Tracing ROS

Christophe Bourque Bédard

Progress Report Meeting
January 8, 2021

Polytechnique Montréal
DORSAL Laboratory

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

Summary

1. Introduction
2. What is ROS?
3. Previous work: Message Flow Analysis for ROS Through Tracing
4. ROS 2
5. Tracing ROS 2
6. Upcoming work and conclusion
7. Questions

Introduction

- Robotics
  - Commercial or industrial applications
  - Safety-critical applications
  - Can be connected over a network (e.g. 5G)
- Key elements
  - Message passing and Remote Procedure Call (RPC)
  - Real-time constraints
- Robotics software development can greatly benefit from tracing

What is ROS?

- Robot Operating System
  - Not an OS!
- Open source framework and set of tools for robotics software development
  - Development started in 2007
  - Well-known in robotics
- Message passing between “nodes”
  - Publish/subscribe
  - Service/action calls (~RPCs)
- Modular
  - Each node generally accomplishes a very specific task
  - Nodes are put together to perform complex tasks
- Intra-process, inter-process, and distributed

Message Flow Analysis for ROS Through Tracing

- Undergraduate research conducted in 2018-2019
- Motivation
  - Time is one of the main concerns for robotics application
  - Towards a critical path analysis
- Tools
  - Instrumentation with LTTng
  - Analysis with Trace Compass
- Result
  - View showing states of publisher and subscription queues
  - View showing flow of a message across queues and nodes

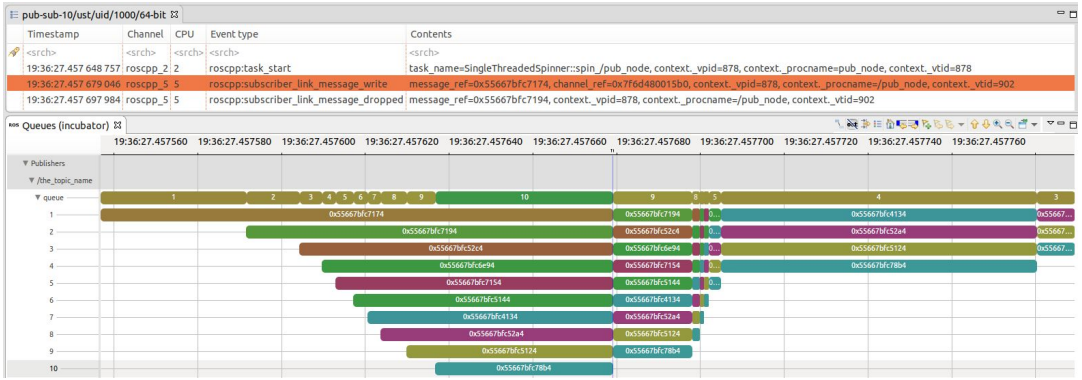

Figure 1. Queues state view.

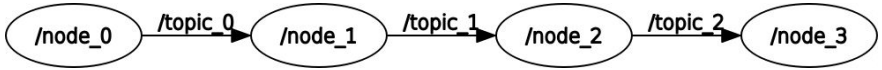

Figure 2. Simple pipeline test case.

Message Flow Analysis for ROS Through Tracing (2)

- Proof of concept
- Towards a ROS-aware critical path analysis (?!)
- Available in the Trace Compass Incubator
- See full write-up
  - christophebedard.com/ros-tracing-message-flow/
- Well received by the ROS community

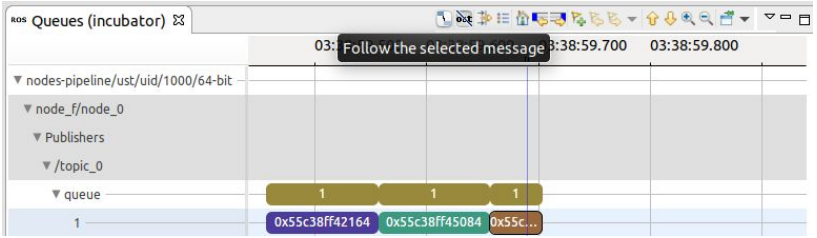

Figure 3. Selecting a specific message to follow.

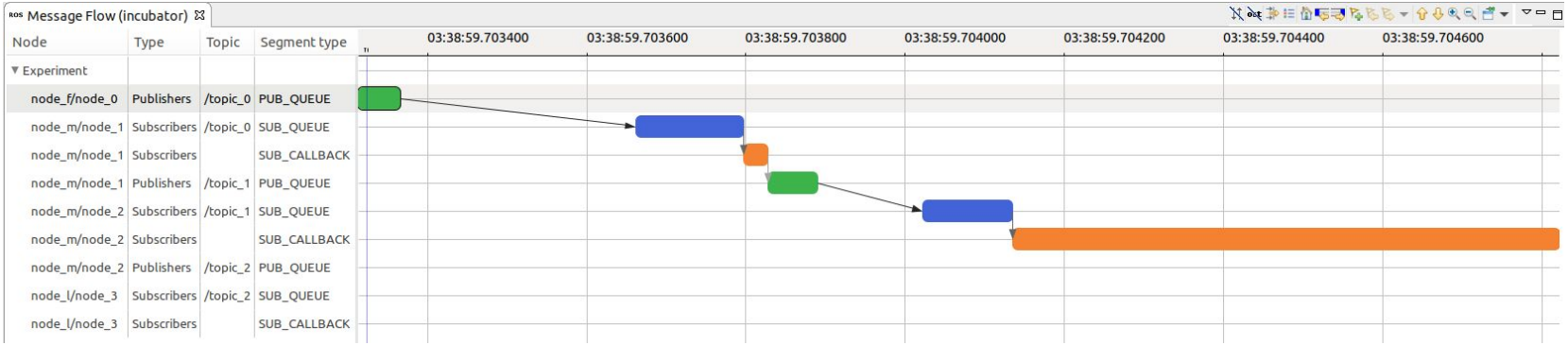

Figure 4. Message Flow view.

ROS 2

- index.ros.org/doc/ros2/
- ROS (1) was more of a research tool for academia
- ROS 2 aims to solve its problems and address new use cases
  - Real distributed structure (master-less)
  - Support for embedded platforms
  - Real-time capability
  - Etc.
- Built on DDS
  - Data Distribution Service, a proven OMG standard
  - Although layers of abstraction allow ROS 2 to use any middleware
  - ROS 2 & DDS support various per-publisher/subscriber QoS settings
- Has seen a lot of interest from companies
  - Technical Steering Committee composed of many tech leaders
  - And many more are actively contributing

Tracing ROS 2

- LTTng instrumentation part of the ROS 2 core
  - gitlab.com/ros-tracing/ros2_tracing
- Instrumentation not part of the distributed binaries
  - However, there are plans to change that
- Closely integrated with ROS 2
  - To encourage use/adoption
  - ROS 2 CLI tools
  - ROS 2 launch/deployment system

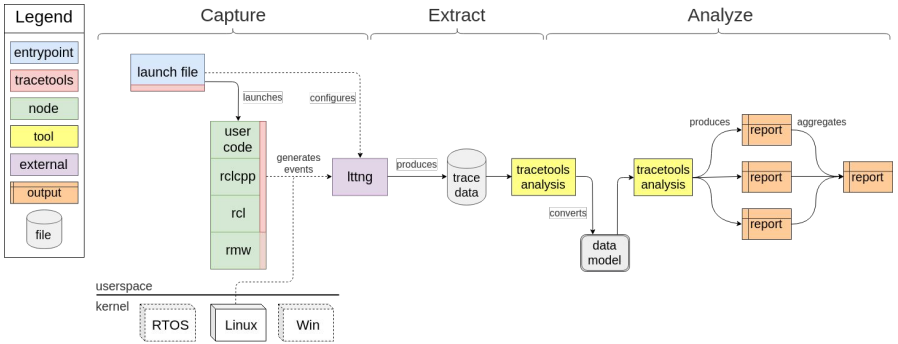

Figure 5. General architecture and workflow.

Tracing ROS 2 (2)

- Analysis tools
  - gitlab.com/ros-tracing/tracetools_analysis
  - Python-based
  - Uses babeltrace Python bindings
  - Currently using Jupyter notebooks
  - Goal: simple tools to write quick and simple analyses
- Interest in supporting Trace Compass
  - Especially Theia Compass

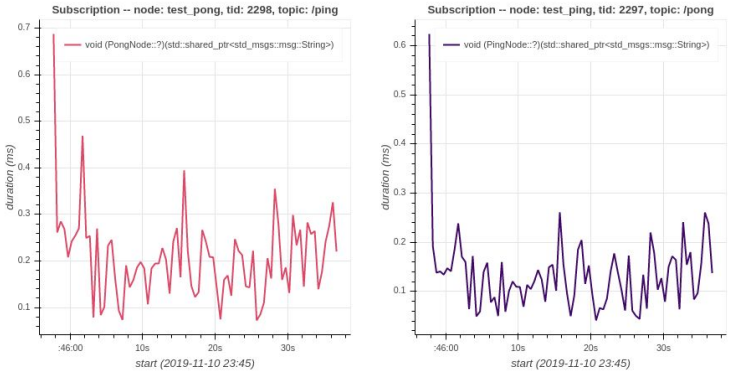

Figure 6. Example: subscription callback durations.

Upcoming work and conclusion

- A lot of interest and potential for ROS 2
- ROS 2 supports quality of service settings through DDS
  - Look into how 5G interacts with that
- Some different options
  - General QoS analyses/views
  - QoS-aware model for message-passing services
  - Starting with ROS 2

Questions?

- christophe.bedard@polymtl.ca