CPS: Medium: GOALI: Enabling Safe Innovation for Autonomy: Making Publish/Subscribe Really Real-Time

PI: Jim Anderson (UNC). Co-PIs: Ron Alterovitz (UNC) and Shige Wang (Motional Inc.)

Students: Rohan Wagle, Sizhe Liu, and Angelos Angelopoulos

https://www.cs.unc.edu/~anderson/projects/pubsub.html

UNC REAL-TIME SYSTEMS GROUP

Motional

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Motivation

In the automotive industry today, there is fierce pressure to innovate with respect to autonomous features. **ROS**, or the **Robot Operating System**, has emerged as a key technology to fuel such innovation. Despite its name, ROS is actually **publish/subscribe (pub/sub)** middleware that facilitates the modular construction of graph-based robotics applications. This modularity enables rapid innovation through code reuse.

However, for full autonomy to ever become a reality, **real-time safety certification** will be needed to ensure that critical computations complete **on time**. Here, ROS is fundamentally broken because it lacks sufficient features for supporting verifiable real-time execution.

Project Focus

Autonomous vehicles (AVs) typically use only a few multicore machines, augmented with accelerators like GPUs, with most time-critical processing happening on individual machines. Using ROS in this context incurs baggage, because it was developed largely for single robots that distribute their processing across multiple networked computers. **For AVs, pub/sub should be optimized first for multicore+accelerator platforms, with networked contexts introduced later.**

Objectives

1. Provide a sound OS foundation for real-time pub/sub.
2. Devise graph response-time analysis.
3. Devise techniques for arbitrating GPUs in processing graphs.
4. Devise and implement a real-time pub/sub framework.
5. Evaluate the produced framework.

Activities

- Summer internships at Motional Inc.
- Bi-weekly meetings with industrial partners to obtain industry perspectives on AV Pub/Sub.

Goal 1: Provide a sound OS foundation for real-time pub/sub

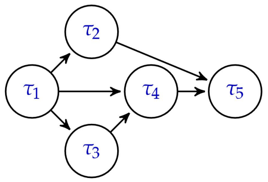
In AV contexts, ROS is almost always applied atop of Linux. Of the scheduling options available in Linux, *SCHED_DEADLINE* is the most real-time capable. *SCHED_DEADLINE* was originally adopted into Linux in part because of relevant response-time analysis produced by our group.

However, its implementation has since been altered to handle new use cases, and these alterations have compromised its ability to uphold response-time guarantees. We will devise patches to it that provably restore these guarantees.

Goal 2: Devise graph response-time bounds

While there has been considerable work directed at real-time processing graphs, the question of how to best support such graphs atop of SCHED_DEADLINE has never been addressed. We will do so by determining how to best break a graph into schedulable entities, assign them deadlines, and obtain graph response-time bounds.

Additionally, *SCHED_DEADLINE* requires execution-time budgets for executable code. Budget management in processing graphs is a complex topic that we are currently exploring under an ongoing NSF Small project. The results obtained there will be leveraged in the proposed effort.



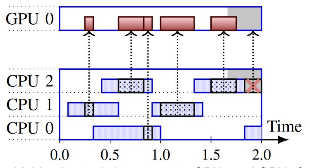
$$\begin{tikzcd} & \tau_2 \arrow[drr] & & \\ \tau_1 \arrow[ur] \arrow[r] \arrow[dr] & & \tau_4 \arrow[r] & \tau_5 \\ & \tau_3 \arrow[ur] & & \end{tikzcd}$$

**A DAG processing graph*

Goal 3: Arbitrating GPU accesses in processing graphs

In AV contexts, accelerator usage is ubiquitous, and GPUs are likely the most commonly used accelerator type. GPUs require arbitration to ensure predictable execution and mitigate hardware interference.

We will devise such arbitration strategies and assess their impact (observed and analytical) on graph response times.

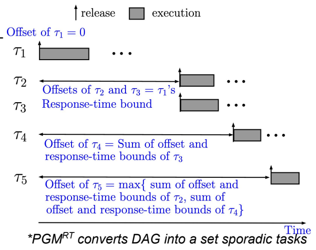


**Arbitrating tasks' accesses to CPUs and GPUs for real-time scheduling*

Goal 4: Produce a real-time pub/sub reference implementation

Through source-code reviews of ROS-based systems, we will produce a list of features that a real-time pub/sub framework should have from an application perspective. Then, based on the analytical results from Goals 1–3, we will design and implement our own pub/sub middleware.

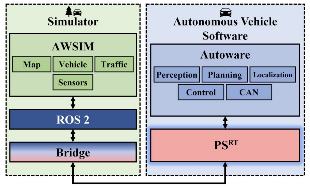
We plan to achieve this by extending *PGM^{RT}* (processing graph method for real time), which is a multicore+GPU middleware developed by our group for executing general real-time processing graphs on POSIX-compliant OSs (e.g. Linux).


release
execution
Offset of $\tau_1 = 0$
τ_1
τ_2
Offsets of τ_2 and $\tau_3 = \tau_1$'s Response-time bound
τ_3
τ_4
Offset of τ_4 = Sum of offset and response-time bounds of τ_3
τ_5
Offset of τ_5 = max{ sum of offset and response-time bounds of τ_2, sum of offset and response-time bounds of τ_4}
Time

**PGM^{RT} converts DAG into a set sporadic tasks*

Goal 5: Evaluate the produced framework

We will evaluate our framework by porting existing ROS-based implementations of AV functions such as perception and planning to use our pub/sub middleware instead. We will then compare the two based on analytical response-time bounds and observed response times. In this work, we will use a detailed AV simulator for Autoware called AWSIM.



Broader Impacts

TOPICS (Talking Over Papers In Computer Science)

CS Undergrad Women's Reading Group

- We read papers out loud (you read that right) and discuss them.
- We've read papers on everything from quantum computing to AI to computer security to Turing Award lectures.
- We also talk about writing tips, applying to grad school, and other things.
- We have only two rules:
  - There's no such thing as a dumb question.
  - We do absolutely no work outside of our one hour per week.
- It's a fun group with interesting discussions.

Award ID#: 2333120
Jan 2024 to Dec 2026 (est.)