RETIS OPEN DAY – JUNE 2022

Daniel Casini

Assistant Professor, Scuola Superiore Sant'Anna, Pisa, Italy

Sant'Anna
Scuola Universitaria Superiore Pisa

Retis
Real-Time Systems Laboratory

Let me introduce myself

- Name: Daniel
- Surname: Casini
- Born in Livorno on 19/07/1990

Education and Working Experience

- Assistant Professor at the RETIS Lab. Dec 2021 – current.
- PostDoc Researcher at the RETIS Lab. July 2020 – Dec. 2021.
- International Ph.D. in Emerging Technologies - Embedded Systems, Scuola Superiore Sant'Anna, Ph.D. degree obtained on July 2nd, 2020, with honors.
- Masters of Science in Embedded Computing System, by University of Pisa and Scuola Superiore Sant'Anna, 110/110, with honors (2013-2016).
- Bachelor Degree in Computer Engineering (Italian), University of Pisa, 110/110 (2010-2013).
- High School Degree, I.T.I.S. Galileo Galilei Livorno, Computer Science, 100/100.

Let me introduce myself

- My research interests include:
  - predictability in applications leveraging middleware frameworks (e.g., TensorFlow and ROS) and artificial intelligence (e.g., autonomous driving)
  - optimization methods
  - software predictability in multi-processor and heterogeneous systems
  - schedulability analysis
  - synchronization protocols
  - predictable I/O virtualization and memory access techniques
  - design and implementation of real-time operating systems and hypervisors
  - analysis of end-to-end delays
  - and more…

Real-Time Research at the RETIS Lab

Retis Real-Time Systems Laboratory

Goal: Provide timing guarantees for modern real-time workloads on COTS platforms

Methodologies

Real-Time Analysis

$\max f(x)$
s.t. …

Design Optimization

Monitoring and Control

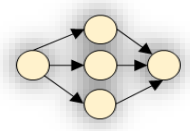
(Some) Topics

I/O virtualization

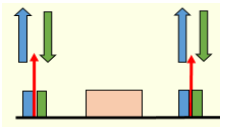
Memory Contention Analysis and Control

TensorFlow
ROS

Timing predictability in Middleware Frameworks
(ROS, DDS, etc.)



Timing analysis of tasks with dependencies



Time Determinism in communications

Frameworks for Autonomous Driving

Two main open-source projects: Apollo and Autoware

- Started in 2015 by Shinpei Kato at Nagoya University (Japan).
- 2300+ stars on GitHub and 500+ accounts on Slack (10/2018).
- Based on ROS 2.

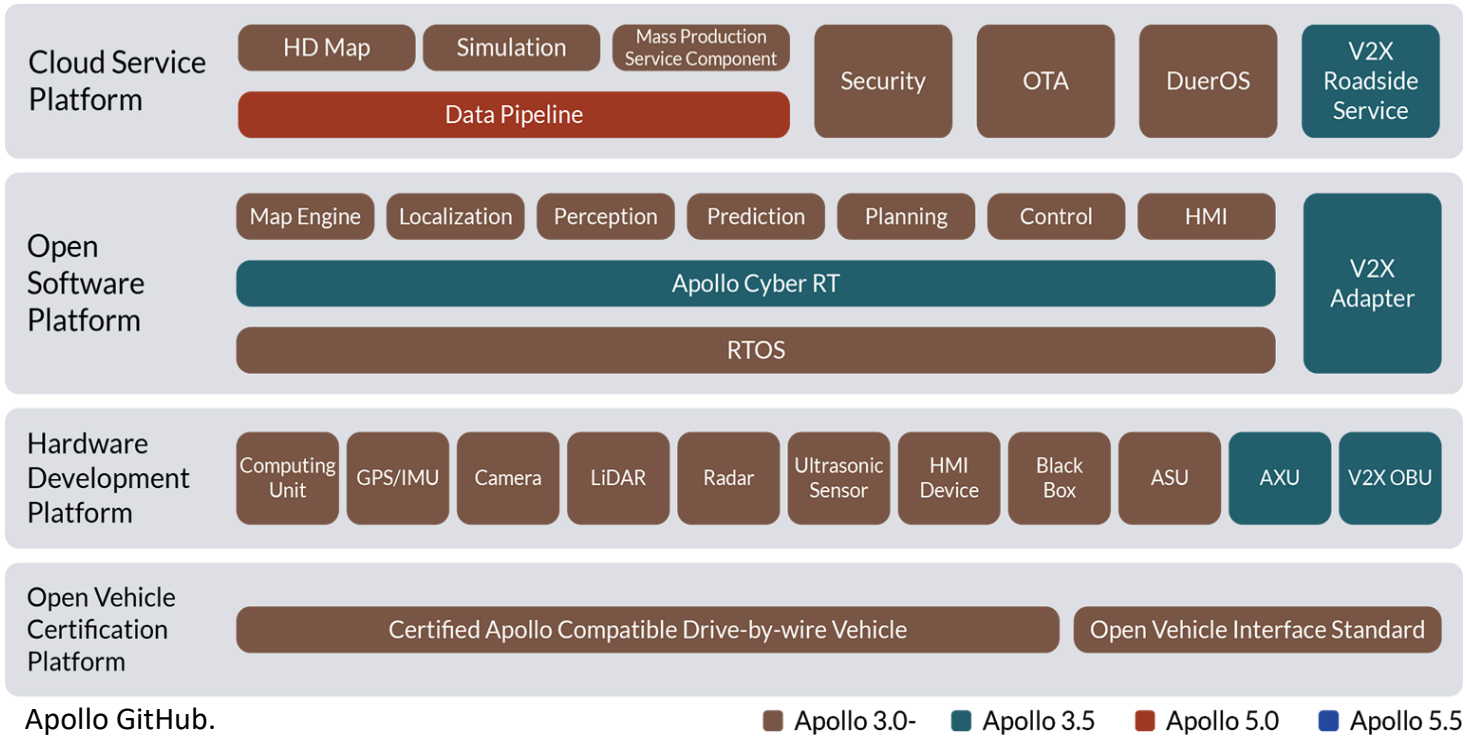
- Started in 2017 by Baidu (China).
- The only company to obtain the first batch of road test licenses in China.
- Based on CyberRT.

The Apollo Stack

Real-Time Systems Laboratory

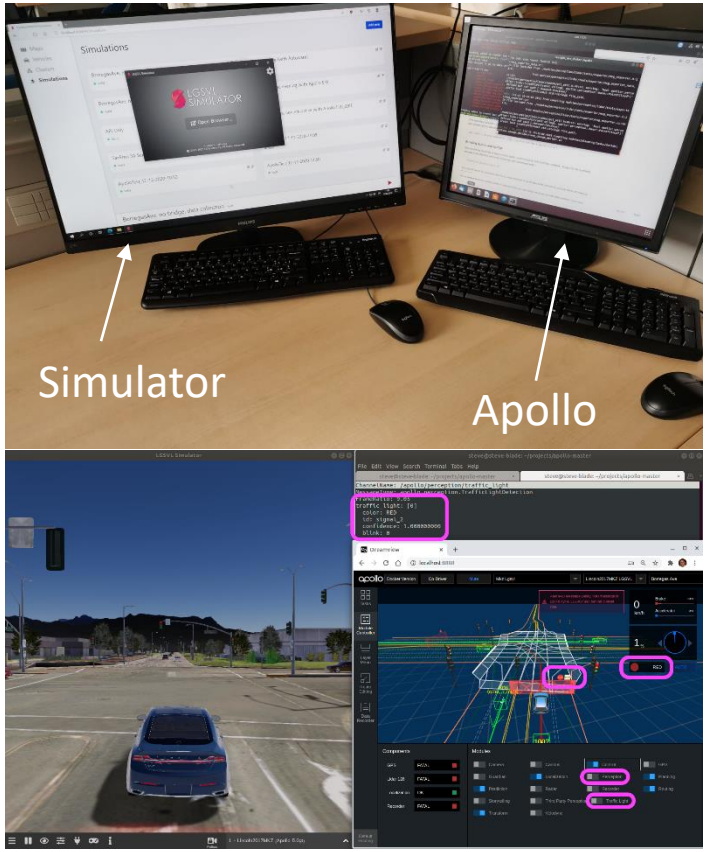
- Four layers

1. Reference Vehicle Platform
2. Reference Hardware Platform
3. Open Software Platform
4. Cloud Service Platform



HW-in-the-loop Simulation

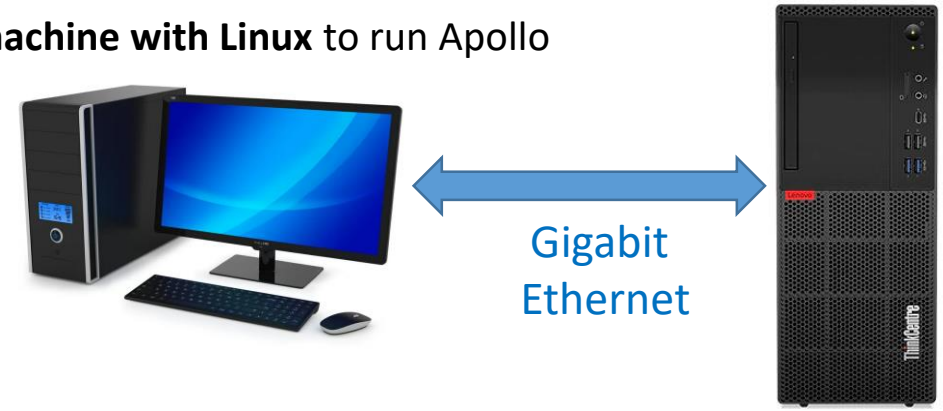
- **HW-in-the-loop simulation environment** to work with Apollo



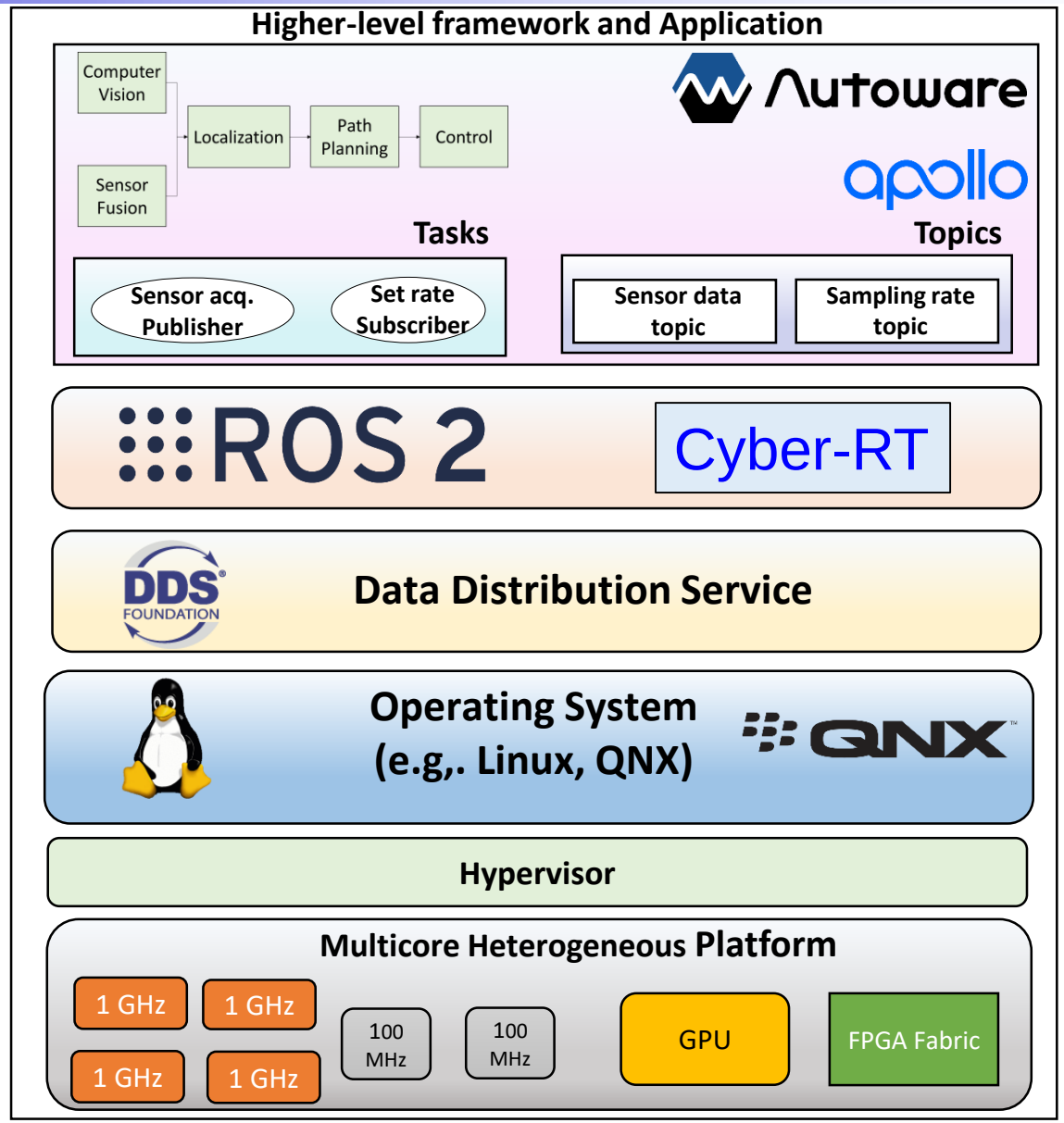
LGSVL Simulator
(https://www.lgsvlsimulator.com/)
on a powerful desktop computer

4 GHz Quad-core CPU, 32GB of RAM,
Nvidia GTX 2080 16GB, Win 10 64-bit

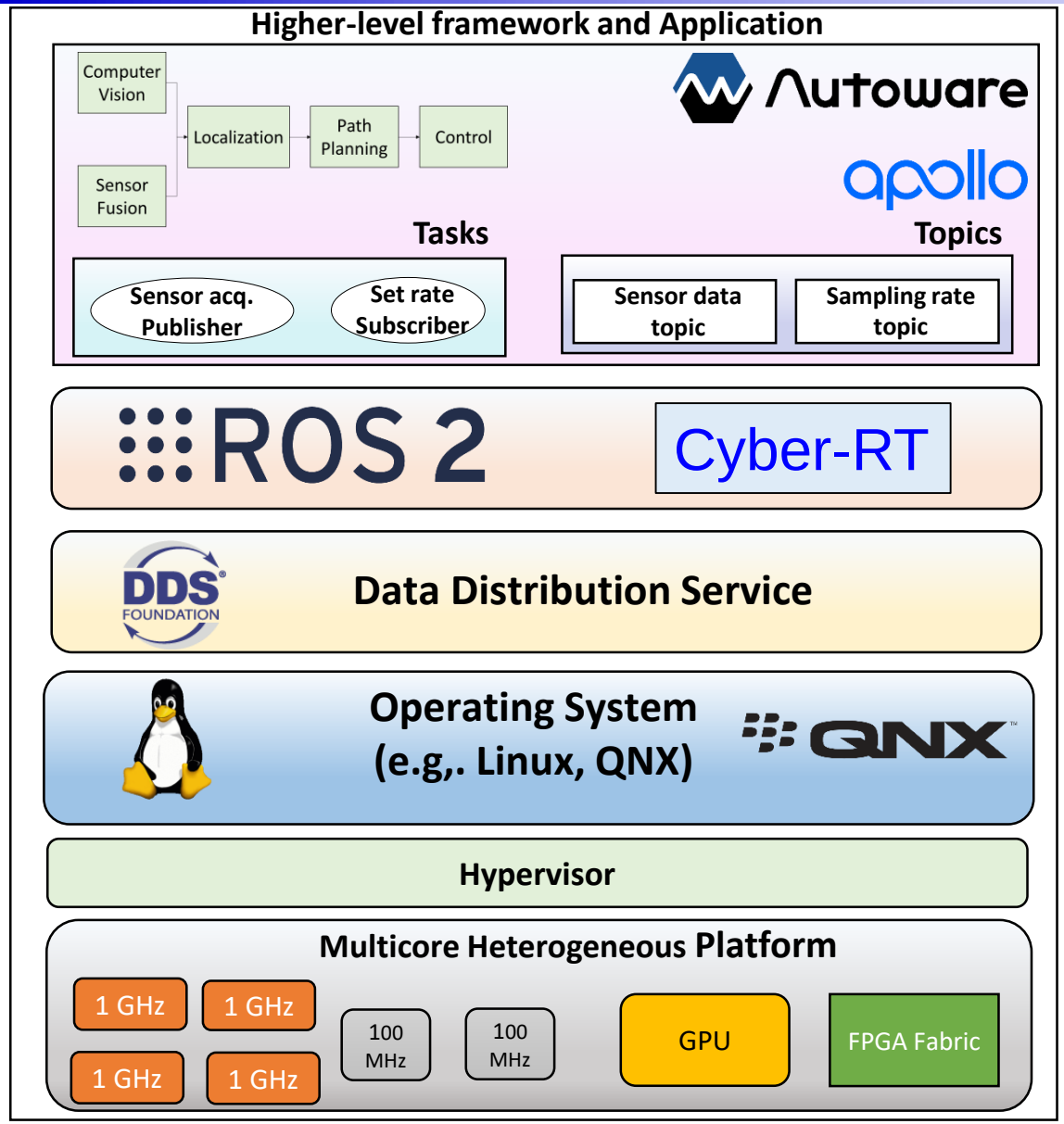
X86 machine with Linux to run Apollo



Threats to Predictability and Challenges

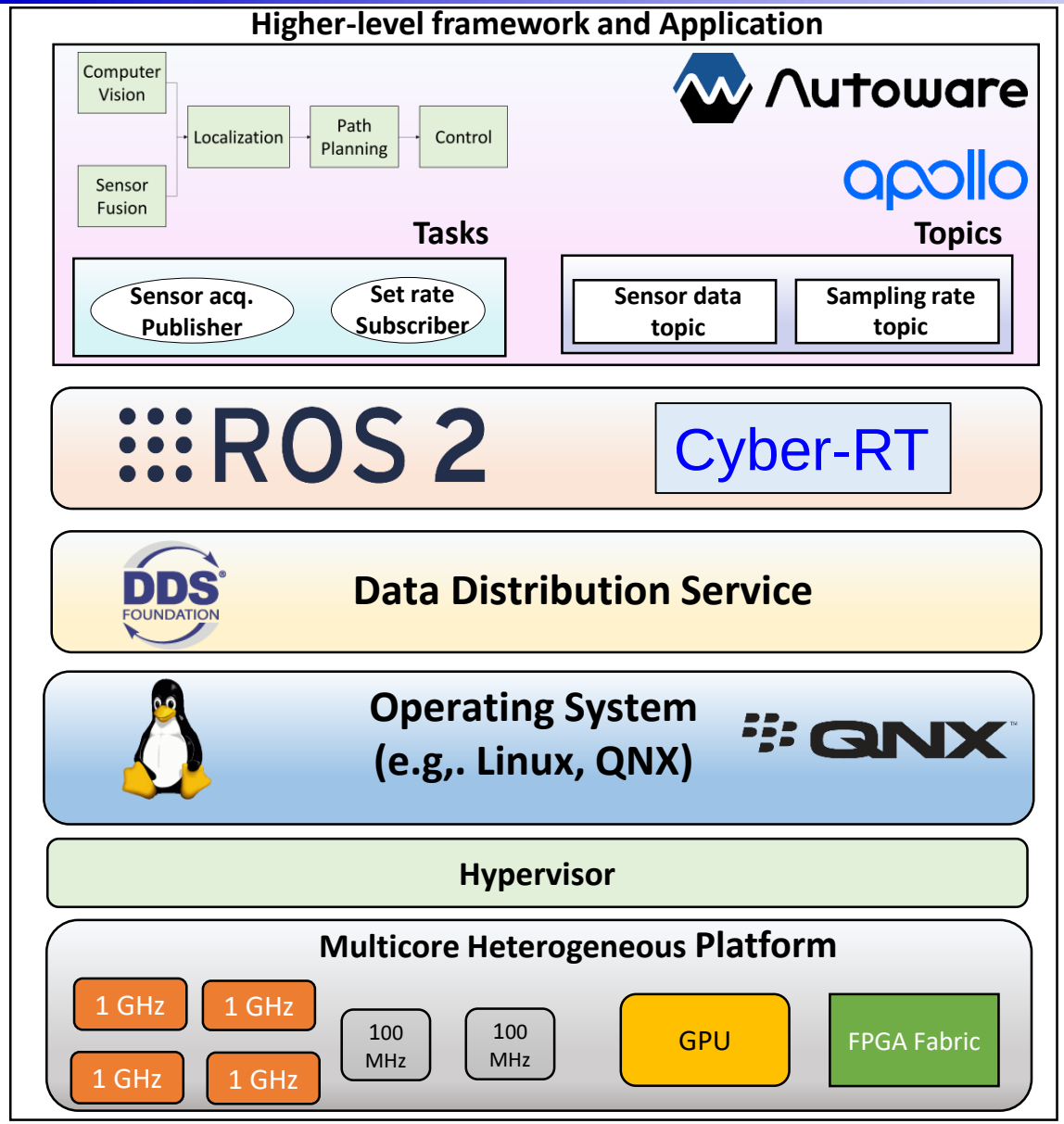


Threats to Predictability and Challenges



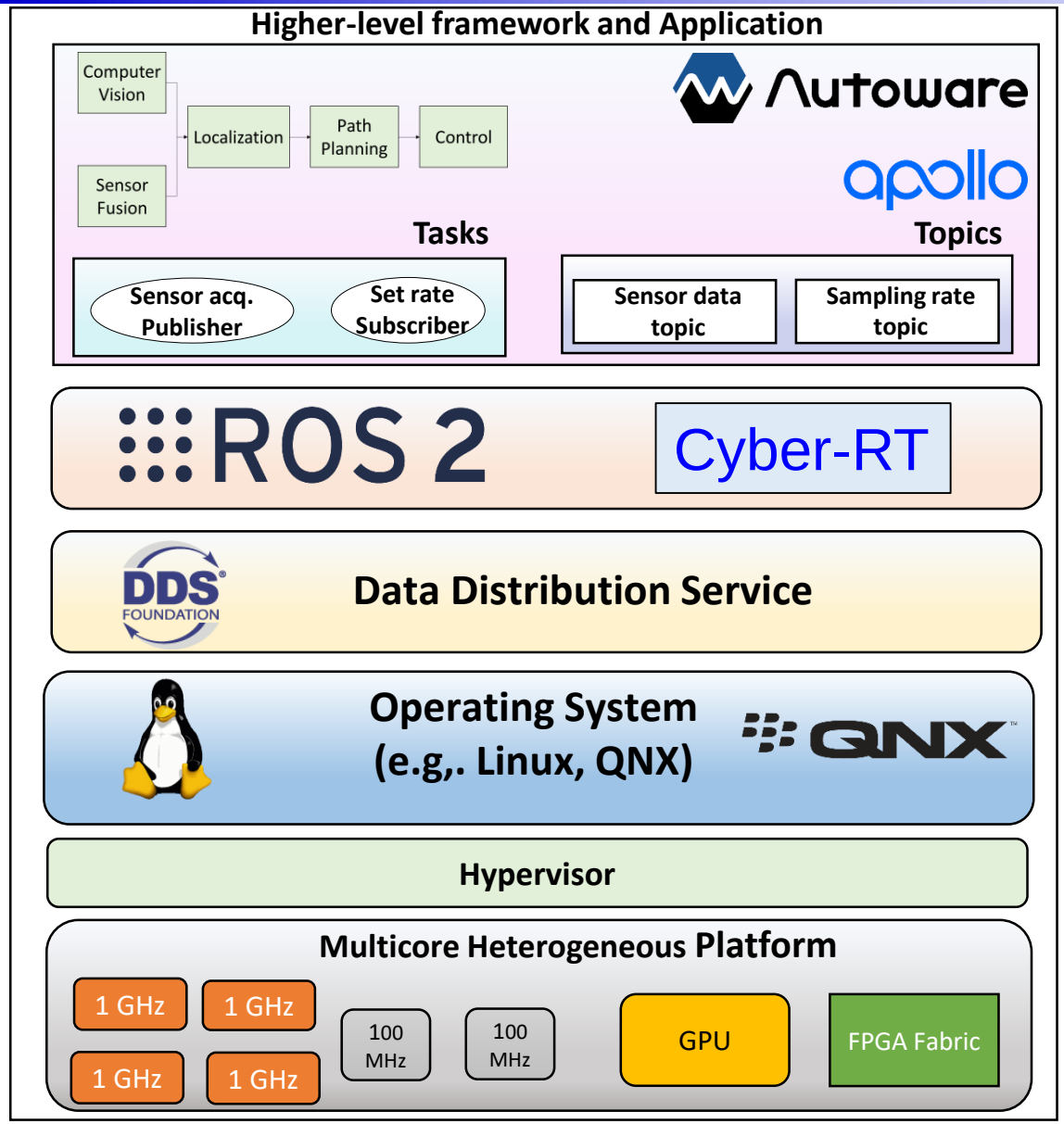
A very complex autonomous driving application that needs to satisfy timing constraints

Threats to Predictability and Challenges



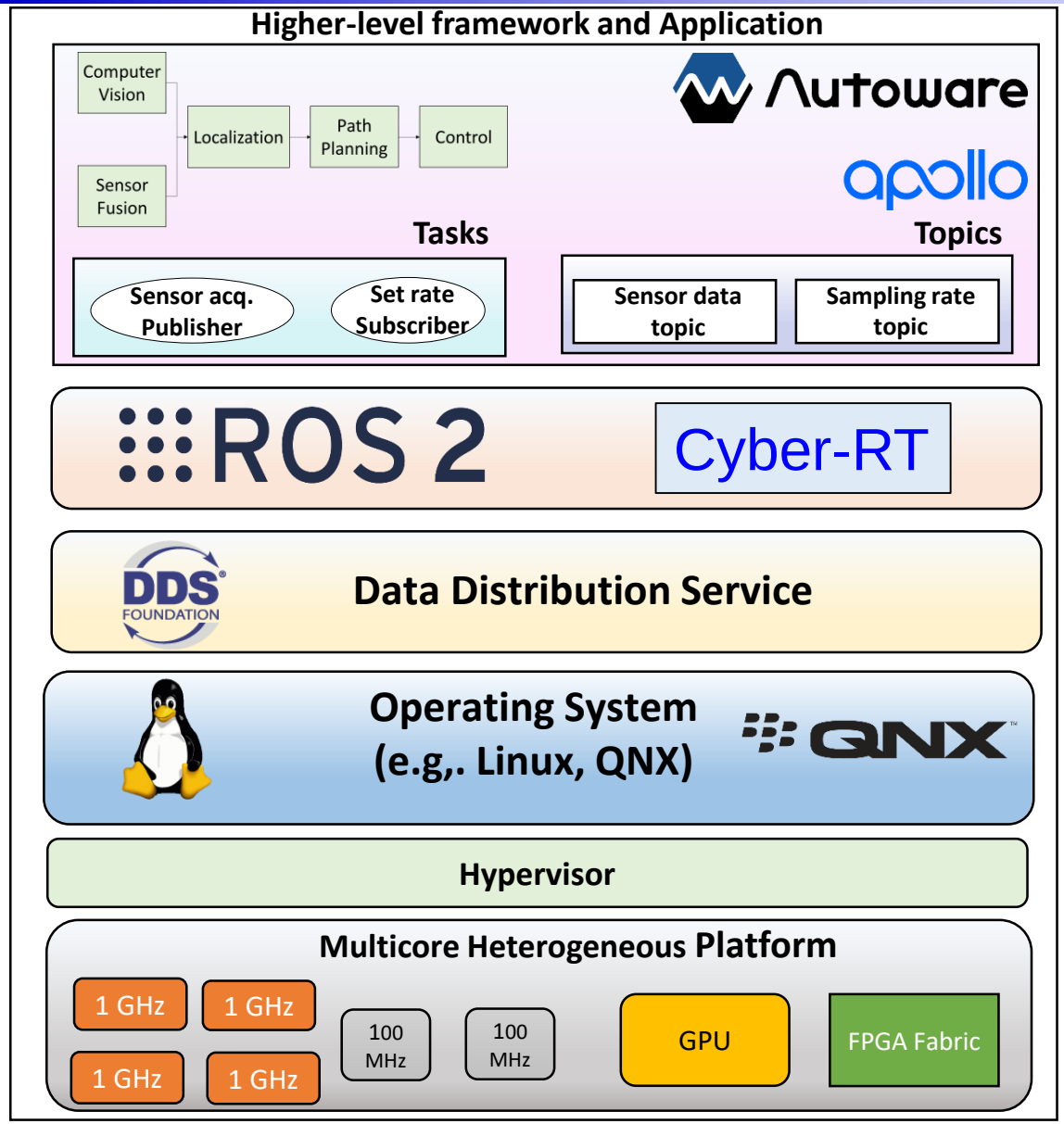
ROS 2 and CyberRT are used to conveniently implement autonomous driving, thanks to the pub/sub infrastructure and the large set of pre-implemented functionalities they provide.

Threats to Predictability and Challenges



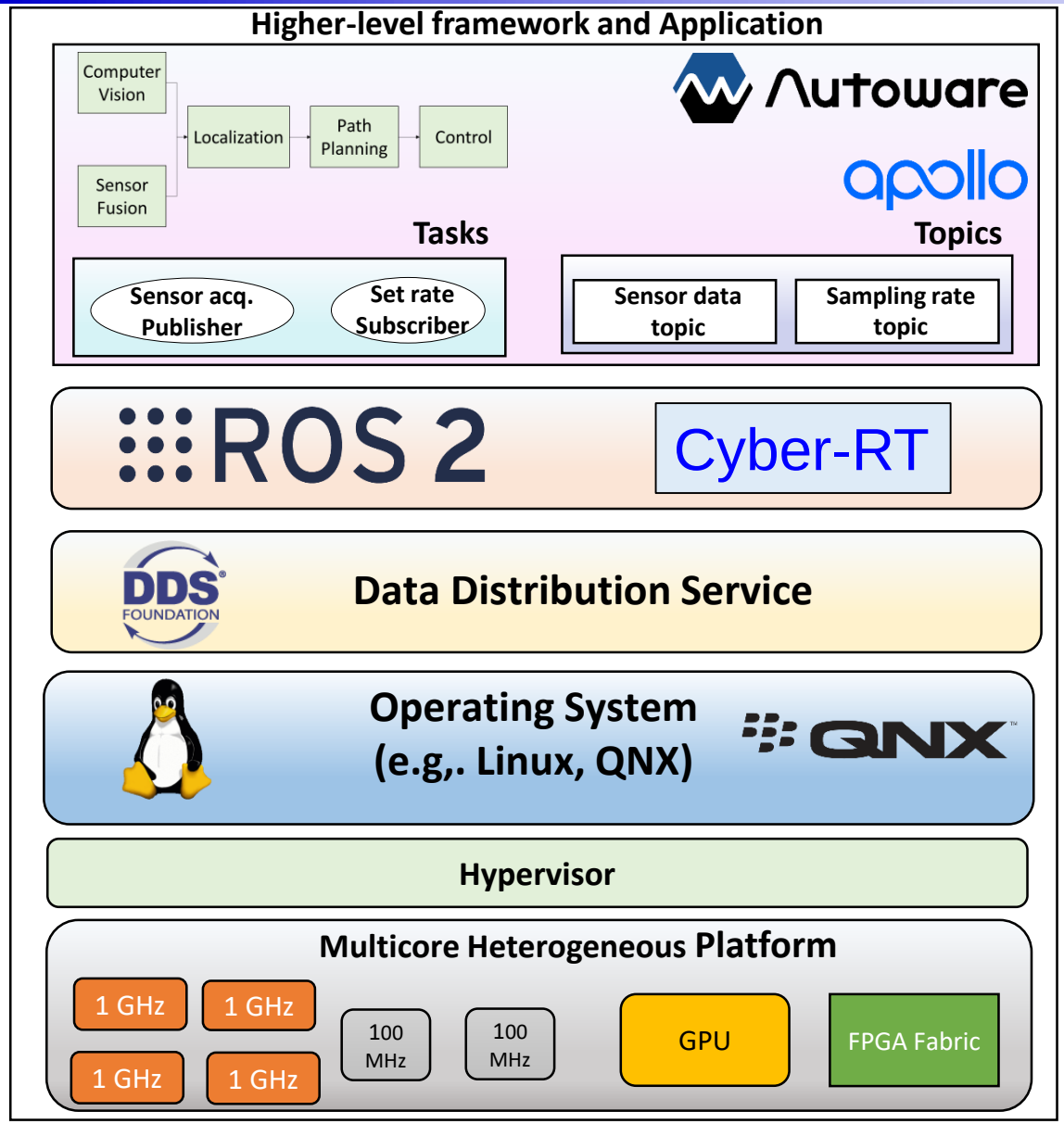
The DDS manages the communication among callbacks, with a complex multi-thread software that needs to be properly scheduled by the OS

Threats to Predictability and Challenges



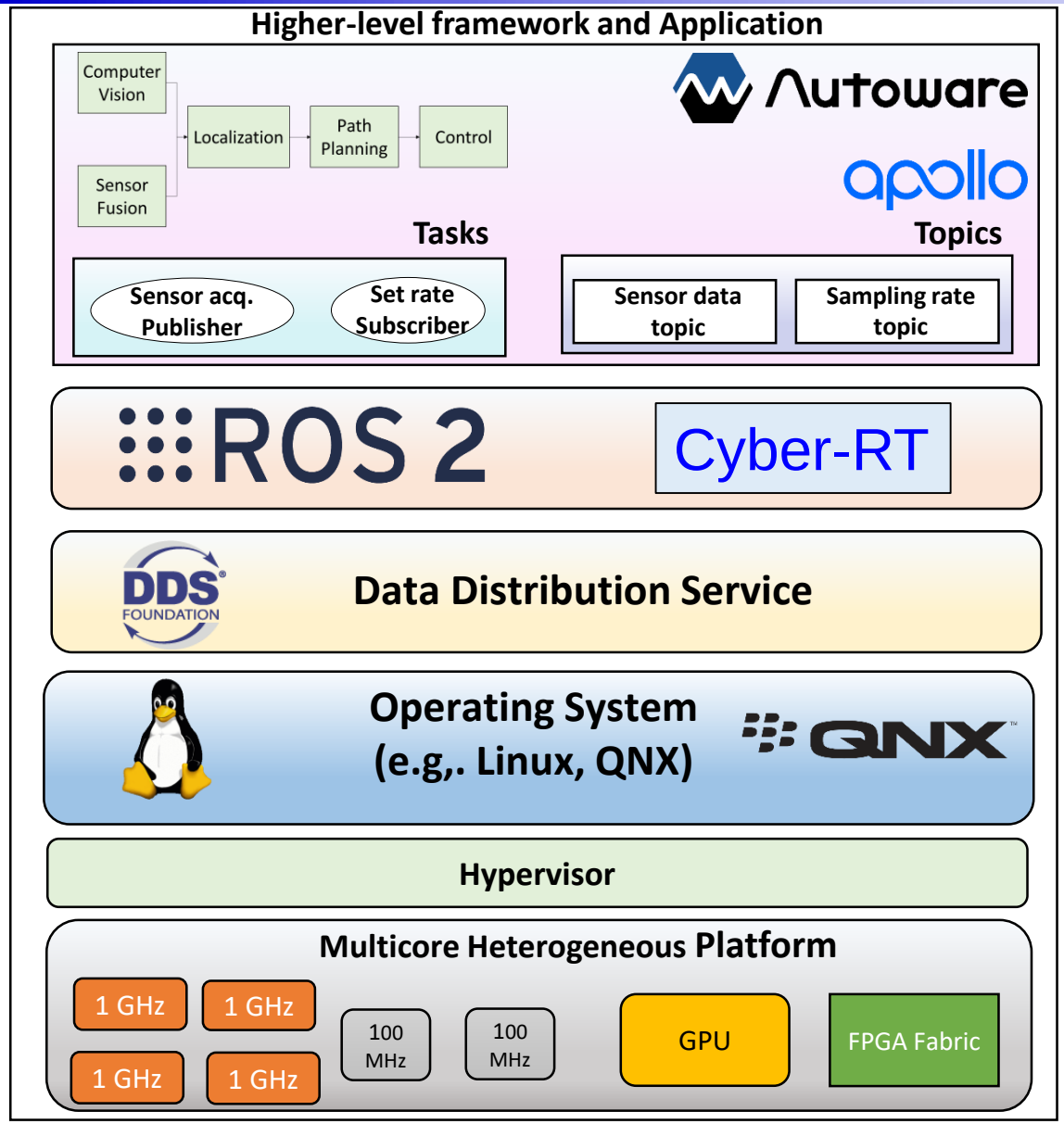
Application-level, ROS2-level, and DDS-level threads needs to be properly scheduled by the OS to meet timing constraints

Threats to Predictability and Challenges



An hypervisor, if present, can greatly help in implementing temporal and spatial isolation between virtual machines (VM), but it needs to be properly configured

Threats to Predictability and Challenges



The underlying hardware platform can be very complex. Usually is equipped with heterogeneous cores and hardware accelerators and it has complex memory hierarchies, introducing further challenges

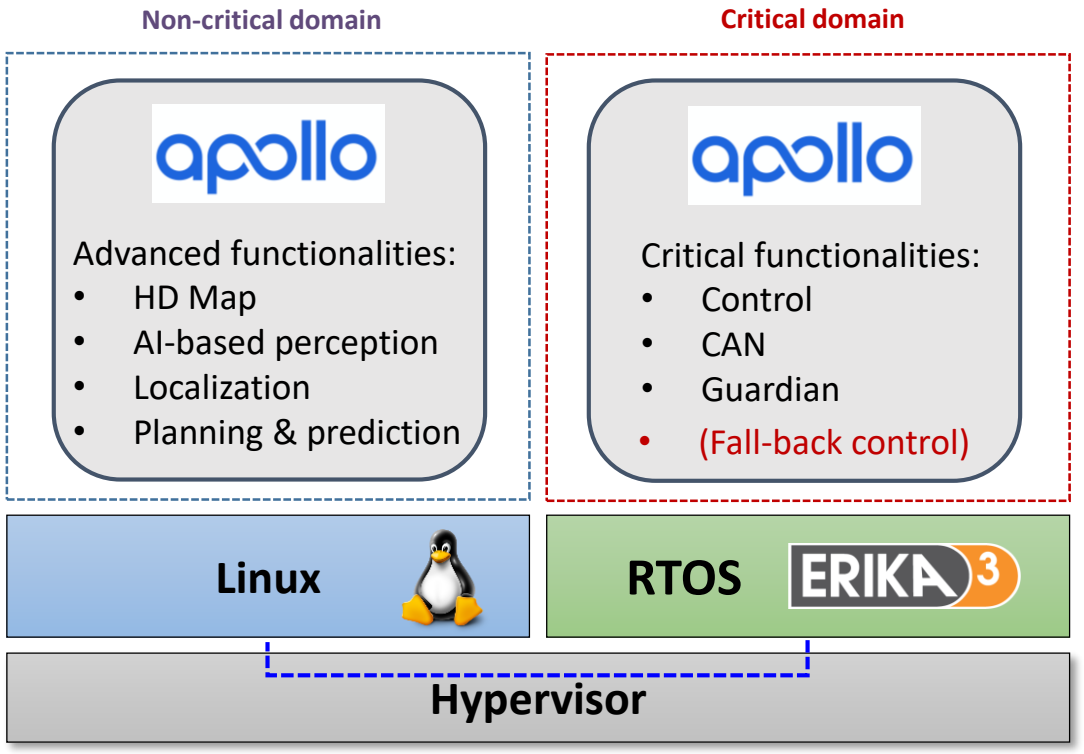
Multi-Domain Architecture for Apollo

OBJECTIVE

Making Apollo safer and more secure by designing a multi-domain architecture

HOW?

1. Separating Apollo's modules between a non-critical and a critical domain, running different OSes
2. Using a hypervisor to separate them
3. Restoring the pub/sub CyberRT-based communication



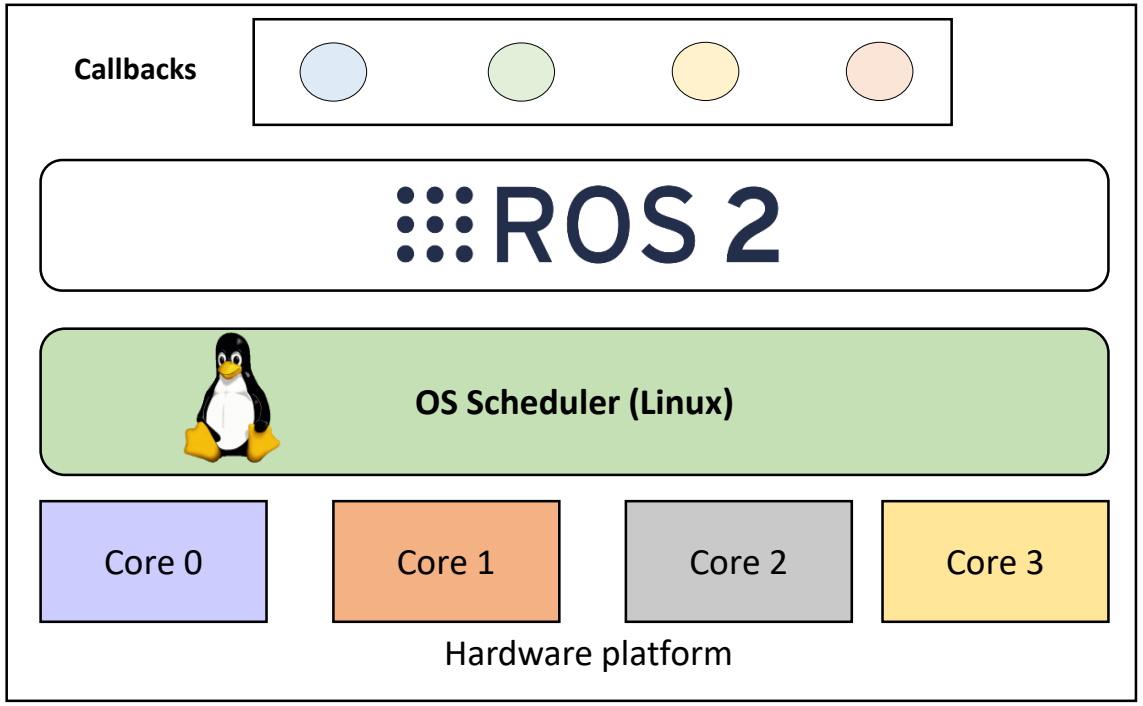
Luca Belluardo, Andrea Stevanato, **Daniel Casini**, Giorgiomaria Cicero, Alessandro Biondi, and Giorgio Buttazzo, "A multi-domain software architecture for safe and secure autonomous driving", *Proc. of the 27th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2021)* , August 18-20, 2021.

Real-Time Scheduling with ROS 2

Retis Real-Time Systems Laboratory

Two levels of scheduling:

- ROS processes are scheduled by the Linux operating system
- Callbacks (e.g., C++ functions) are in turn scheduled by the ROS executor implemented by ROS



ROS-level analysis accounting for the ROS's quirks

Based on CPA [1]
(Compositional Performance Analysis)

Linux-level scheduling using reservations (e.g., SCHED_DEADLINE)

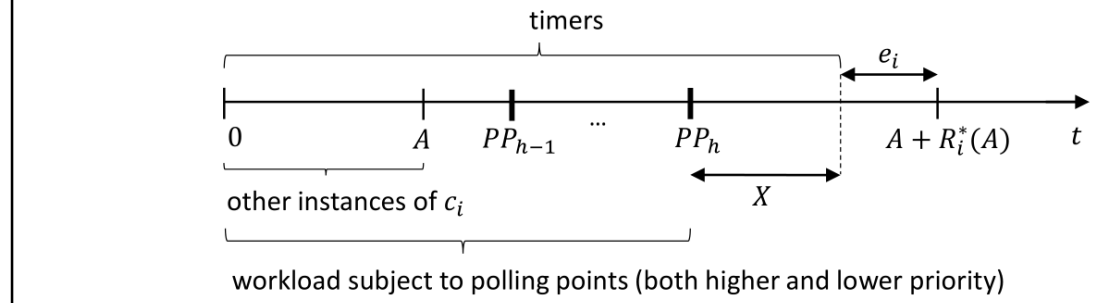
[1] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the SymTA/S approach," IEEE Proceedings - Computers and Digital Techniques, March 2005.

Response-Time Analysis

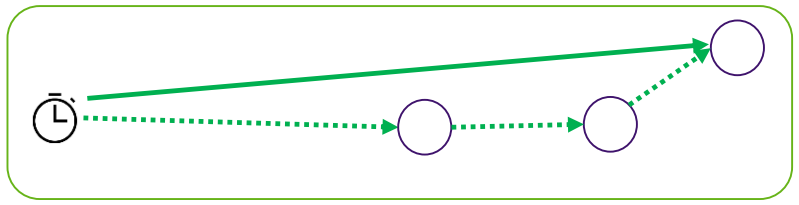
$$sbf_k(A + R_i^*(A)) = rbf_i(A+1) + RBF(hp_k(c_i), A + R_i^*(A) - e_i + 1) + B_i$$

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

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



Response-Time Analysis that accounts for the ROS's peculiarities



Optimization for **intra-executor** chains

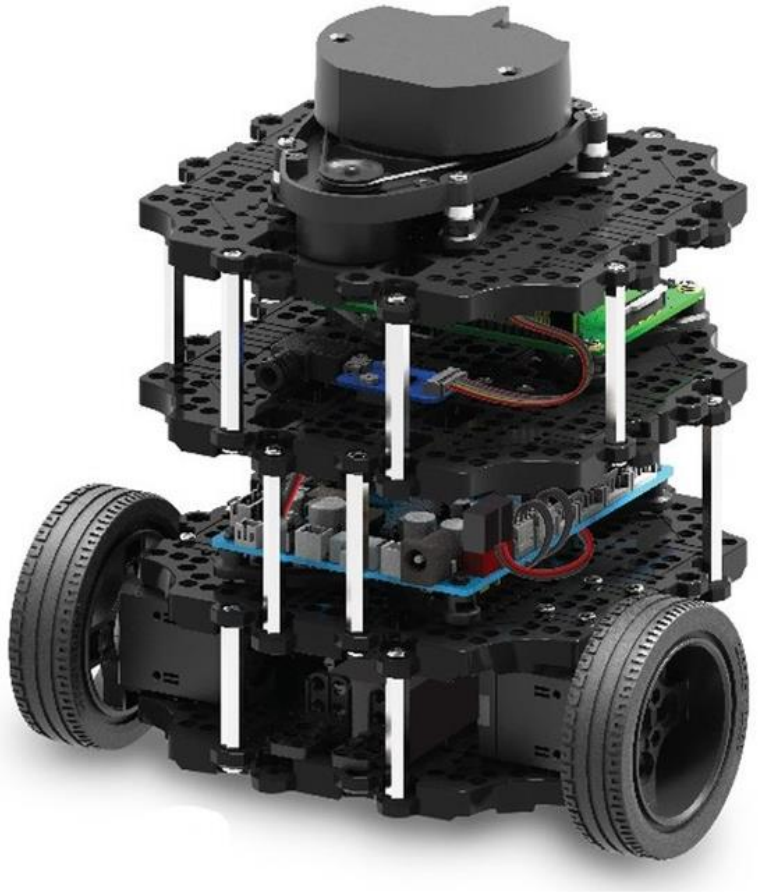
- **Daniel Casini**, Tobias Blaß, Ingo Lütkebohle, and Björn B. Brandenburg, "Response-Time Analysis of ROS 2 Processing Chains under Reservation-Based Scheduling", *In Proceedings of the 31th Euromicro Conference on Real-Time Systems (ECRTS 2019), Stuttgart, Germany,* July 9-12, 2019.
- Tobias Blaß, **Daniel Casini**, Sergey Bozhko, and Björn B. Brandenburg, "A ROS 2 Response-Time Analysis Exploiting Starvation Freedom and Execution-Time Variance", *In Proceedings of the 42nd IEEE Real-Time Systems Symposium (RTSS 2021)*, Dortmund, Germany, December 7-10, 2021.

Evaluation

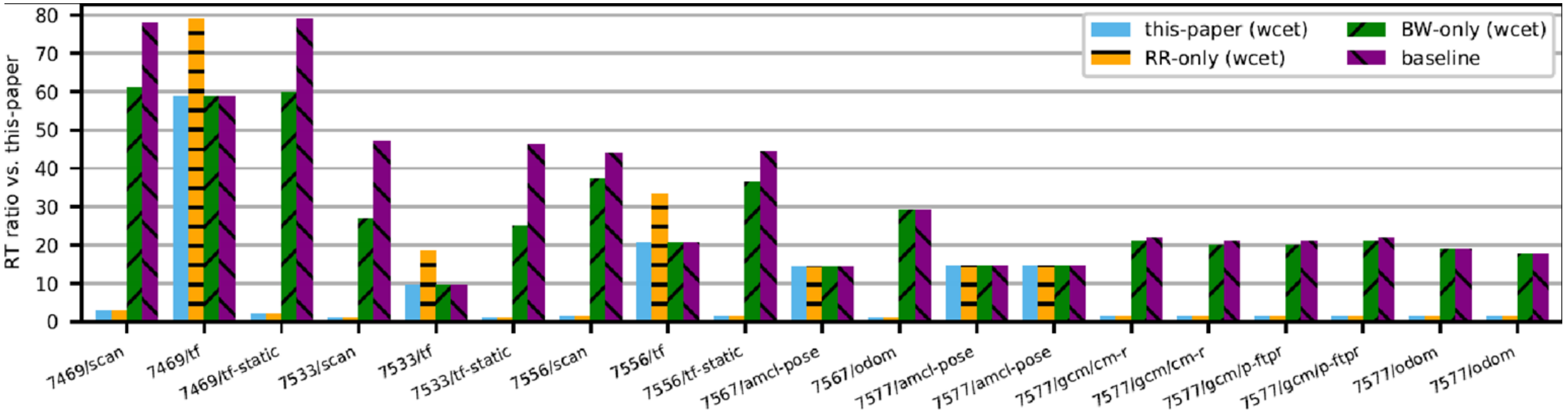
- **Turtlebot 3** “Burger” controlled by a **Raspberry Pi 4B**

- Running various **ROS packages**
  - Navigation 2 packages
  - Turtlebot 3 drivers

- Callback graph extracted from measurements
  - See Blass et al., “Automatic Latency Management for ROS 2: Benefits, Challenges, and Open Problems”, RTAS *2021*

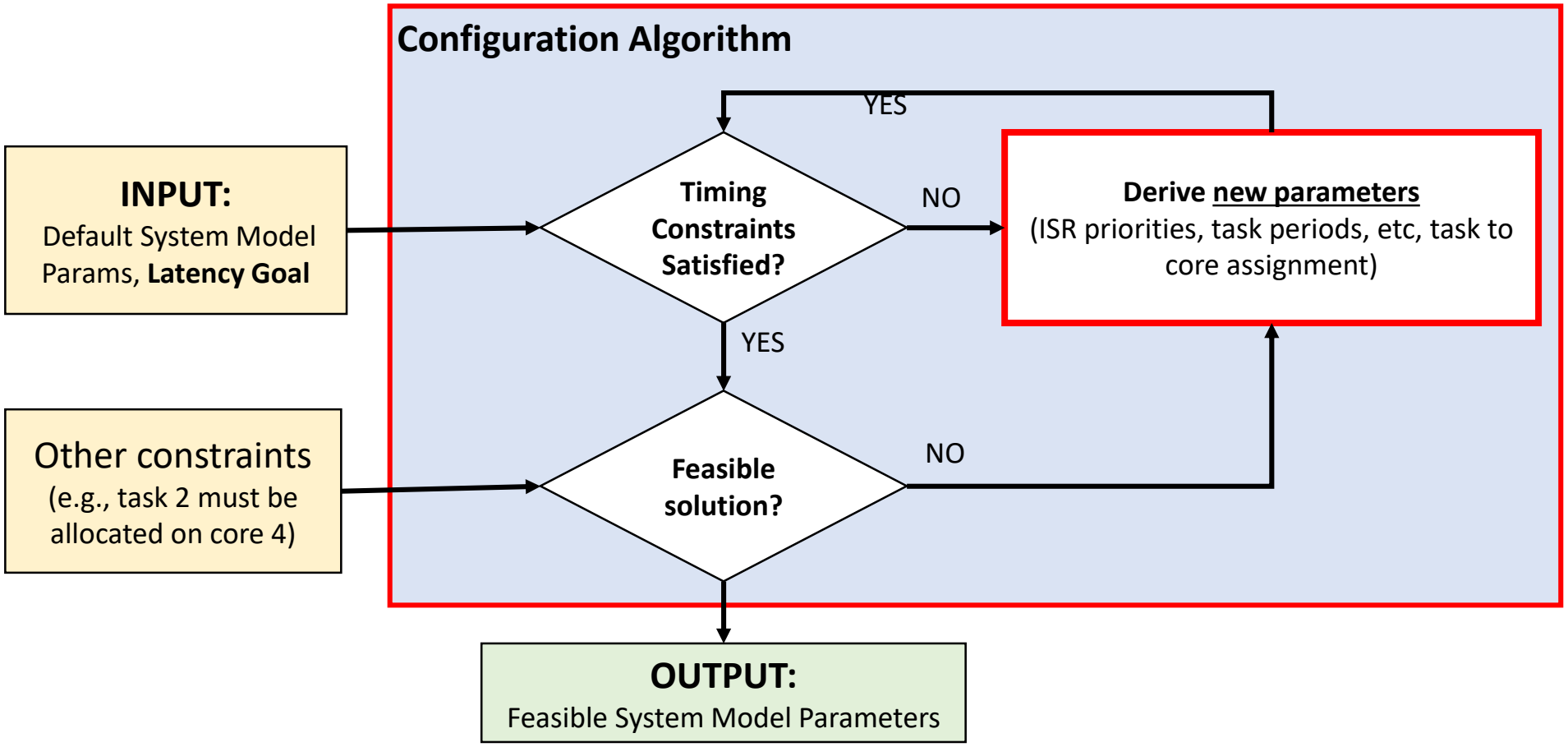


Evaluation


RT ratio vs. this-paper
0
10
20
30
40
50
60
70
80
this-paper (wcet)
RR-only (wcet)
BW-only (wcet)
baseline
7469/scan
7469/tf
7469/tf-static
7533/scan
7533/tf
7533/tf-static
7556/scan
7556/tf
7556/tf-static
7567/amcl-pose
7567/odom
7577/amcl-pose
7577/amcl-pose
7577/gcm/cm-r
7577/gcm/cm-r
7577/gcm/p-ftpr
7577/gcm/p-ftpr
7577/odom
7577/odom

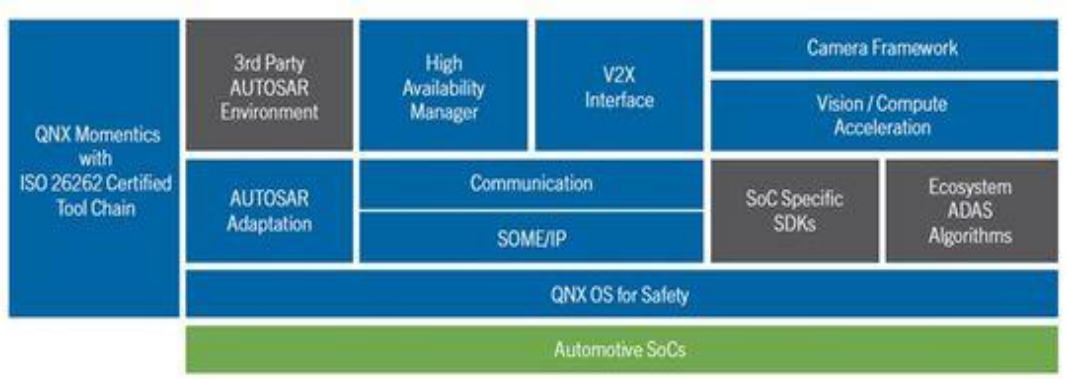
Design-Space Exploration

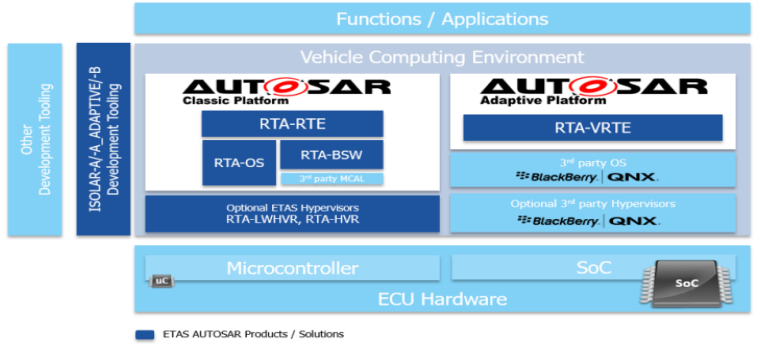
The analysis enables the development of tools for the design-space exploration of the system parameters



The opportunity of using QNX

- Preferred base operating system by many automotive OEMs
- ISO-26262 certified at the highest level of assurance (ASIL-D)
- POSIX Compliant, commercial, proven: AUTOSAR Adaptive needs POSIX
- Commercial OS supporting CPU reservations!





Interestingly, the QNX reservation-based scheduler supports multiple threads in the same reservation, thus offering opportunities for the coordinate scheduling of ROS and DDS threads

However, it is first necessary to understand the behavior of the **QNX reservation-based scheduling algorithm**

Dakshina Dasari, Matthias Becker, **Daniel Casini**, and Tobias Blaß, "End-to-End Analysis of Event Chains under the QNX Adaptive Partitioning Scheduler", *In Proceedings of the 28th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2022)*, Milan, Italy, May 4-6, 2022.

Thesis Opportunities

- Design and implementation of predictable schedulers for ROS 2
- Real-time analysis and design of methods to improve the real-time performance of the DDS
- Analysis and performance comparison of different IPC mechanisms in the QNX real-time operating systems
- End-to-end analysis of response times in autonomous driving frameworks
- Protocols to improve the real-time performance in autonomous driving frameworks
- Scheduling algorithms and optimization for parallel tasks (e.g., OpenMP-like)
- Locking protocols to ensure mutual exclusion when accessing shared resources under real-time constraints
- Memory-aware scheduling algorithms and resource control policies
- Protocols to predictably access hardware accelerators
- Design and implementation of AI-based algorithms to perform design optimization and scheduling decisions
- Predictability in real-time edge systems
- and more…

If you would like to work on some other related topics, just write me, and we can discuss it

Thank you!

Daniel Casini
daniel.casini@santannapisa.it

https://retis.sssup.it/~d.casini/index.html