19. October 2021

ROS 2 Executor: How to make it efficient, real-time and deterministic?

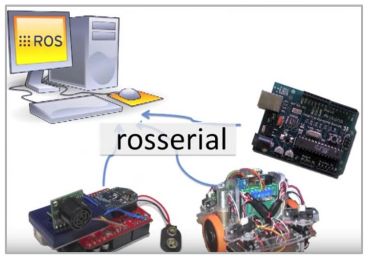
Micro-ROS: The rclc Executor

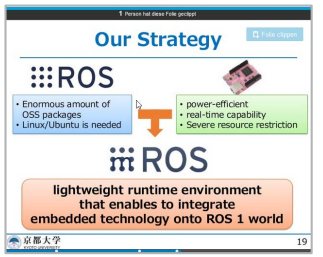
Dr. Jan Staschulat
Bosch Corporate Research

BOSCH

Rclc Executor: how to make it deterministic?
Why micro-ROS?

- ROS 2 dominant framework for powerful devices
  - ROS version 2 (ROS 2)
    - quality of service, security, ...
    - but large memory footprint
    - not real-time safe, non-deterministic Execution behavior





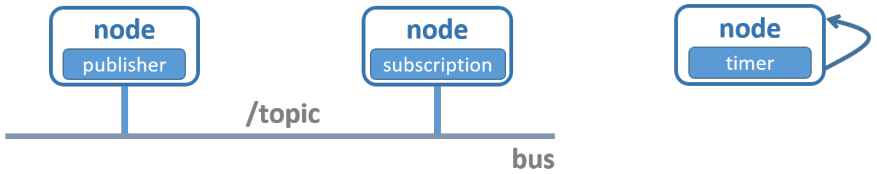
- Approaches for integrating microcontrollers
  - rosserial (limited features, only for ROS 1)
  - mROS (targets 400 MHz MCU, 10MB RAM)

- **micro-ROS puts ROS 2 on microcontrollers**
  - Medium size controllers (100 KB RAM)
  - ROS 2 API with C client library
  - Integration of uC into ROS 2 much less effort

Rclc Executor: how to make it deterministic?
Execution management in ROS 2 and micro-ROS

ROS 2 concepts:

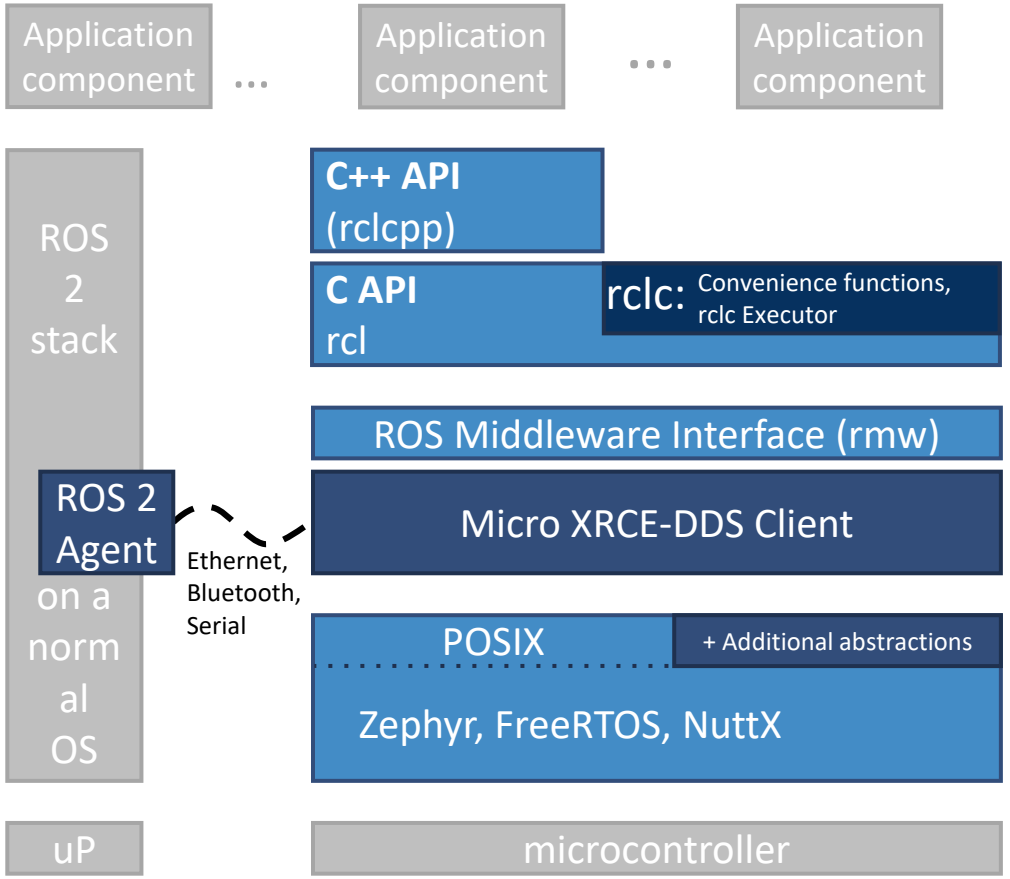
- Publish and subscribe middleware



- Executor

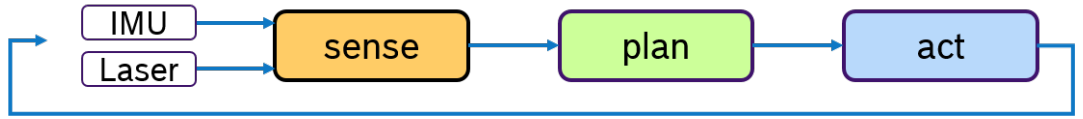
spin()

- Check for new messages
- Execute callbacks of subscriptions and timers

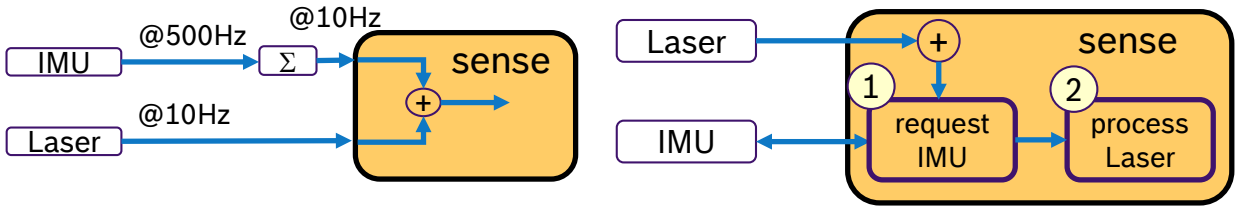


Rclc Executor: how to make it deterministic?
Robotic software design patterns

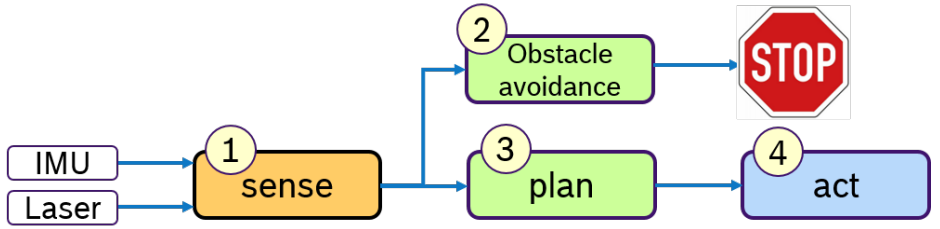
- Sense-plan-act control loops
  - **Phased execution**, e.g. start plan-phase only after all sensors have been processed in previous phase



- Sensor fusion with multiple rate inputs
  - Explicit control **when to start processing** depending on availability of messages
  - **Pre-defined order** of processing



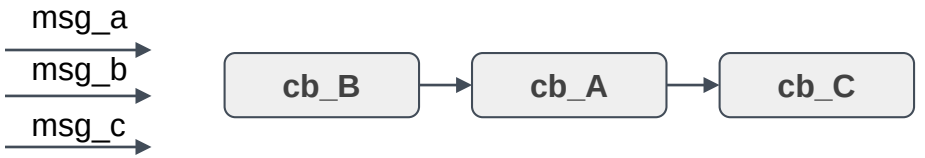
- High priority processing path
  - **Pre-defined order** of callback processing



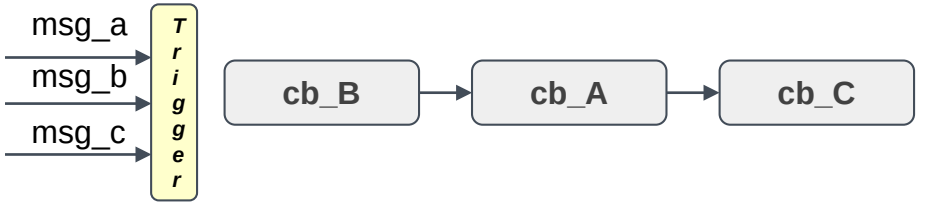
BOSCH

Rclc Executor: how to make it deterministic?
Key concepts

- Deterministic behavior
  - User-defined order of callback processing (round-robin execution)

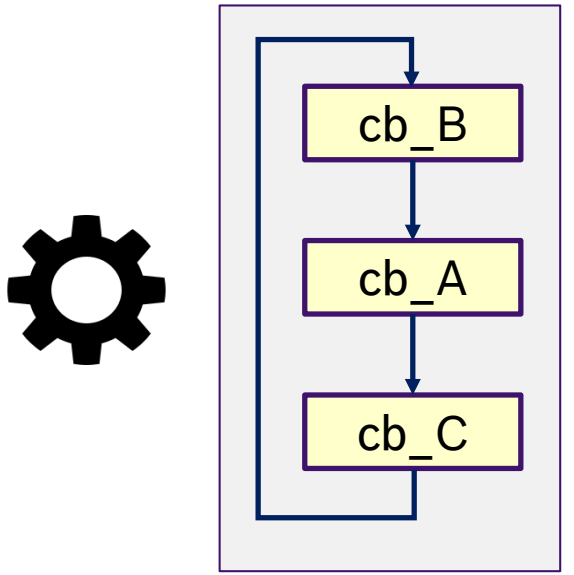


- Domain-specific scheduling
  - Trigger condition determines when processing of callback functions start (e.g. AND, OR, ONE)



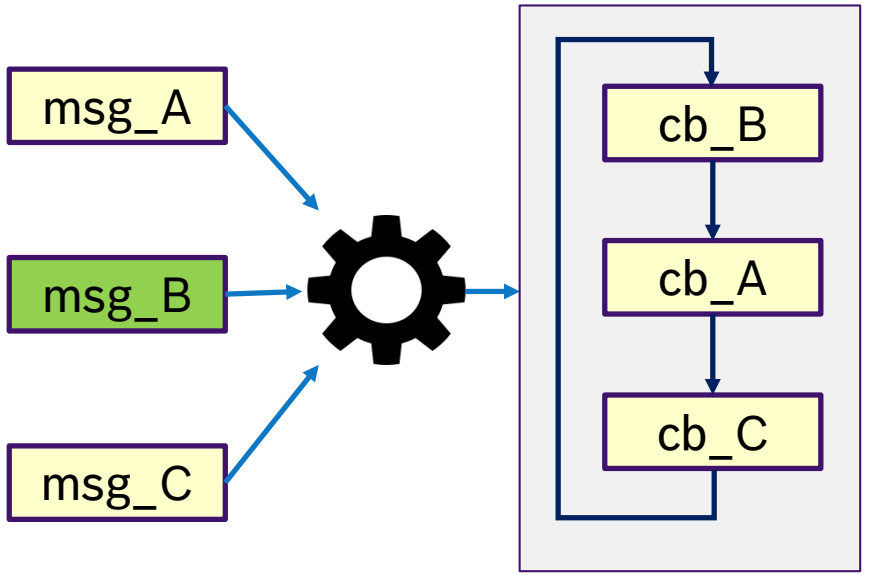
Rclc Executor: how to make it deterministic?
rclc Executor: concept 1: fixed processing order

- Fixed sequential execution of callbacks
  - Deterministic semantics
  - Bounded response time (round-robin)



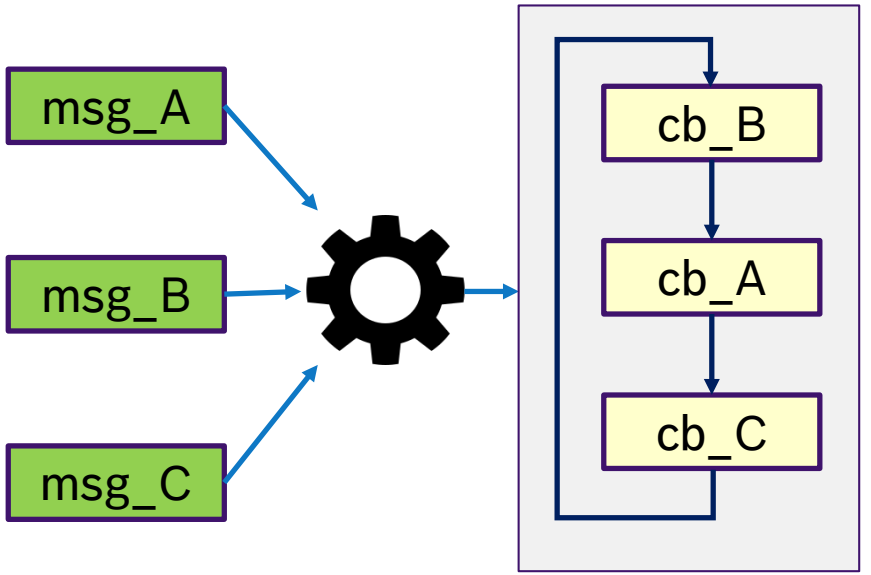
Rclc Executor: how to make it deterministic?
rclc Executor: concept 2: trigger condition

- Fixed sequential execution of callbacks
  - Deterministic semantics
  - Bounded response time (round-robin)
- Trigger
  - Determines start of execution
  - Pre-defined triggers:
    - **ONE - one particular subscription with new data**



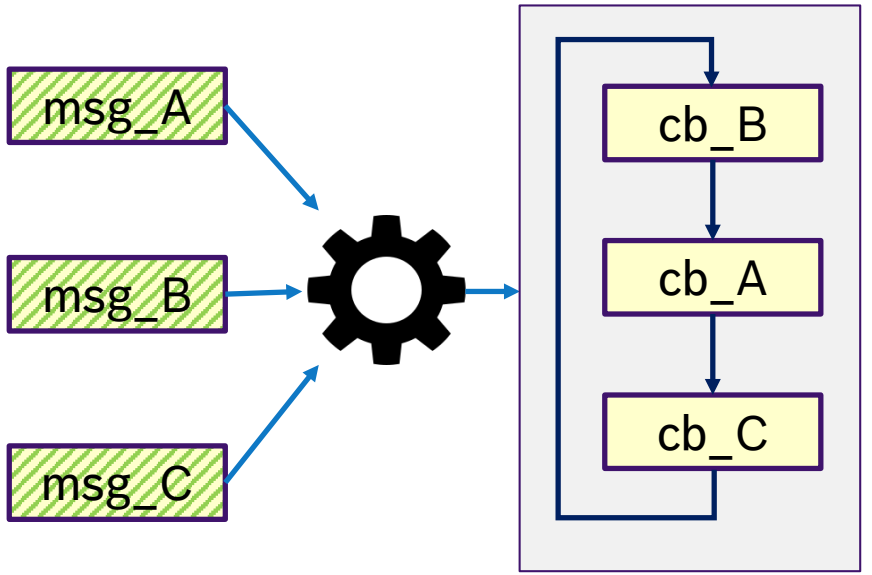
Rclc Executor: how to make it deterministic?
rclc Executor: concept 2: trigger condition

- Fixed sequential execution of callbacks
  - Deterministic semantics
  - Bounded response time (round-robin)
- Trigger
  - Determines start of execution
  - Pre-defined trigger:
    - ONE - One particular subscription with new data
    - **AND - all subscriptions with new data**



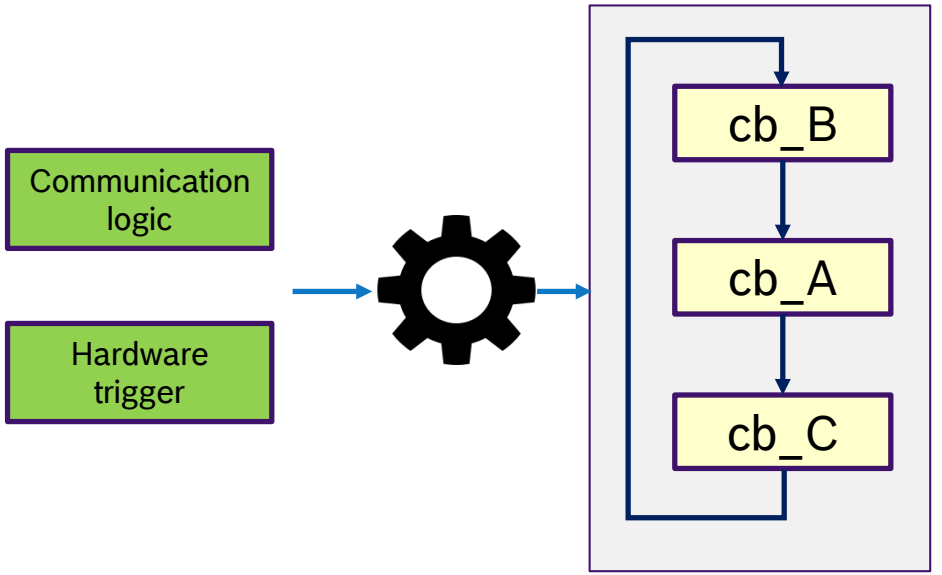
Rclc Executor: how to make it deterministic?
rclc Executor: concept 2: trigger condition

- Fixed sequential execution of callbacks
  - Deterministic semantics
  - Bounded response time (round-robin)
- Trigger
  - Determines start of execution
  - Pre-defined trigger
    - ONE - One particular subscription with new data
    - AND - All subscriptions with new data
    - **OR - Any subscription with new data (rclcpp default Executor)**



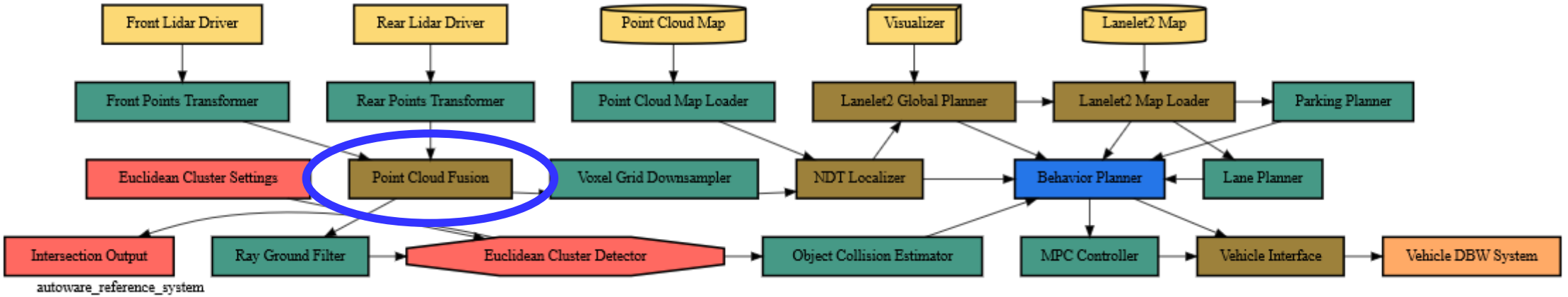
Rclc Executor: how to make it deterministic?
rclc Executor: concept 2: trigger condition

- Fixed sequential execution of callbacks
  - Deterministic semantics
  - Bounded response time (round-robin)
- Trigger
  - Determines start of execution
  - Pre-defined trigger
    - ONE - One particular subscription with new data
    - AND - All subscriptions with new data
    - OR - Any subscription with new data (rclcpp default Executor)
  - **User-defined trigger**
  - Callbacks can be configured to be executed with/without new data



Rclc Executor: how to make it deterministic?

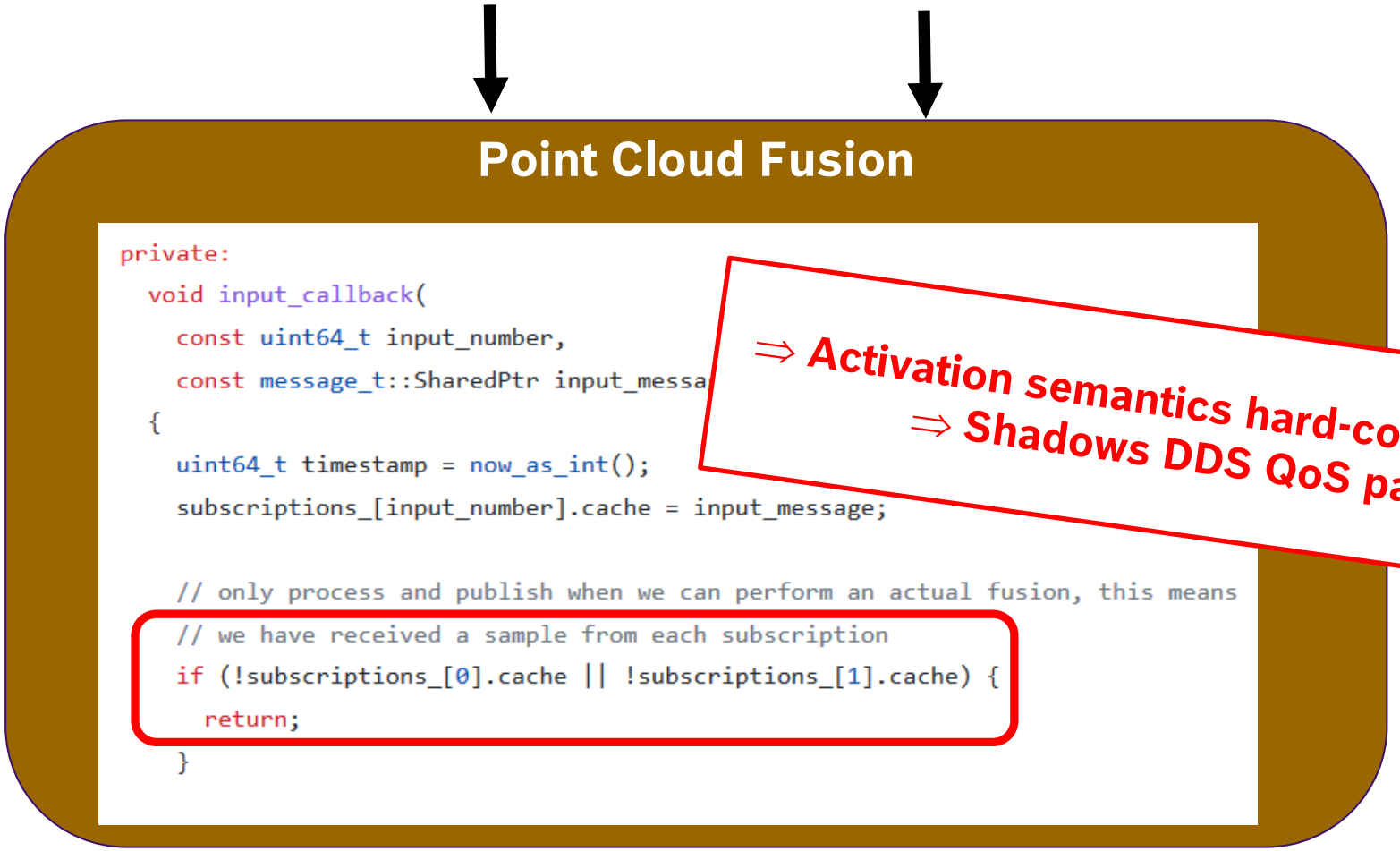
Autowareauto-reference system



- Fusion node has two inputs
- Is executed if both inputs are available

Rclc Executor: how to make it deterministic?

Autowareauto-reference system: default executor



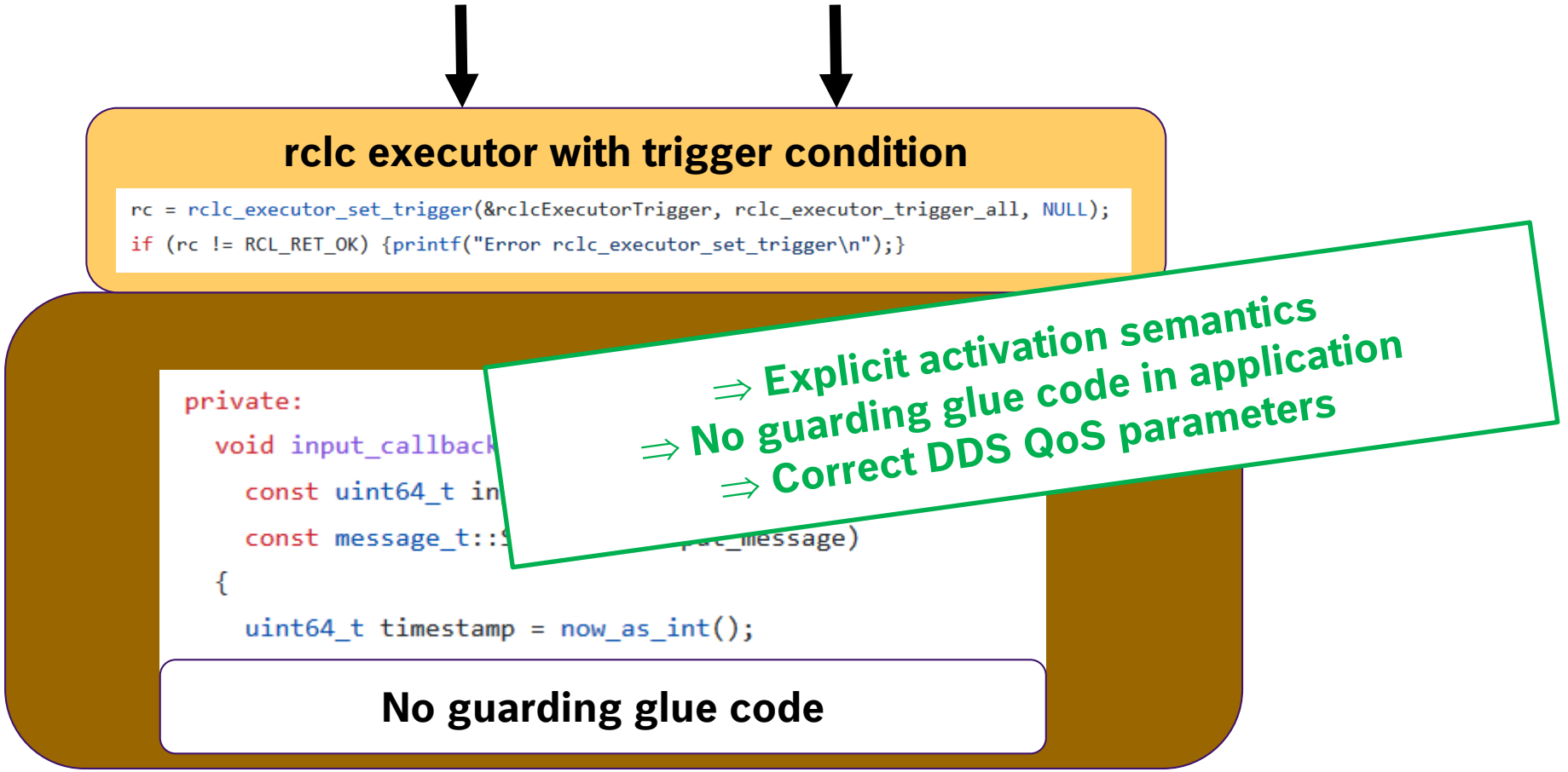
⇒ **Activation semantics hard-coded in application!**

⇒ **Shadows DDS QoS parameters!**

BOSCH

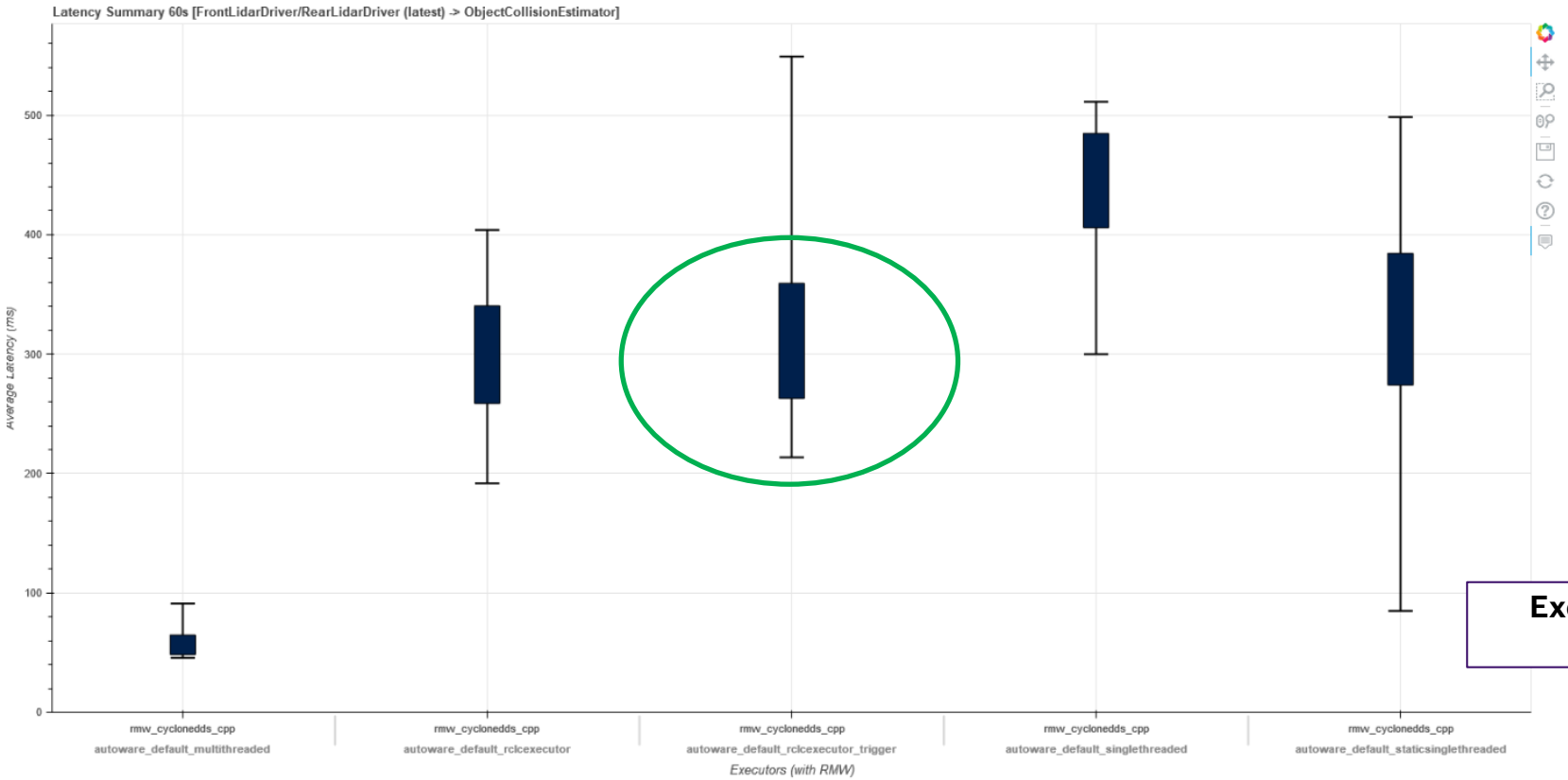
Rclc Executor: how to make it deterministic?

Autowareauto-reference system: rclc executor with trigger



Rclc Executor: how to make it deterministic?

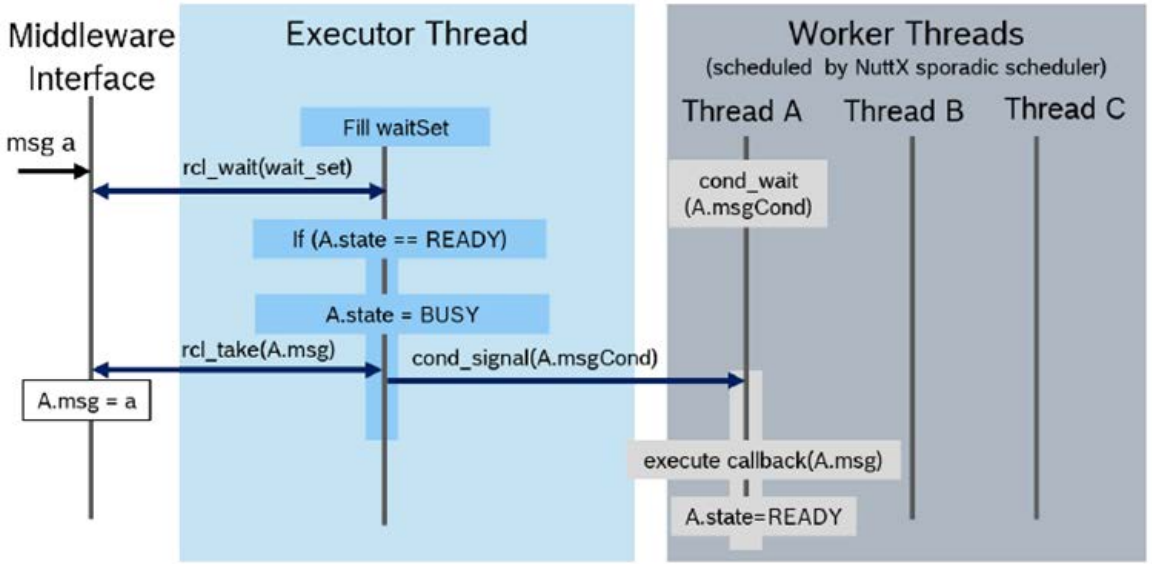
Autowareauto-reference system: latency results



Executed on Rasp on 4 cores for 60s

Rclc Executor: how to make it real-time?
Real-time scheduling and multi-threading

- Expose scheduling features of RTOS
  - ROS 2 API for subscriptions enables assignment of an RTOS scheduling policy (e.g. priority)
  - Executor thread manages data exchange with middleware (DDS)
  - Callbacks are processed in worker threads
  - Status: proof-of-concept with budget-based scheduling of NuttX-OS (arXiv paper)



Rclc Executor: how to make it deterministic?
Conclusion

- Embedded safety-critical applications require real-time and deterministic behavior
- Typical robotic software patterns can be implemented with the rclc-Executor semantics by
  - User-defined execution order
  - Trigger condition
  - Real-time scheduling
- Benefits of these features:
  - Separation of concerns (activation semantics vs application code)
  - Deterministic behavior
  - Real-time guarantees of end-to-end latencies

- **A ROS 2 Executor with trigger condition would support the development of deterministic ROS 2 applications.**

Questions?