A First Look at ROS 2 Applications Written in Asynchronous Rust (Artifact)

Martin Škoudlil
Czech Institute of Informatics, Robotics and Cybernetics,
Czech Technical University in Prague, Czech Republic

Michal Sojka[1]
Czech Institute of Informatics, Robotics and Cybernetics,
Czech Technical University in Prague, Czech Republic

Abstract

Deterministic real-time performance is critical for robotic applications, many of which are developed in C++ using the ROS 2 framework. Recently, Rust has emerged as a compelling alternative to C++ in various domains, including robotics. This study explores whether ROS 2 applications written in Rust can achieve real-time performance comparable to their C++ counterparts. We focus on the R2R library, which provides Rust bindings for ROS 2 with a modern asynchronous programming interface.

This artifact includes source code for several ROS 2 nodes and supporting Python scripts. The nodes implement simple publishers or subscribers and vary in programming language (Rust, C++), Rust asynchronous runtime (futures, Tokio), and callback-to-OS-thread mapping with different thread priorities. The nodes are run with tracing enabled and by post-processing the traces we obtain statistics about end-to-end latencies. Our results demonstrate that specific configurations enable Rust R2R applications to match the real-time performance of C++ applications.

Running the artifact as is does not require any ROS 2 or Rust knowledge. Following our README and running mentioned shell commands should be sufficient. However to verify that the commands do what is promised, some familiarity with ROS 2 workspace structure and Rust's build tool cargo would be beneficial.

2012 ACM Subject Classification Software and its engineering → Real-time systems software

Keywords and phrases ROS, Rust, Real-time, Response time

Digital Object Identifier 10.4230/DARTS.11.1.2

Related Article Martin Škoudlil, Michal Sojka, and Zdeněk Hanzálek, "A First Look at ROS 2 Applications Written in Asynchronous Rust", in 37th Euromicro Conference on Real-Time Systems (ECRTS 2025), LIPIcs, Vol. 335, pp. 1:1–1:21, 2025.
`https://doi.org/10.4230/LIPIcs.ECRTS.2025.1`

Related Conference 37th Euromicro Conference on Real-Time Systems (ECRTS 2025), July 8–11, 2025, Brussels, Belgium

1 Scope

This artifact includes source code of eleven ROS 2 nodes and supporting Python scripts. The ROS 2 nodes implement ROS publishers and nine variants of ROS subscribers. Python scripts allow running all combinations of ROS nodes, measure their timing properties and plot results into figures. The results demonstrate that the configuration of Rust R2R applications proposed in the paper meets all timing requirements and matches the real-time performance of C++ ROS applications, while other configurations violate timing requirements.

[1] corresponding author

2 Content

The artifact package includes:
- `README.md` – instructions how to run the artifact
- `runner` – supporting Python scripts
- `src` – source code of all ROS nodes

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://zenodo.org/records/15446666`, the development repository is located at `https://gitlab.ciirc.cvut.cz/skoudmar1/ros-r2r-analysis`.

4 Tested platforms

Average laptop computer running Linux should be sufficient.

- **Minimal requirements:** 4 CPU cores, 4 GB of memory (16 GB preferred for parallel compilation).
- **Linux distribution:** Ubuntu 24.04 (recommended for ROS compatibility) or any other distribution with the Nix package manager installed.

PREEMPT_RT Linux kernel is an advantage, but the results can be well reproduced with a non-rt kernel on an idle system. Similarly, the results should be reproducible even when running Linux in a virtual machine, but this can lead to more jitter in the measurements and idle host system is recommended.

5 License

The artifact is available under the MIT license.

6 MD5 sum of the artifact

ad707c64ef8fced102c589cd5a53b2c0

7 Size of the artifact

118 KiB