open robotics

Open-RMF

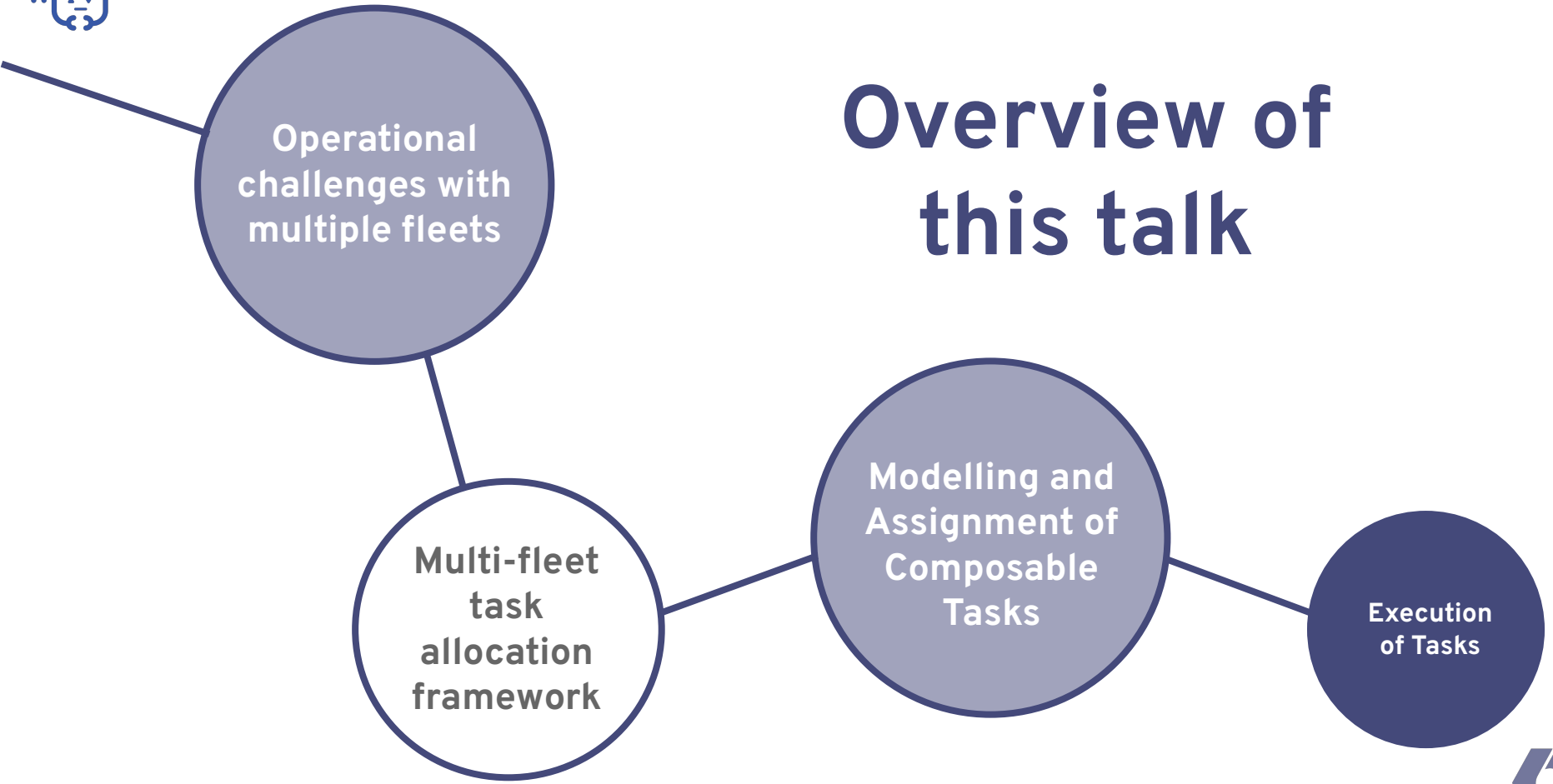
# How custom tasks are defined, assigned, and executed in Open-RMF

Grey, Yadunund

ROSCon Kyoto 2022

Day 1- 20 Oct

# Overview of this talk



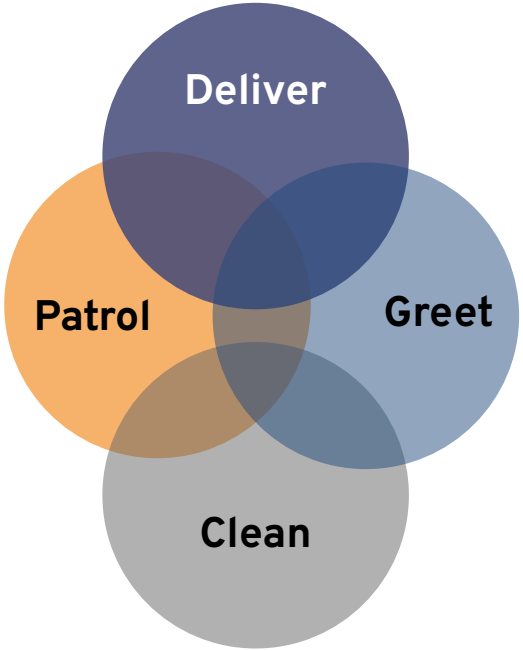



# The need for robotic **interoperability** is on the rise



*Image credits:*
*LG, AP, IIAC, Imaginechina, AFP, Ottonomy*

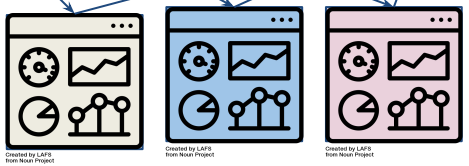
# Managing daily operations of multiple robot fleets is challenging




Capabilities of fleets may be specialized or shared


Operators

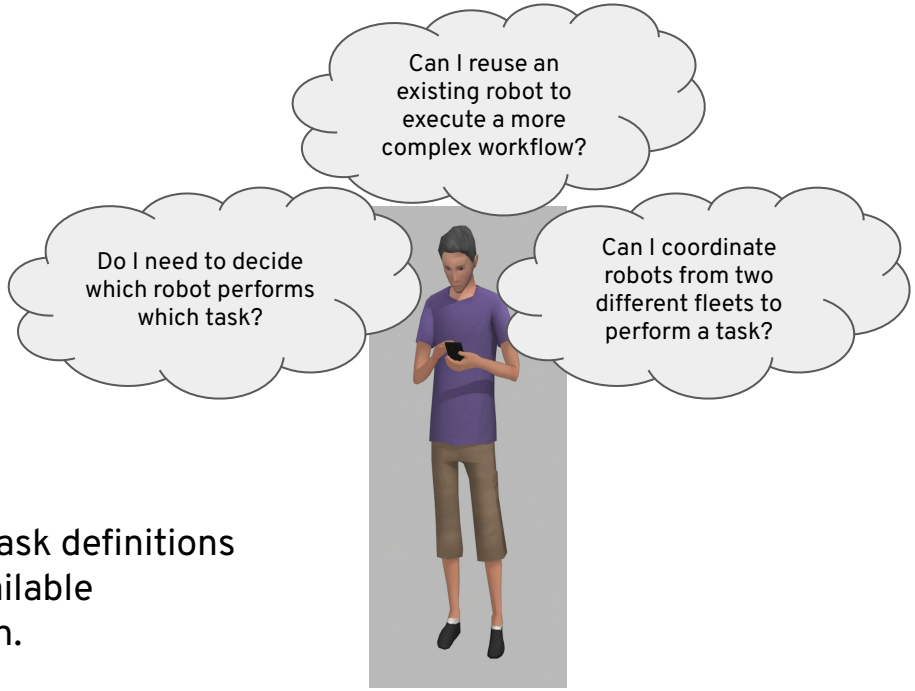

Robotic fleets in the facility



Dashboard for each fleet

# Functional requirements for a task management framework

- Flexible
- Predictive modelling
- Platform agnostic

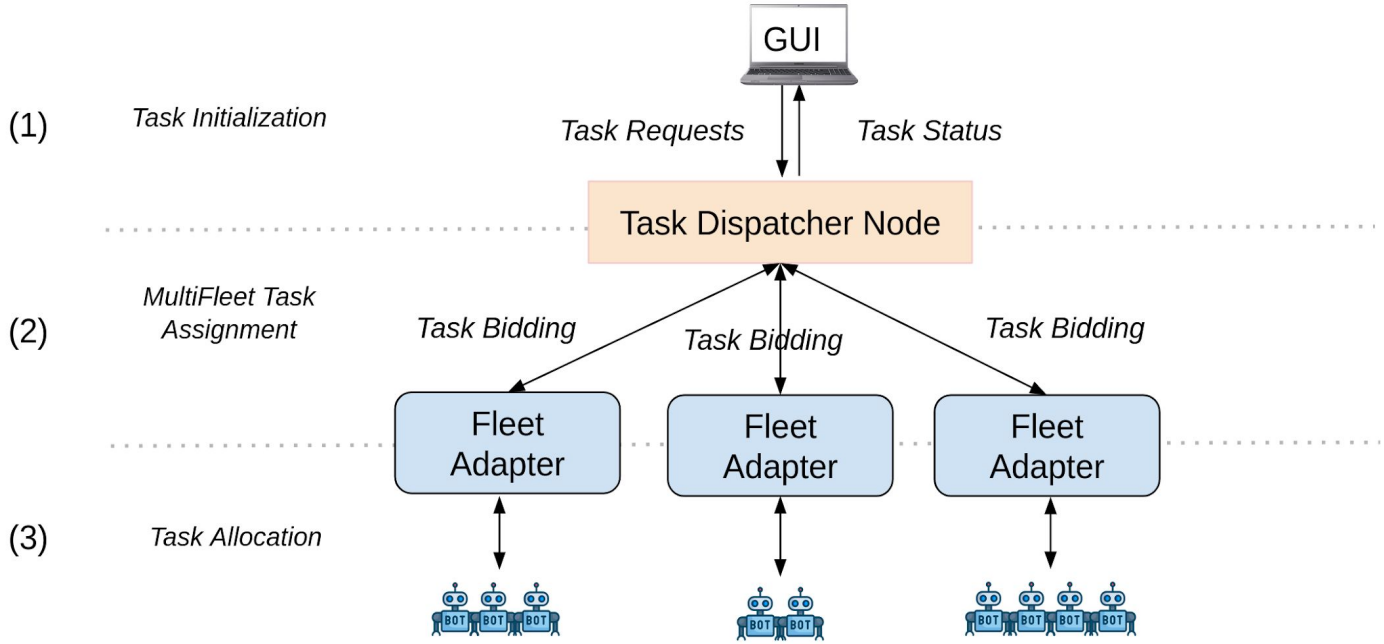



**Requirement**: A framework for constructing task definitions at runtime, assigning the task to the most available fleet/robot and managing the task's execution.

# Overview

## Fleet Adapters Bid for Tasks

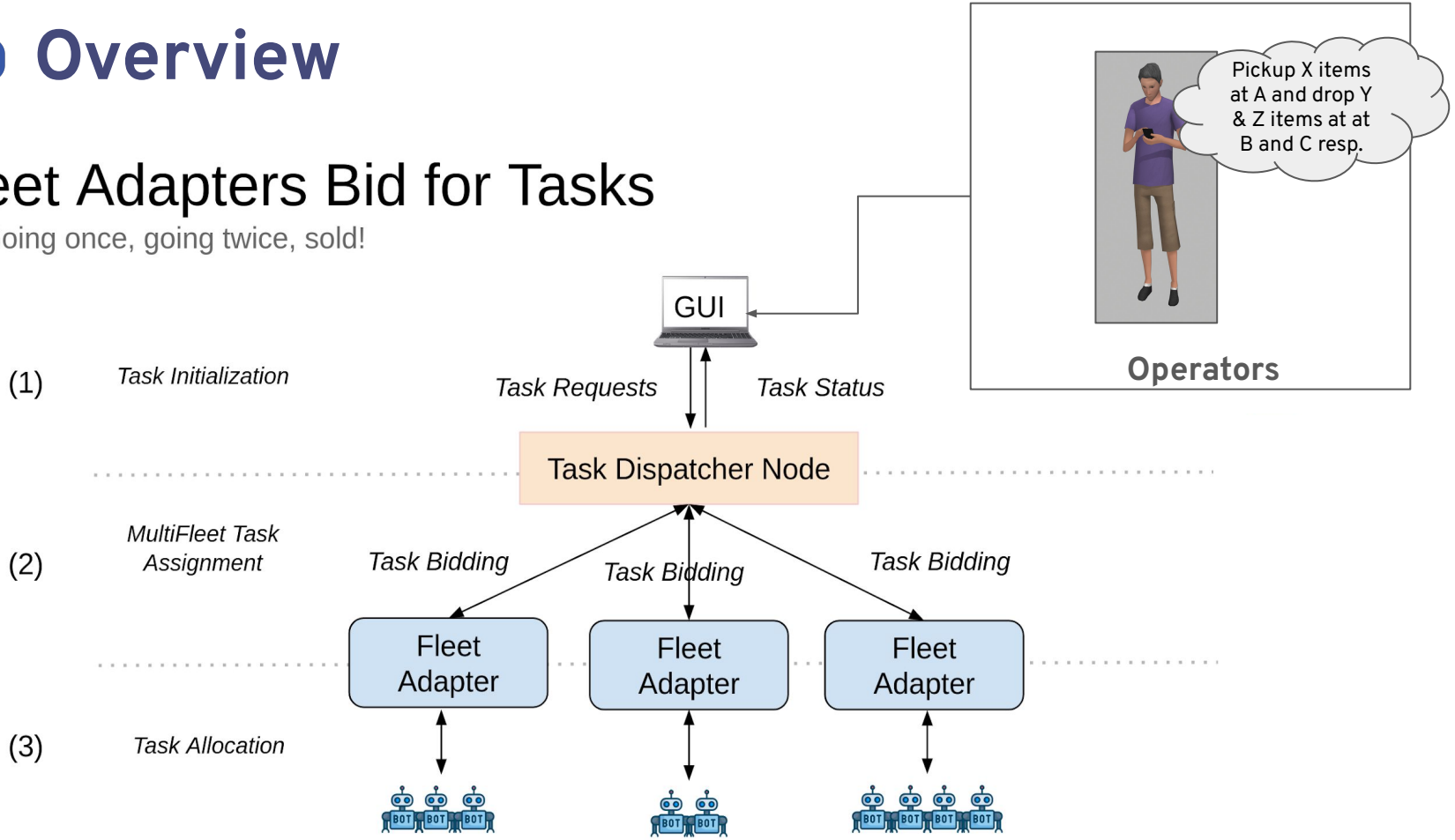
Going once, going twice, sold!



# Overview

## Fleet Adapters Bid for Tasks

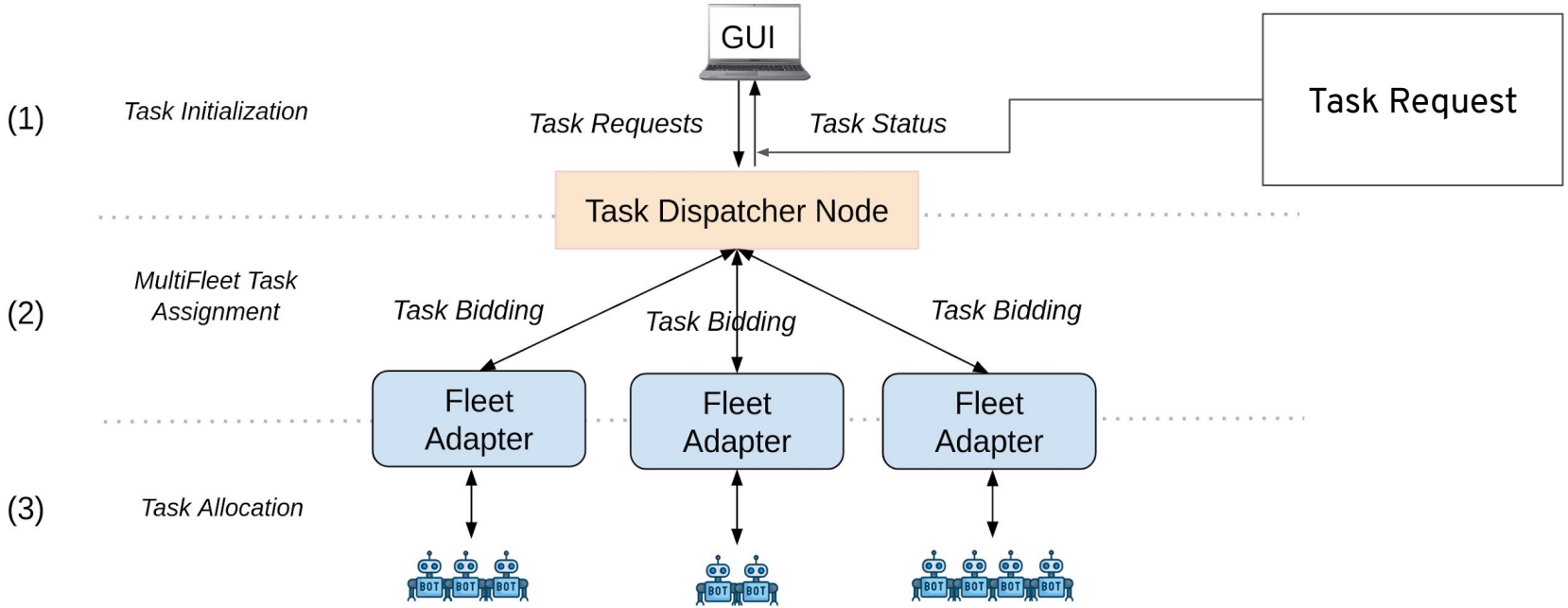
Going once, going twice, sold!



# Overview

## Fleet Adapters Bid for Tasks

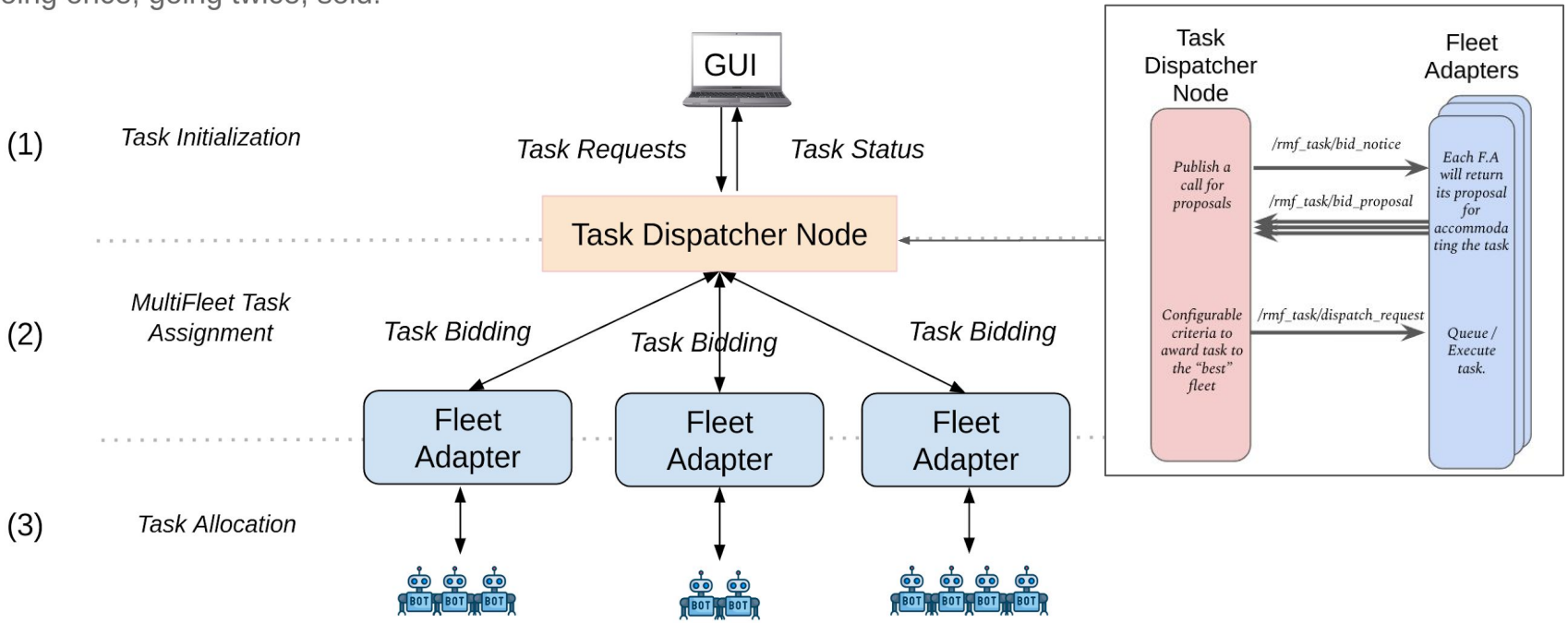
Going once, going twice, sold!



# Overview

## Fleet Adapters Bid for Tasks

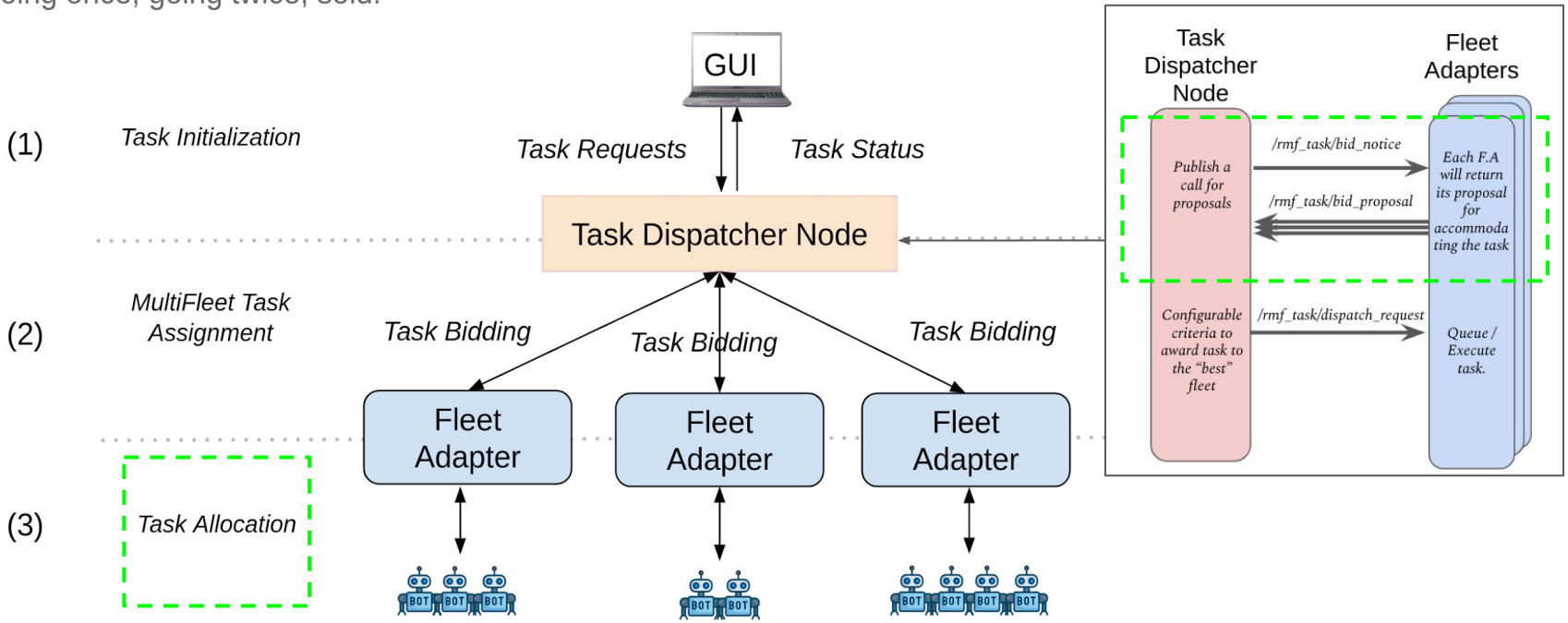
Going once, going twice, sold!



# Overview

## Fleet Adapters Bid for Tasks

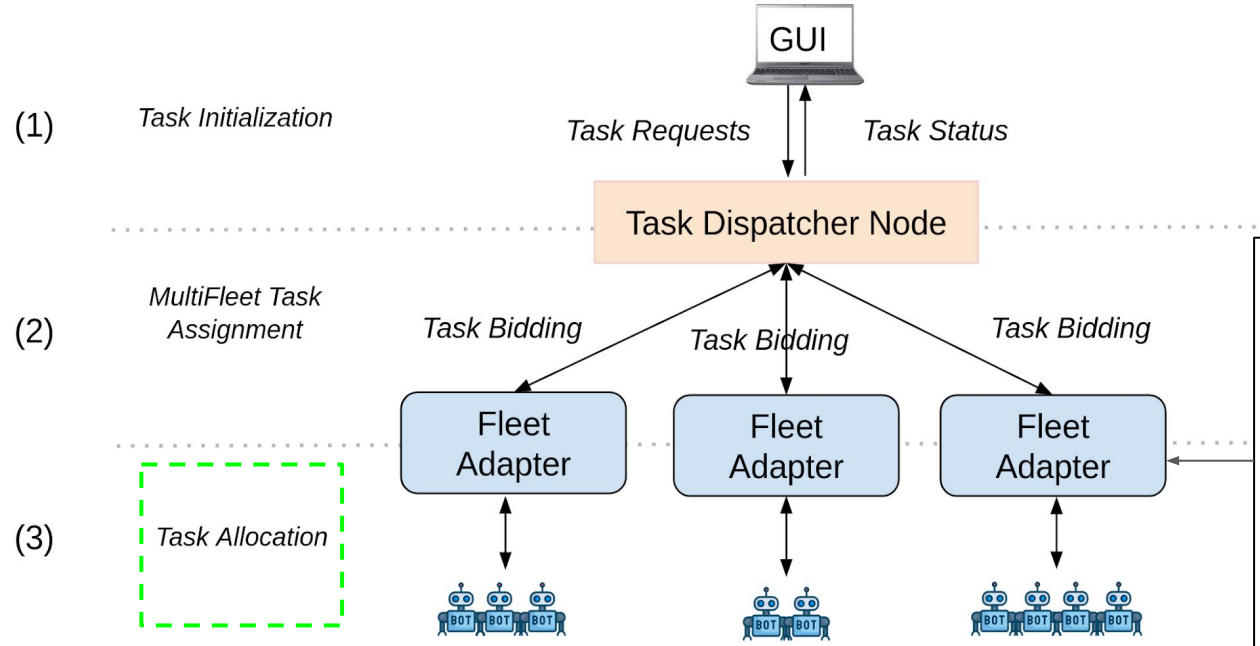
Going once, going twice, sold!

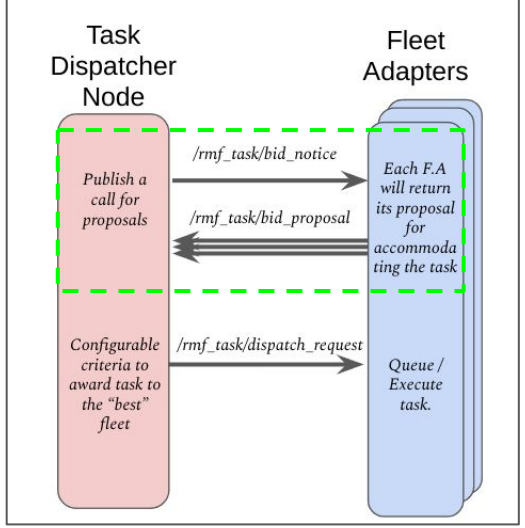


# Overview

## Fleet Adapters Bid for Tasks

Going once, going twice, sold!



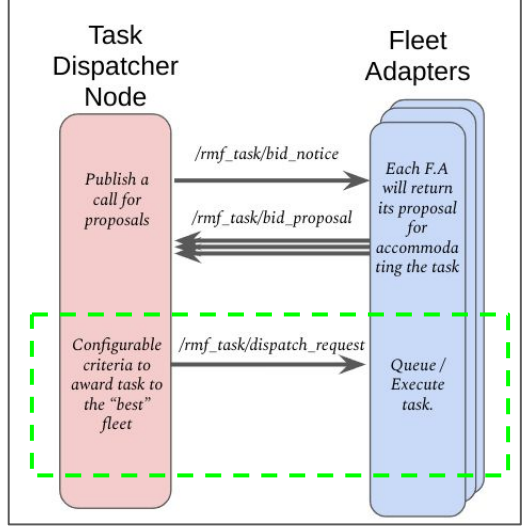

Each fleet adapter is capable of:

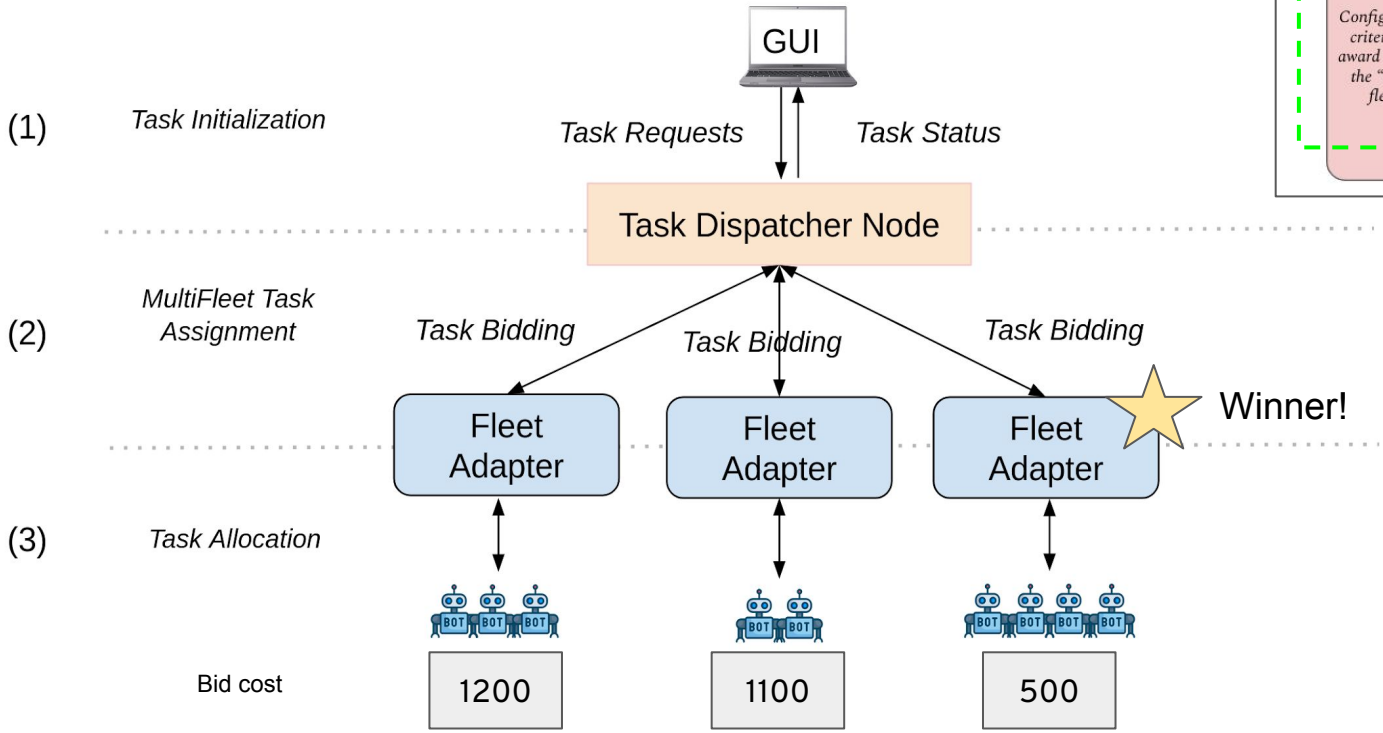
- Validating whether its fleet can perform the task
- Solving a Multi-robot Task Allocation problem

# Overview

## Fleet Adapters Bid for Tasks

Going once, going twice, sold!



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

# What is a "Task"?

## Task Description

serializable data structure that can be interpreted into...

## Predictive Model

inputs: (initial state prediction, robot description)
output: predicted state after task completion

Provided to a multi-agent task planner to search for a "minimum-cost" assignment of tasks to robots

⚠️ **Current Scope** ⚠️

The current implementation assumes **each task** is assigned to **one mobile robot** and that individual tasks **do not depend** on each other.

Future versions of RMF will support **multi-agent tasks** and **constraints between tasks**.

## Runtime

generates a sequence of task "phases"

## Task Phase

monitors state of robot and infrastructure to issue commands (e.g. navigate to location, open door, summon elevator) to fulfill an objective of the task

Human operators or external systems can request that a phase is skipped or repeated. This is helpful if a phase did not go as intended.

# Task Descriptions

## Simple, premade

```
{
    "category": "delivery",
    "description": {
        "pickup": {
            "place": "L2_pharmacy",
            "payload": [
                {"sku": "48052", "quantity": 2},
                {"sku": "37981", "quantity": 1}
            ]
        },
        "dropoff": {
            "place": "L3_ward32_bed4"
        }
    }
}
```

Common tasks can be given simple premade description schemas with a minimal set of parameters to fill in

Each **category** is associated with its own **description** schema that can be interpreted by task planners and executors.

More detailed instructions: https://osrf.github.io/ros2multirobotbook/task_new.html

## Custom, composed

```
{
    "category": "compose",
    "description": {
        "detail": "Drop off medication and then greet the patient",
        "phases": [
            {
                "activity": {
                    "category": "pickup",
                    "description": {
                        "place": "L2_pharmacy",
                        "items": [{"sku": 48052, "quantity": 2}]
                    }
                }
            },
            {
                "activity": {
                    "category": "dropoff",
                    "description": {
                        "place": "L3_ward10_bed4",
                        "items": [{"sku": 48052, "quantity": 2}]
                    }
                },
                "on_cancel": [{
                    "category": "dropoff",
                    "description": {"place": "L2_pharmacy"}
                }]
            },
            {
                "activity": {
                    "category": "greet",
                    "description": {
                        "place": "L3_ward10_bed4",
                        "language": "Hokkien"
                    }
                }
            }
        ]
    }
}
```

# Task Descriptions

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

## Predictive Models for Composed Tasks

Each leaf-node activities need to be either:

- an activity primitive with built-in support implemented in RMF
- a custom activity that the system integrator has plugged in an interpreter for

A predictive model for the whole task is assembled by chaining together the predictive models of the leaf-node activities

# Task Descriptions

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

## Task Acceptance Criteria

Not all robots can perform all tasks...

Each different robot platform is integrated with its own RMF Adapter

- The adapter knows the **description** schema of each **category** that the platform can support
- The adapter knows robot-specific parameters, e.g. battery, speed, navigation graph, payload capacity, and other capabilities like cleaning, scanning, greeting

If none of an adapter's robots can perform a task because of incompatibility, the task is rejected.

# Allocation of tasks

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

Given M tasks of varying start times and descriptions, and $N_i$ robots in F fleets,

- Distribute M tasks across F fleets such that
  - Robots are only assigned tasks they are capable of performing
  - Robots have sufficient resources (e.g. battery) to perform all assigned tasks
  - Overall optimality of assignments
  - Assumptions
    - Each task is executed by only one robot (no collaboration)
    - A robot will perform a task only after fully completing the previous task
    - Each robot is assigned a charger

# Allocation of tasks

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

Given M tasks of varying start times and descriptions, and $N_i$ robots in F fleets,

- Distribute M tasks across F fleets such that
  - Robots are only assigned tasks they are capable of performing
  - Robots have sufficient resources (e.g. battery) to perform all assigned tasks
  - Overall optimality of assignments
  - Assumptions
    - Each task is executed by only one robot (no collaboration)
    - A robot will perform a task only after fully completing the previous task
    - Each robot is assigned a charger

**rmf_task::TaskPlanner**

A* based search algorithm to determine the right sequence in which tasks should be executed within the fleet to minimize overall time.

# Allocation of tasks

https://github.com/open-rmf/rmf_battery
https://github.com/open-rmf/rmf_task

RMF Task Allocation Planner- Other features

- Priority Assignment
  - Add a **Priority** field to task request
  - During node expansion, check if new node assignments are valid
    - Valid = high priority tasks are assigned prior to low priority ones
  - **If invalid, f(n) = g(n) + h(n) * penalty**
- Finishing Task
  - Automatically include a task that the robot has to perform at the end of its assignments
  - Park, ChargeBattery, etc

- Fleet adapters automatically replan task assignments when a task is cancelled

- Battery charging tasks are automatically inserted when needed

# Execution

https://github.com/open-rmf/rmf_ros2/tree/main/rmf_fleet_adapter/schemas

## Activity Hierarchy

Execution is broken down into a hierarchy of "activities"

- **delivery:** medicine from pharmacy to ward31
  - **pickup:** medicine from pharmacy
    - **go_to_place:** pharmacy
      - **move_to:** atrium door-entry wait point
      - **pass_through_door:** atrium door
        - **open_door:** atrium door
        - **move_to:** atrium door-exit wait point
        - **close_door:** atrium door
      - **move_to:** pharmacy door-entry wait point
      - ...

Each activity is publishing requests (e.g. navigation requests, open/close door requests) and subscribing to state updates to manage the progress of the task

⚠️ Currently activities at the same hierarchy level are treated as sequential, but future versions of RMF will support parallel activities, conditional execution, branching, and activity loops

# Execution

https://github.com/open-rmf/rmf_ros2/tree/main/rmf_fleet_adapter/schemas

## Phase Management

Each hierarchy is contained within a "Phase"

**delivery task:** medicine from pharmacy to ward31

**pickup:** medicine from pharmacy

- **go_to_place:** pharmacy
  - **move_to:** atrium door-entry wait point
  - **pass_through_door:** atrium door
    - **open_door:** atrium door
    - **move_to:** atrium door-exit wait point
    - **close_door:** atrium door
  - **move_to:** pharmacy door-entry wait point
  - ...

**on success**

**dropoff:** medicine to ward31

- **go_to_place:** ward31
  - **move_to:** pharmacy door-exit wait point
  - **pass_through_door:** pharmacy door
    - **open_door:** pharmacy door
    - **move_to:** pharmacy door-entry wait point
    - **close_door:** pharmacy door
  - **move_to:** service lift Lobby A2
  - ...

**on cancel**

**dropoff:** medicine to pharmacy

Phases are always sequential

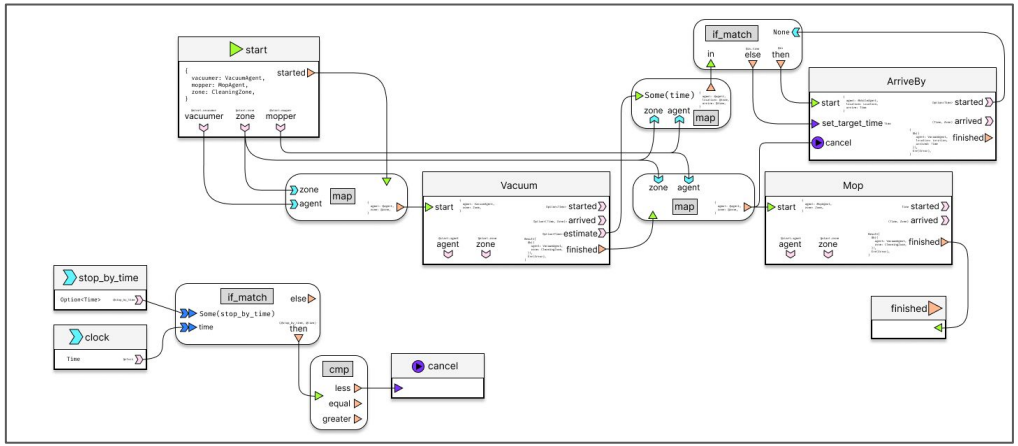
** this limitation will be eased in future versions*

A web-based API can be used to skip, restart, or retry phases

Each phase can be assigned a cancellation sequence that it will follow if the task is cancelled while the phase is active

# Future work

https://github.com/open-rmf/rmf/discussions/169




**Created by the oldman creative from the Noun Project**

- Generalized task compositions
- GUI for designing workflows

- Backend for scheduling recurring tasks
- Endpoints for modifying the schedule

# Questions