The State of Open-RMF

ROSCon 2024

Quick Recap

Quick Recap

The interoperability dilemma

Many Amazing Robots

Cleaning

Food Serving

Security

Logistics

Delivery

but they need to communicate with each other

Guten Tag!

こんにちは

and resources have already been spent to develop and deploy disparate platforms

We can'~~t~~ all get along

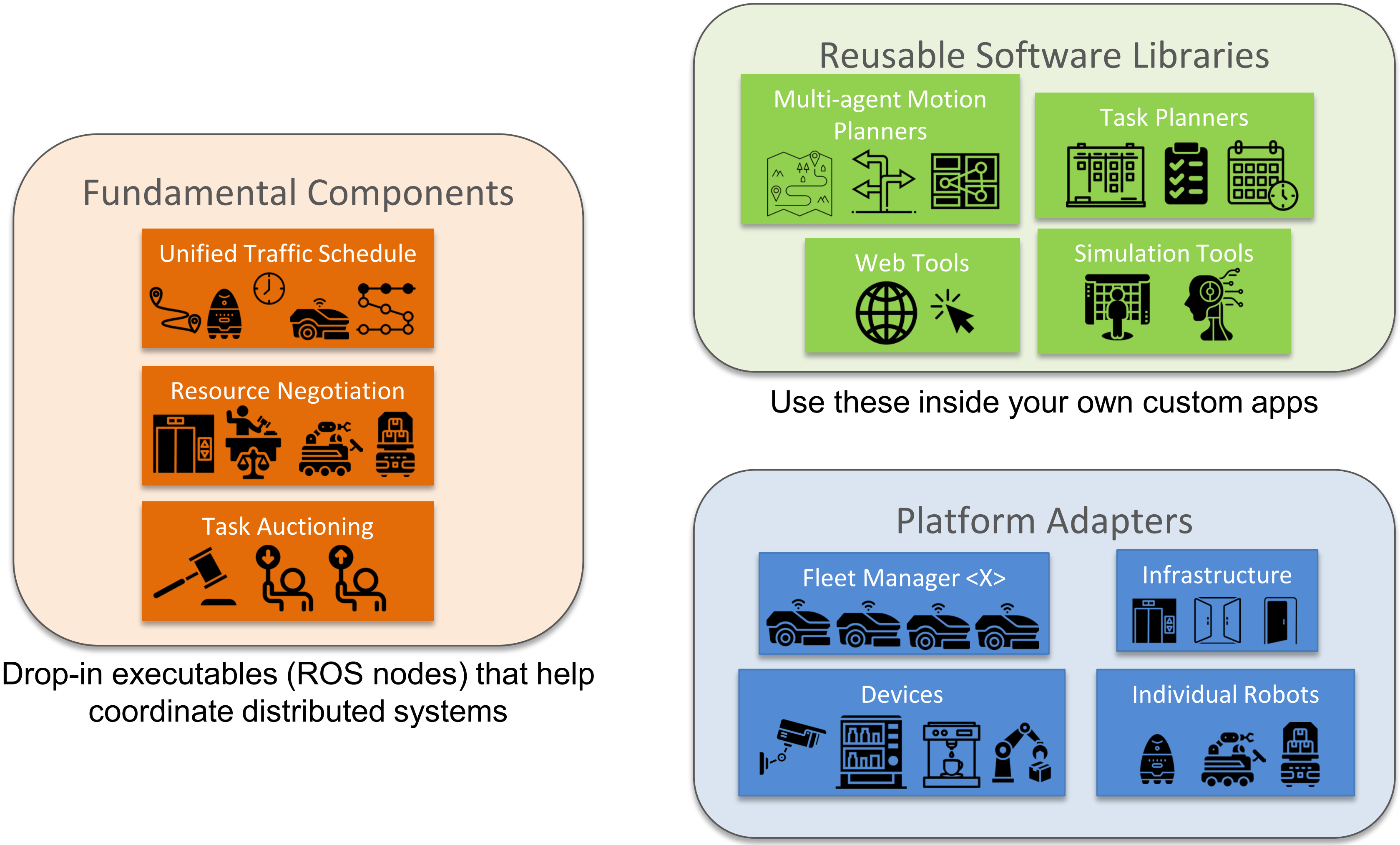
Quick Recap: Assumptions

- Robots from multiple vendors will be needed
  No single vendor offers a robot that can do everything
- Not all mobile robots can offer the same inputs/outputs or capabilities
  Different vendors have different APIs and levels of control available
  Some platforms may have more or less “intelligence” than others, e.g. AGV vs AMR
- Dynamic environments with unpredictable elements
  The robots need to operate amidst human traffic
  Furniture or other items may be moved, creating unanticipated static obstacles
- Centralized command & control is not (always) an option
  Different platforms might have their own fleet management tools
  They might get installed and maintained by different system integrators

Quick Recap: Design

- Shared traffic schedule
  All robots openly communicate their intended trajectories through space-time
- Negotiate to resolve conflicts
  A **peer-to-peer** negotiation process, similar to “conflict-based search” is used to fix traffic conflicts when they arise
- Auction tasks to find the “most available” agent
  If multiple different robot platforms can perform a task, they will “bid” to see who can get it done at the “lowest cost”
- Provide a stable (while evolving) SDK
  We are constantly improving the implementation and algorithms used
  We maintain stable APIs in the C++ and Python SDKs so the efforts of early adopters are not wasted
  New features/capabilities become available by expanding the APIs of the SDKs

✕ No specification or stable wire protocol (for now)
The wire protocol often needs to change as the algorithms improve
Users should integrate using the SDK, which hides the details of the wire protocol and maintains compatibility



Drop-in executables (ROS nodes) that help coordinate distributed systems

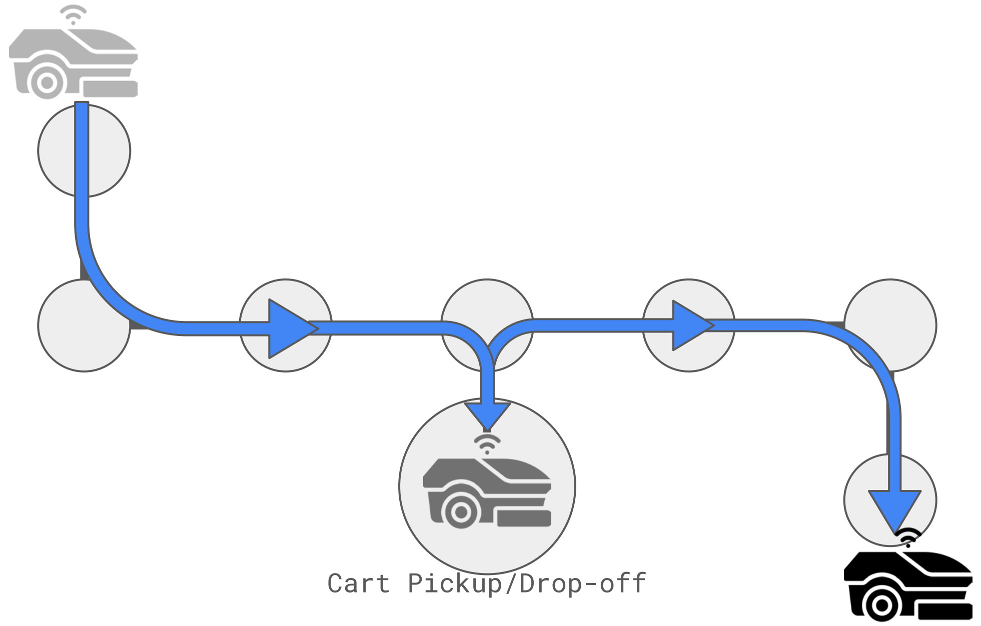
Use these inside your own custom apps

Connect hardware to your system

Present

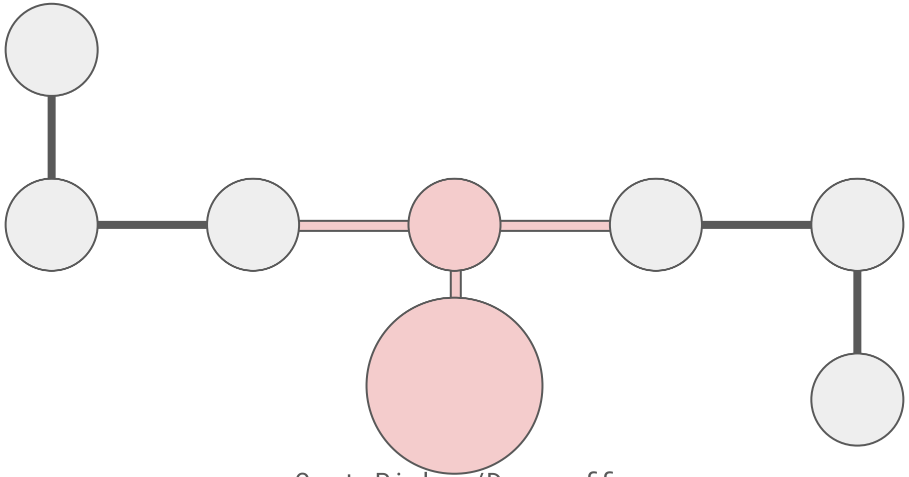
Recent Additions: Mutex Groups

Some operations may need specific maneuvers that require more clearance than what the usual traffic negotiation system can provide.



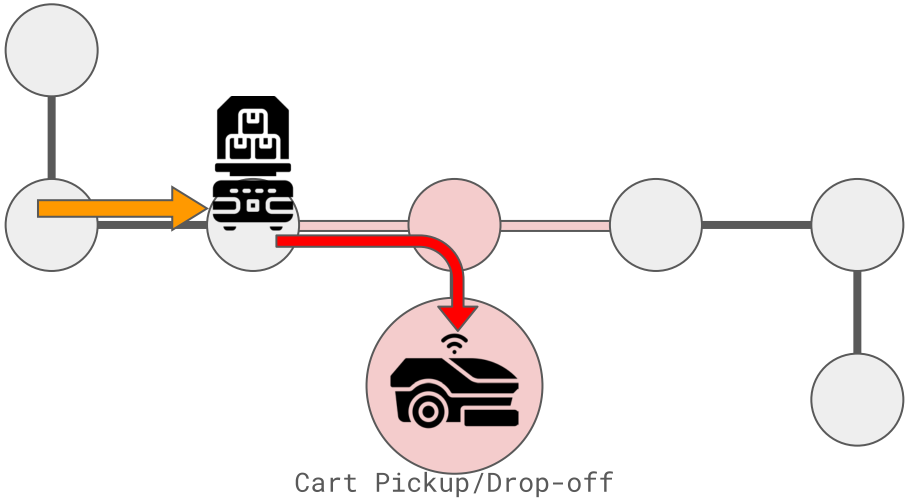
Recent Additions: Mutex Groups

With mutex groups, we can mark this cluster of lanes and vertices as belonging to one mutex group (marked red).



Recent Additions: Mutex Groups

Any other robots that want to enter this cluster must wait until the mutex group is unlocked, regardless of what the traffic negotiation system would allow.



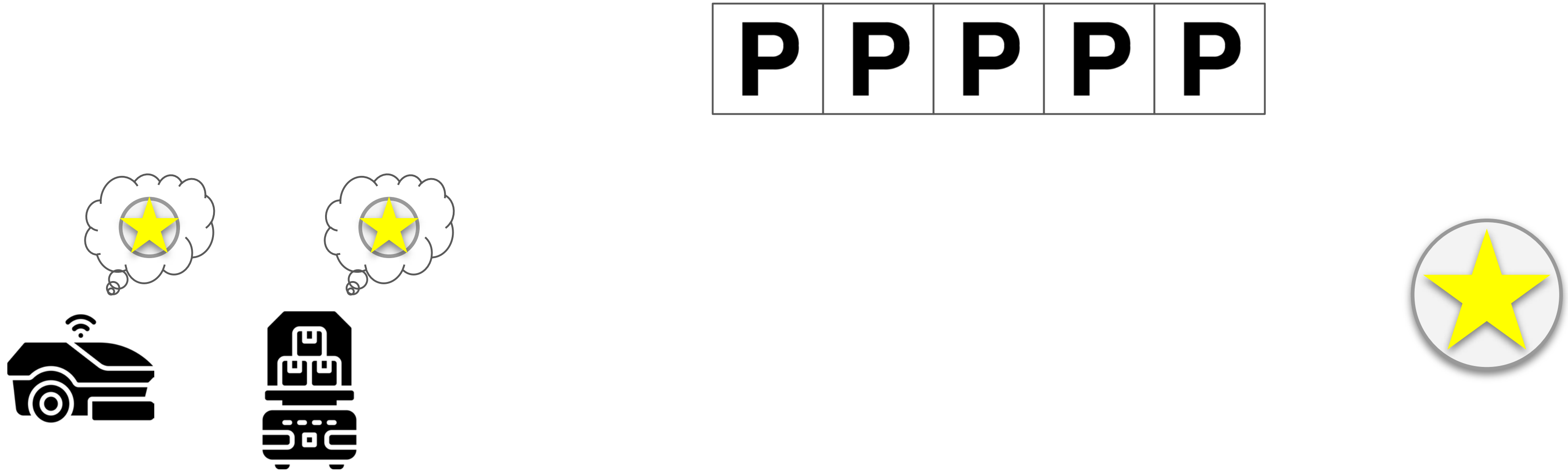
Recent Additions: Robot Commission

New “commission” feature allows robots to be temporarily taken in and out of service

- A decommissioned robot can automatically redistribute its queued tasks to other compatible robots in the same fleet
- A decommissioned robot will stop receiving new task requests from Open-RMF
- You can choose to recommission a robot at any time
- A recommissioned robot may immediately receive compatible task requests that were queued up for other robots in the same fleet

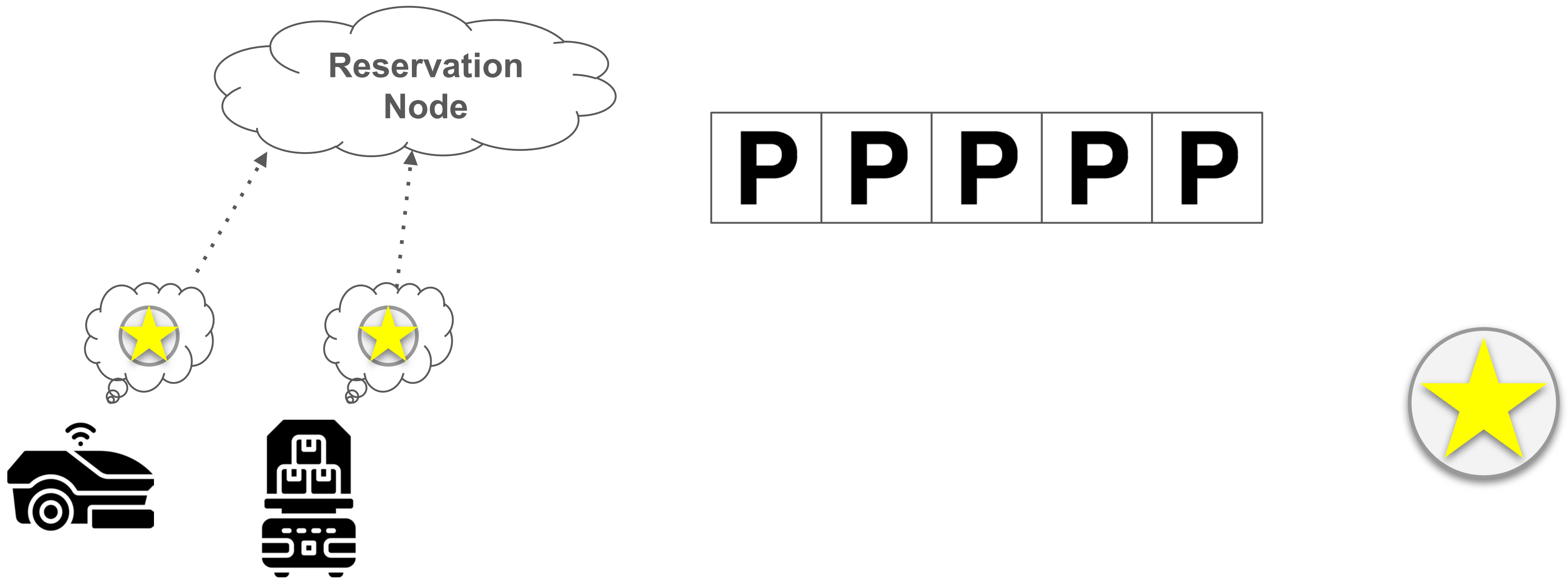
Recent Additions: Reservation System

There are often cases where multiple robots have the same destination at the same time, e.g. needing to receive a payload at a pickup point.



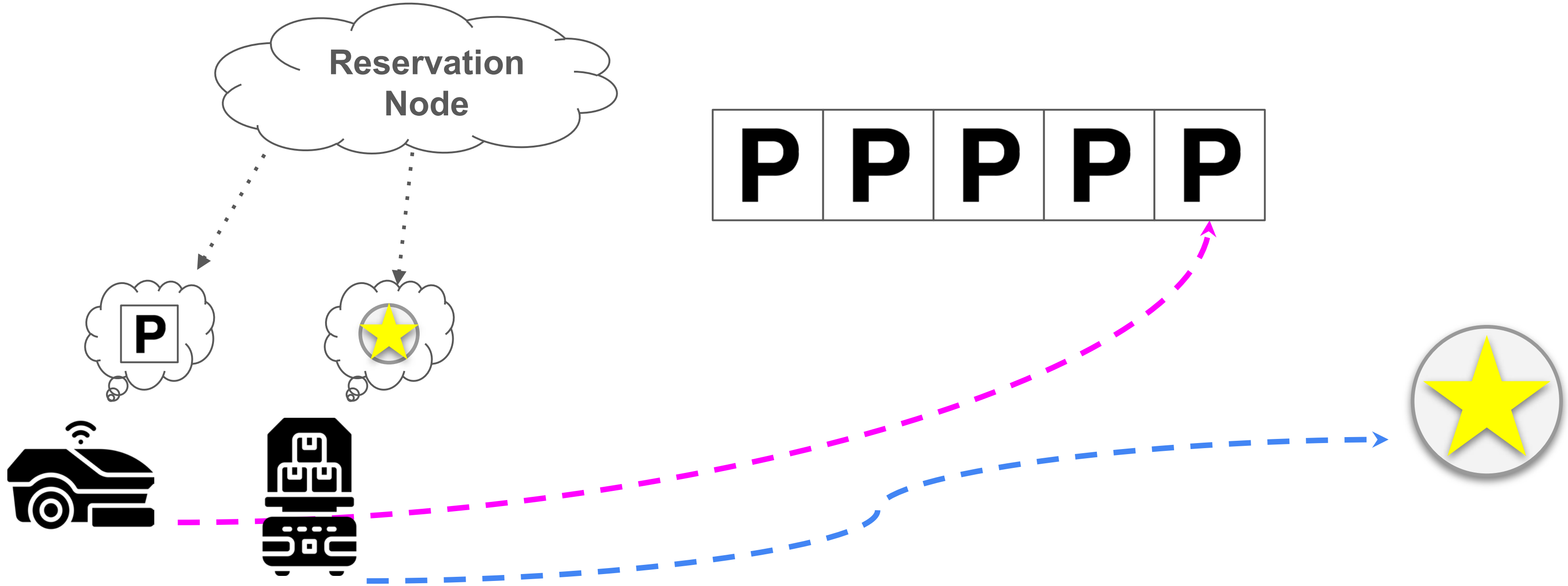
Recent Additions: Reservation System

They will each ask the Reservation Node to reserve the spot for them



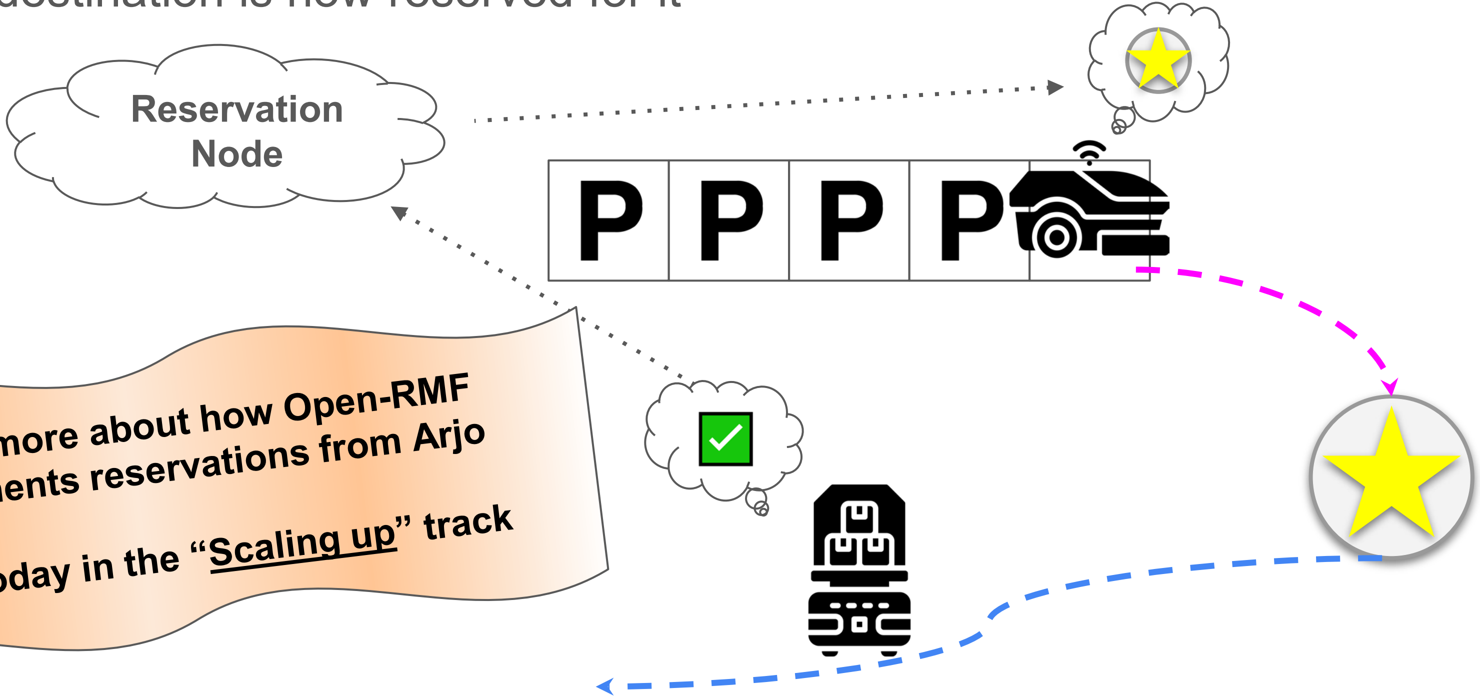
Recent Additions: Reservation System

One will be granted the reservation and the other will be given a parking spot to wait at



Recent Additions: Reservation System

When the first robot is finished, the reservation node will notify the parked robot that the destination is now reserved for it



Learn more about how Open-RMF implements reservations from Arjo

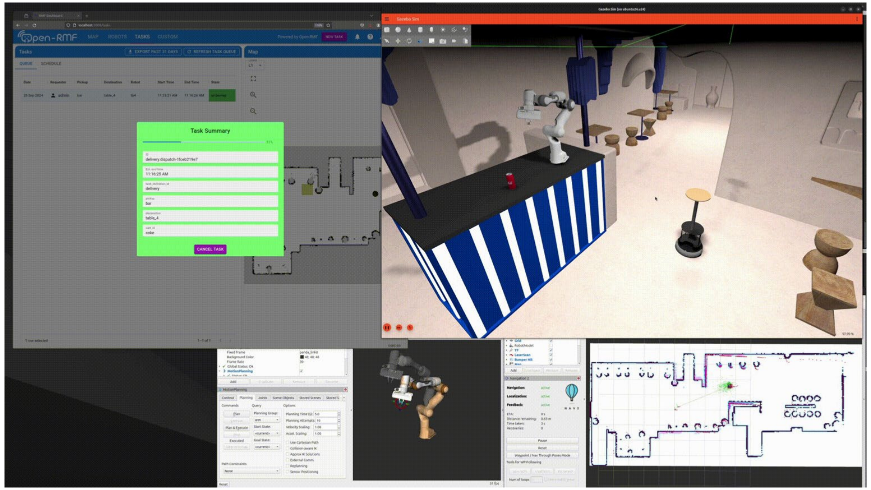
14:20 today in the "Scaling up" track

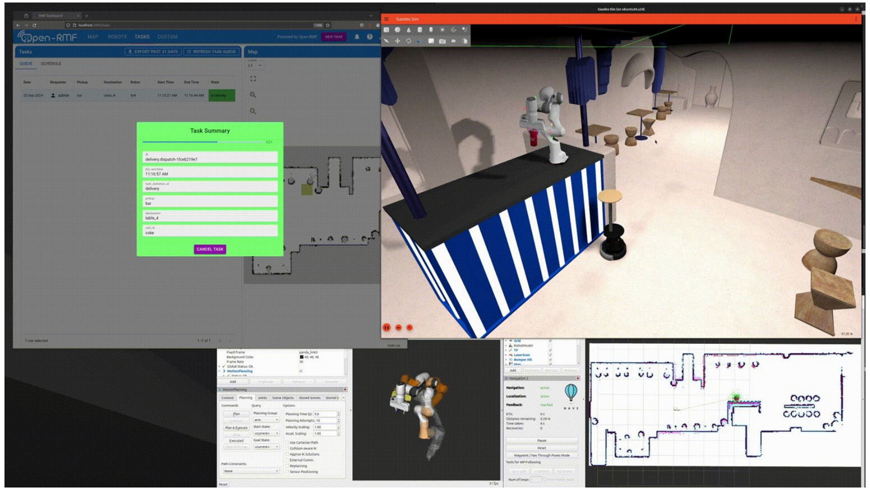
Recent Additions: Ionic Demo

The new demo world for Gazebo Ionic has Open-RMF integration!

- Navigation performed by Nav2
- Workcell motion planning done by MoveIt!
- Actions coordinated by Open-RMF

https://github.com/gazebosim/ionic_demo





Future

“next generation”

Past Scope

Emphasis on mobile robots:

- How do they coordinate the sharing of space? e.g. crossing paths in a corridor
- How do they share infrastructure? e.g. automated doors and elevators
- How do they synchronize with other devices? e.g. a robot arm that loads a delivery

Modest scale—not too much robot density (for now):

☑ Hospitals (clean, telepresence, schedule and ad hoc deliveries)
☑ Malls, hotels, airports (clean, deliver, security, assistance)
☑ Libraries (clean, deliver, scan bookshelves)
✕ Dense warehouses

Future Scope

Orchestration of automated devices in general:

- How do devices coordinate their activities?
  e.g. devices work in parallel when possible, and sync when needed
- How do devices share resources?
  e.g. optimizing how resources are allocated to minimize the overall delay to all devices
- How do we make the overall system understandable to humans?
  e.g. human operators should not usually be surprised by the system's behavior, and when a surprise does happen, a good explanation should be readily available

Any scale:

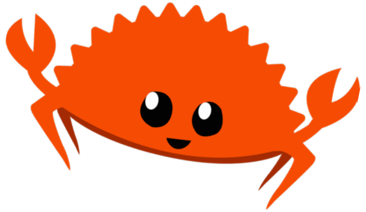
- ✓ Dense warehouses
- ✓ Factories
- ✓ Entire cities

Complaints we’ve received and issues we’ve observed

- Despite being contained in just a few headers, the public API is **difficult for users to find**. And if they do find it, they might not know **how to begin** with it.
- Difficult to contribute / improve / modify the core because of the complexity and the risk of data races, undefined behavior, etc.
- Difficult to handle special situations
- Path planning is not customizable enough
- Traffic negotiation is not customizable enough
- Task behaviors are not customizable enough
- Open-RMF tries to do too much
- Open-RMF doesn’t do enough

Both of these at the same time and in the same situation

Is this even solvable??



New paradigms:

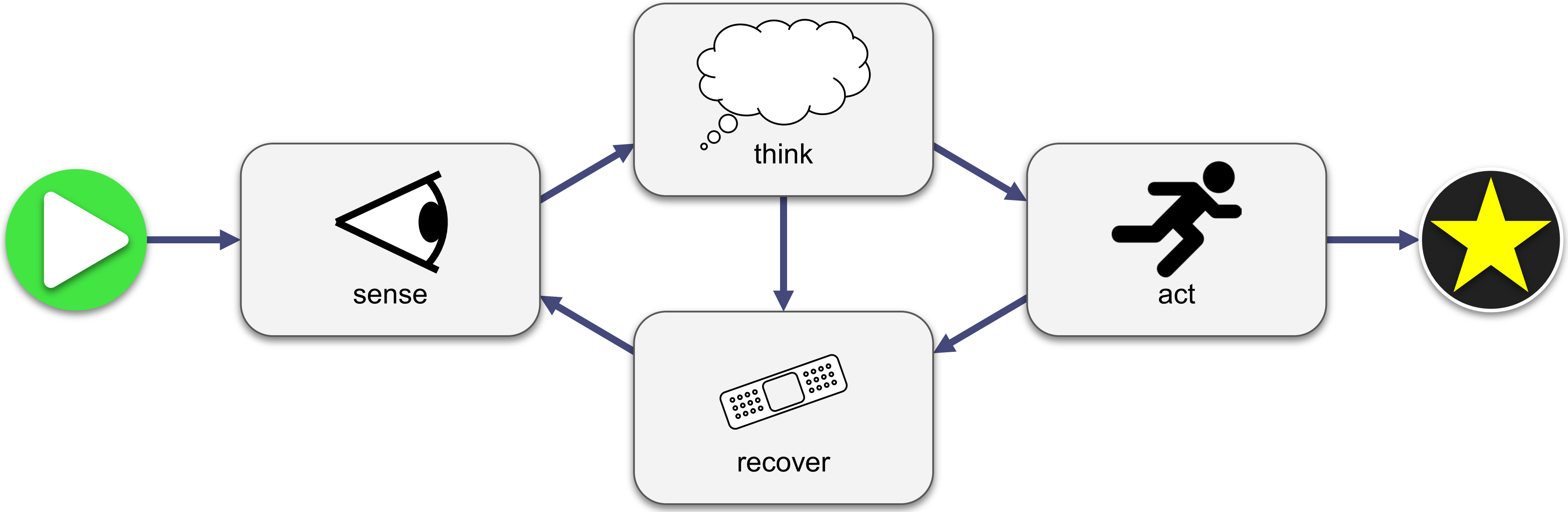
- High-performance memory-safe language (Rust)
  - Eliminate risk of data-races, undefined behavior, and inexplicable crashes
  - Still get maximum performance, minimal overhead, and full benefit of multi-threading with none of the risks
  - Easier to contribute since each new LoC doesn’t risk introducing new undefined behavior
- “Entity Component System” implementation (Bevy)
  - Extreme modularity
  - Extreme extensibility
  - Higher performance than traditional use of interfaces and smart pointers (better CPU cache optimization)
- Service Workflow Architecture
  - All behaviors are defined in terms of workflows built out of services and actions
  - Every workflow is, itself, a service / action
  - Users can define custom workflows and insert them into the system

Workflows

Inspired by “Workflow Nets”, a special case of Petri Nets

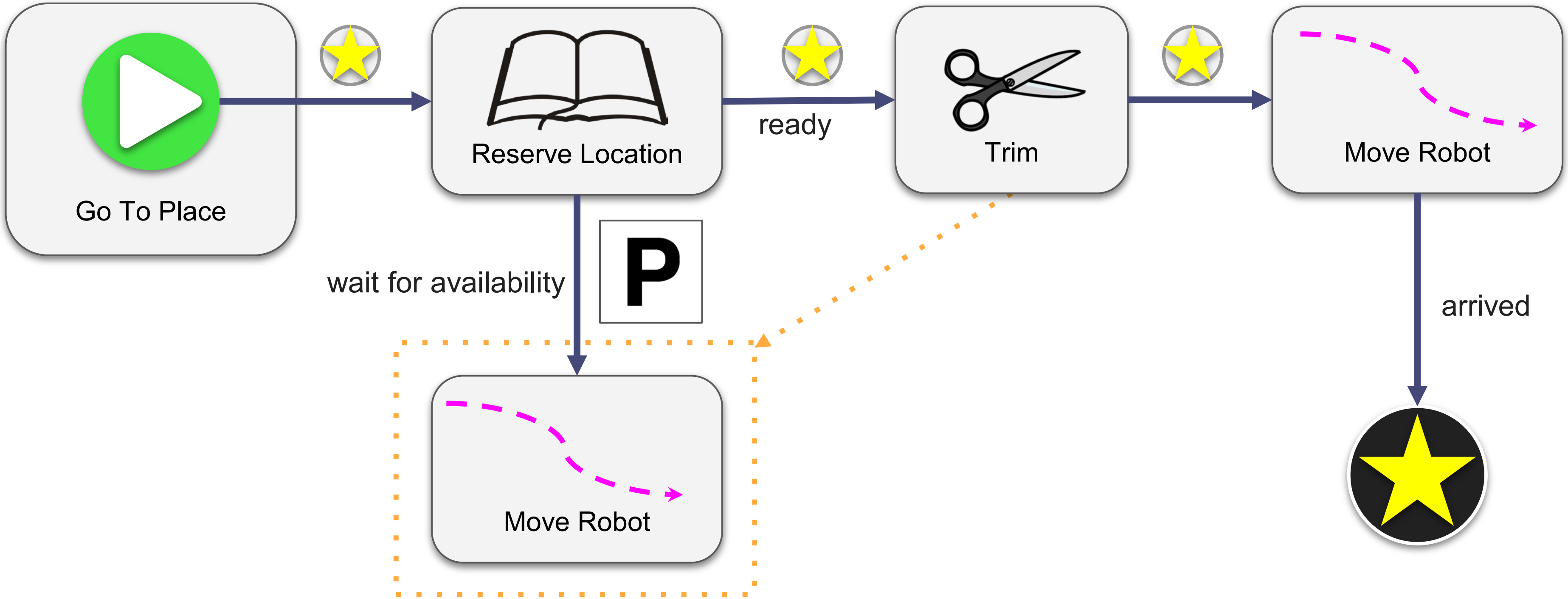
(thanks to Thomas Horstink for presenting on Petri Nets at ROSCon last year)

Define the flow of activities as a workflow diagram and it can be executed



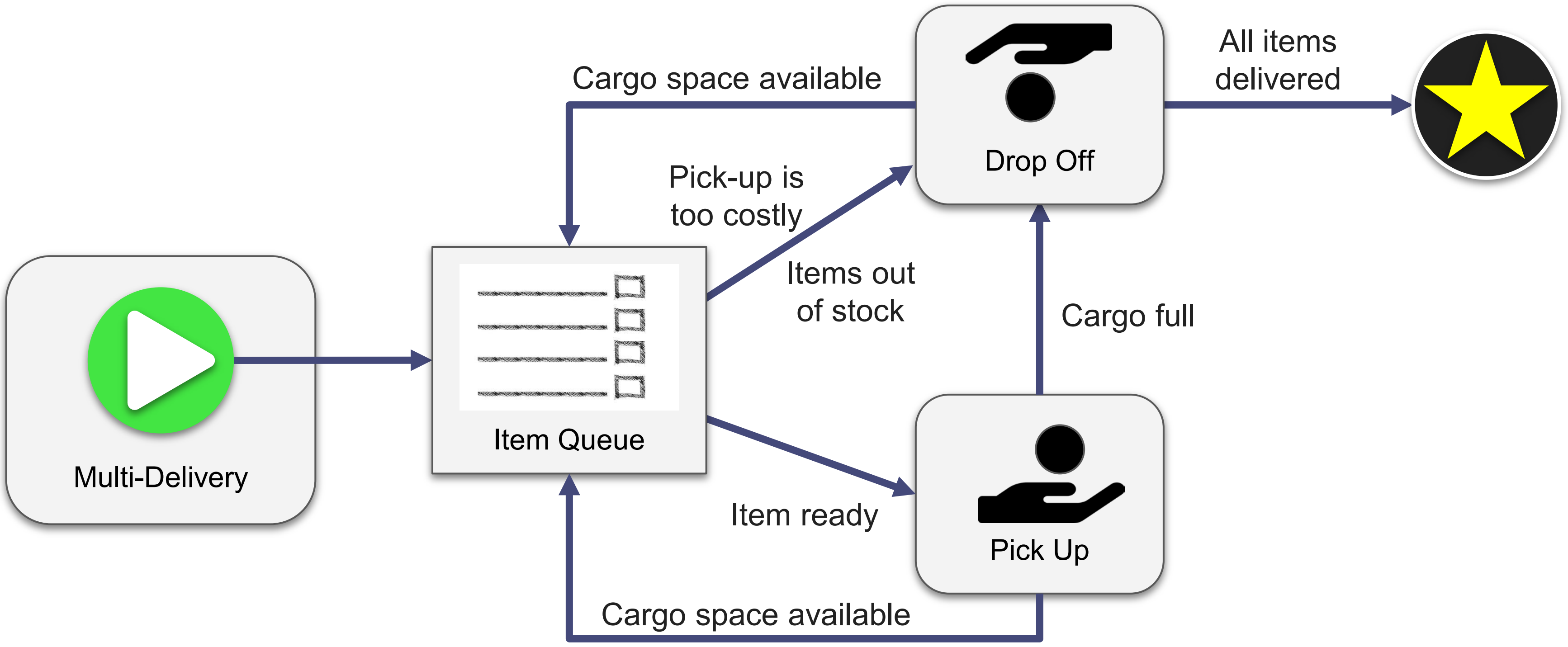
Workflows

Good at intuitively expressing parallel activities and syncing them as needed



Workflows

Good for expressing cycles



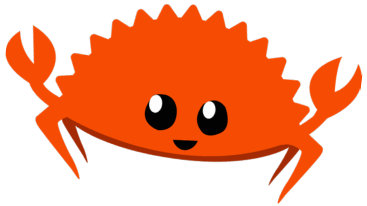
Workflows

Implemented as bevy impulse: https://github.com/open-rmf/bevy_impulse

Primitive operations supported out of the box:

- **Scope** create a sub-workflow within a larger workflow
  - When the termination node of a scope is reached, all remaining activity in the workflow will be forced to stop
  - Good for racing different branches against each other
- **Fork Clone** clone (copy) an output and send it down multiple branches at the same time
- **Filter** check the value of the input and dispose it if a condition is not met
- **Unzip** split one Output<(A, B, C, …)> into Output<A>, Output<B>, Output<C>, …
  - Each output can then be sent down its own branch in the workflow
- **Join** merge multiple branches into one, i.e. Output<A>, Output<B>, Output<C> into Output<(A, B, C)>
- **Listen** trigger a node each time one or more buffers in the workflow is modified
- **Spread** if an output contains a collection of elements (e.g. an array, a vector), give each element its own thread along the same branch
- **Collect** gather up some (or all) of the upstream activity into a collection before sending off the whole collection as one output
- **Gate** temporarily block some branch of the workflow from running, collecting the inputs for that branch into a buffer
- **Trim** cancel all current activity in some part of a workflow

ROS Client Library



Several ROS 2 client libraries exist for Rust, but we're committing to rclrs

- https://github.com/ros2-rust/ros2_rust
- Community-driven
- Monthly working group meetings
- Practices are aligned with the broader ROS ecosystem
- Aims to become a first-class and fully featured ROS 2 client library

Enhancing rclrs is officially on the Open-RMF roadmap

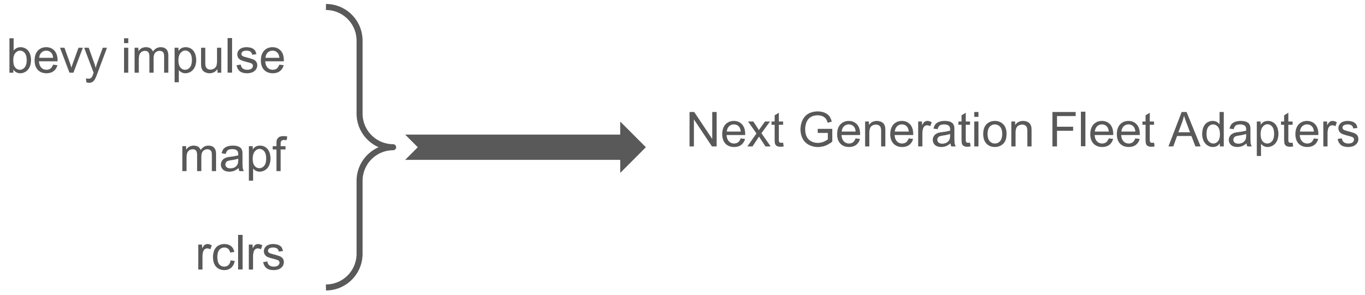
Multi-agent Path Finding

More advanced MAPF

https://github.com/open-rmf/mapf

- Implemented in Rust
- Customizable
- Scales well to large and complex problems
- Out-of-the-box Continuous Conflict-Based Search planner

What’s coming



First Targets

→ MiR API
  ↳ MiR has an open specification which the Open-RMF team is very familiar with

→ VDA 5050
  ↳ Open specification which is popular in the European auto manufacturing industry

→ Native Integration
  ↳ We’ll provide a plugin for the ROS 2 nav stack (Nav2)
  ↳ Any Nav2 robot that uses the plugin would immediately integrate seamlessly with traffic negotiated by Open-RMF

We love feedback!

Open-RMF Project Management Committee (PMC) every two weeks

- 01:00 UTC+0 Tuesday (folks on the west side of the Atlantic will see this Monday)
- Next session on 5 November 2024

Special Interest Group on Interoperability

- Not exclusively about Open-RMF, but it's always an appropriate topic
- Takes place this **first Thursday** of every month at 15:00 UTC+0

Project discussion board

- https://github.com/open-rmf/rmf/discussions

All sessions are on the OSRF Official Events Calendar