Priority-Driven Real-Time Scheduling in ROS 2: Potential and Challenges

Hyunjong Choi, Daniel Enright, Hoora Sobhani,

Yecheng Xiang, and Hyoseung Kim

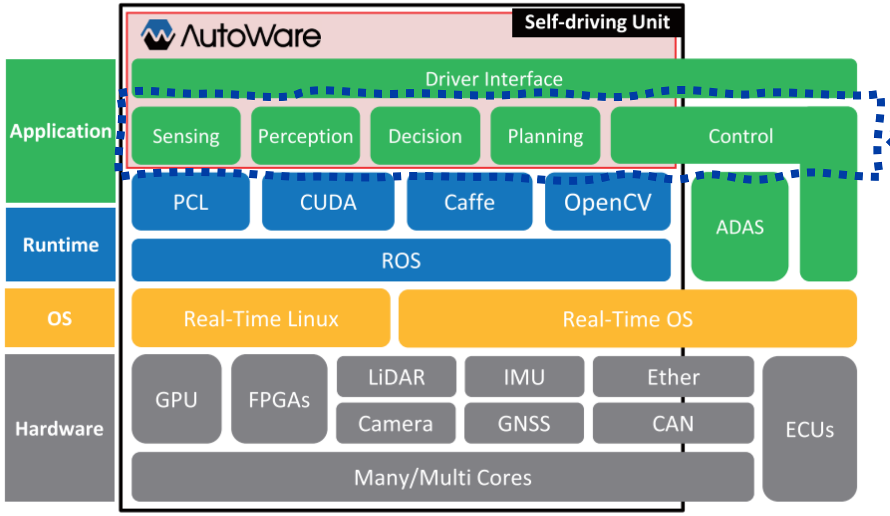
ROS

- One of the most prevalent robotic middleware frameworks

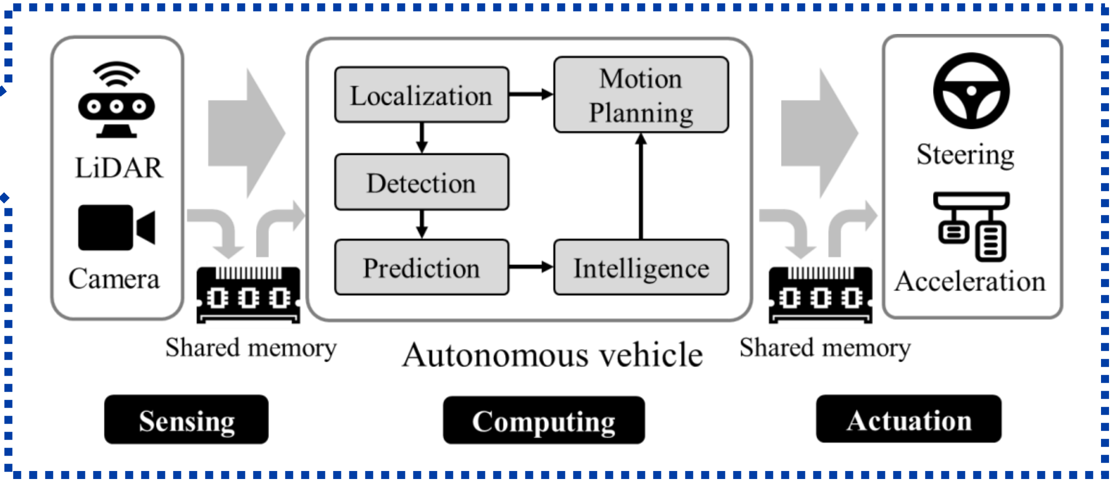
Galactic Geochelone, released May 2021

- ***Predictable end-to-end behavior*** of systems is essential for robotic applications

Revealed shortcomings in real-time support for safety-critical applications



< Example of ROS-based robotic framework (Autoware.Ai) > †



< Chain in self-driving application >

Violating timing constraints (e.g., end-to-end latency) can cause catastrophic accidents.

DESIGN AUTOMATION CONFERENCE 59

†S. Kato et al. "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems", ICCPS, 2018

Limitations of current ROS 2

- Priority-unaware complex layers of abstractions
  - Round-robin like callback scheduling behavior
  - Prone to priority inversion

  ➡ **Ignores criticality or urgency of processing chains**

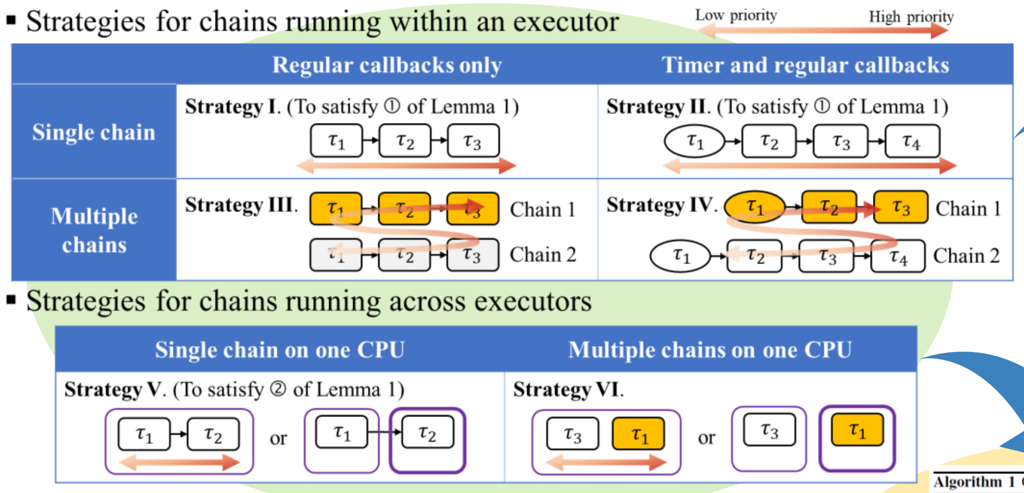
- Lack of systematic support for resource allocation
  - All nodes compete for resources in a nondeterministic way

  ➡ **Long end-to-end latency and poor resource utilization**

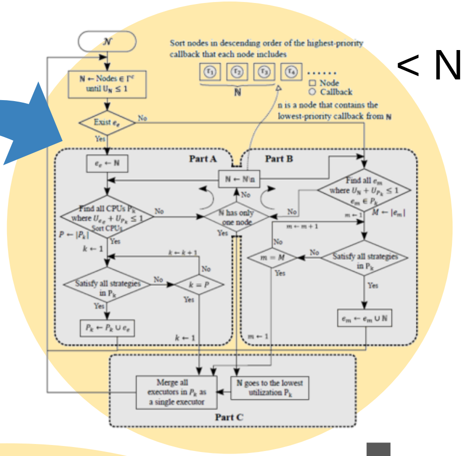
➡ **We need a priority-driven paradigm for real-time support in ROS 2!**

Priority-driven scheduling framework for ROS 2

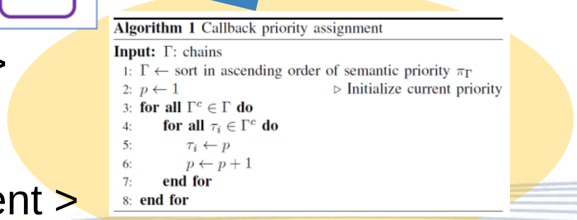
- Priority-driven chain-aware scheduling (PiCAS)†: enables *prioritization of critical computation chains* across complex abstraction layers of ROS 2
  - Minimizes end-to-end latency
  - Ensures predictability even when the system is overloaded



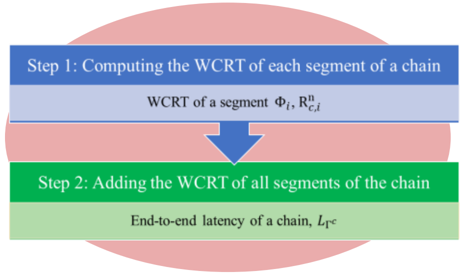
< Chain-aware scheduling strategies >



< Node-to-Executor allocation >



< Priority assignment >

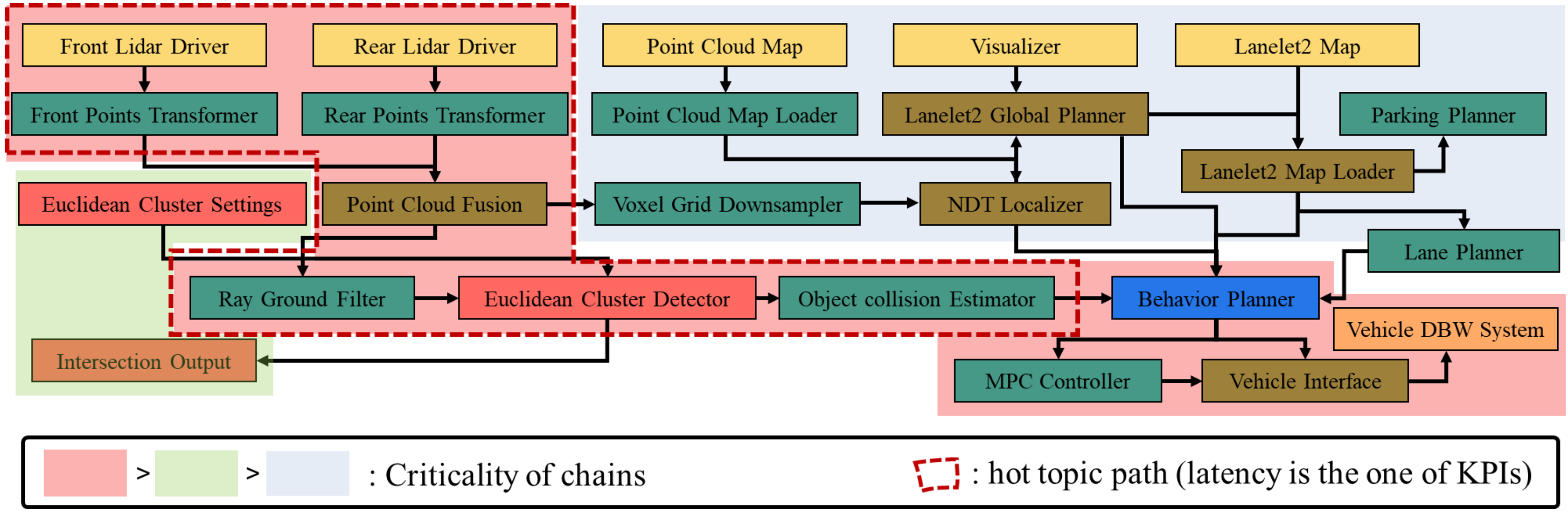


< End-to-end timing analysis >

†H. Choi et al. "PiCAS: New design of priority-driven chain-aware scheduling for ROS2." *RTAS*, 2021.

PiCAS on the reference system (1/2)

- We integrated PiCAS into the open-source *reference system*† for evaluation



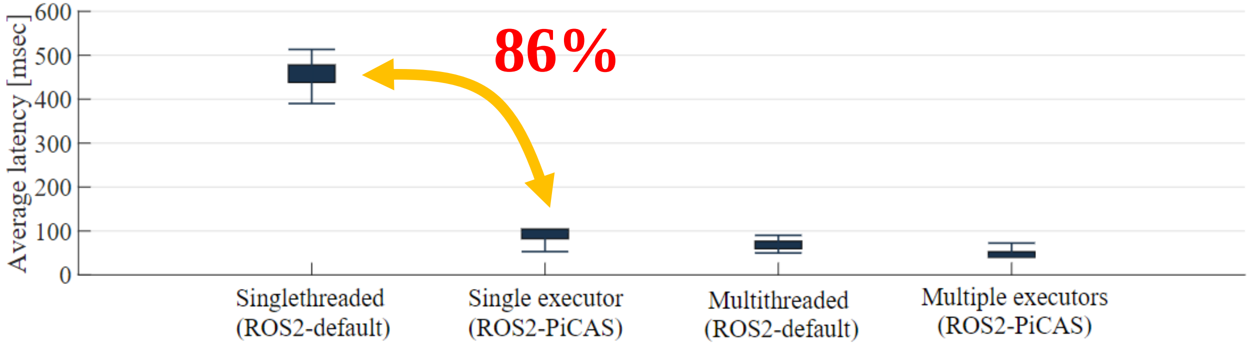
< Autoware model of the reference system >

- Evaluation criteria: Key Performance Indicators (KPIs)
  - Average end-to-end latency of hot topic path
  - Number of dropped messages
  - Jitter of periodic node, e.g., Behavior Planner

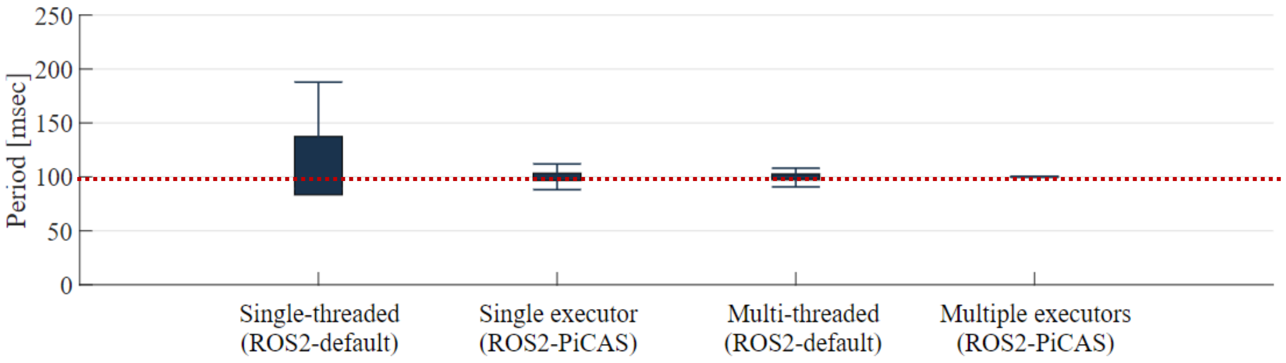
†ROS2 Real-Time Working Group. Reference system. https://github.com/ros-realtime/reference-system

PiCAS on the reference system (2/2)

- Evaluation environment
  - Raspberry Pi 4 with a fixed CPU frequency of 1.5GHz
  - 4 CPU cores for multiple executors (ROS2-PiCAS) and multi-threaded executor (ROS2-default)



< End-to-end latency of hot topic path >



< Behavior Planner jitter >

	Singlethreaded (ROS2-default)	Single executor (ROS2-PiCAS)	Multithreaded (ROS2-default)	Multi. executors (ROS2-PiCAS)
Mean	0.8681	0.0282	0	0
STD	0.3347	0.1651	0	0

< Number of dropped messages >

Real-time support for multi-threaded executors

- Challenges
  - Runtime callback distribution across multiple threads
  - Unsynchronized polling points of the threads

  ➡ **Existing ROS 2 analyses are not directly applicable to multi-threaded executors**

- Our ongoing efforts
  - Develop real-time analysis for the *default* multi-threaded executors of ROS 2
    - Revise conventional *non-preemptive global scheduling analysis* by considering semantic differences, e.g., callback dependencies, chains, polling points, and ready set management
  - Extend PiCAS to multi-threaded executors
    - Enable priority-driven scheduling for better end-to-end latency and predictability
  - Explore the effects of *callback groups*, e.g., *mutually-exclusive* vs. *reentrant*

Real-time GPU acceleration

- Challenges
  - Asynchronous and unstructured models for kernel execution on GPU accelerators
  - Blocking time and priority inversion by GPU kernel execution from low-priority chains

➡ **Unpredictable real-time behavior of ML/AI workloads**

- Our ongoing efforts
  - Build a GPU server node in the ROS 2 software stack
    - Priority-driven control of GPU requests to shared hardware accelerators
    - Concurrent kernel execution with real-time spatial multitasking and prioritized CUDA streams
  - Develop an architecture to support a low-overhead accelerator resource management framework
    - Minimizing data copy delays with efficient zero-copy IPC methods, e.g., Iceoryx

Conclusion & Future work

- Conclusion
  - Presented the benefit of enabling priority-driven scheduling in the ROS 2 framework
    - Integrated our PiCAS framework into the reference system
    - Demonstrated that PiCAS *outperforms* the existing ROS 2 scheduling scheme w.r.t. key performance indicators, e.g., *average end-to-end latency, dropped messages, and jitter of periodic node,* under practical scenarios
  - Discussed challenges and issues for *multi-threaded executors* and *real-time support of ROS 2 with shared accelerators*

- Future work
  - Evaluate the effectiveness of PiCAS against other executors, e.g., cbg executor

Q & A

Priority-Driven Real-Time Scheduling in ROS 2: Potential and Challenges

- ROS 2 PiCAS source
  - https://github.com/rtenlab/ros2-picas
- PiCAS with the reference system
  - https://github.com/rtenlab/reference-system