Response-Time Analysis of ROS 2 Processing Chains under Reservation-based Scheduling

Daniel Casini, **Tobias Blass**, Ingo Lütkebohle, Björn Brandenburg

Sant'Anna Scuola Universitaria Superiore Pisa

SIC Saarland Informatics Campus

MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS

BOSCH Invented for life

This Paper in a Nutshell

Robots are **complex cyber-physical systems**, subject to real-time constraints

https://robots.ros.org/robonaut2/

ROS, the most popular robotics framework is **not based on real-time system models** and **no response-time analysis exists**

https://robots.ros.org/innok-heros/

This work:

Understand the surprising and undocumented timing behavior of ROS

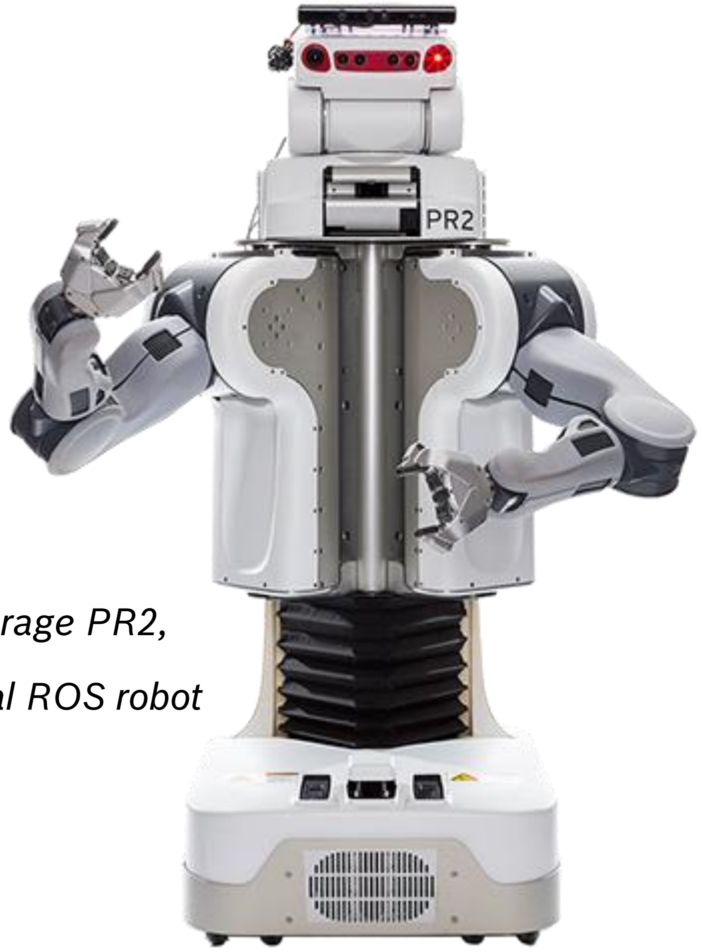
Develop a response-time analysis that is aware of ROS's quirks

The Robot Operating System (2007)

Popular robotics **framework** in academia and industry

- > 10 000 users
- > 1000 packages
- > 500 attendees at ROSCon

12 years of development exposed many **limitations** in the original design



Willow Garage PR2,

the original ROS robot

http://rasc.usc.edu/pr2.html

BOSCH

From ROS to ROS 2

- Complete refactoring of the ROS framework
- Recently released first long-term support version
- Aims to support **real-time control**

https://www.youtube.com/watch?v=npQMzH3j_d8

"We want to support real-time control directly in ROS, including inter-process and inter-machine communication"

From "*Why ROS 2?*" at *design.ros2.org*

What does this mean in practice?

BOSCH

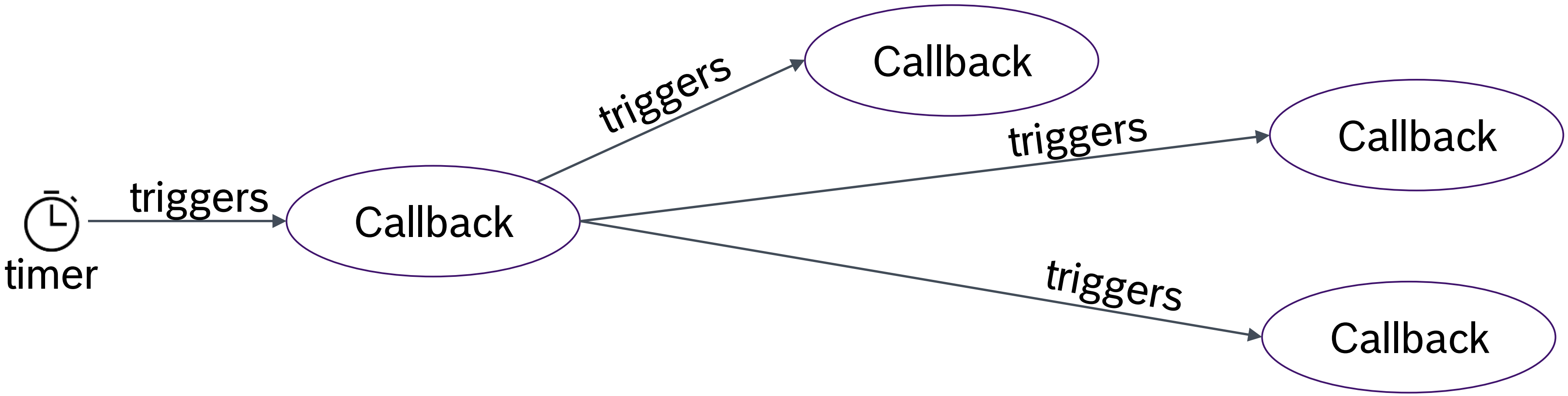
Our Needs for Real-Time Control

1. Support for automated timing **validation**
   - Does this robot react in time?

2. Support for model-based **design-space exploration**
   - Would this robot react in time if I used this hardware?
   - Would this robot react in time if I implemented it that way?

We need a response-time analysis

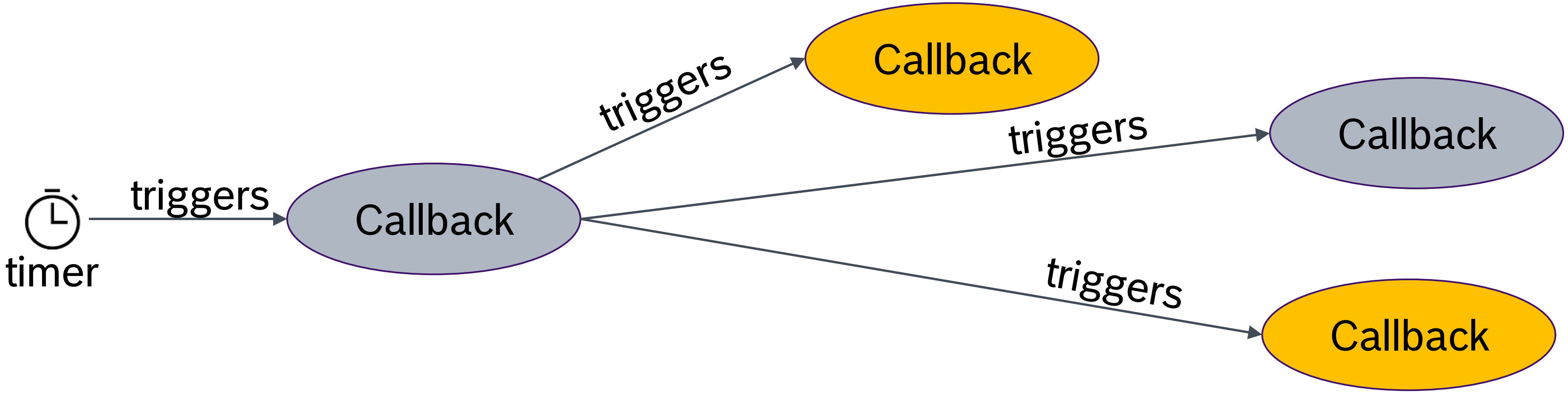
The Quest for a ROS 2 Response-Time Analysis
ROS Systems are distributed networks of callbacks



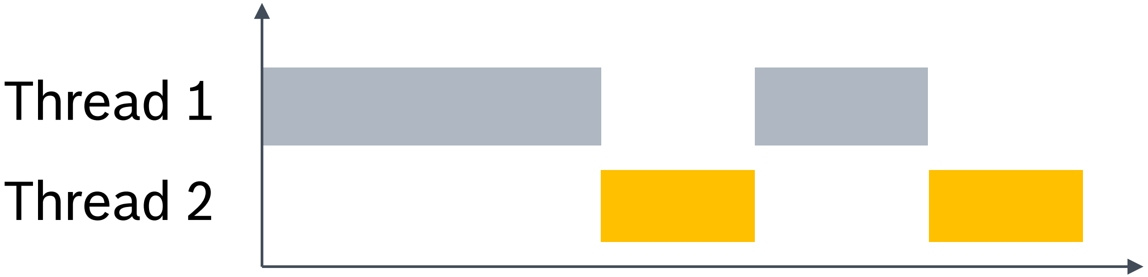
CR/AEE1 | 2018-12-11

The Quest for a ROS 2 Response-Time Analysis
Callbacks are assigned to *executor threads*

ROS-Level Scheduling

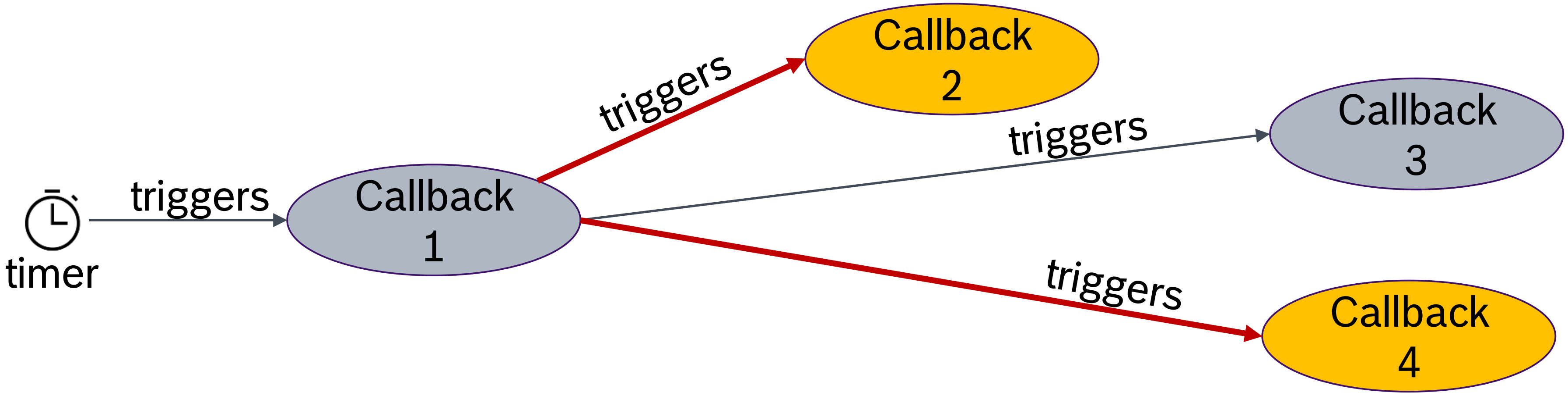


Linux-Level Scheduling



The Quest for a ROS 2 Response-Time Analysis

ROS-Level Scheduling



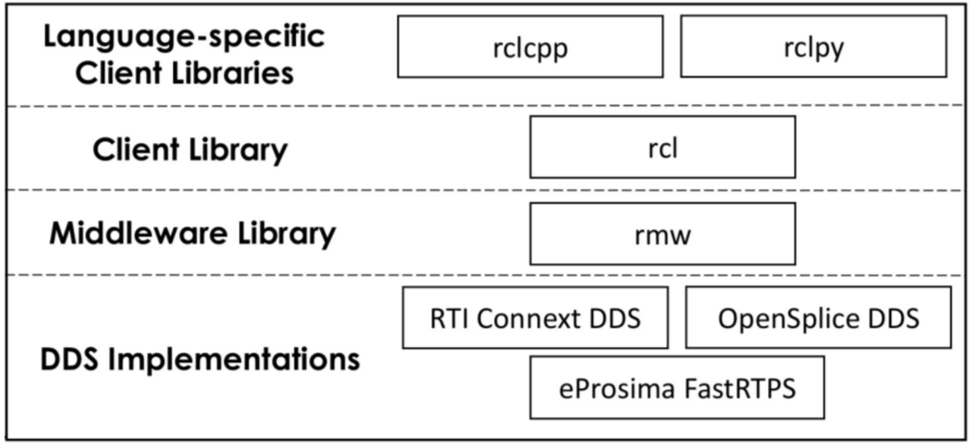
Is callback 2 or callback 4 executed first?

The ROS documentation **does not specify the execution order** of callbacks

BOSCH

The Quest for a ROS 2 Response-Time Analysis
Looking at the source code

The four layers of the ROS implementation

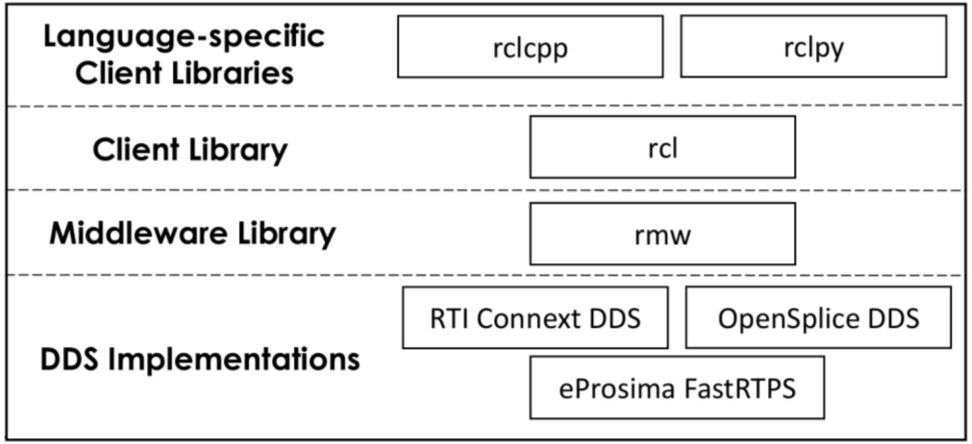


Question: Which layer determines the callback execution order?

Answer: All of them

The Quest for a ROS 2 Response-Time Analysis
Looking at the source code

The four layers of the ROS implementation



Callback execution order is **ROS-specific** and combines properties from

- Fixed-priority scheduling
- FIFO scheduling
- TDMA scheduling

Why not use a framework that prioritizes real-time?

	ROS	real-time robotics frameworks
Community size	**Huge**	**Small**
Effort to integrate third-party software	Low	High
Simulation support	Out-of-the-box, with ready-made models for many parts	None
Hardware Support	Lots of robotics hardware comes with ROS drivers	low-level Linux drivers
Predictability	Difficult	Easy

Guess which one people use?

CR/AEE1 | 2018-12-11

This Work
Make ROS 2 more predictable by

Understanding and documenting the **ROS 2 timing behavior**

Developing an **end-to-end response-time analysis** for ROS 2

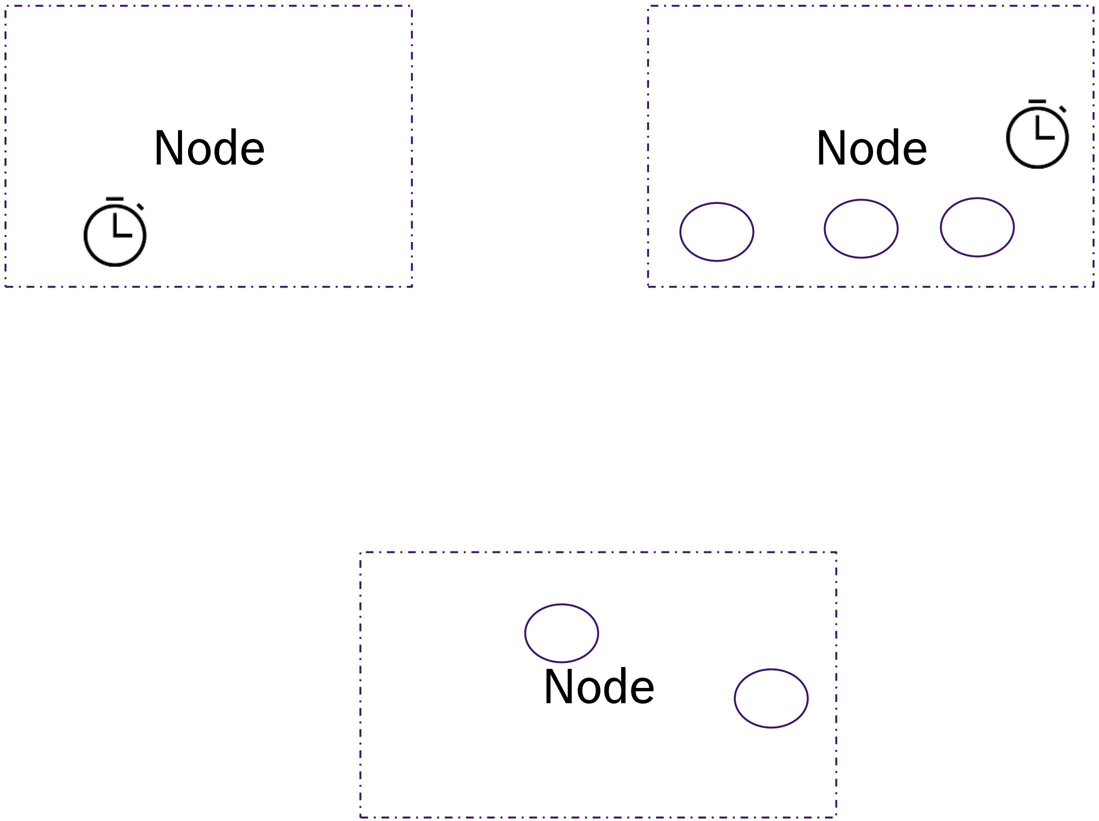
This Work
Make ROS 2 more predictable by

Understanding and documenting the
ROS 2 timing behavior

Developing an **end-to-end response-time analysis** for ROS 2

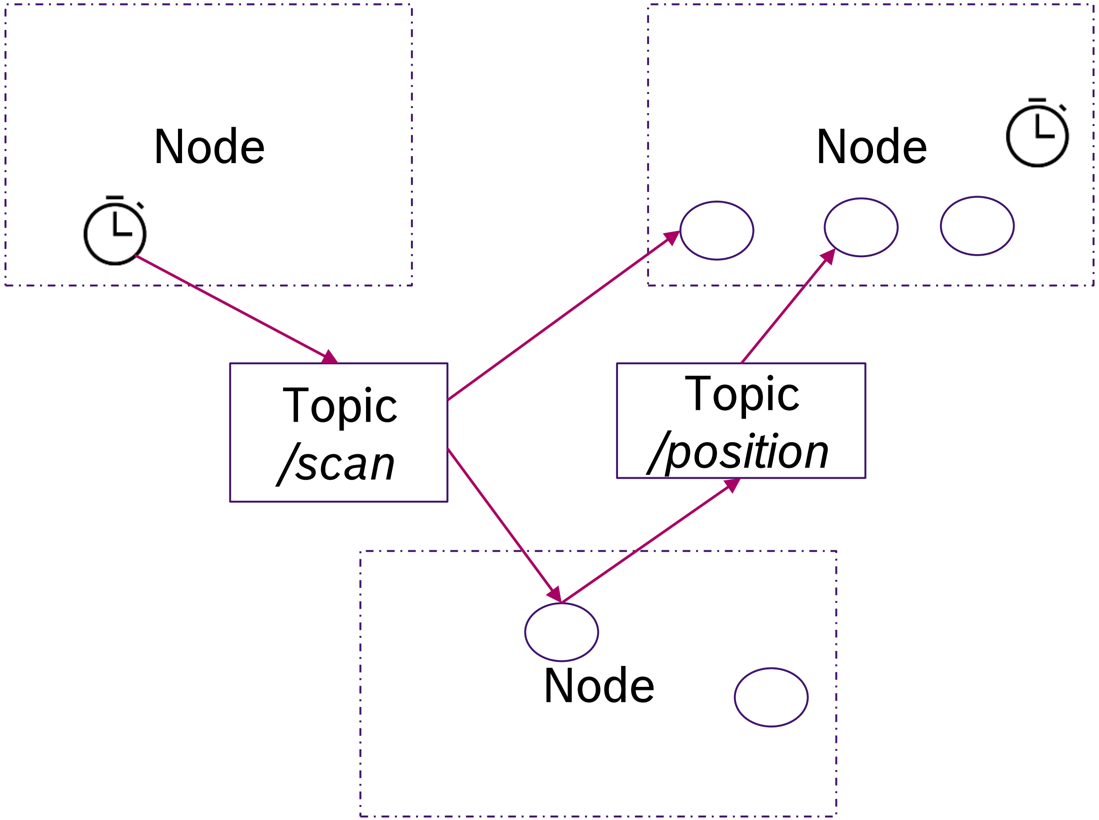
ROS Systems

ROS Systems consist of callbacks, grouped into nodes



ROS Systems

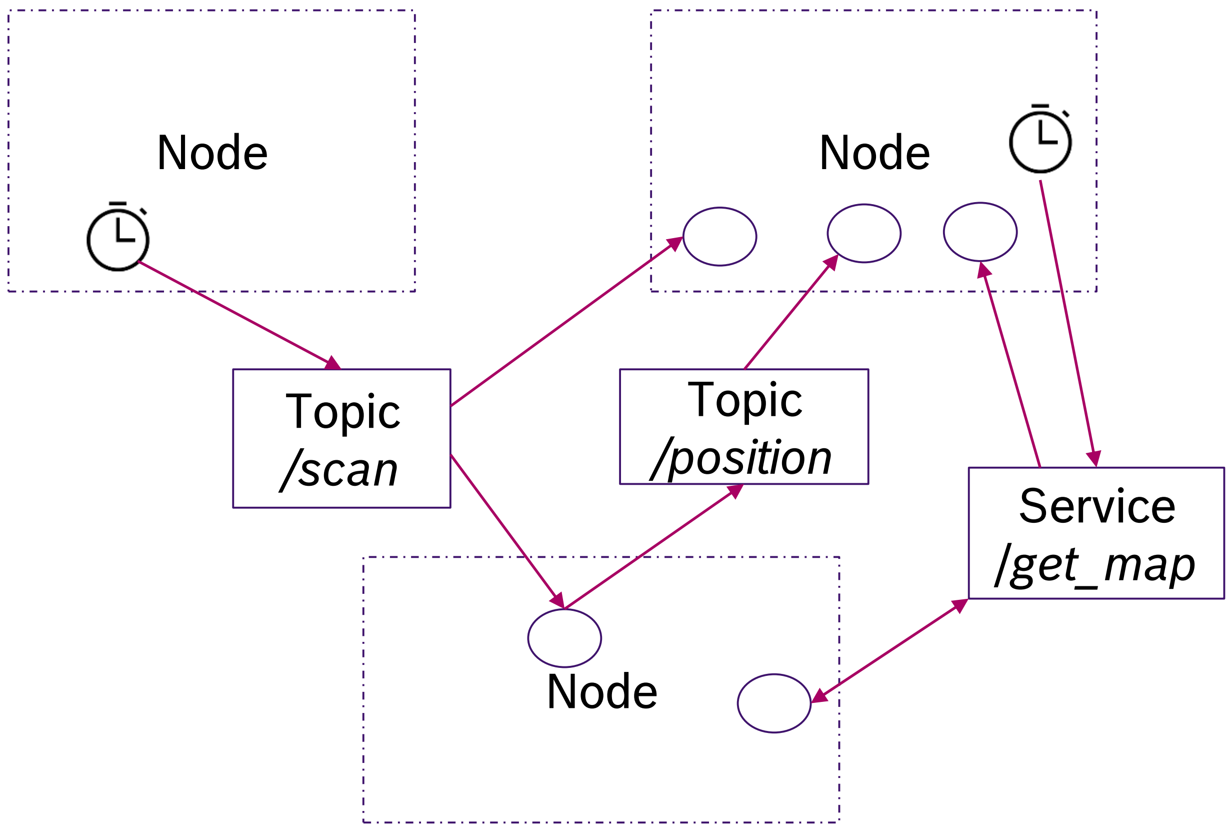
Nodes communicate using topics (a pub/sub mechanism) ...



CR/AEE1 | 2018-12-11

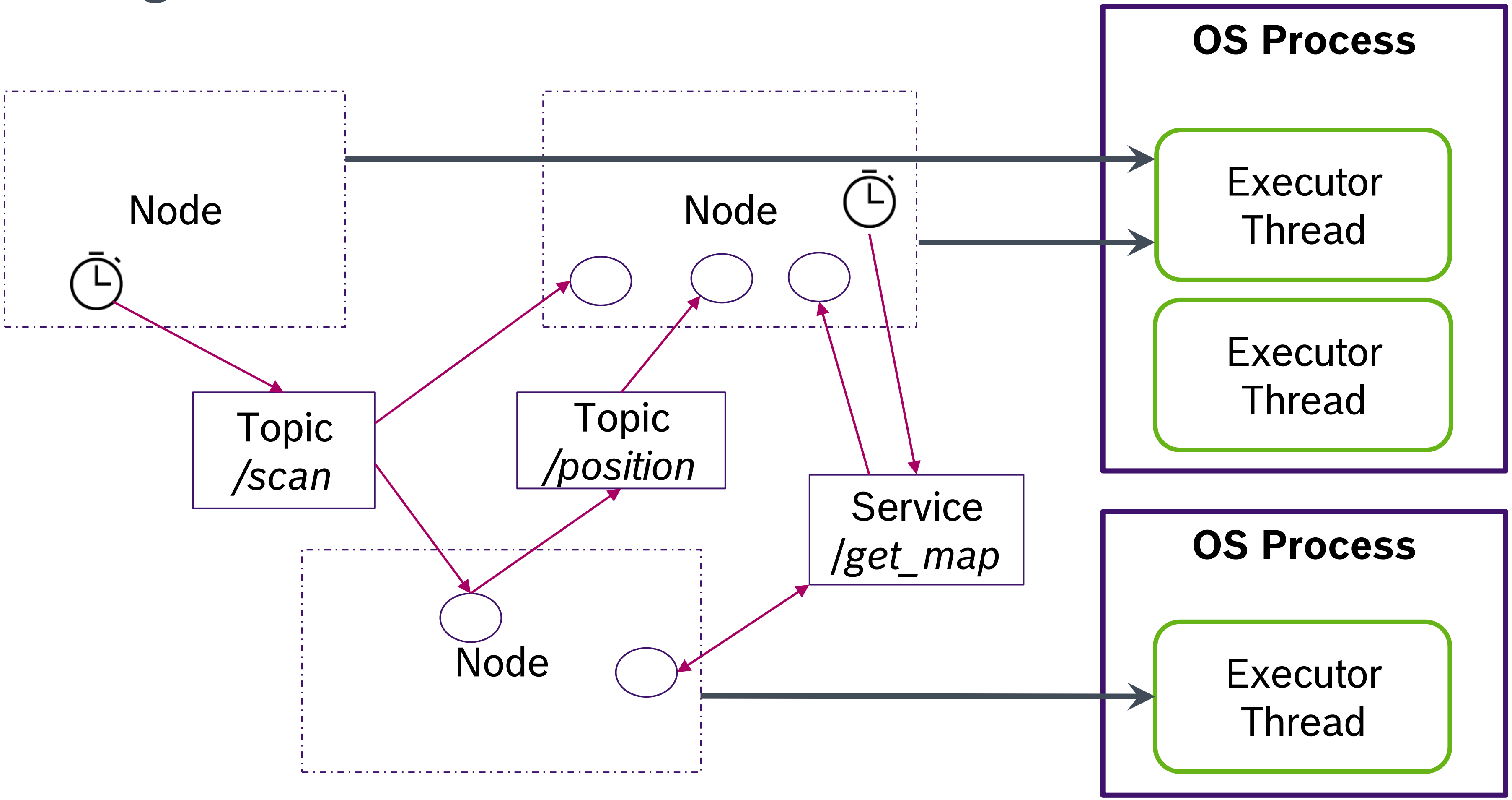
BOSCH

ROS Systems
… and services (remote procedure calls)



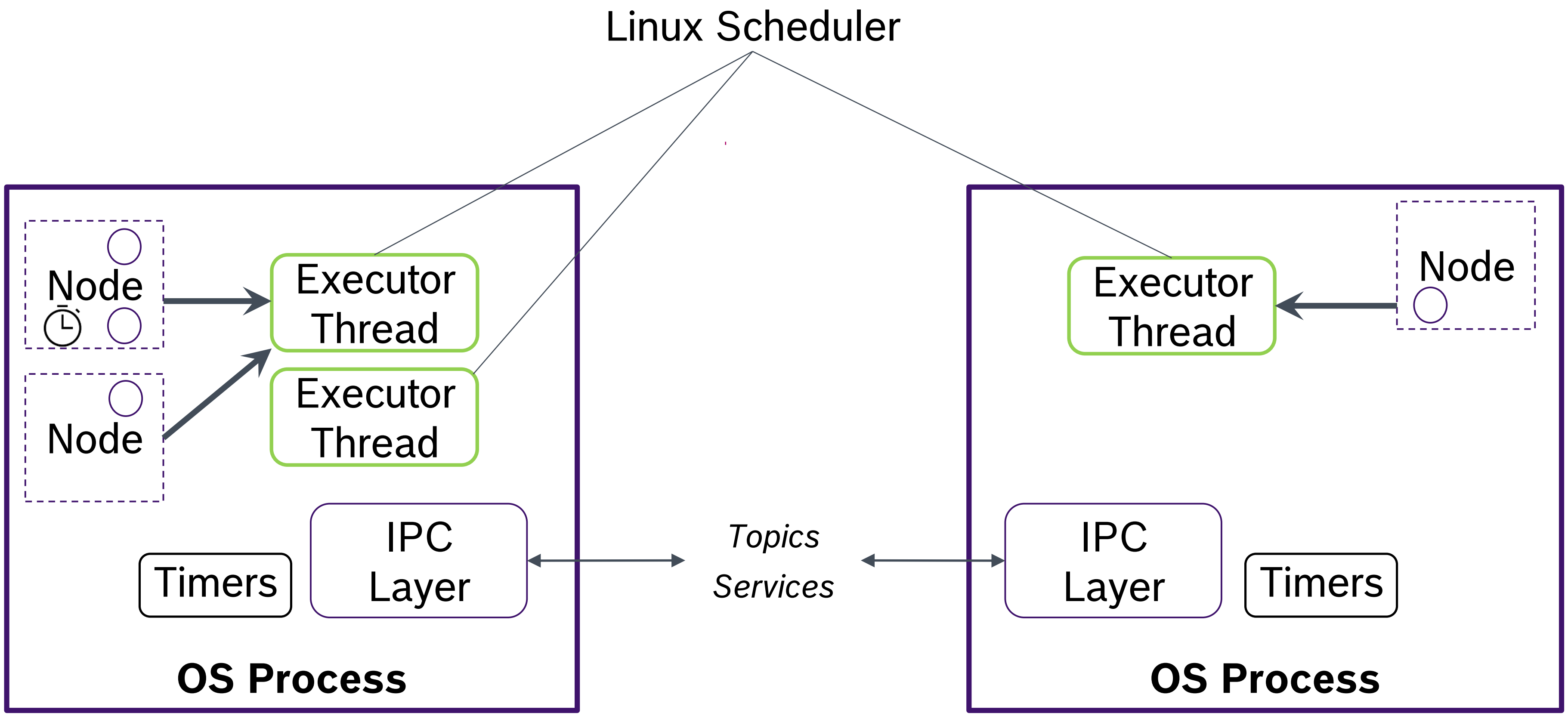
ROS Systems

Nodes are assigned to executor threads



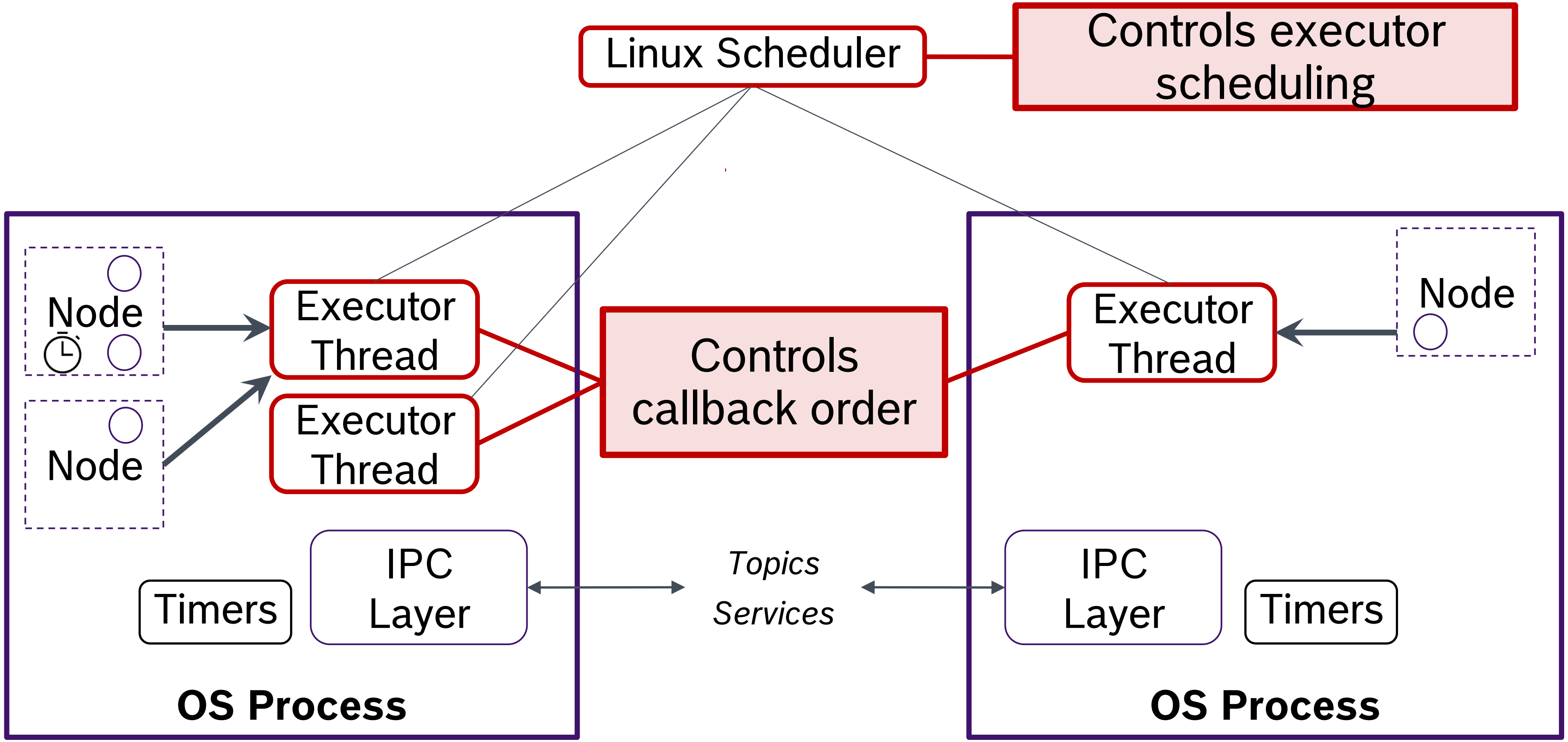
CR/AEE1 | 2018-12-11

ROS Systems
The operating system's view

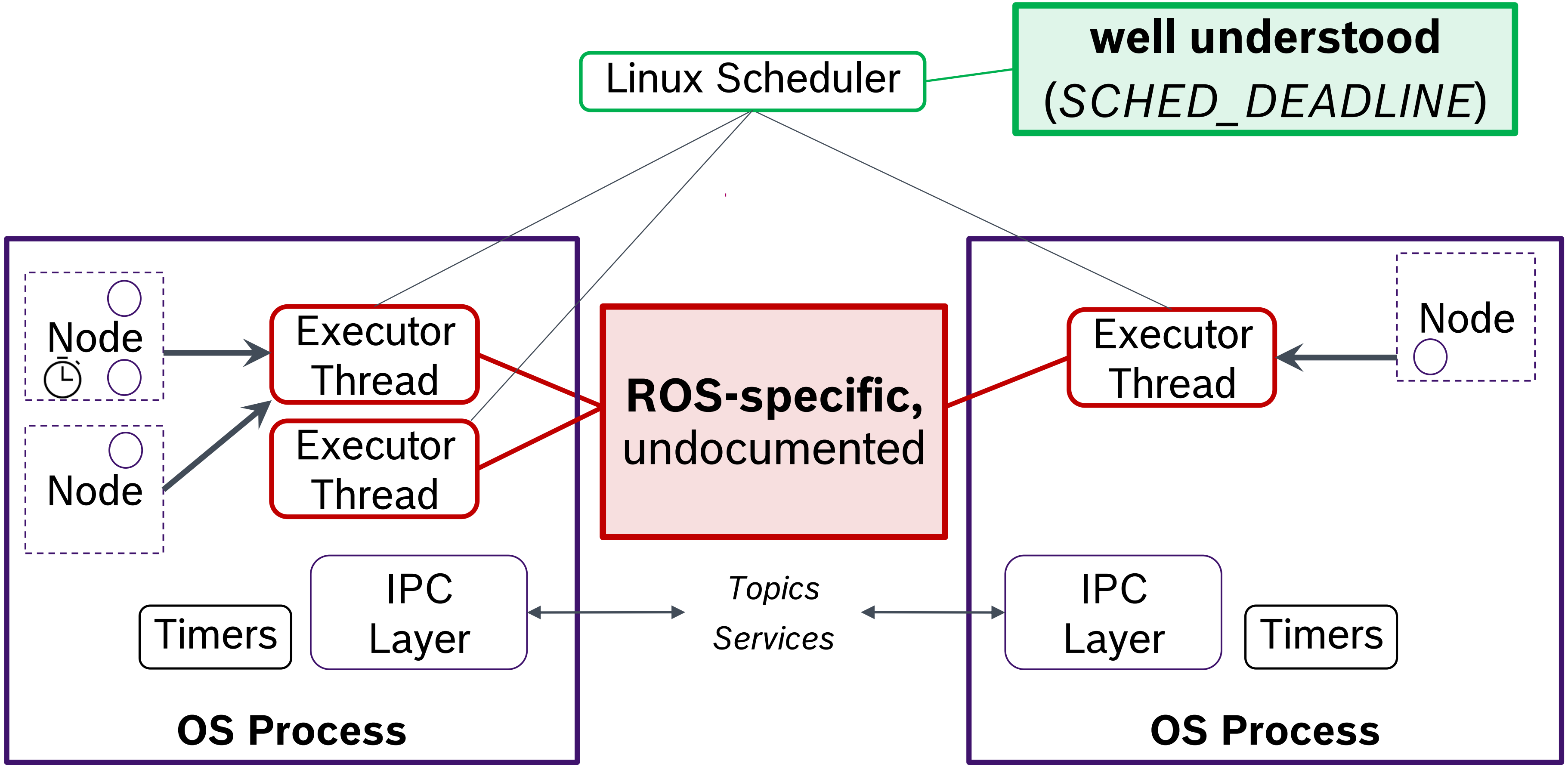


CR/AEE1 | 2018-12-11

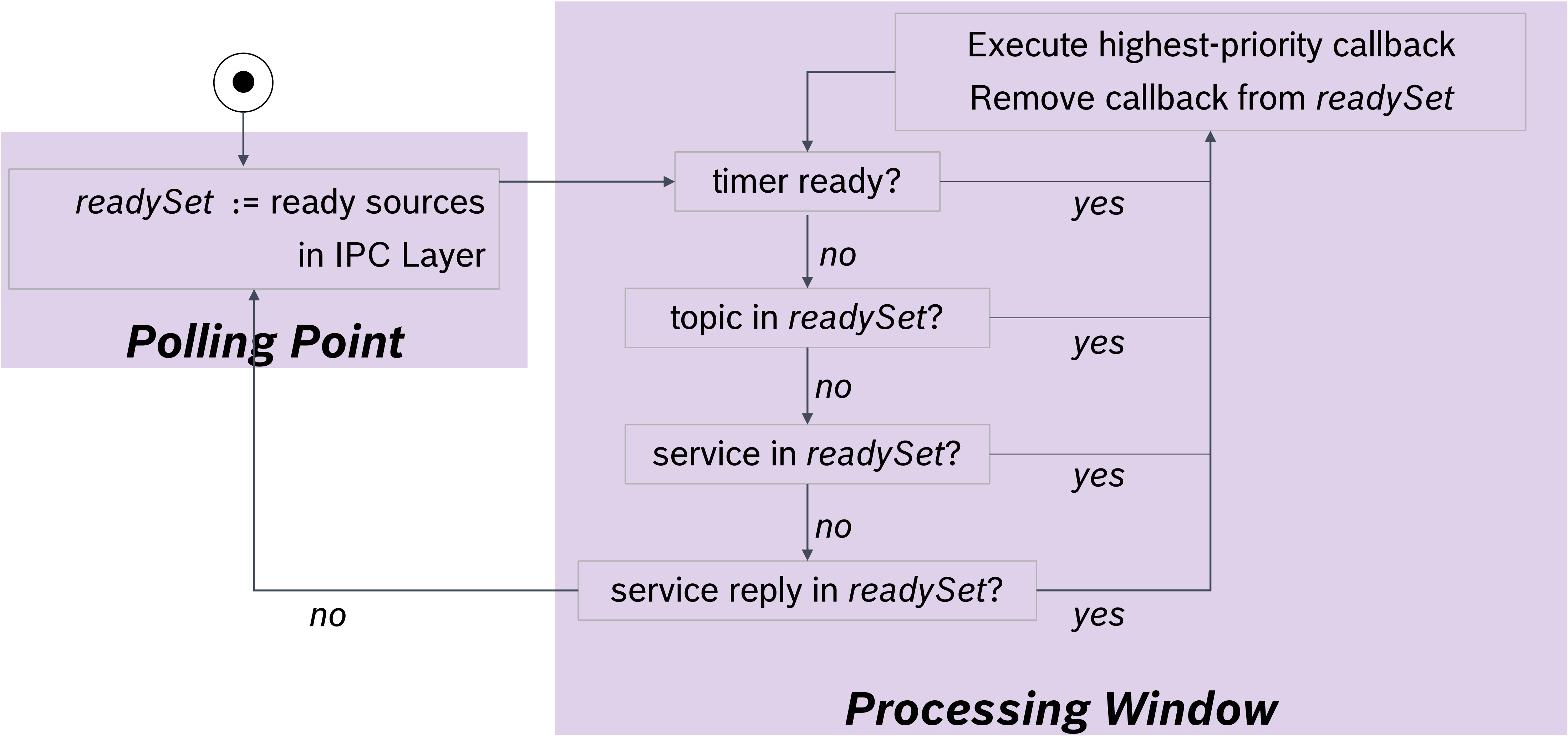
Understanding ROS's Timing Behavior
Who determines the execution order?



Understanding ROS's Timing Behavior
Who determines the execution order?

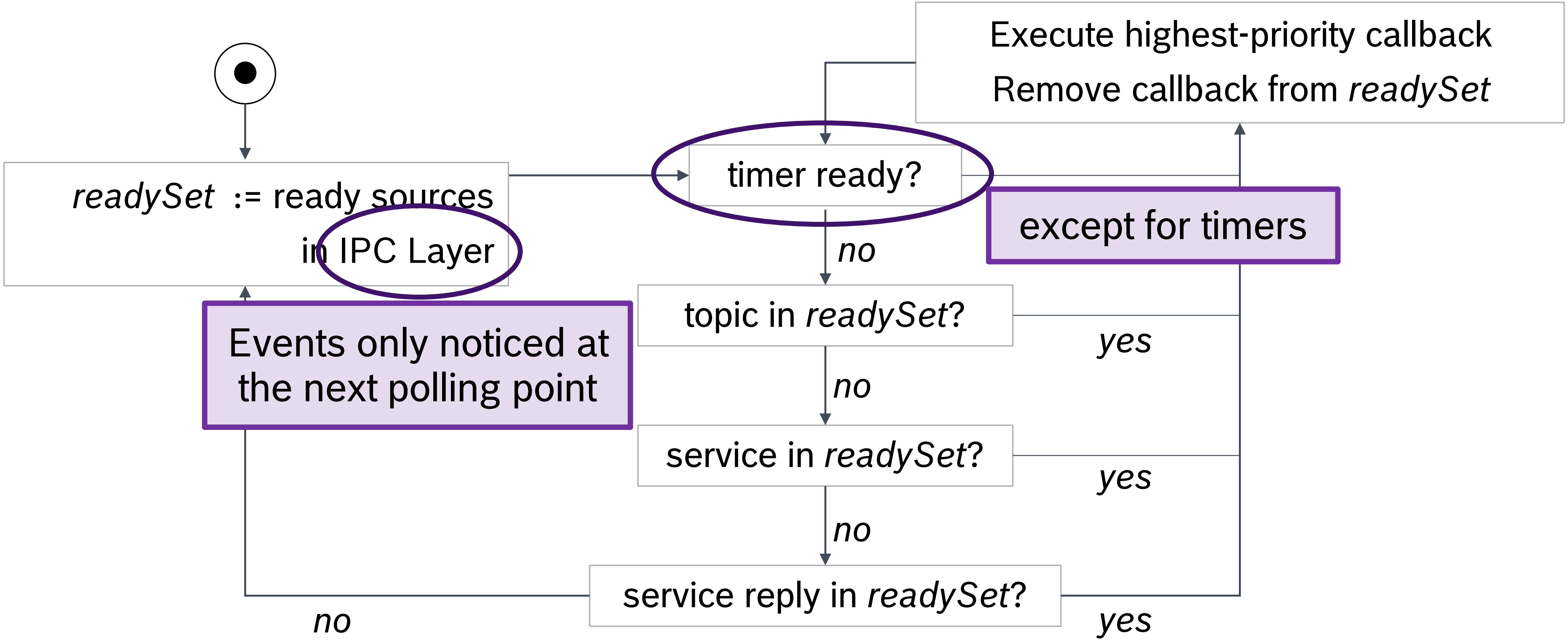


The Executor’s Algorithm



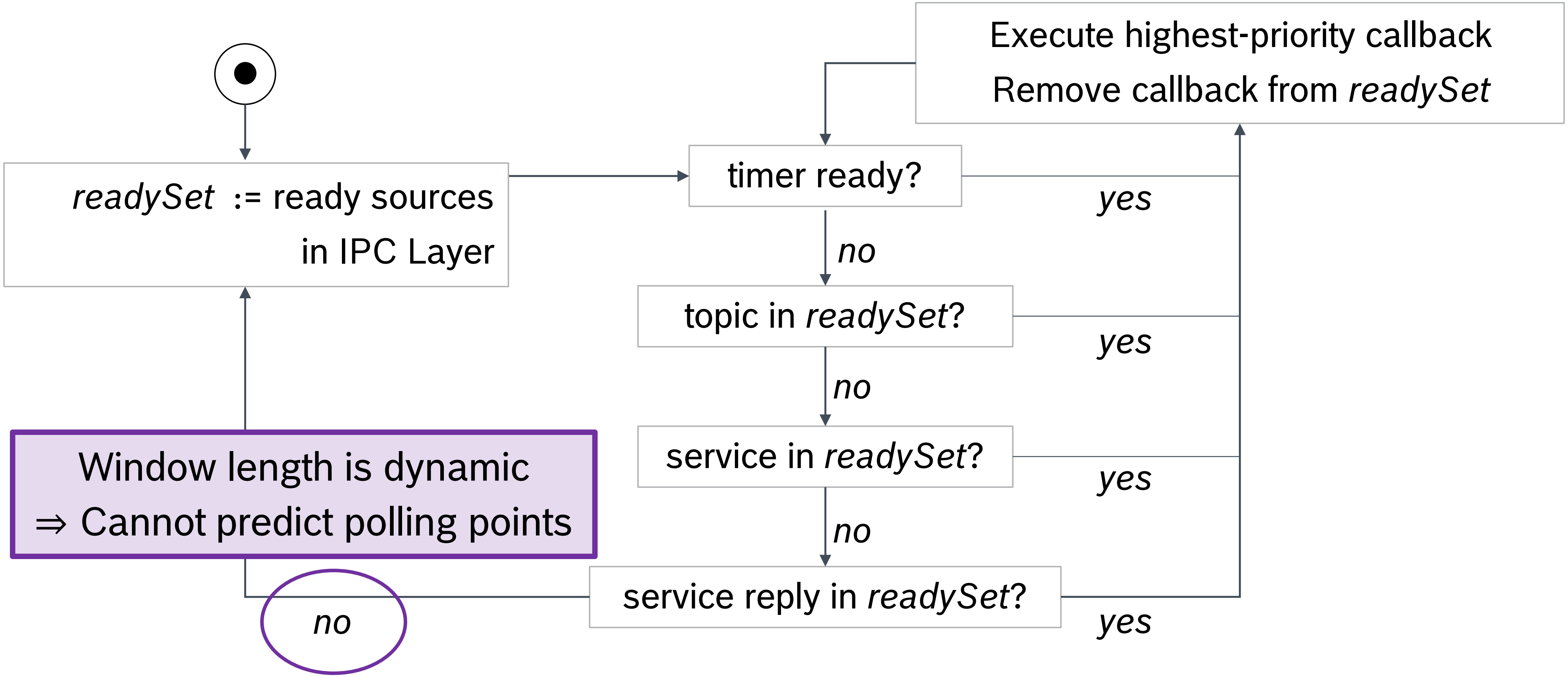
CR/AEE1 | 2018-12-11

Peculiarities of the Executor

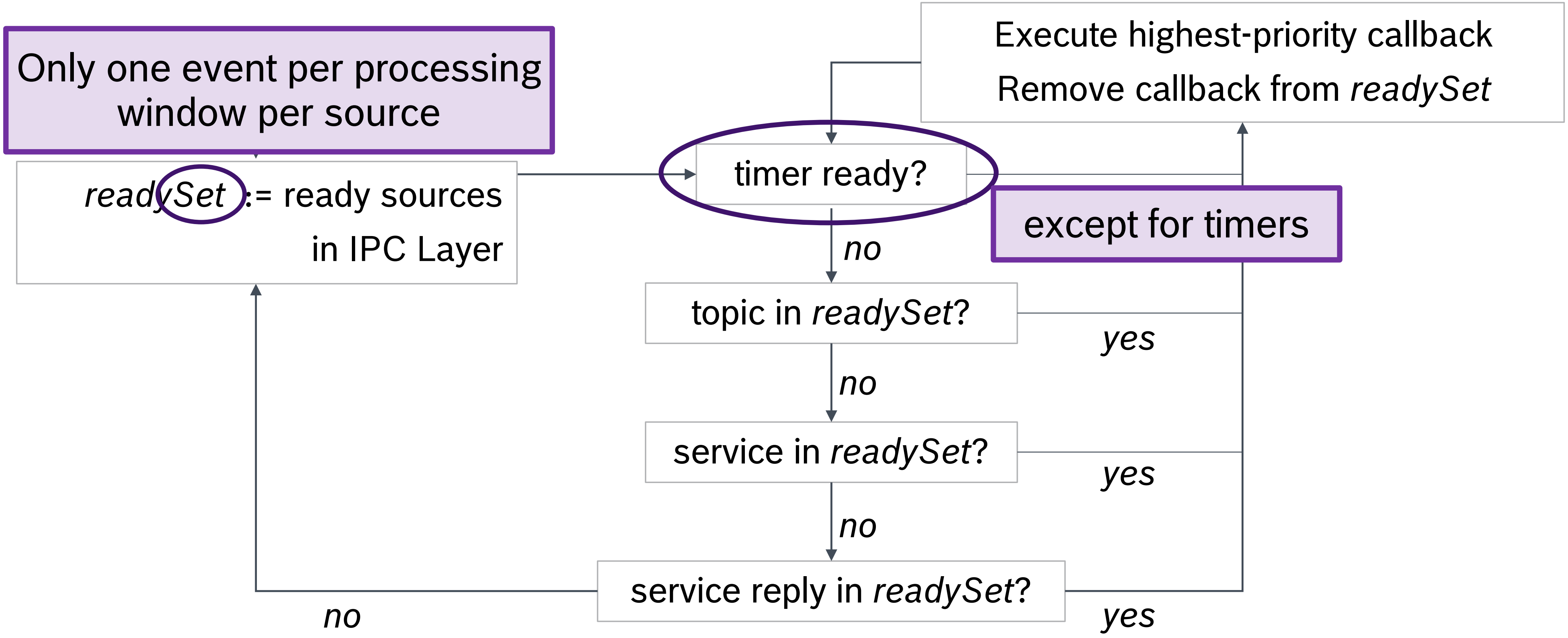


CR/AEE1 | 2018-12-11

Peculiarities of the Executor



Peculiarities of the Executor



CR/AEE1 | 2018-12-11

The Executor: Summary

The ROS 2 Executor **differs significantly from the usual schedulers**

Existing models and analyses cannot be applied directly:

- Existing models don’t capture the necessary information
- Existing analyses don’t consider the quirks of the ROS 2 Executor

The artifact contains an experiment that validates our model of the executor’s behavior

This Work
Make ROS 2 more predictable by

Understanding and documenting the **ROS 2 timing behavior**

Developing an **end-to-end response-time analysis** for ROS 2

Response-Time Analysis
Compositional Performance Analysis (CPA)

Henia et. al., 2005

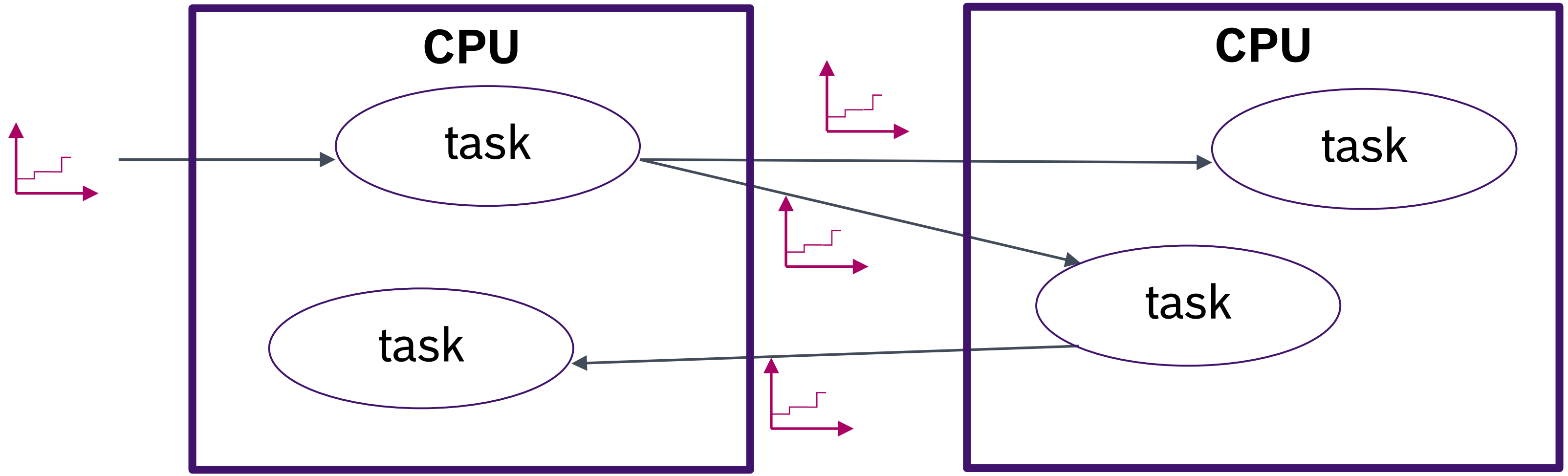
Per-Task Analysis

Computes per-task response times given event arrival curves

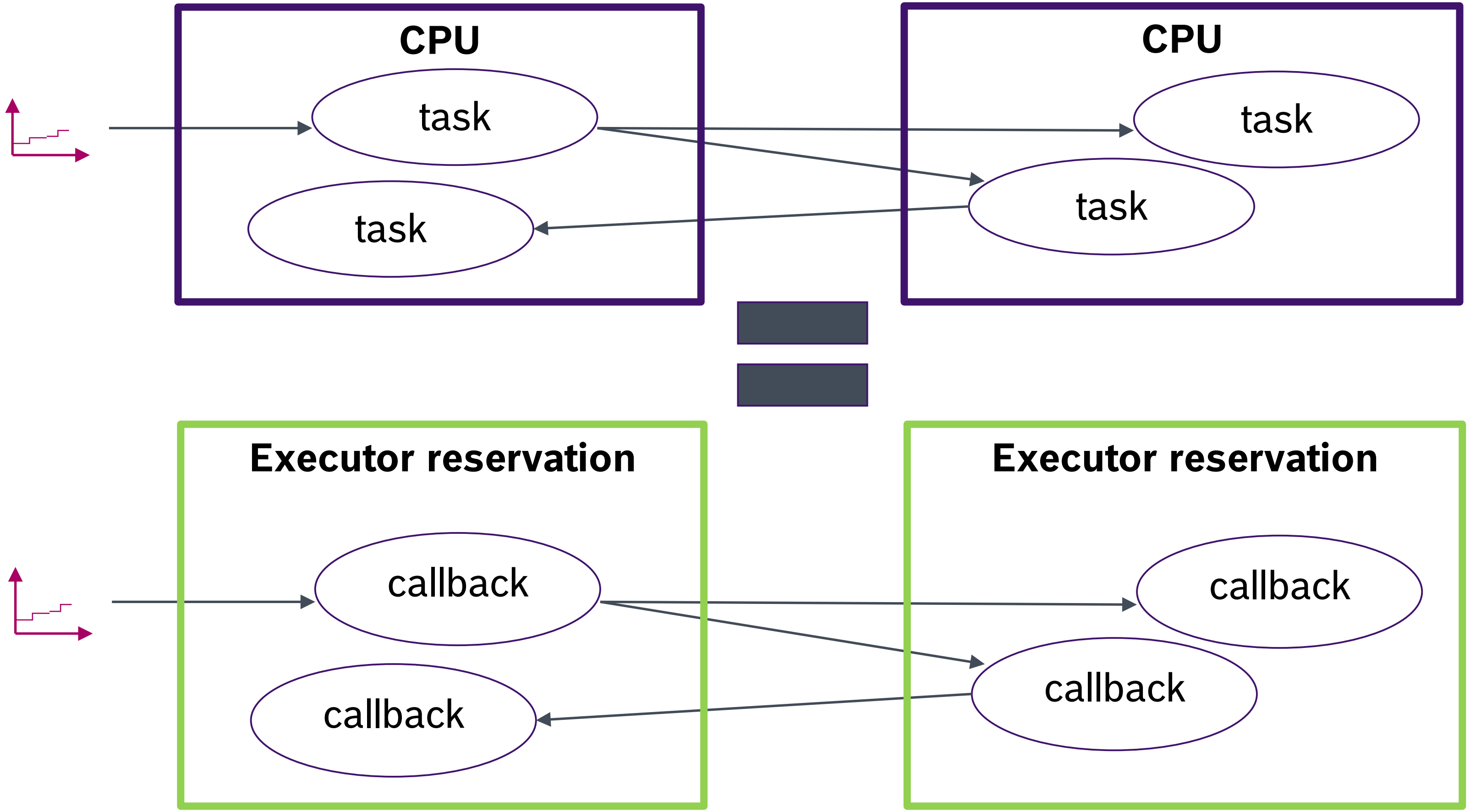
Fixed-point search

Arrival-Curve Propagation

Computes event arrival curves given per-task response times



Response-Time Analysis
The CPA approach fits ROS well



Response-Time Analysis
Extending CPA for ROS

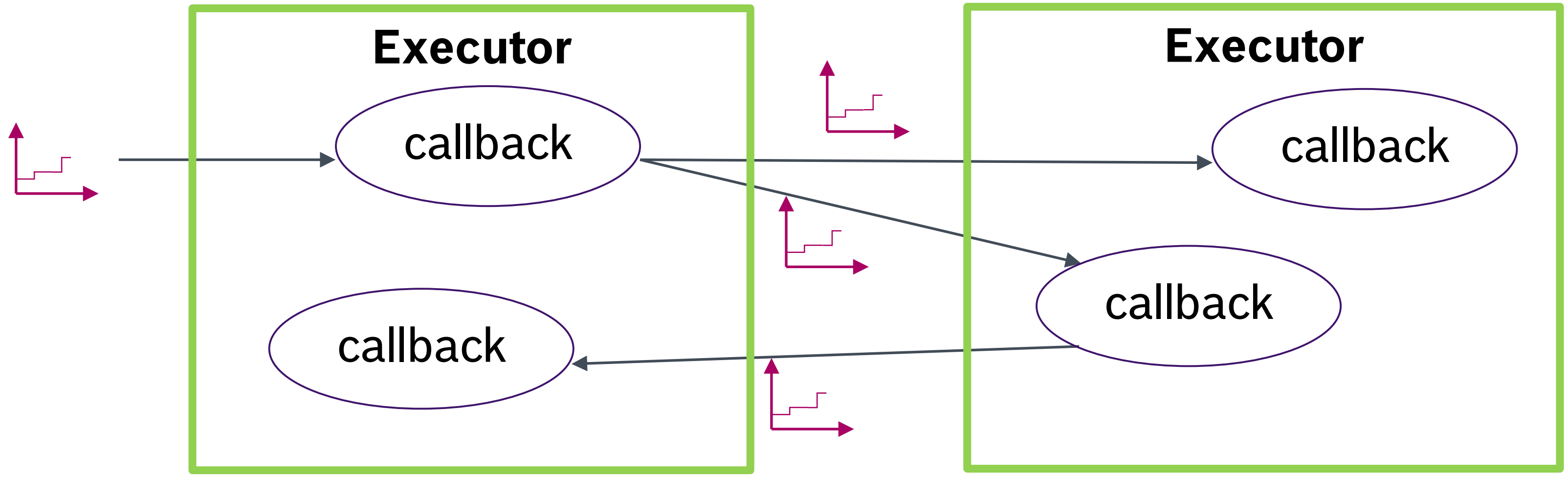
Per-Callback Analysis

Needs to account for ROS's quirks

Fixed-point search

Arrival-Curve Propagation

Inherited from CPA unchanged



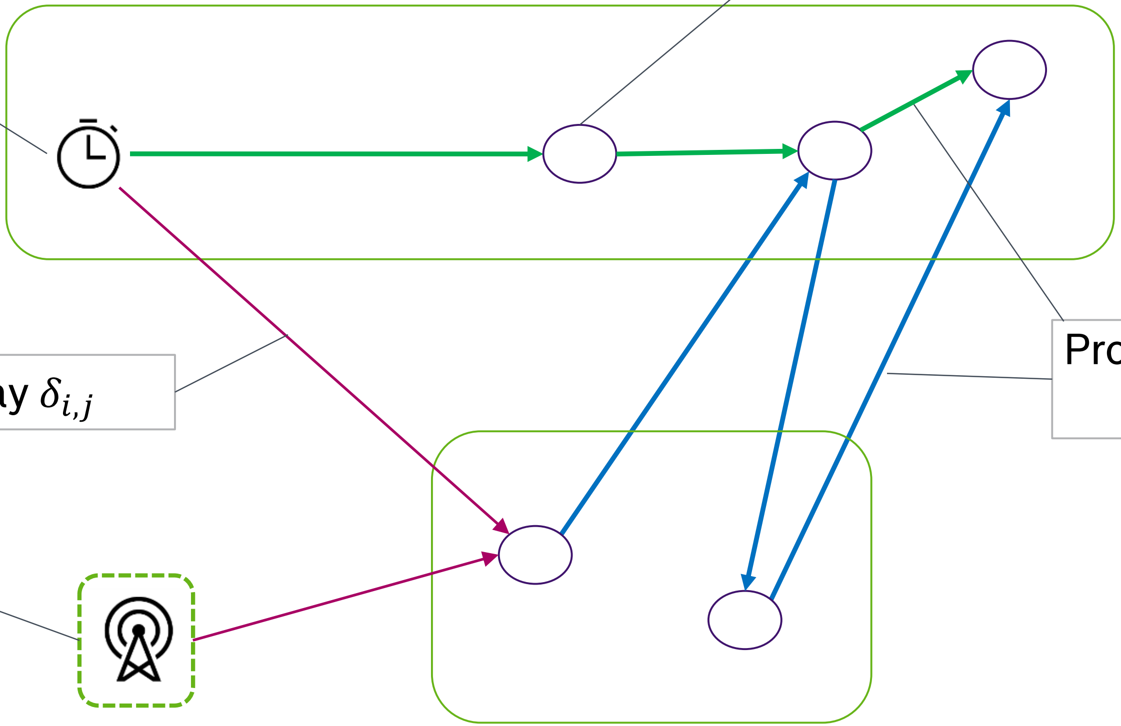
CR/AEE1 | 2018-12-11

BOSCH

Response-Time Analysis
A real-time model for ROS 2

Subscription callback
- WCET bound e_i
- Priority π_i

Timer callback
- WCET bound e_i
- Priority π_i
- Event arrival curve η_i^a



Communication Delay $\delta_{i,j}$

Processing chains of interest

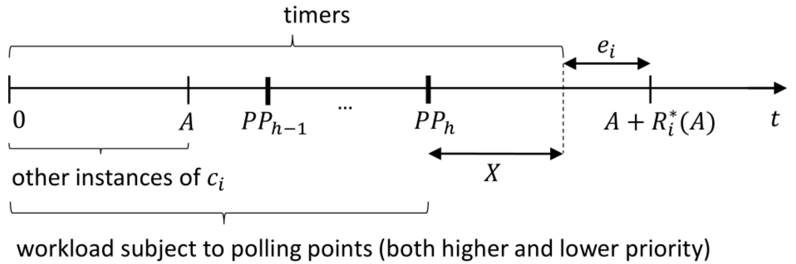
Event source external to ROS
- Event arrival curve η_i^a

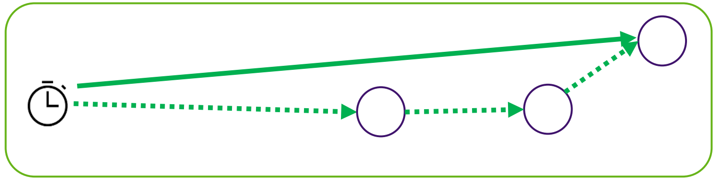
Response-Time Analysis
A per-callback response-time analysis for ROS

$$sbf_k(A + R_i^*(A)) = rbf_i(A+1) + RBF(hp_k(c_i), A + R_i^*(A) - e_i + 1) + B_i$$

$$sbf_k(A + R_i^*(A)) = rbf_i(A+1) + RBF(\{\mathcal{C}_k \setminus c_i\}, A + R_i^*(A) - e_i + 1)$$

Dedicated analyses for **timers and polling-point-based callbacks**



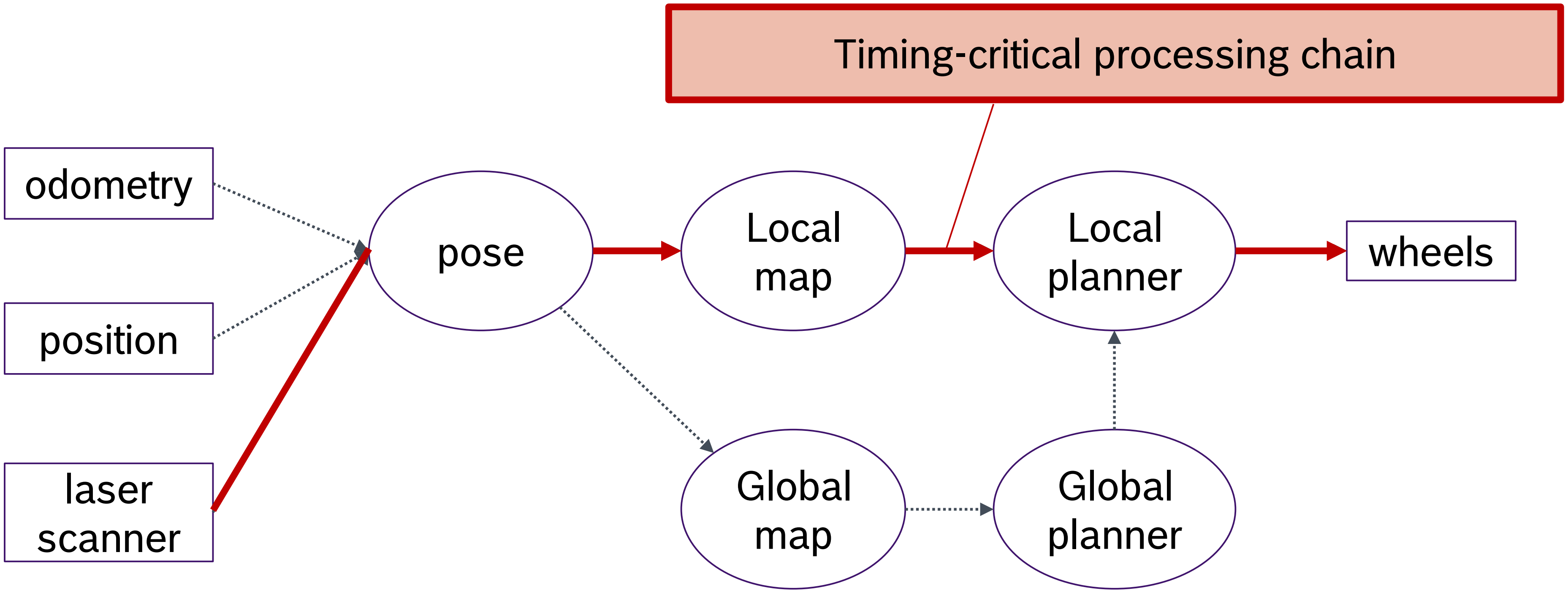


Busy-window analysis **without critical instant assumptions**

Optimization for **intra-executor** chains

Case Study:
The *move_base* Navigation Stack

Processing Steps



Can we implement this safely in ROS 2?

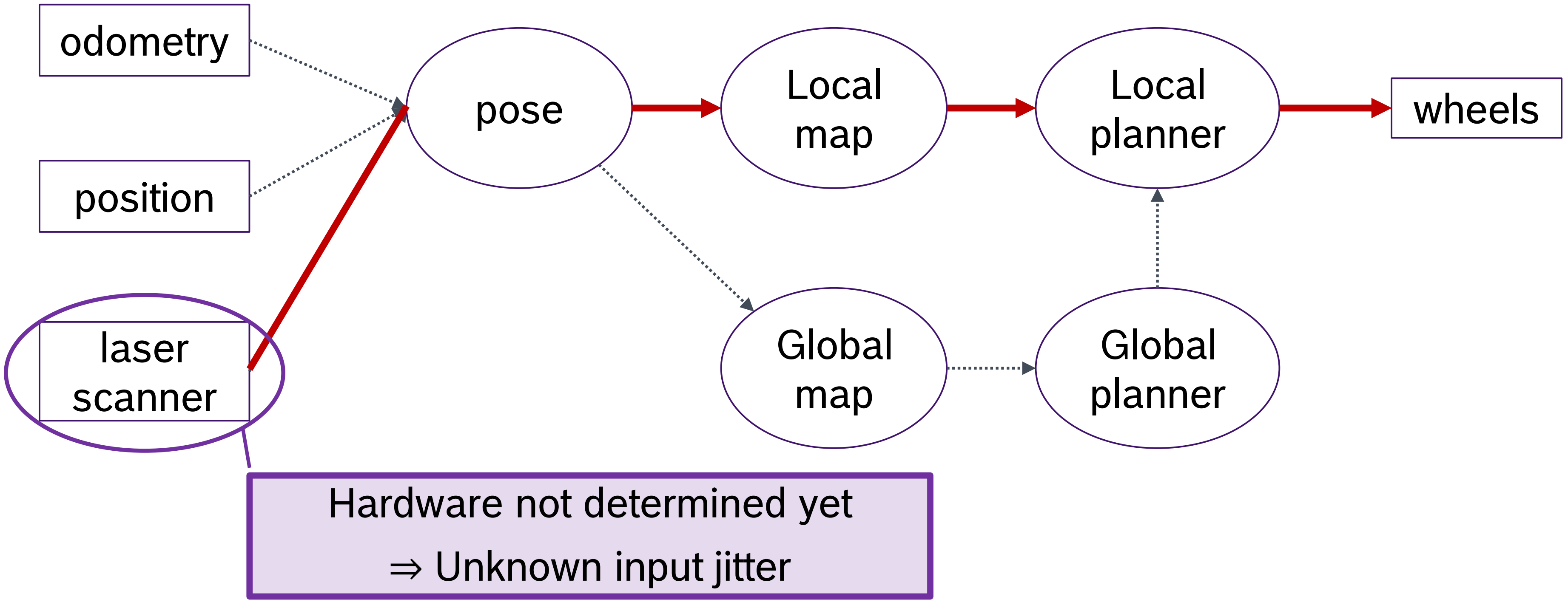
Our Needs for Real-Time Control

1. Support for automated timing **validation**
   - Does this robot react in time?

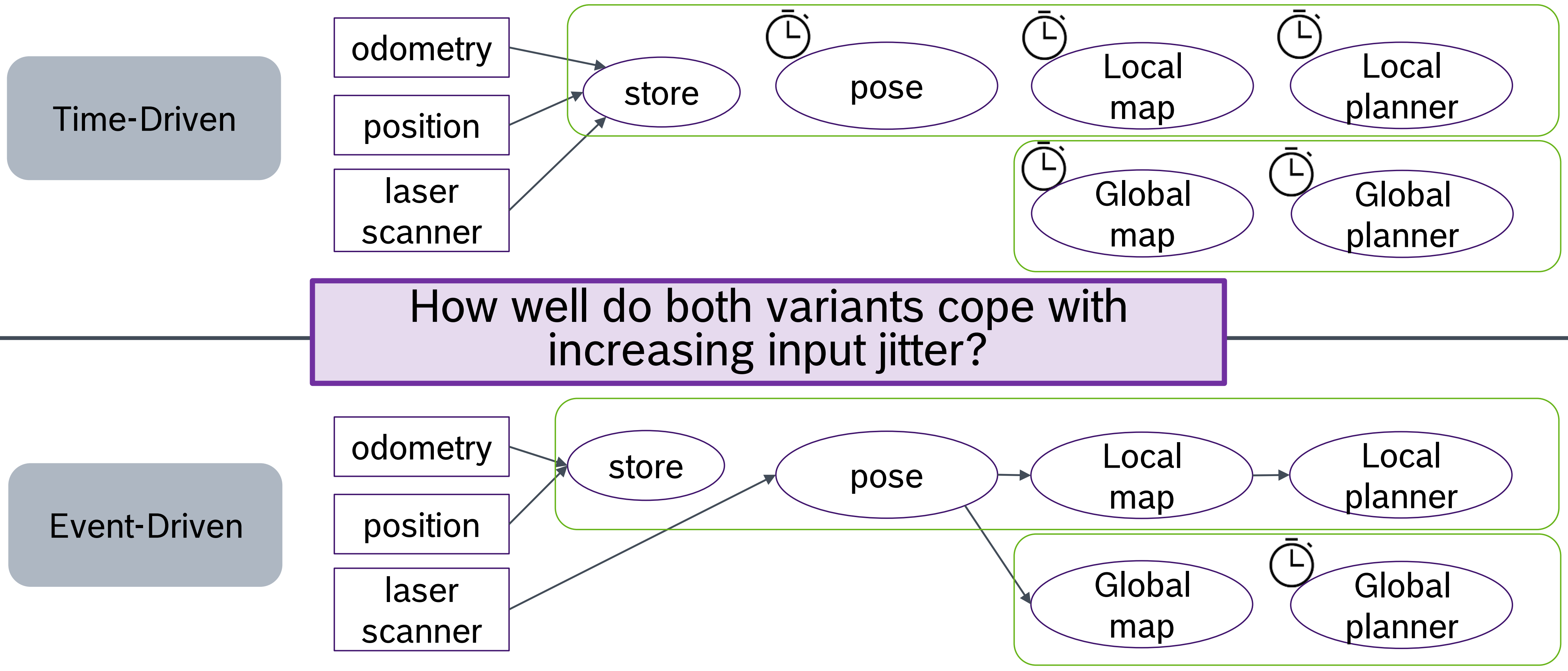
2. Support for model-based **design space exploration**
   - Would this robot respond in time if I used this hardware?
   - Would this robot respond in time if I implemented it that way?

BOSCH

Hardware Uncertainty

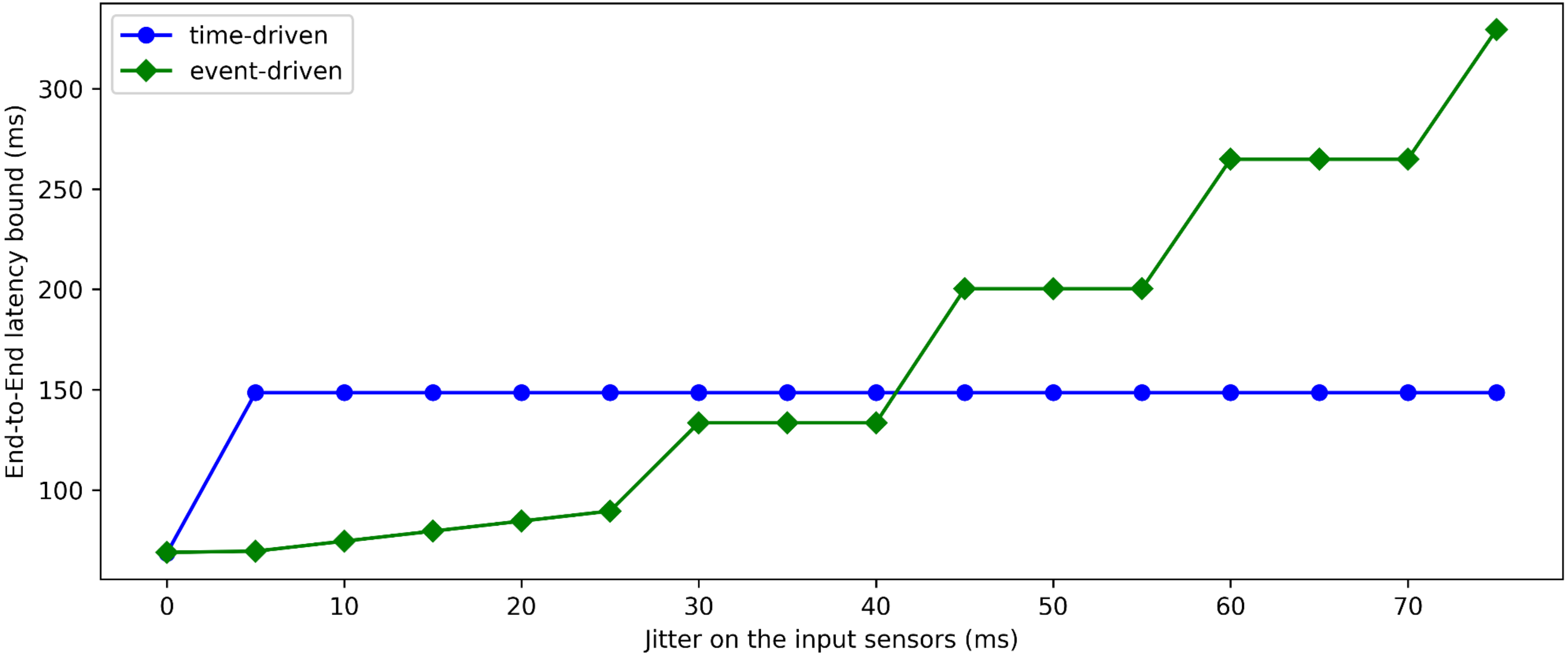


Two Implementation Choices

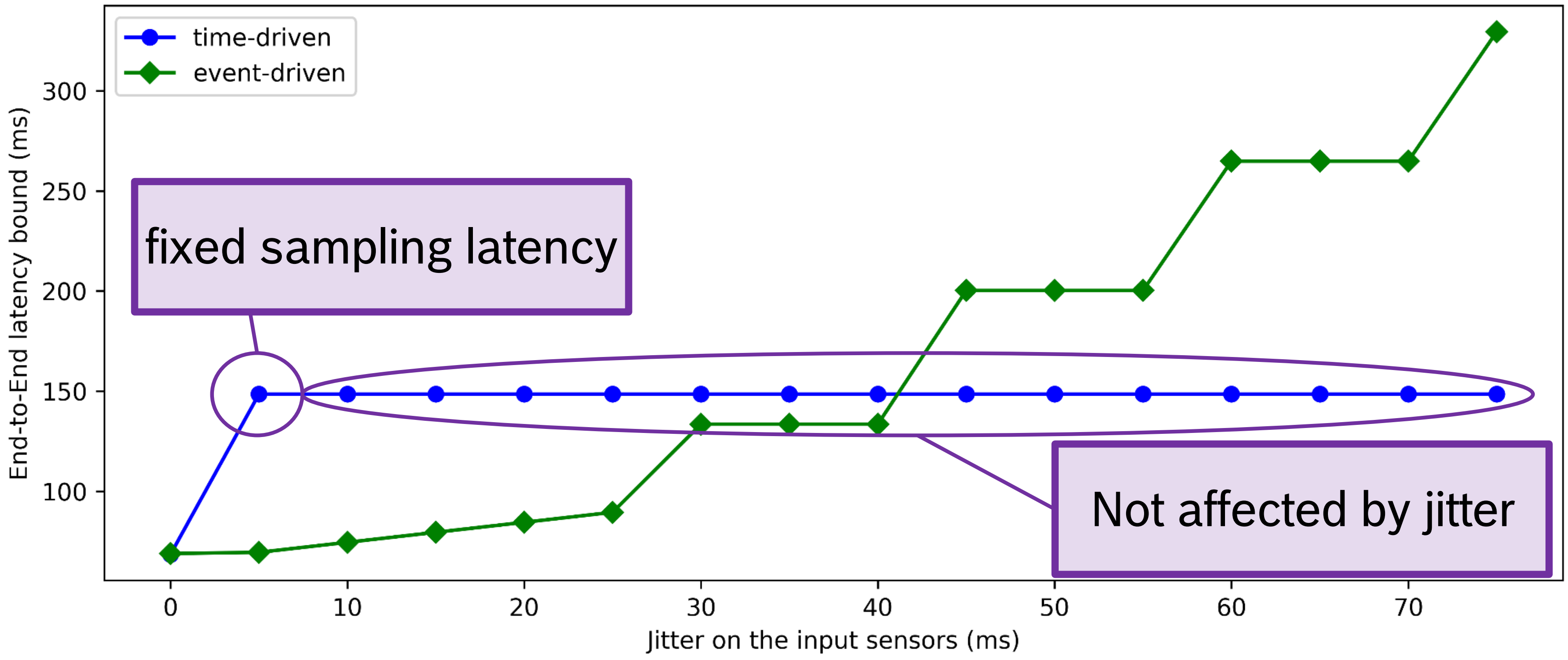


BOSCH

Exploring Jitter Sensitivity



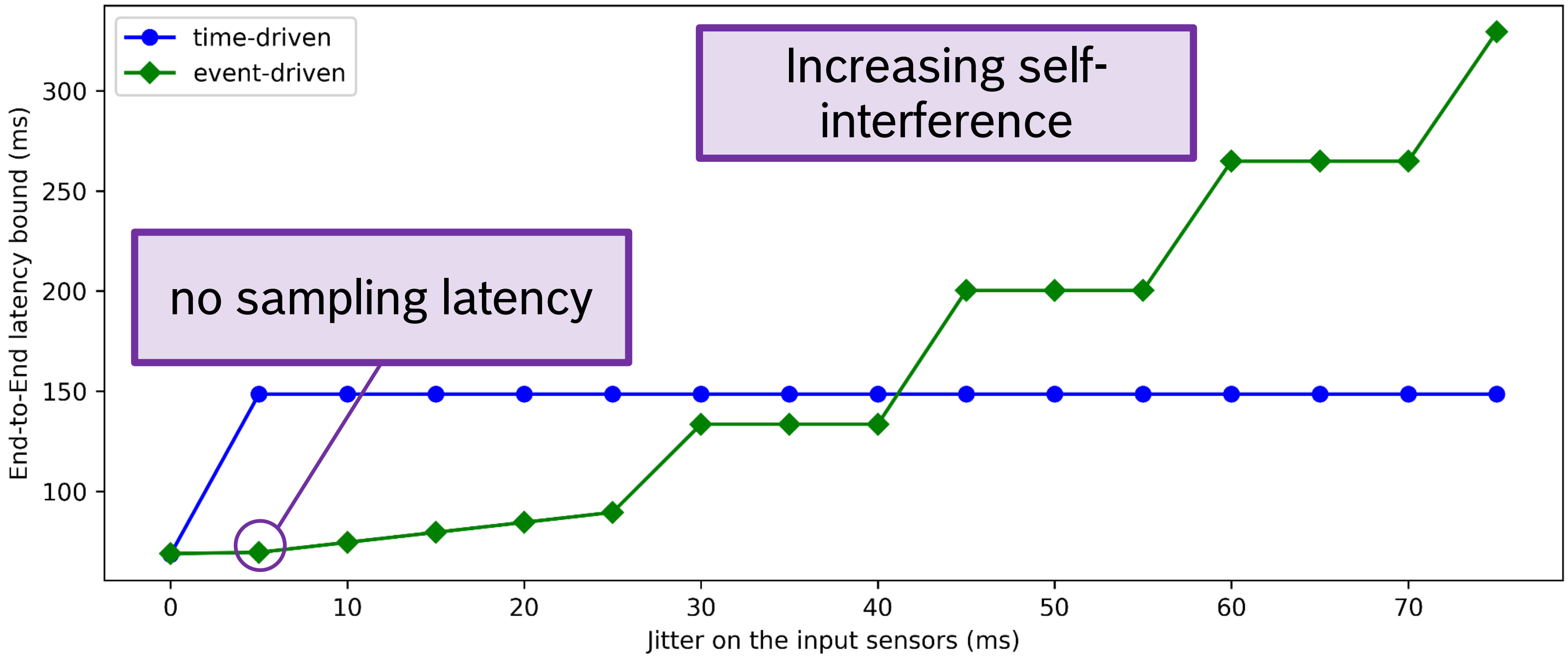
Exploring Jitter Sensitivity



CR/AEE1 | 2018-12-11

BOSCH

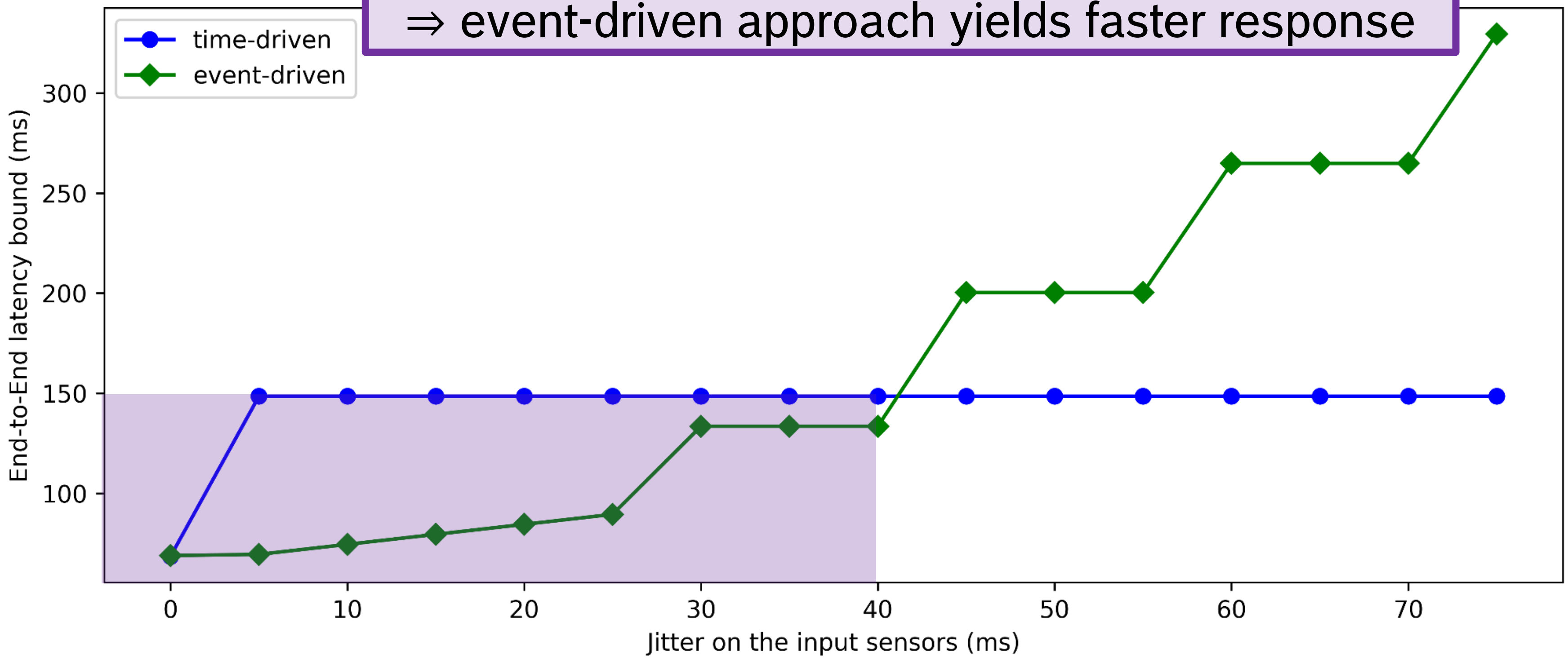
Exploring Jitter Sensitivity



Exploring Jitter Sensitivity

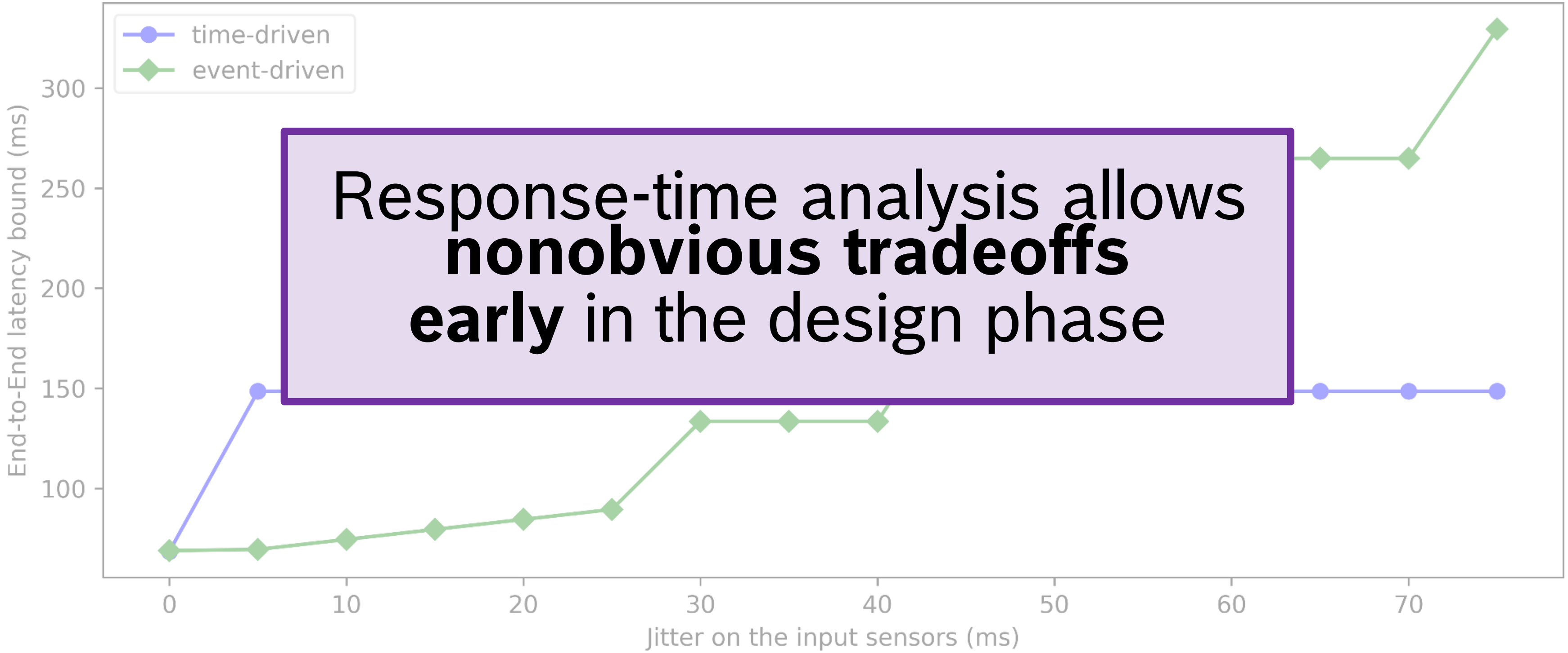
“All laser scanners have jitter <40ms”

⇒ event-driven approach yields faster response



BOSCH

Exploring Jitter Sensitivity



Response-time analysis allows **nonobvious tradeoffs early** in the design phase

BOSCH

Future Work

Extract the timing model automatically

- static analysis or runtime system introspection

Develop a real-time executor

- Discussions with ROS developers underway (Join us!)

Summary

Contributions

- A comprehensive description of the **ROS 2 execution model**
- A **response-time analysis** for ROS 2 applications

This work enables **ROS users** to

- **Explore the design space** cheaply and early
- Provide **safety guarantees** for their applications

This work provides **real-time researchers** with

- A **task model** that **reflects the leading robotics framework**

Source code available at *https://github.com/boschresearch/ros2_response_time_analysis*