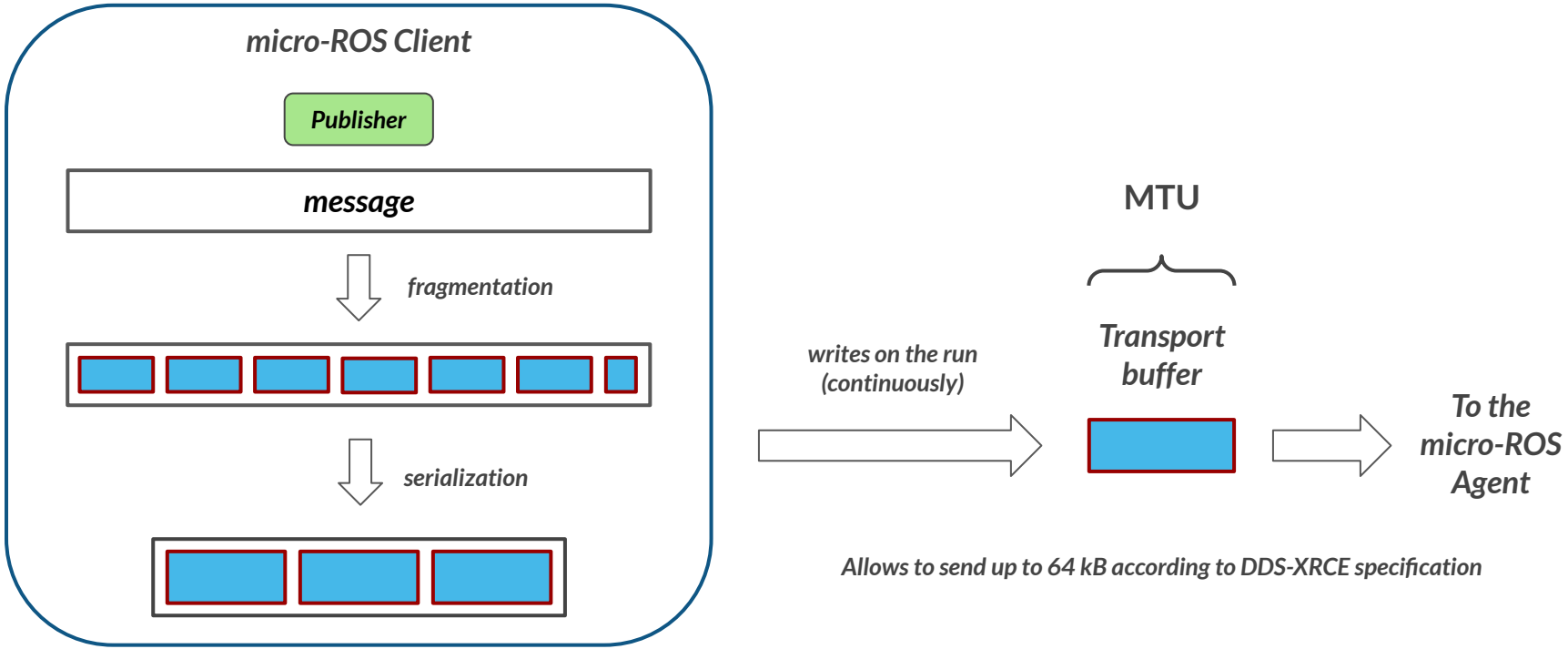
micro-ROS provides user API for interfacing with lowest level transport layer at runtime

This enables to implement custom transports and allows to transmit ***DDS-XRCE*** wire protocol over virtually any protocol, network or communication mechanism.

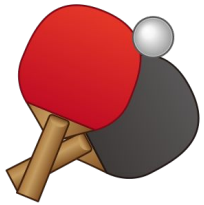
4 functions must be implemented:

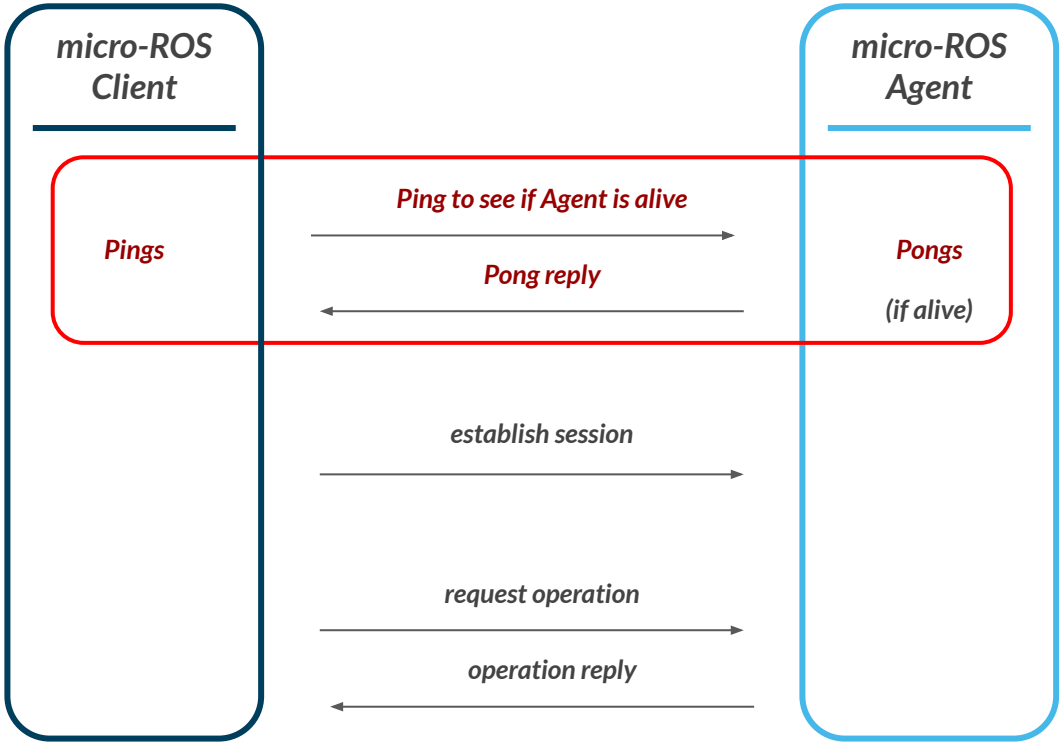
```
my_custom_transport_open(...)
my_custom_transport_close(...)
my_custom_transport_write(...)
my_custom_transport_read(...)
```

Continuous fragment mode

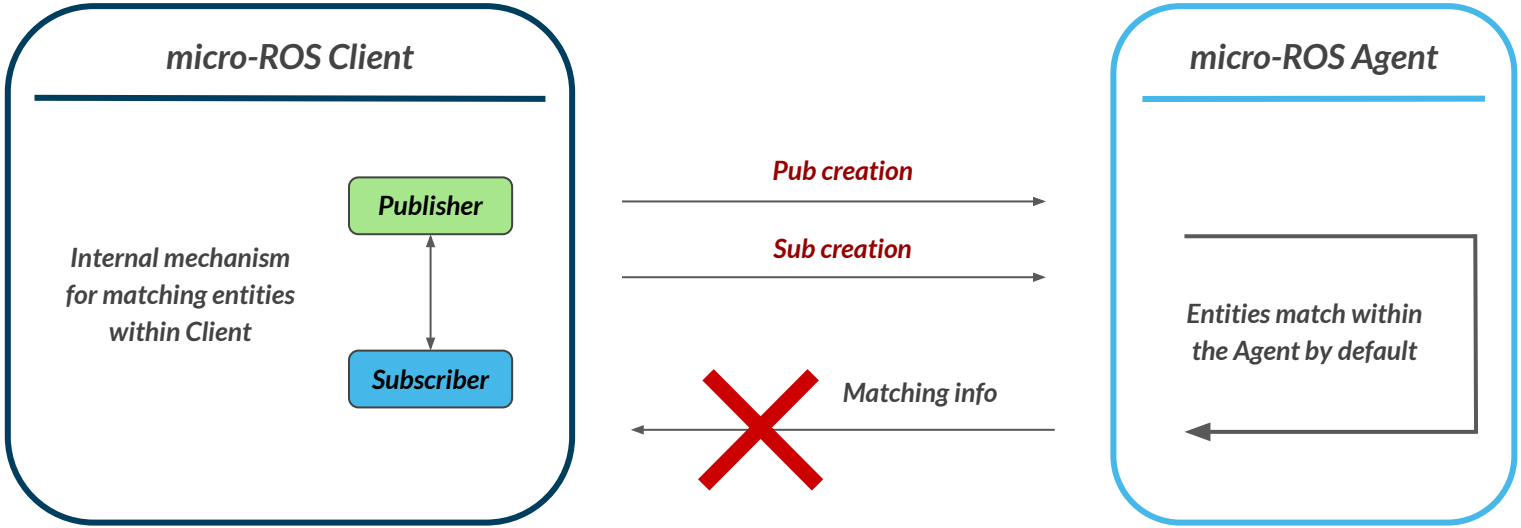


Client-to-Agent Ping

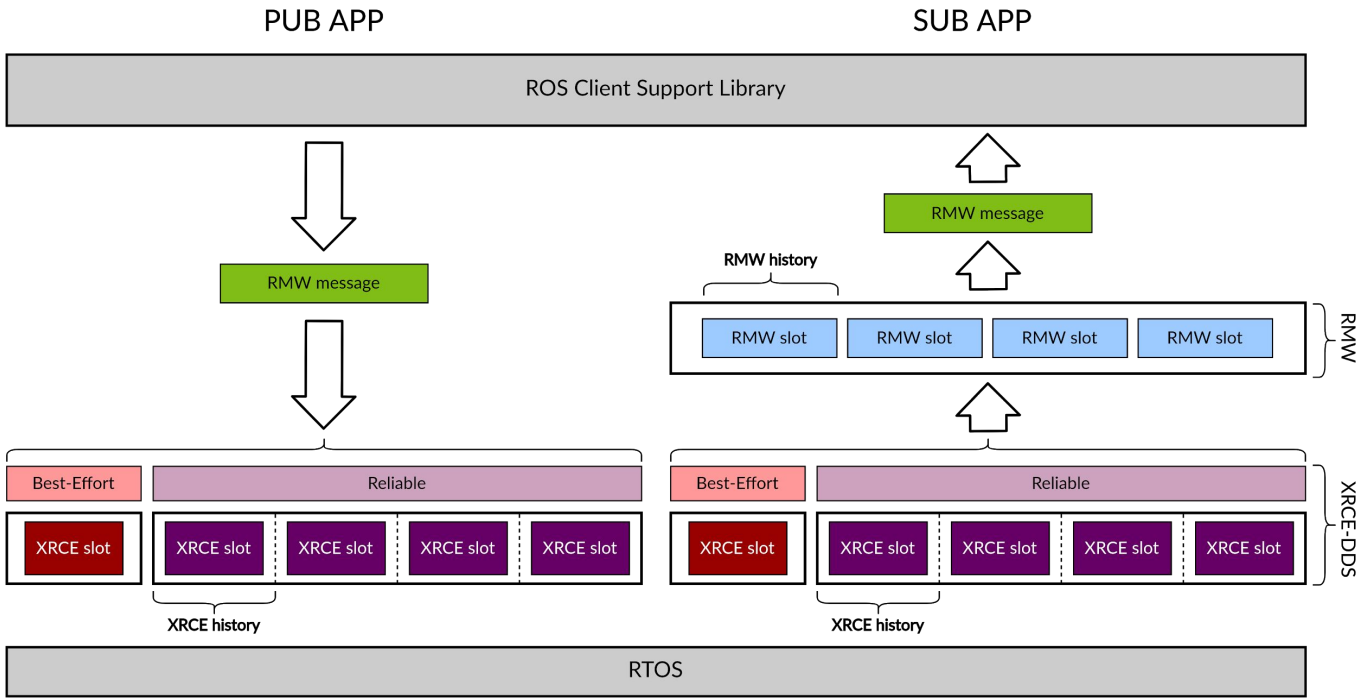




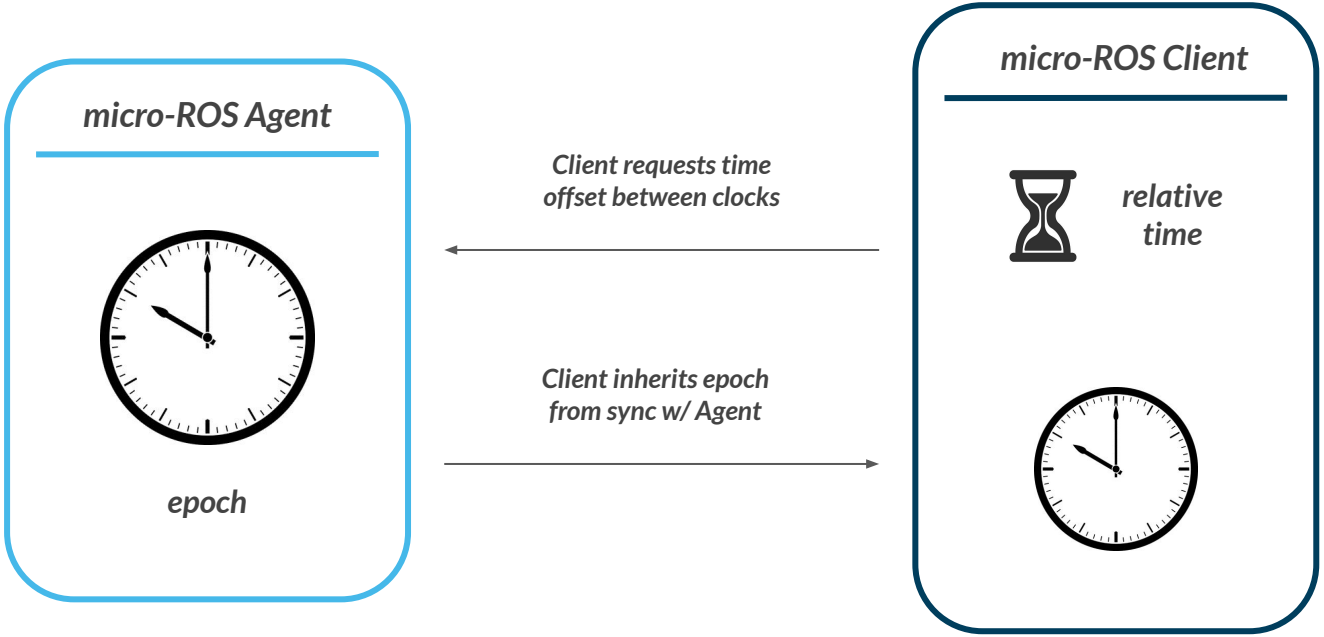
Shared Memory



Static Memory Pools in the RMW



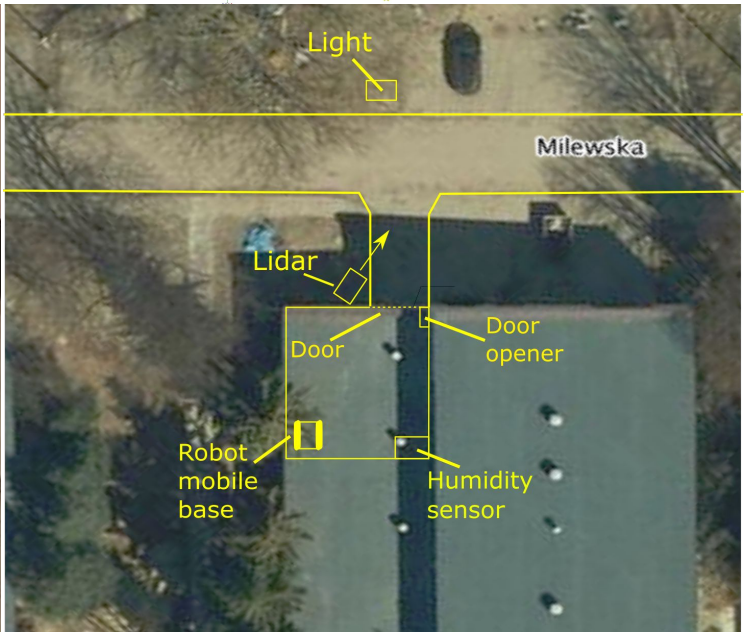
Client-Agent time sync



Demos

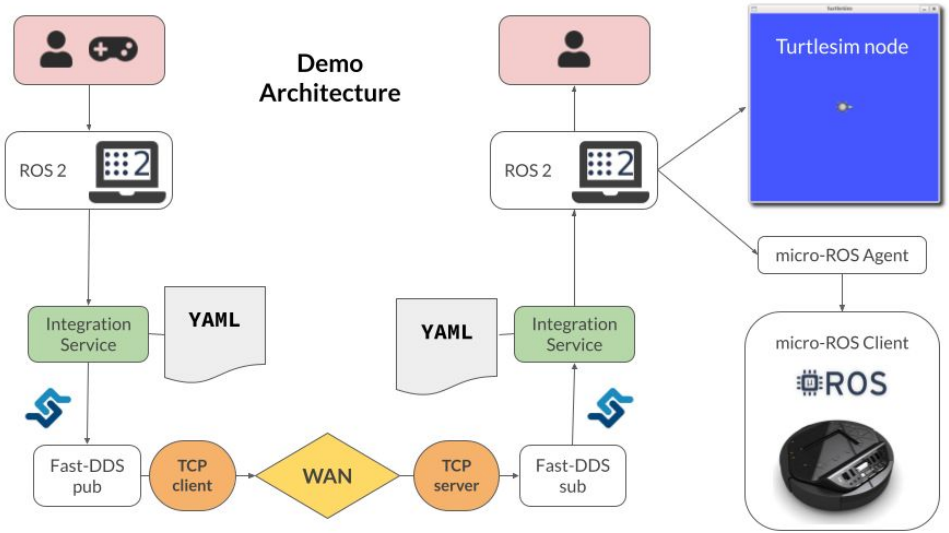
Demo I

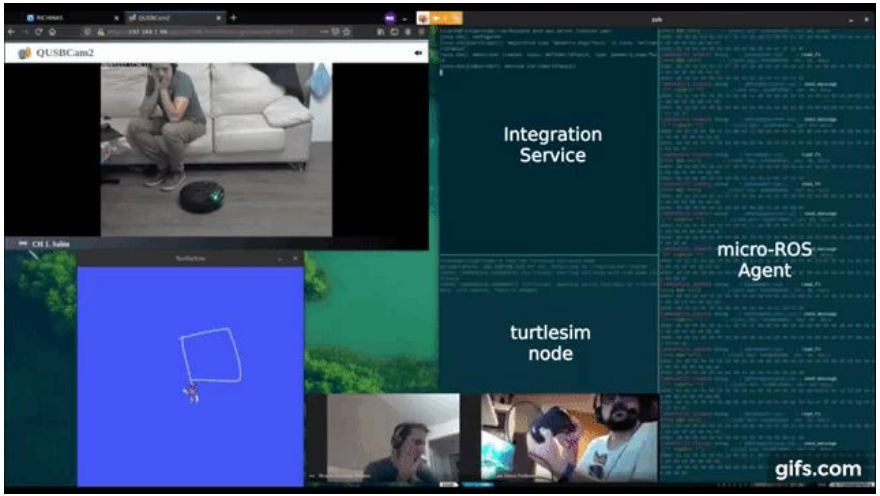
micro-ROS enabling smart warehouses duties



Demo II

ROS 2 to micro-ROS TCP tunneling via Integration Service!





Thanks for your attention!

Back-up slides

RMW

- Implemented using ***Micro XRCE-DDS*** middleware in lower layer
  - Allows static configuration of memory resources

Micro XRCE-DDS configurable parameters

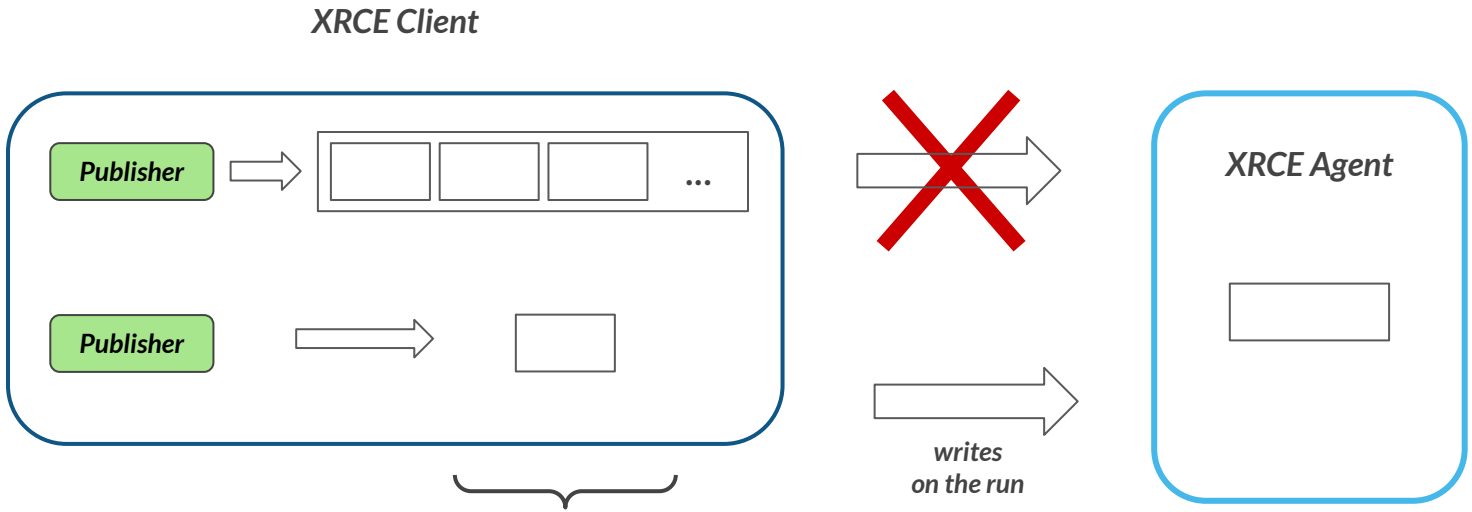
- Transport [UDP, serial, custom]
- Agent IP
- Agent Port
- Creation mode [XML, Ref]
- IP version [IPv4 - IPv6]

micro-ROS configurable parameters

- Max Publishers
- Max Subscriptions
- Max Clients
- Max Services
- Max Topics
- Max History
- Node name max length
- Type name max length
- Max Nodes
- Topic name max length

Configurability of these parameters allows preconfiguring the size of the library and tuning the size of the buffer to the memory available

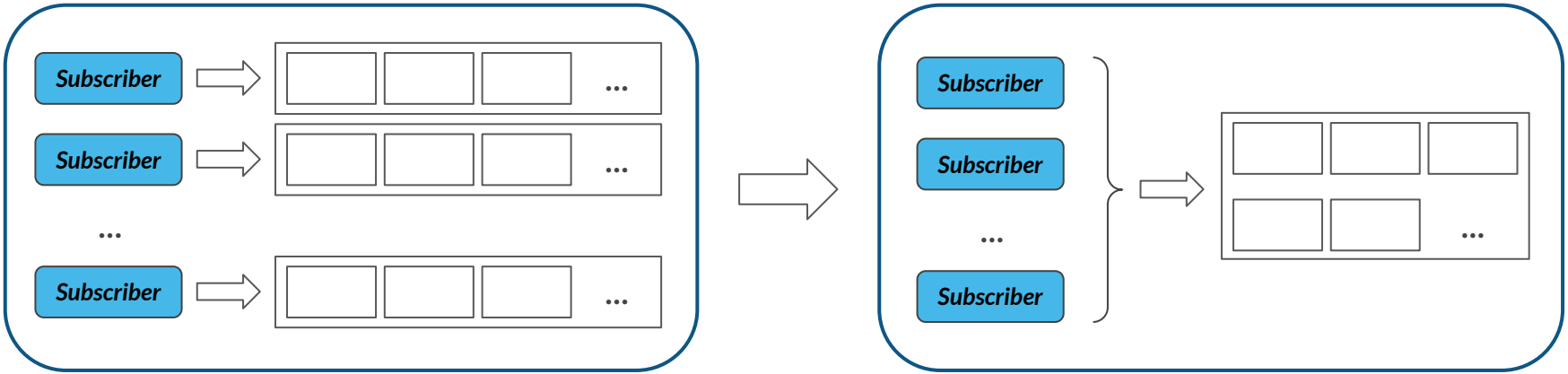
Continuous fragment mode



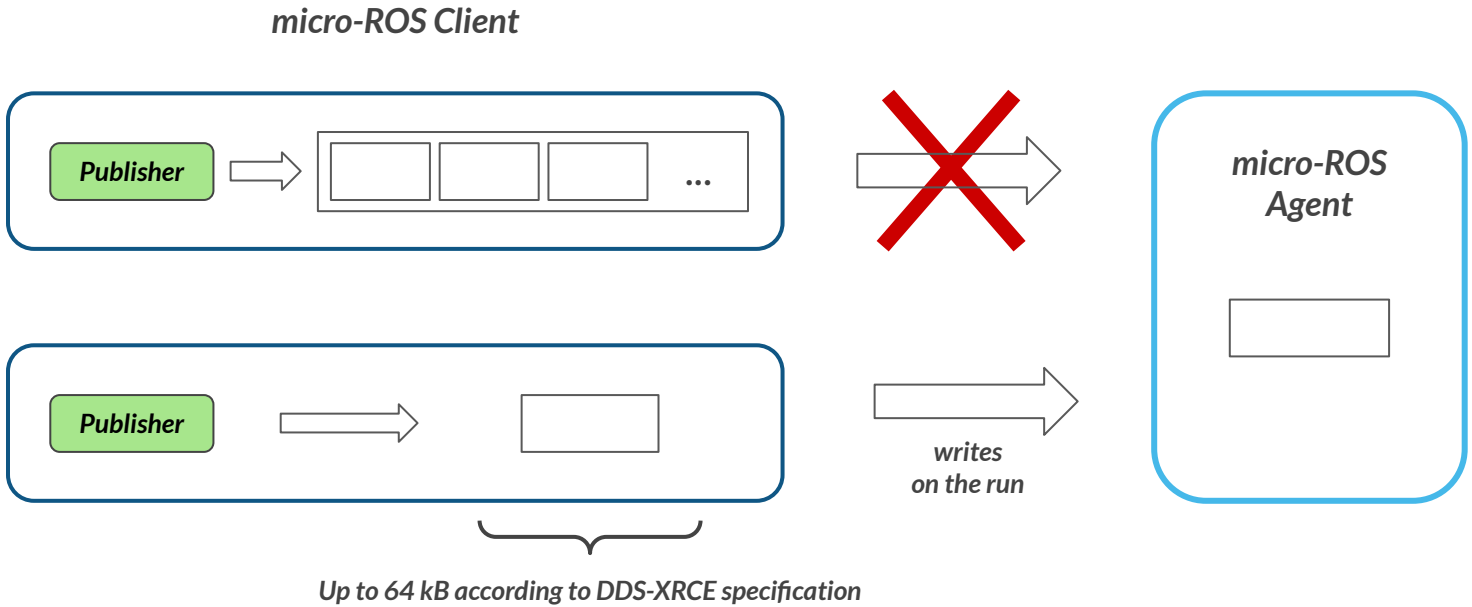
Max: 64 kB according to DDS-XRCE specification

Static Memory Pooling in RMW

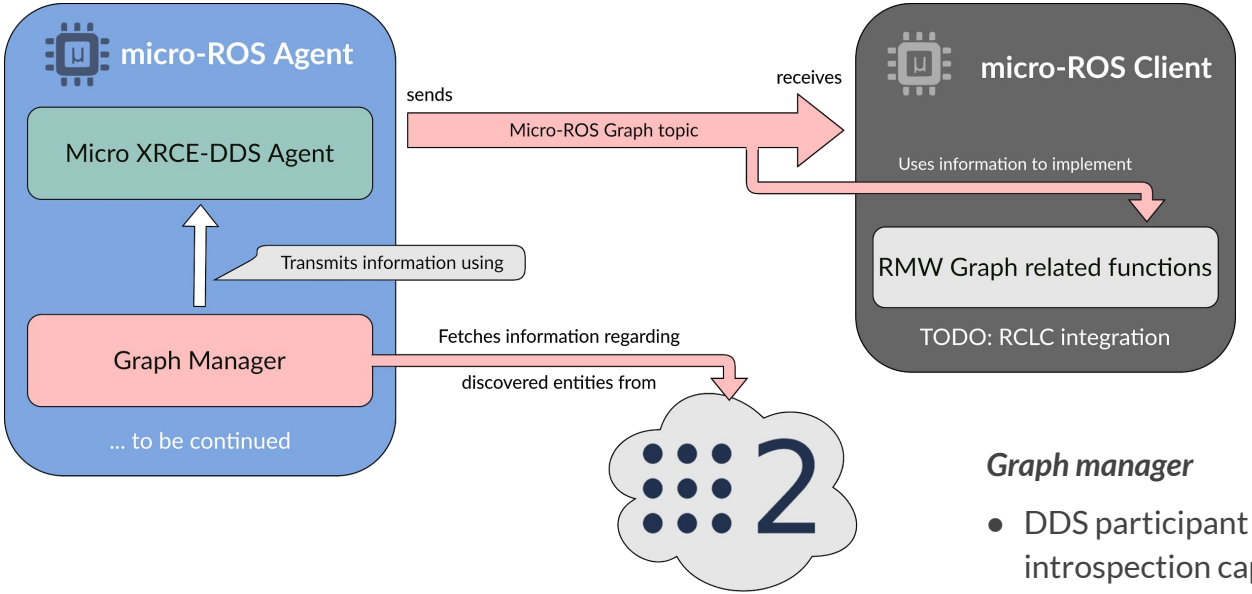
micro-ROS Client's RMW



Continuous fragment mode



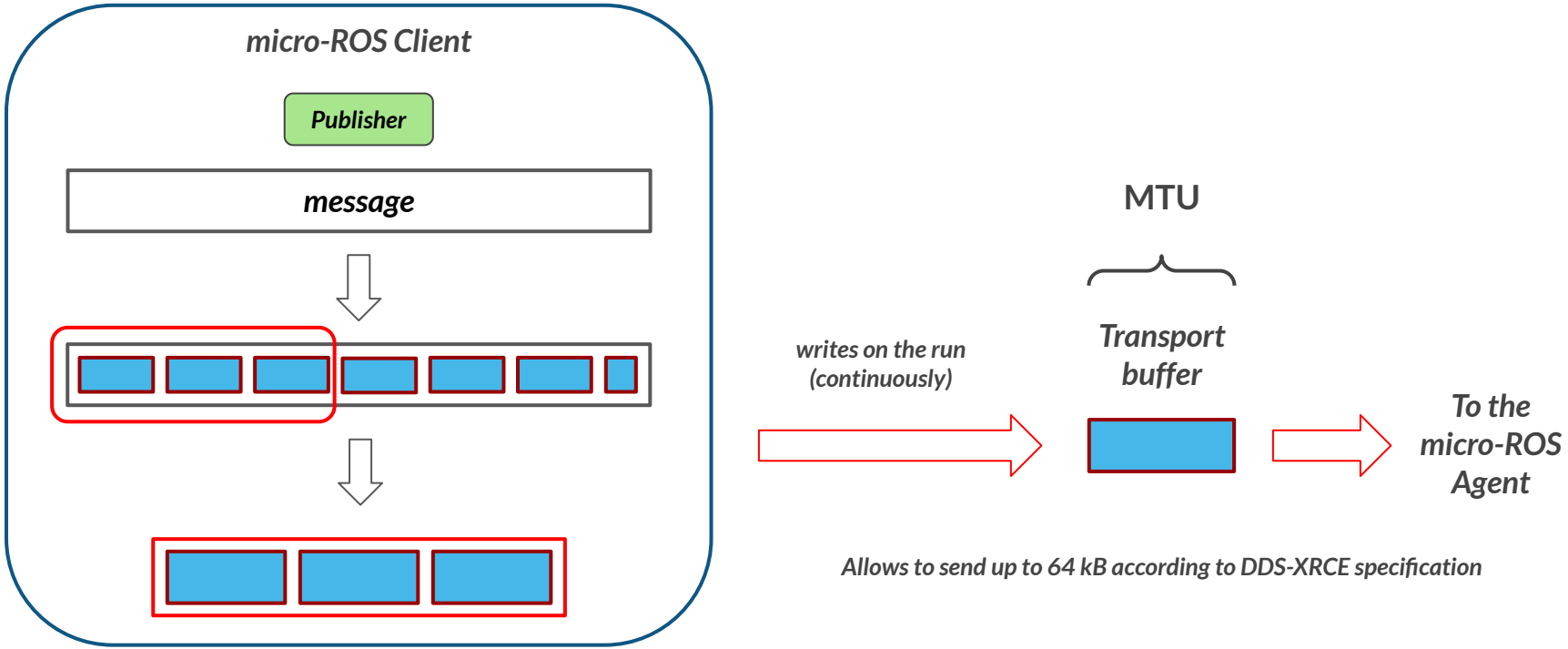
Graph support



Graph manager

- DDS participant scanning the network: provides introspection capabilities to user. ROS 2 topology consumable by micro-ROS
- micro-ROS topology info available to ROS 2

Continuous fragment mode



Continuous fragment mode