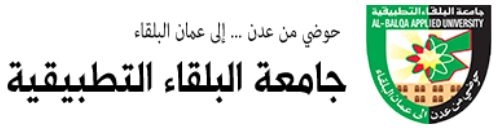


ROS Workshop

Introduction & Overview

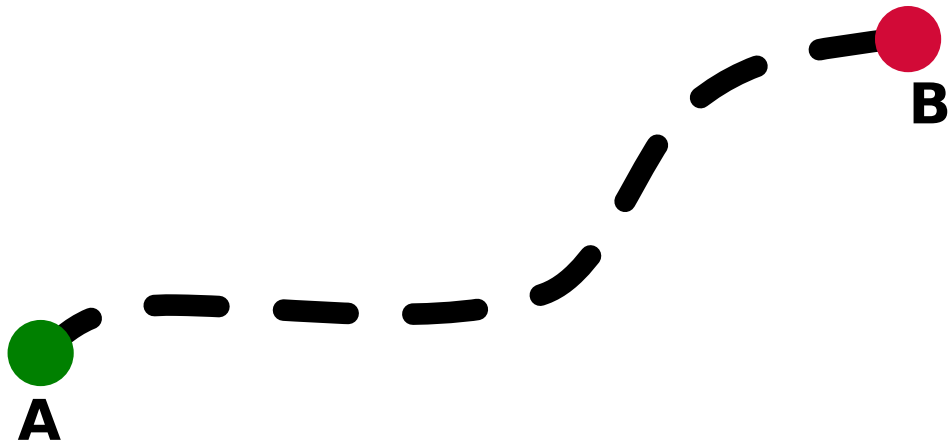
January 11, 2023

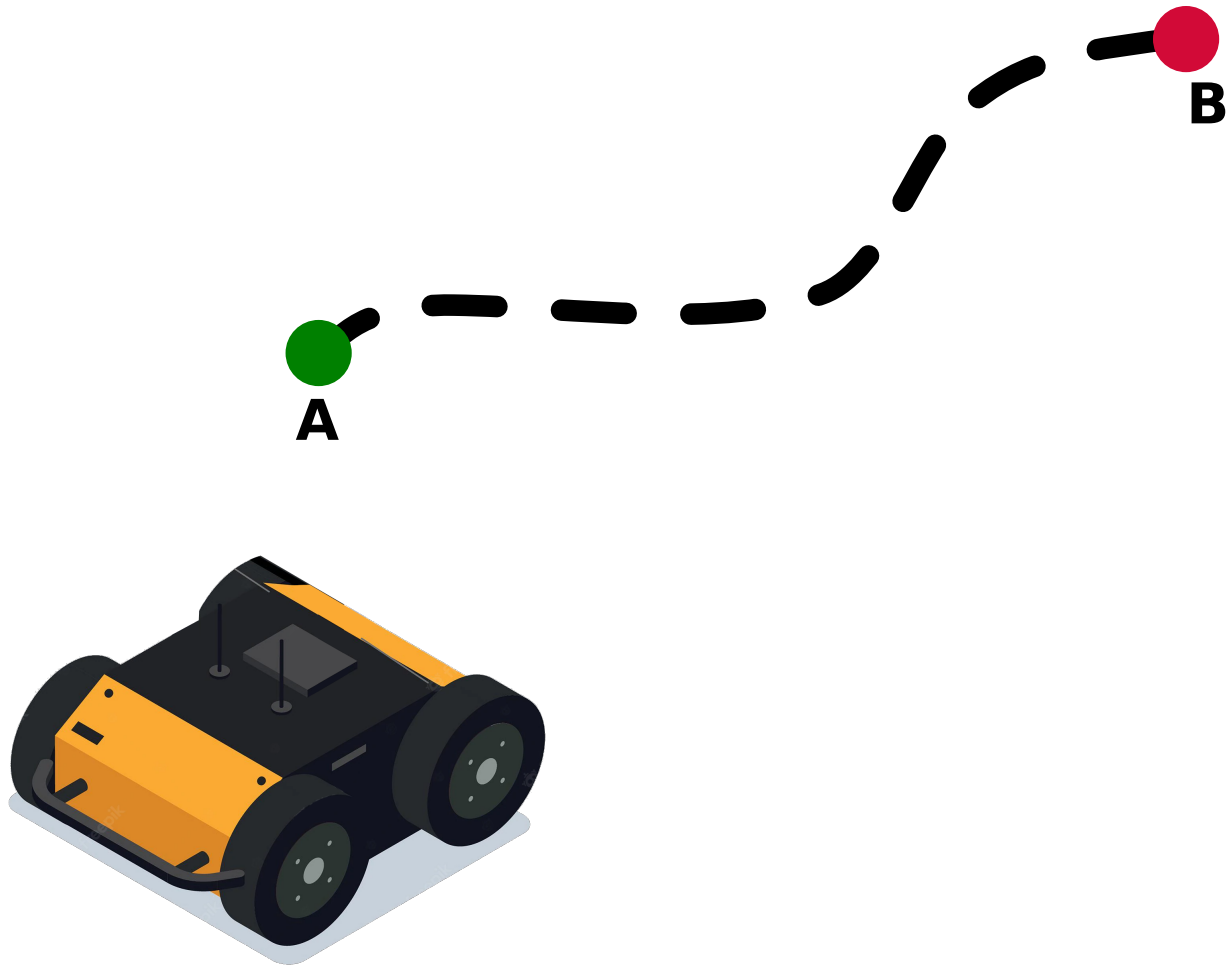
Hassan Umari

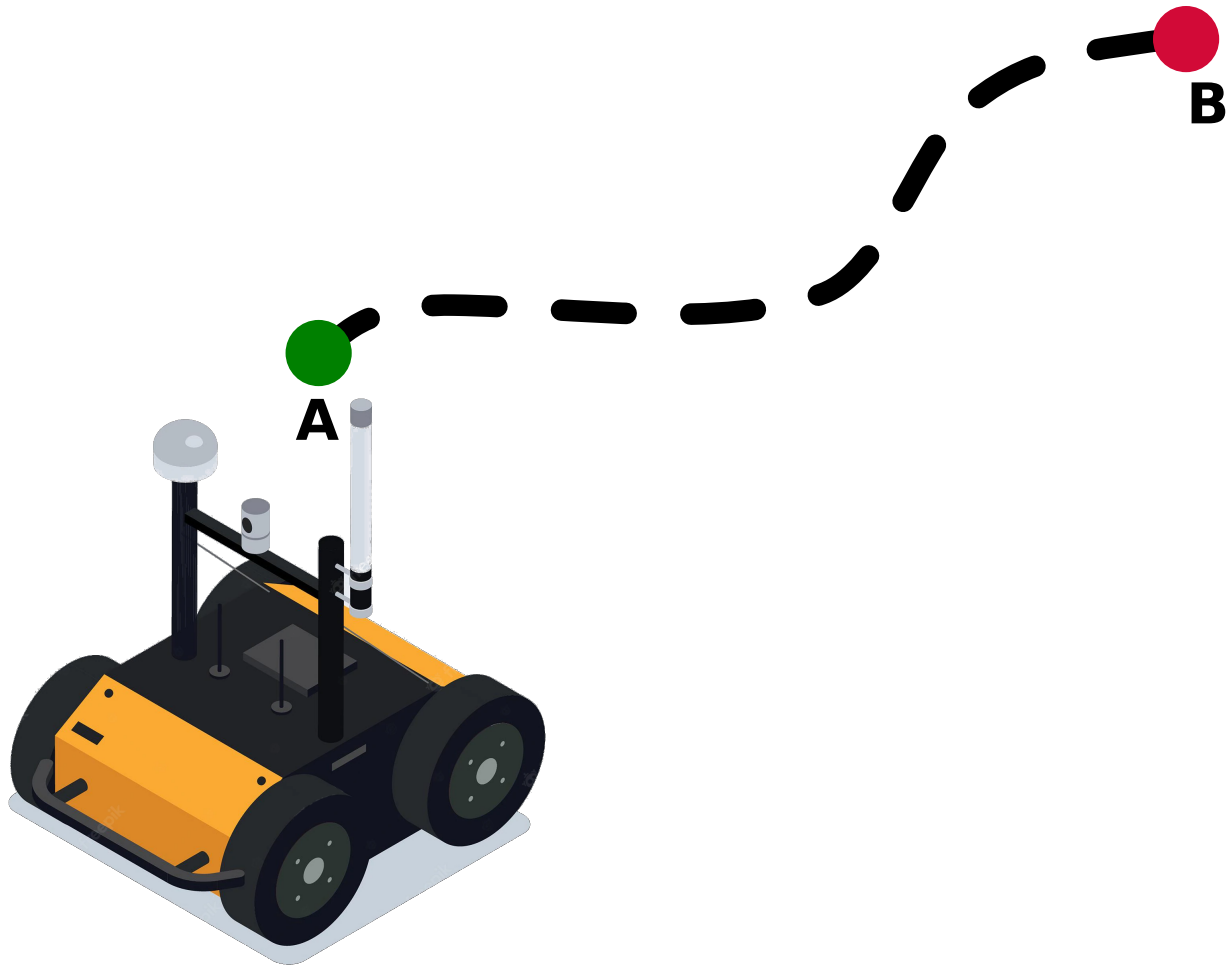
What ROS is

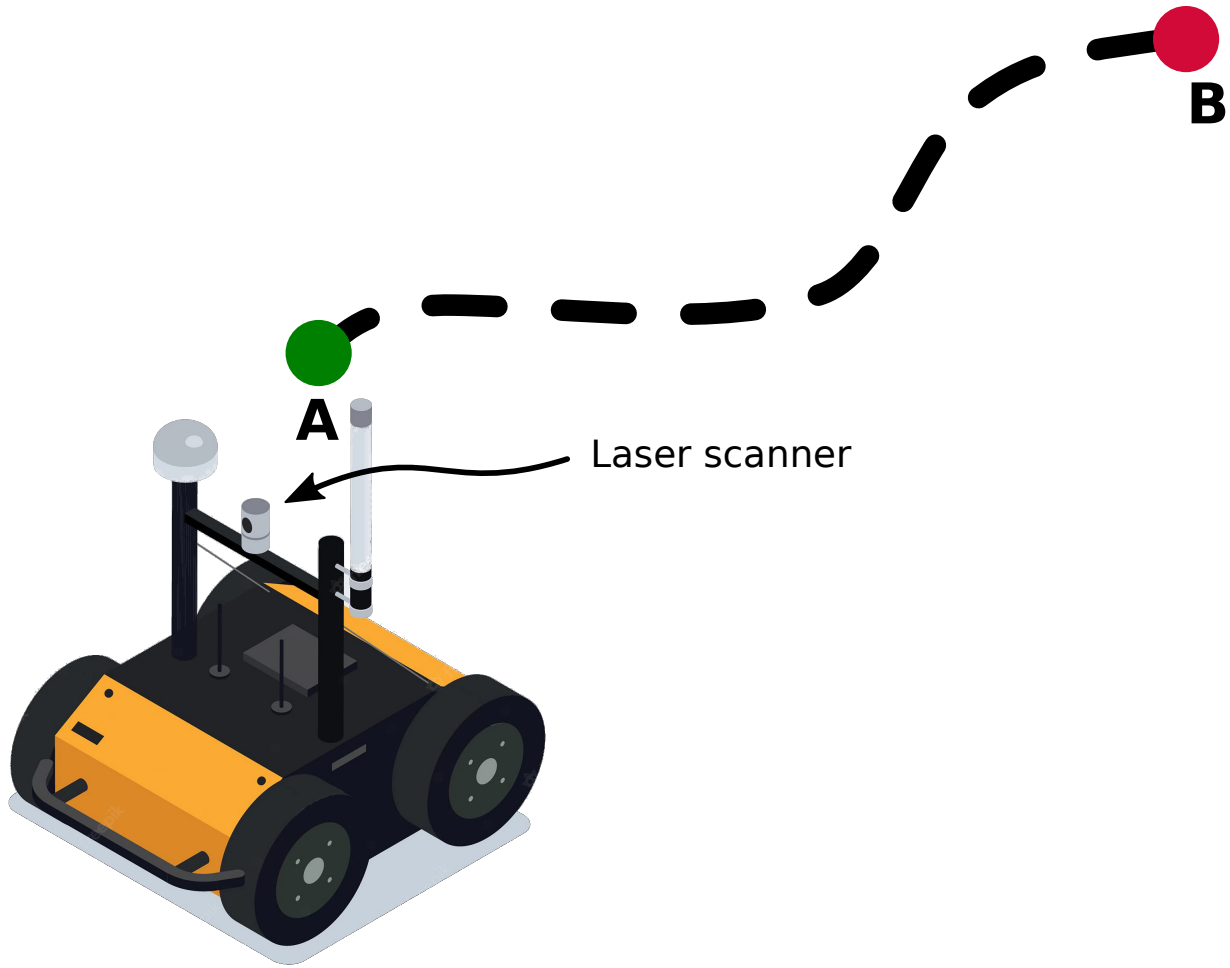
Robot Operating System

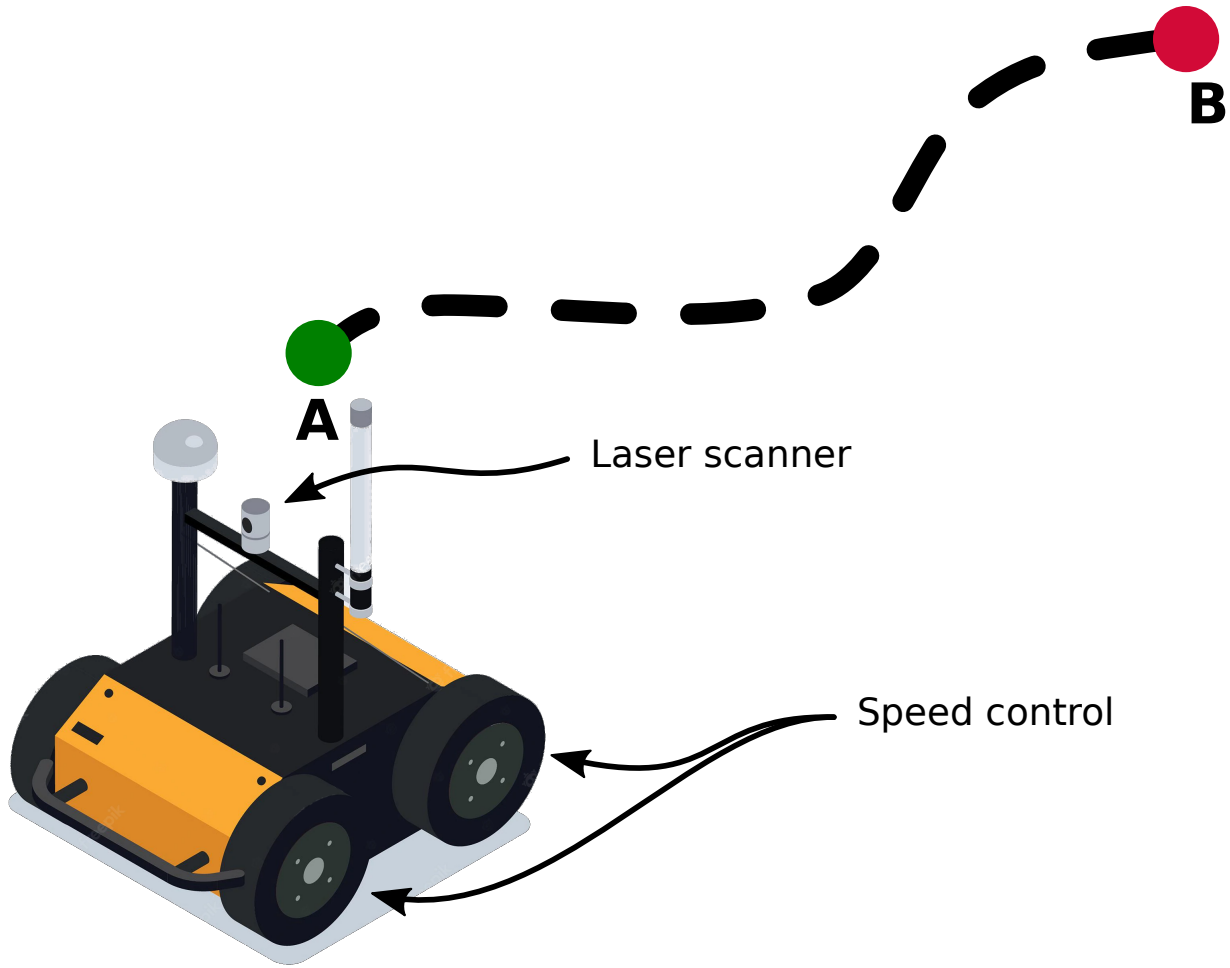
- Short for: Robot Operating System.
- A collection of libraries and tools.
- It helps software developers create robot applications.

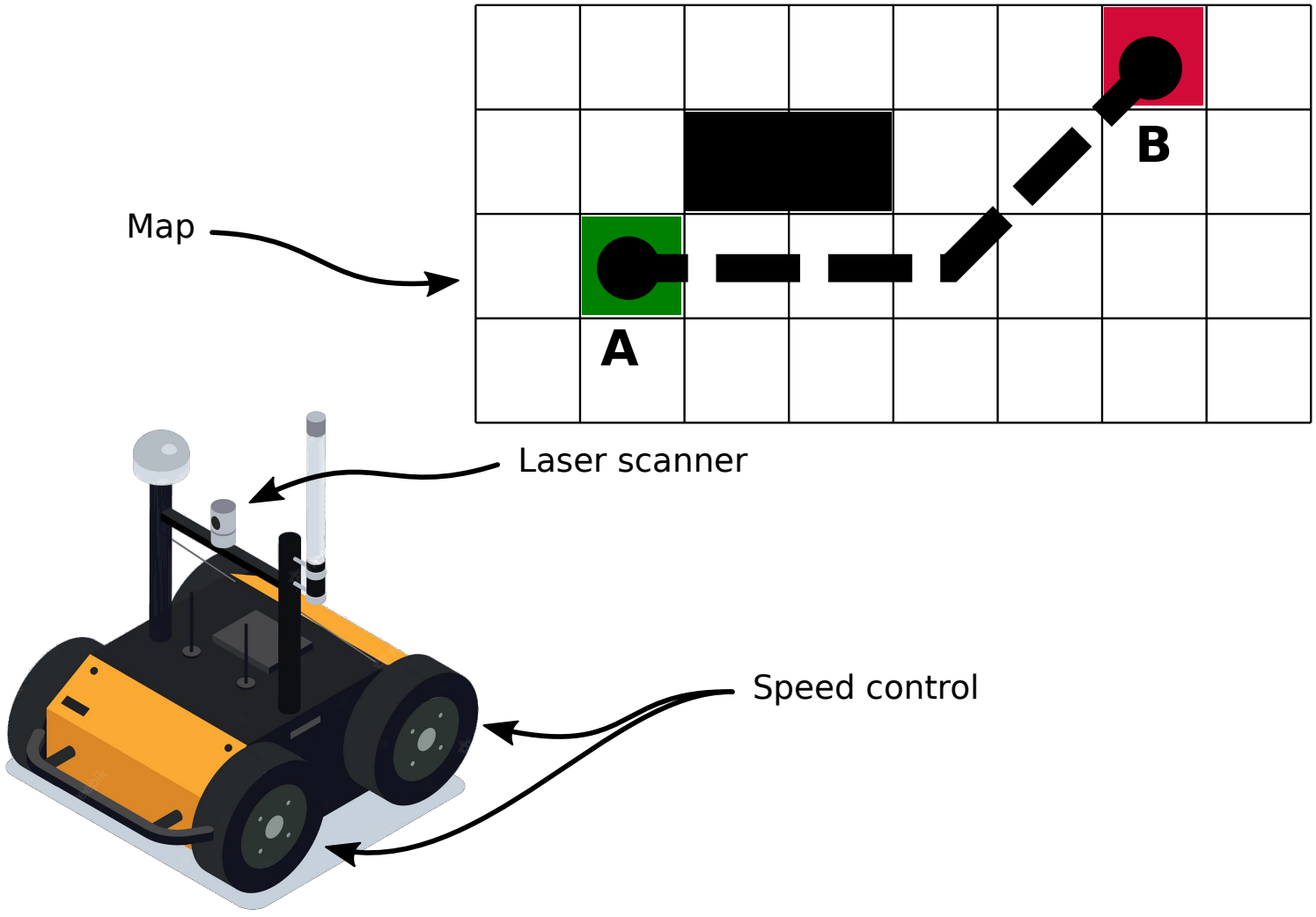

B
A

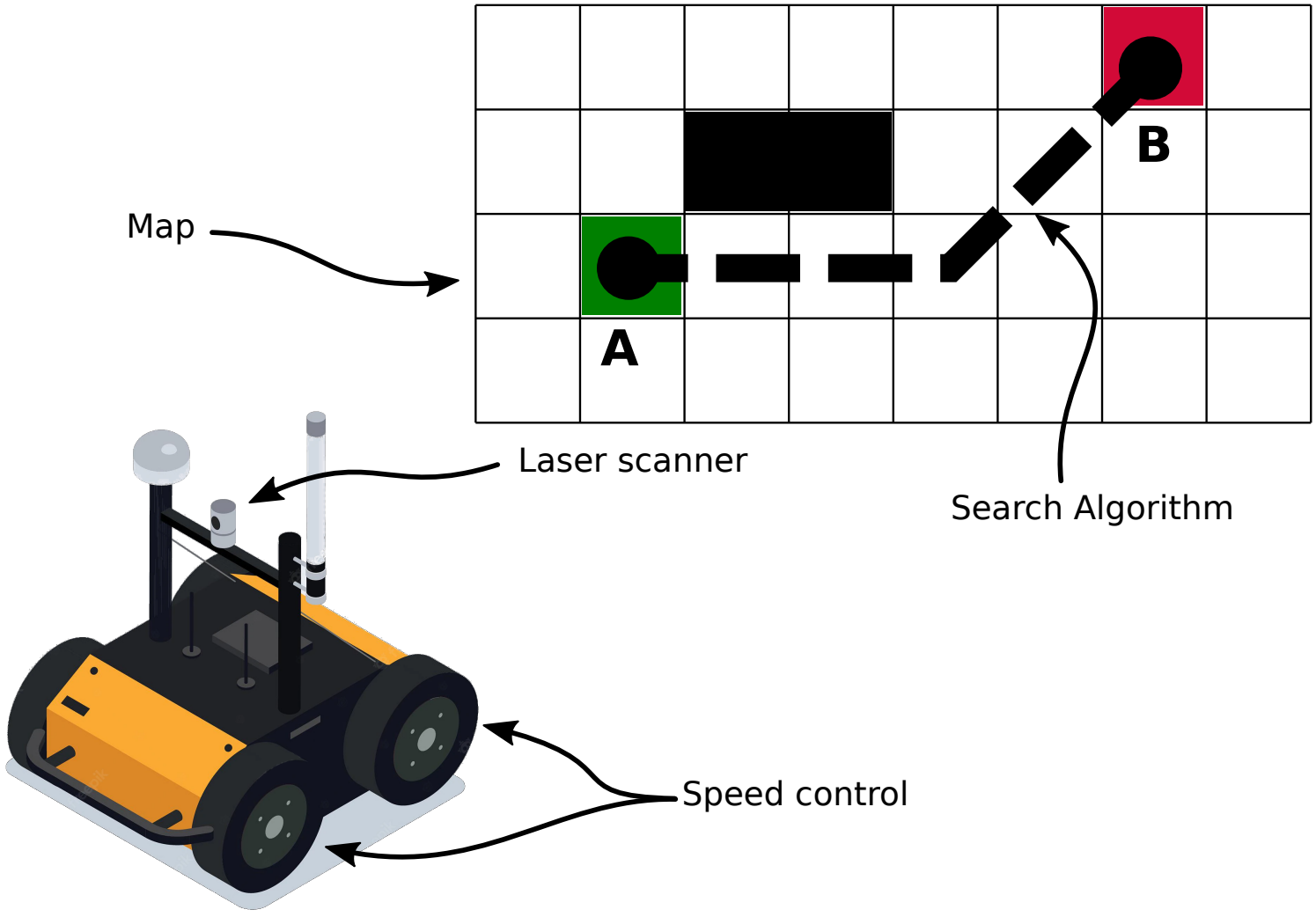

A
B

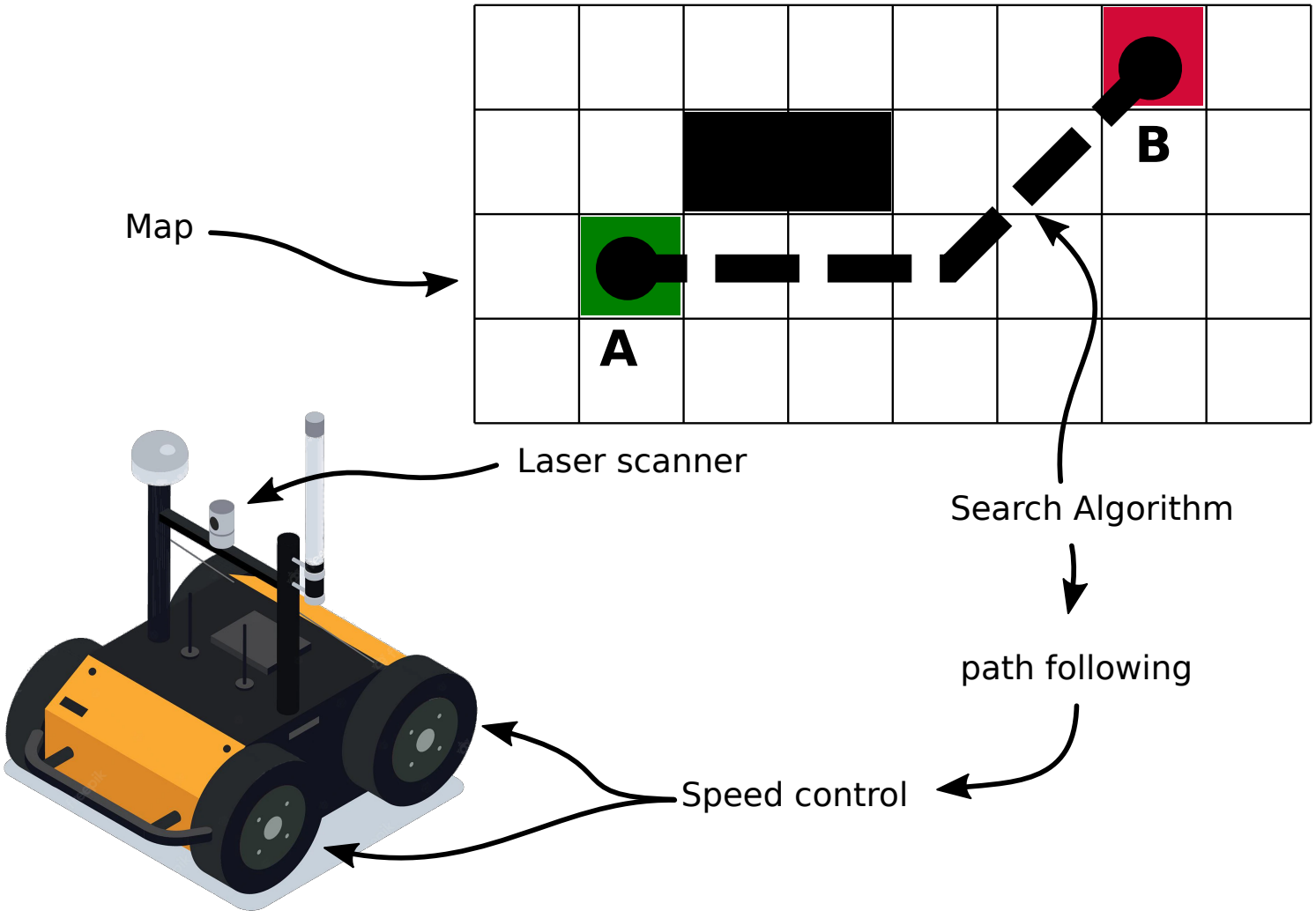

A
B


B
A
Laser scanner


B
A
Laser scanner
Speed control


Map
A
B
Laser scanner
Speed control


Map
A
B
Laser scanner
Search Algorithm
Speed control


Map
B
A
Laser scanner
Search Algorithm
path following
Speed control

robot.py

```
def get_path(start, end):
    return make_plan(start, end)



def follow_path(robot, path):
    for point in path:
        robot.move_to(point)




class Robot:
    def send_speed(self, v, w):
        serial.write(v)
        serial.write(w)
```

robot.py

```
def get_path(start, end):
    return make_plan(start, end)


def follow_path(robot, path):
    for point in path:
        robot.move_to(point)



class Robot:
    def send_speed(self, v, w):
        serial.write(v)
        serial.write(w)
```

Algorithm

robot.py

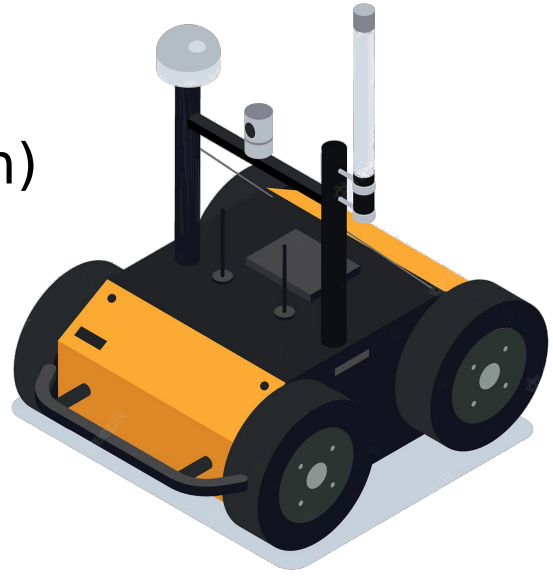
```
def get_path(start, end):
    return make_plan(start, end)


def follow_path(robot, path):
    for point in path:
        robot.move_to(point)


class Robot:
    def send_speed(self, v, w):
        serial.write(v)
        serial.write(w)
```

Algorithm

Interface with Hardware

Mapping

Path Planning
(search algorithm)



Localization

Path following

Robot interface

What ROS is

Robot Operating System

A robot application:

- consists of **multiple** software components.
- part of the application will be responsible for interfacing with the **hardware**
- without a framework like ROS, it's hard to **share** your code with others
- you might have to implement many components from scratch (**re-inventing** the wheel).

What ROS is

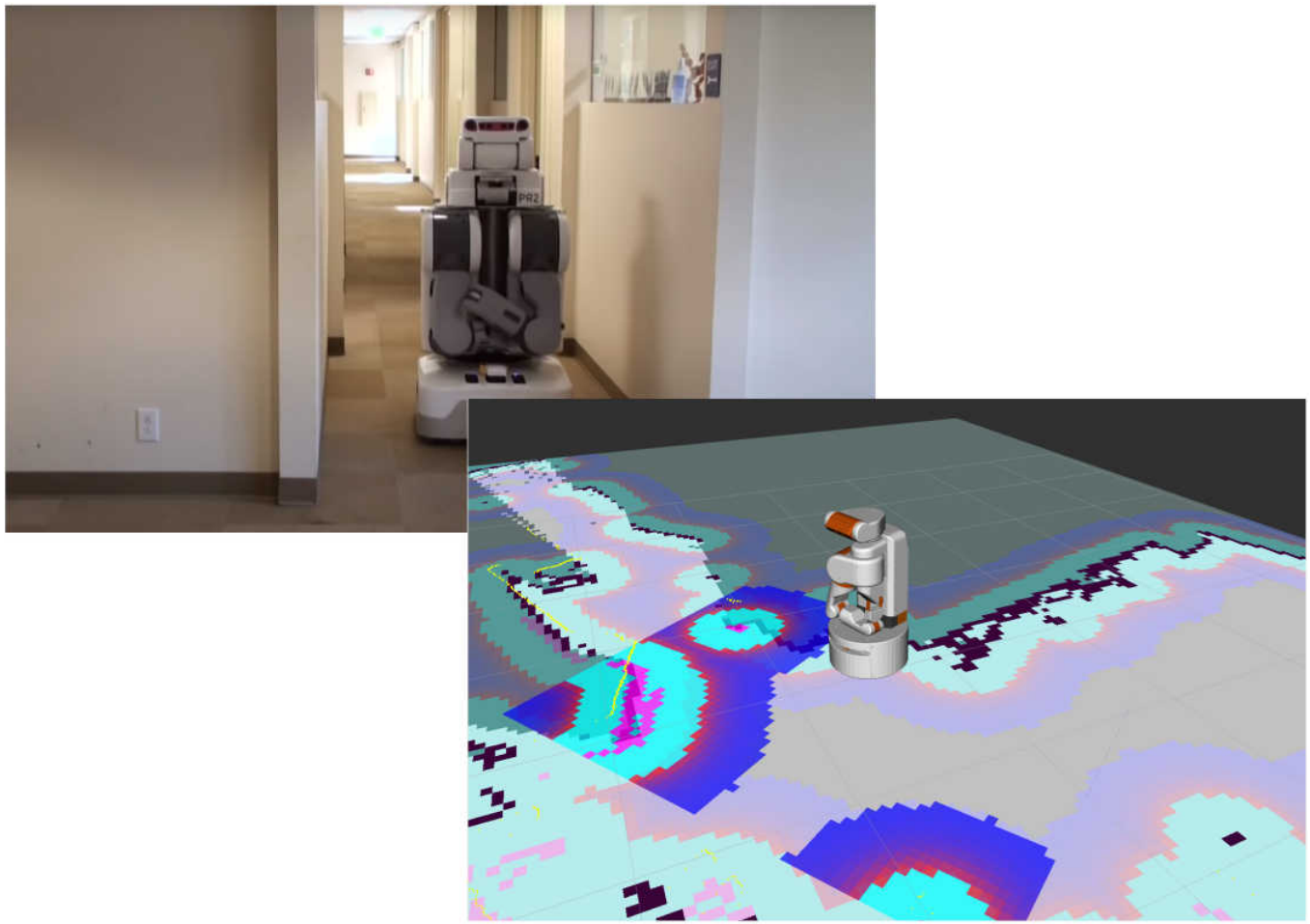
Robot Operating System

- A way to standardize writing software for robots.
- It enhances code reusability  .
- ROS is open-source  .
- ROS is installed on top of Linux. 

Example Projects

Robot Operating System

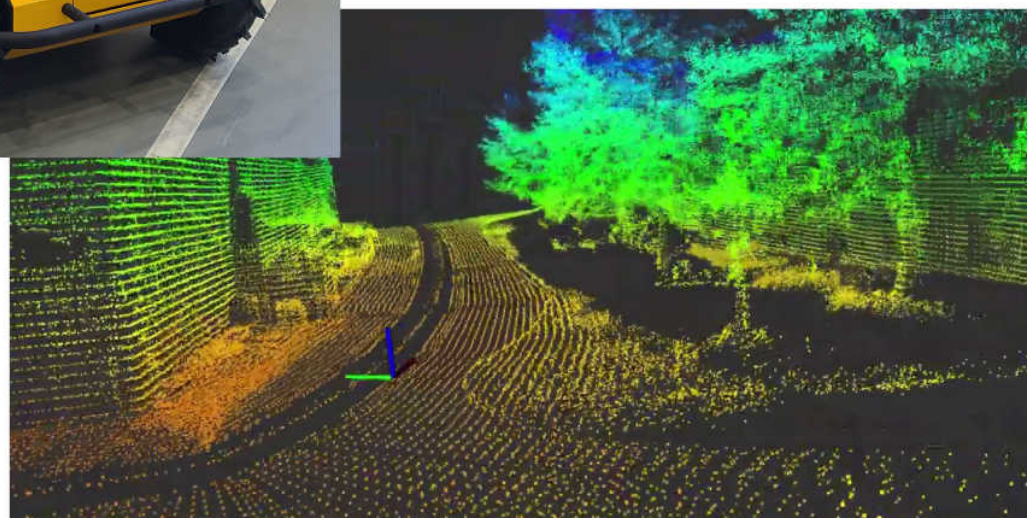
Robot Navigation



Example Projects

Robot Operating System

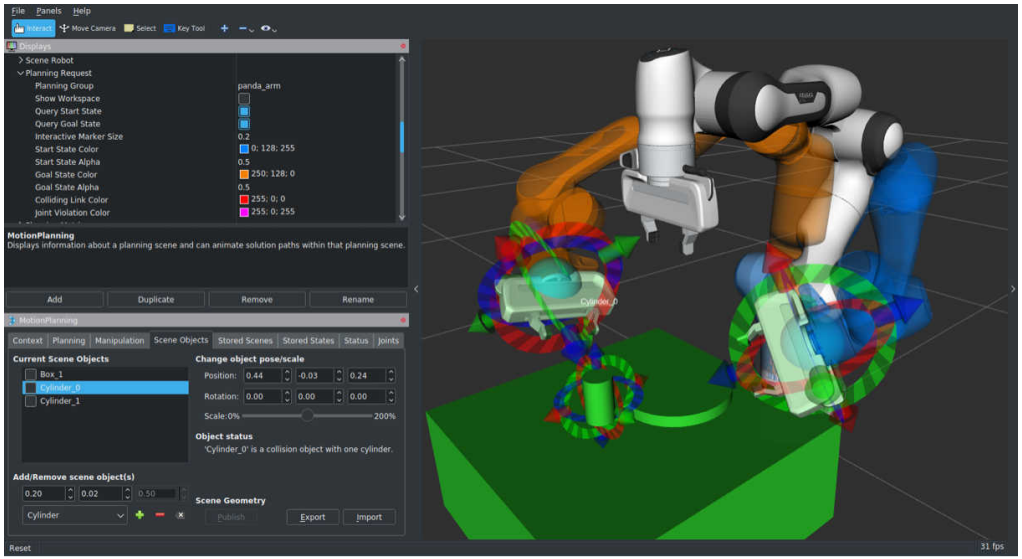
Robot Navigation



Example Projects

Robot Operating System

Robot Manipulation



What ROS is

Robot Operating System

```
ROS = communication middleware + tools + libraries
```

Quick peek

What ROS is

Robot Operating System

```
ROS = communication middleware + tools + libraries
```

What ROS is NOT

Robot Operating System

- It is NOT a programming language.
- It is NOT an integrated development environment (IDE).
- It is NOT a stand-alone operating system

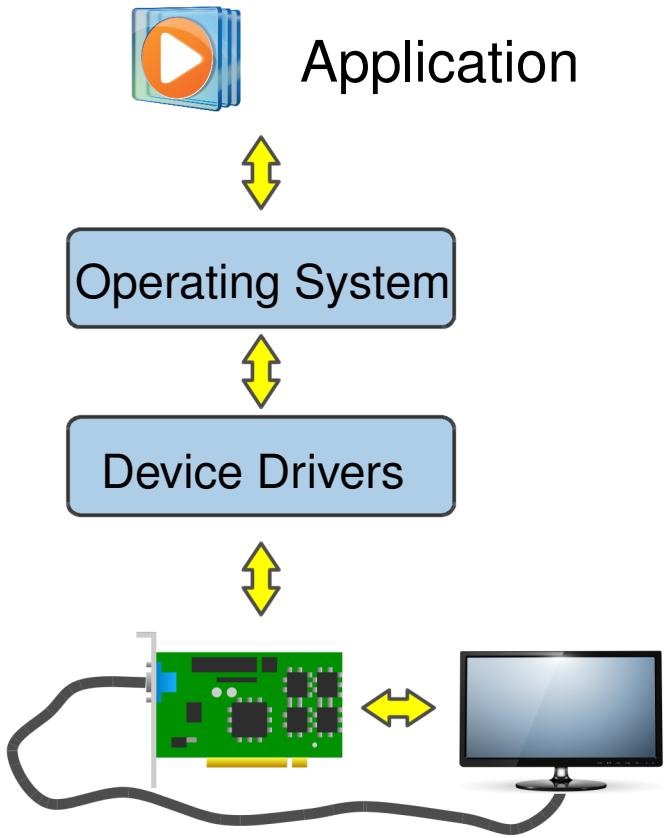
Analogy Between ROS and Operating Systems

Software Applications

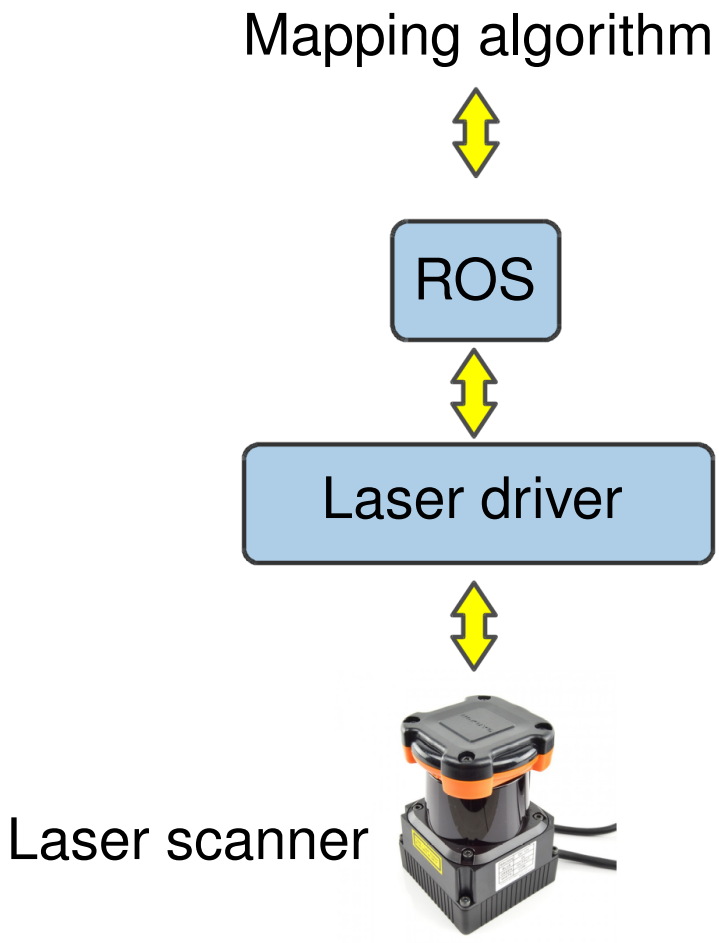
work on

Different hardware

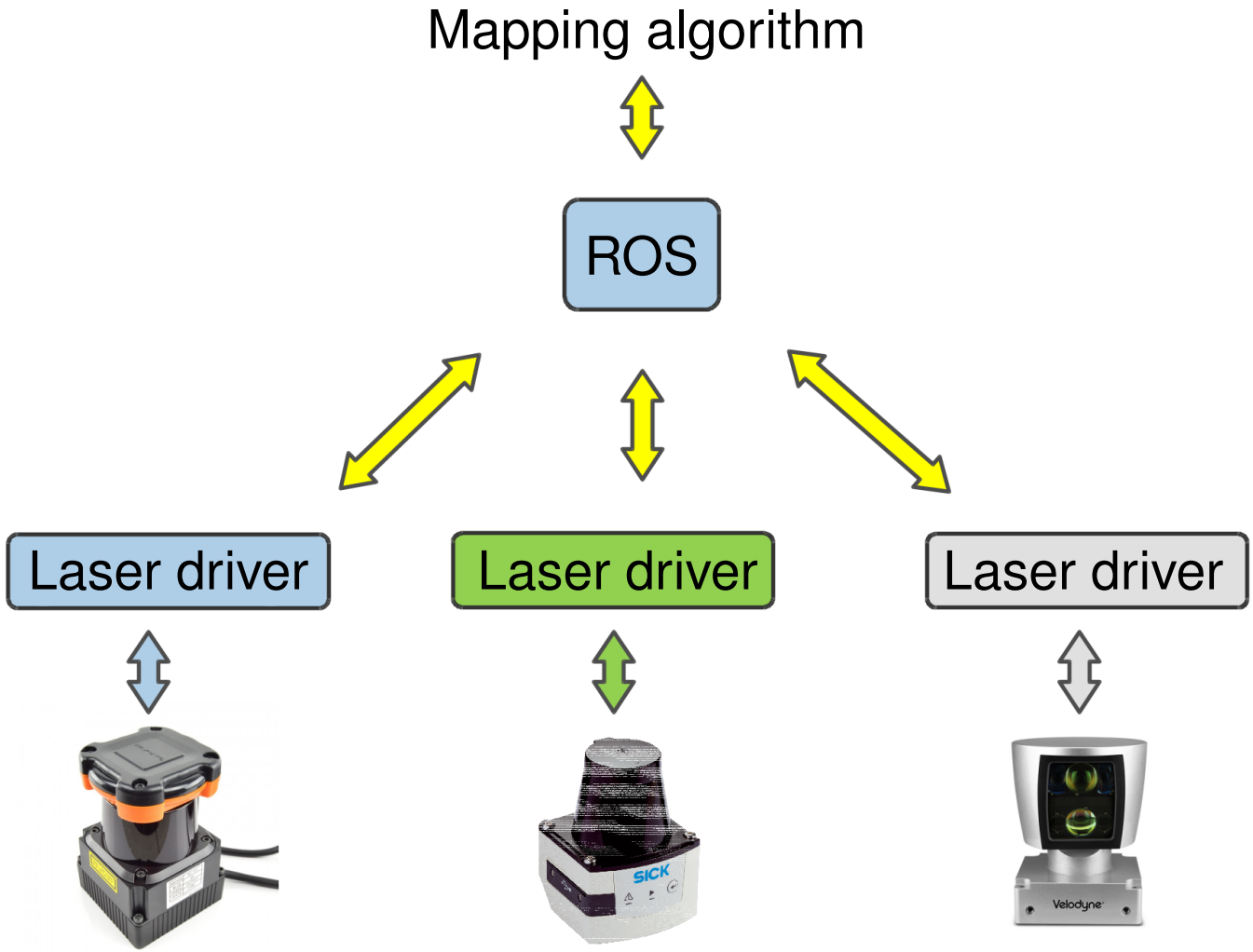
Analogy Between ROS and Operating Systems



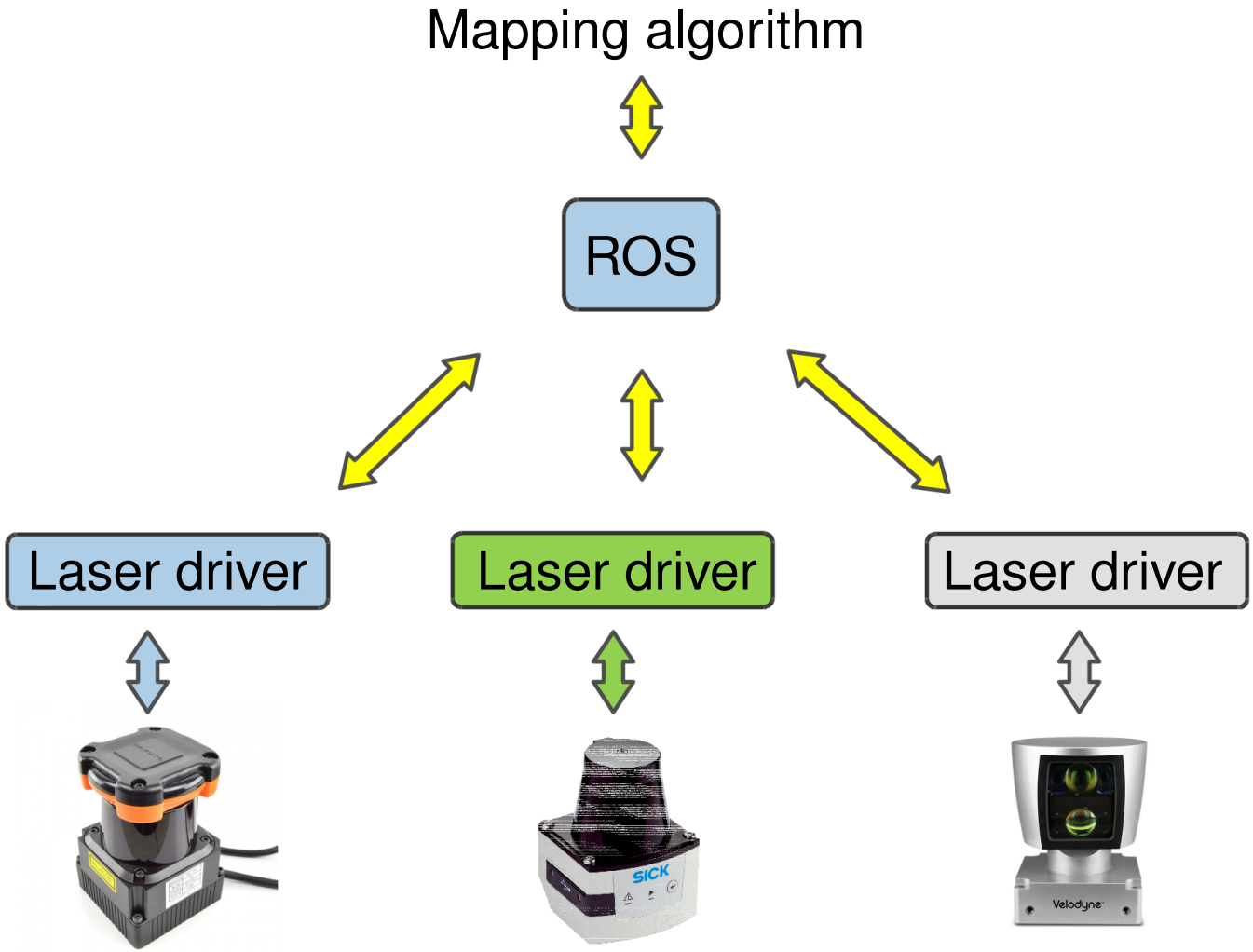
Analogy Between ROS and Operating Systems



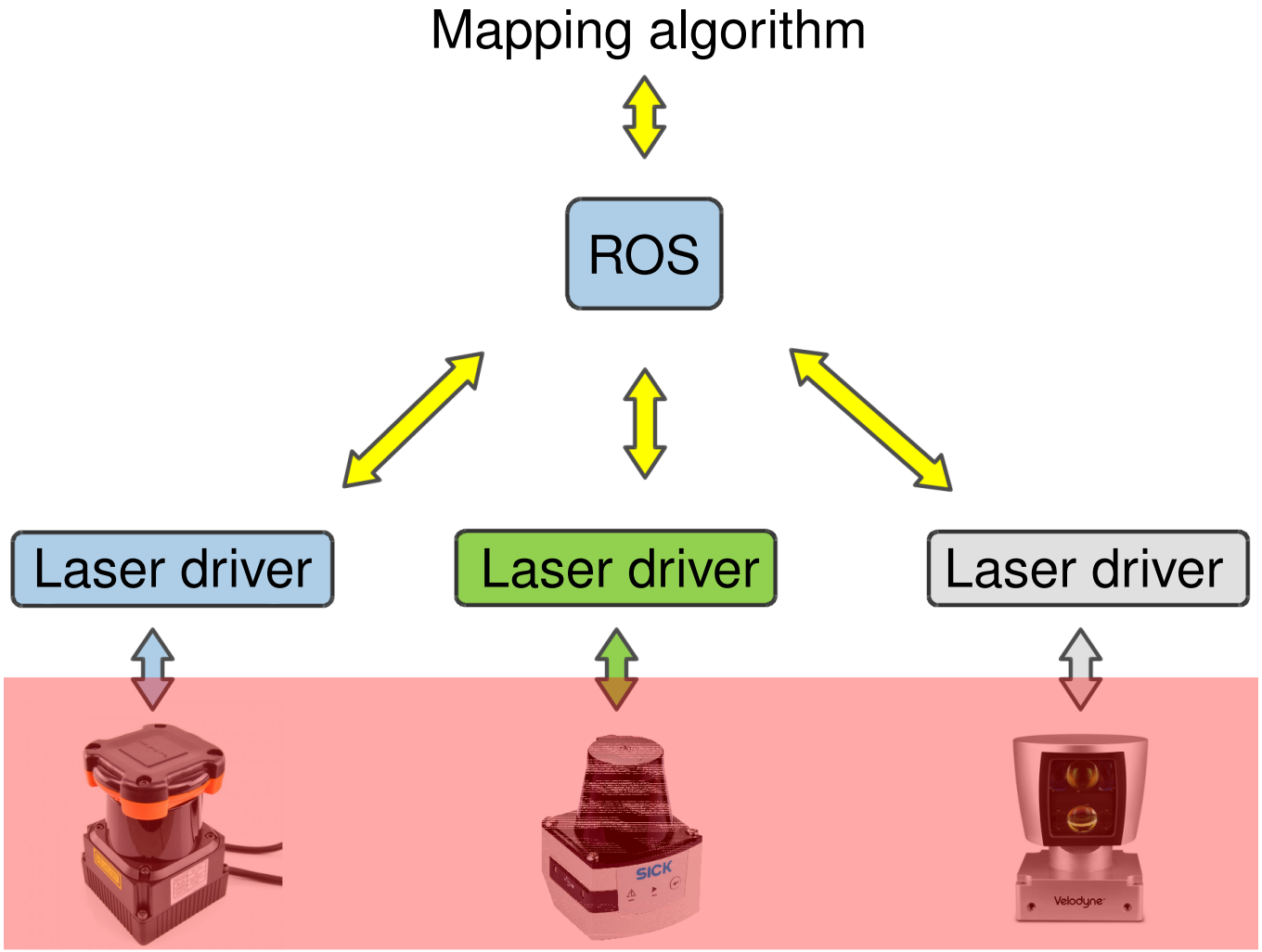
Analogy Between ROS and Operating Systems



Analogy Between ROS and Operating Systems

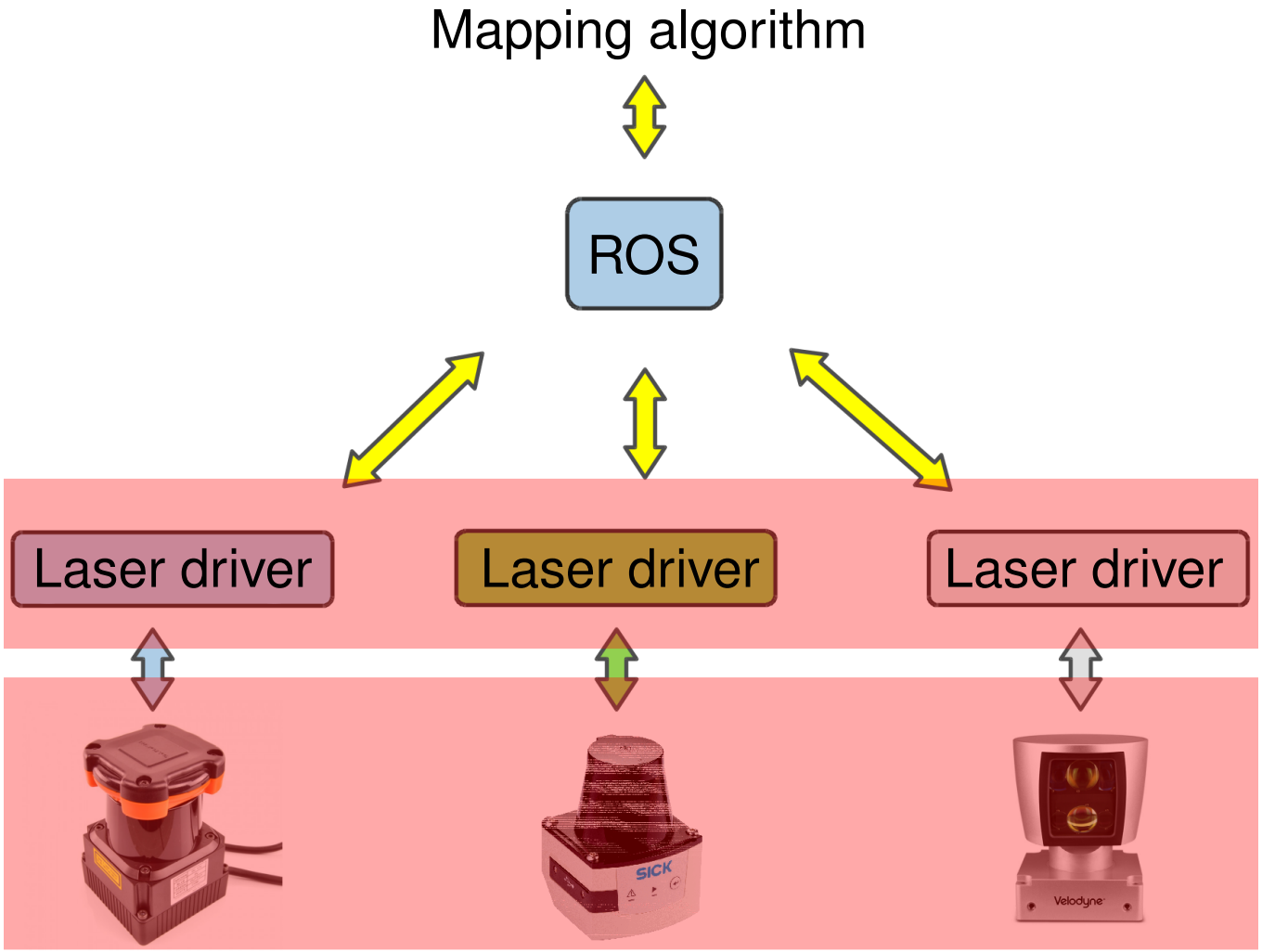


Analogy Between ROS and Operating Systems

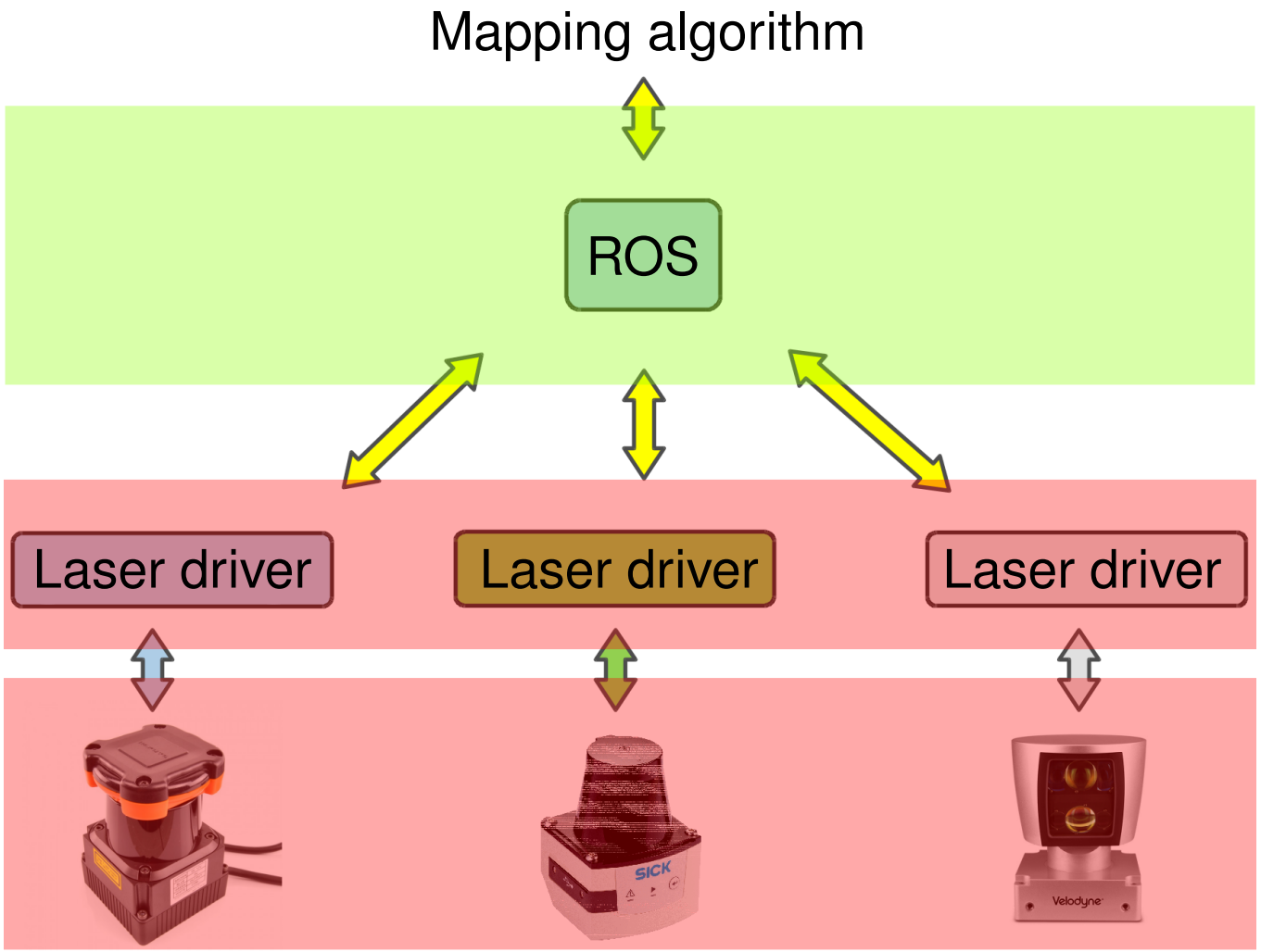


Different hardware

Analogy Between ROS and Operating Systems



Analogy Between ROS and Operating Systems

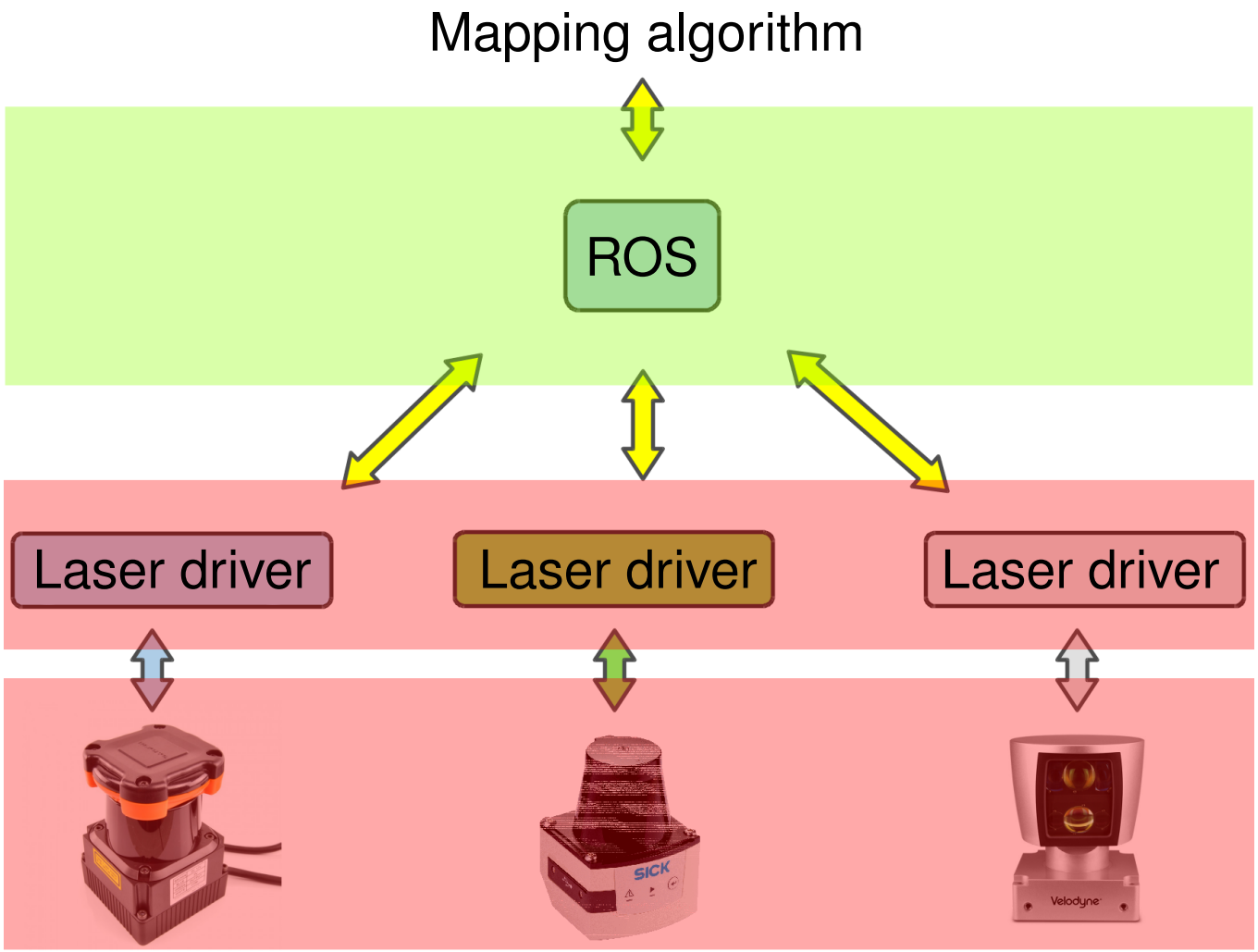


Data received in the same message format

Different drivers

Different hardware

Analogy Between ROS and Operating Systems



Analogy Between ROS and Operating Systems

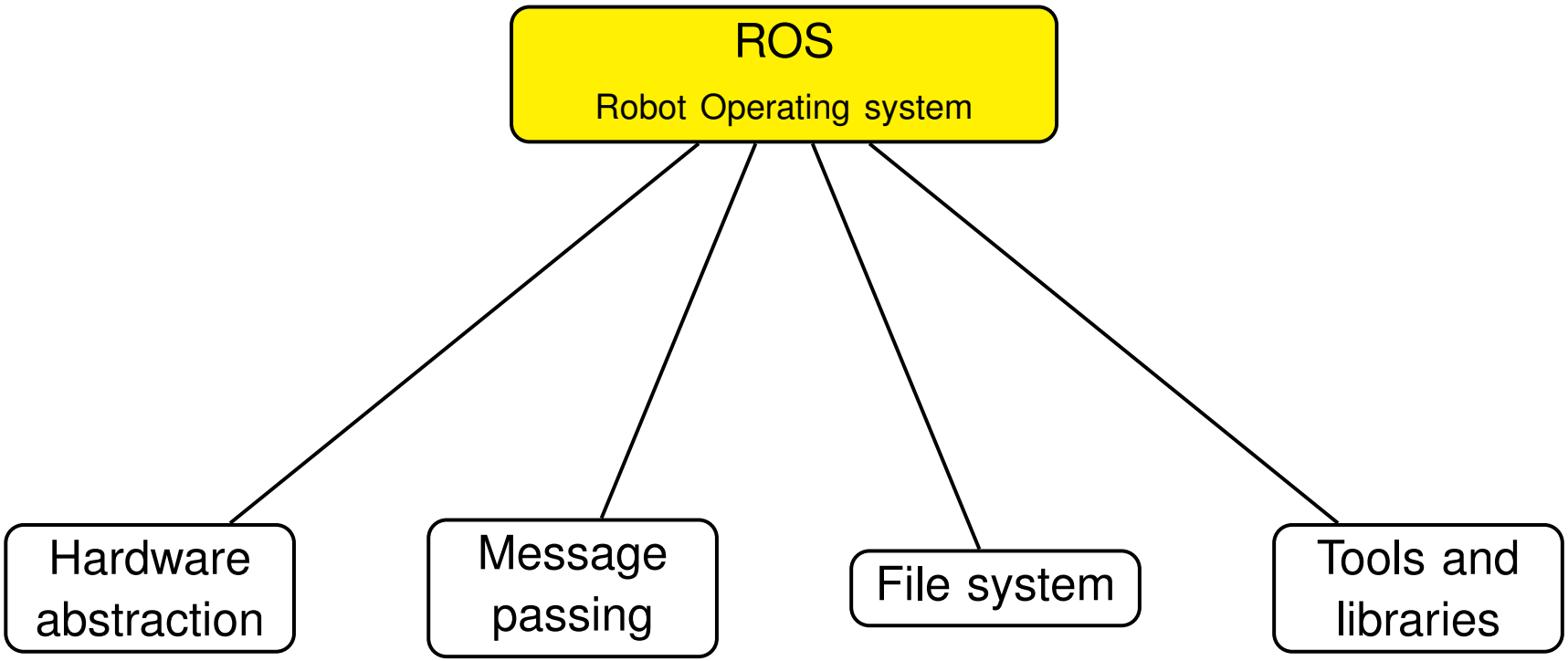
Mapping | Navigation | pick & place

Robot Applications

work on

Different hardware

Analogy Between ROS and Operating Systems



Features of ROS

- Language independent.
- Distributed and Modular.
- A lot of libraries and tools.
- Open Source.
- Active Community.

Language independent

- ROS functionalities are implemented as a library in different programming languages.
- These libraries are referred to as ROS client libraries.

Language independent

ROS client libraries.

- Main ROS Client libraries:
  - roscpp
  - rospy
  - roslisp
- Experimental ROS client libraries:
  - rosjava
  - rosruby
  - and some others..
- ROS support on MATLAB:
  - Robotics System Toolbox




Distributed and Modular

- ROS supports running processes on multiple computers connected together through a LAN.
- In a system running ROS, there will be multiple of processes where each process can do certain task. A process can be changed without altering the remaining processes.

A lot of libraries and tools

- Examples of libraries:
  - Navigation stack.
  - SLAM (gmapping, hector SLAM, etc..).
  - Localization (amcl, etc..).
  - Motion planning for manipulators (MoveIt)
  - Support for popular libraries (OpenCV, PCL).
- Examples of tools:
  - RVIZ:3D Visualization.
  - ROS bag files: Logging Sensor Data.
  - Catkin: A Build System.
  - Command line tools.

Bad Things About ROS

- It needs a computer. Does not work on a microcontroller!
- Not optimized for multiple robots.
- Supported only on Linux, no support for Windows or macOS.
- ROS imposes a communication overhead.

ROS Concepts

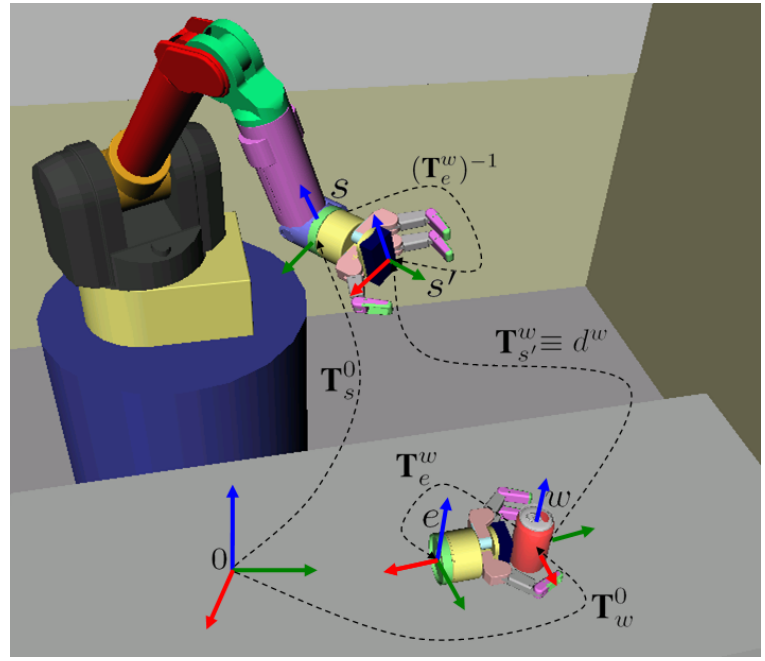
1. Nodes.
2. Topics.
3. Messages.
4. Master.
5. Services.
6. Actions
7. Parameter Server.
8. Bags.

ROS Concepts

Nodes:

- A ROS node is a process that exchanges data with other processes through ROS network.
- It may be written in Python, C++, or even MATLAB.

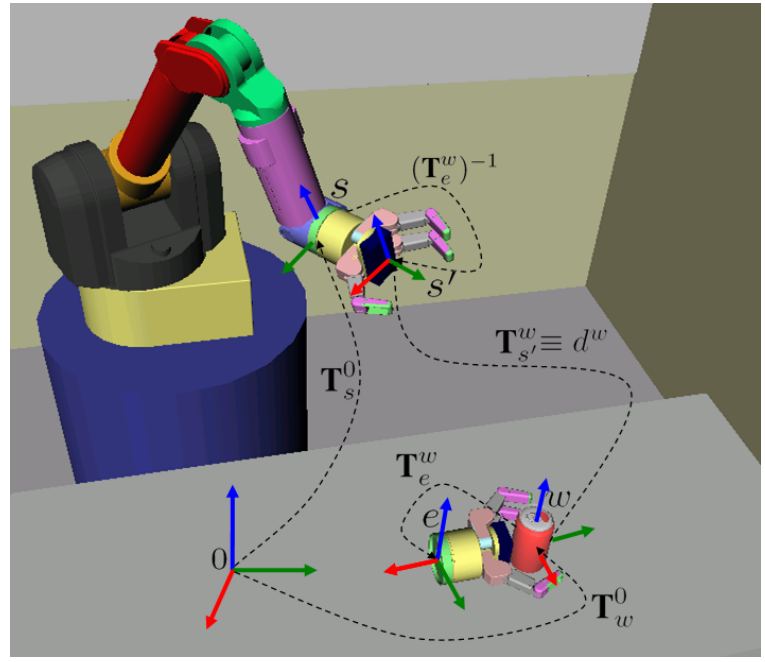

KINOVA

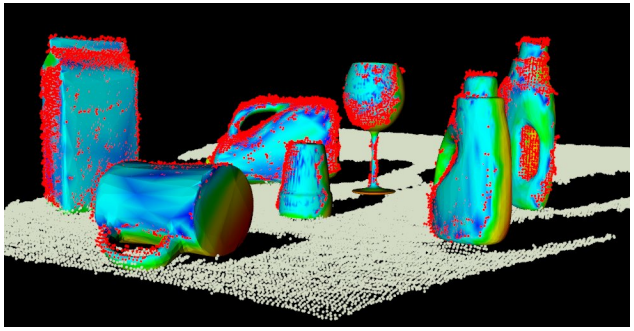


http://arm.eecs.umich.edu/images/TSR.png

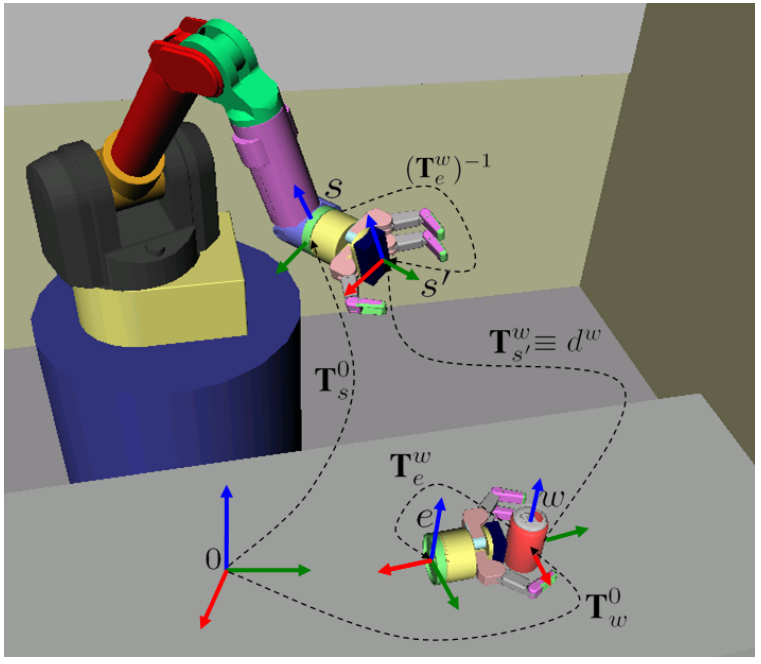

XBOX 360
KINECT


KINOVA


$(\mathbf{T}_e^w)^{-1}$
s
s'
$\mathbf{T}_{s'}^w \equiv d^w$
$\mathbf{T}_s^0$
$\mathbf{T}_e^w$
e
w
0
$\mathbf{T}_w^0$

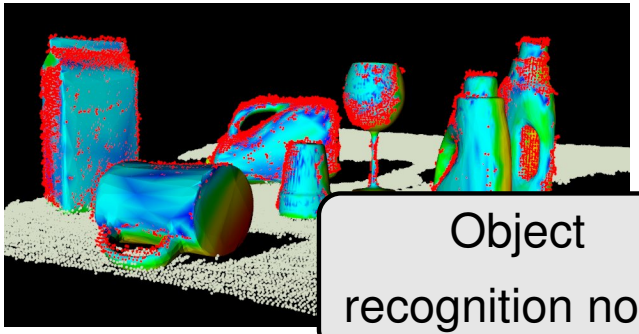


http://www.pointclouds.org/blog/_images/cvfh1.jpg



ROS master node

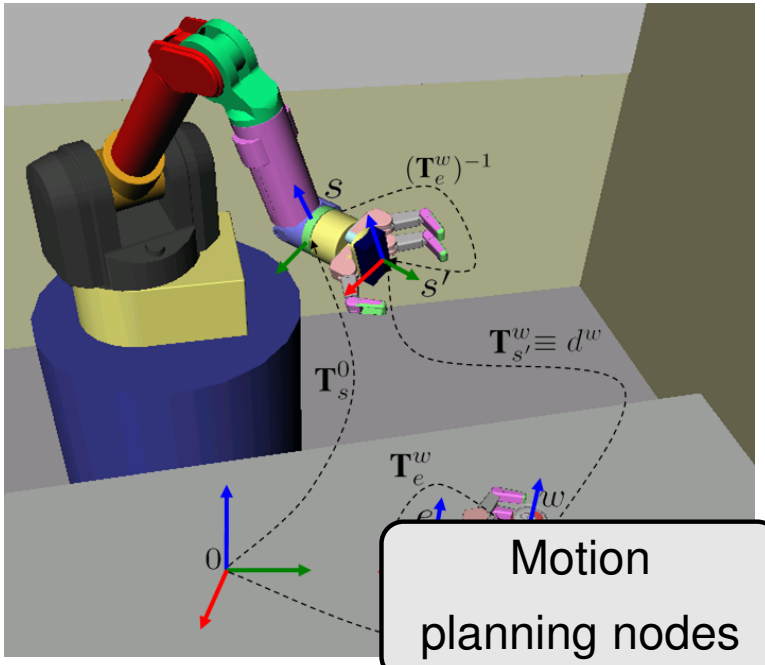
Camera driver node



Object recognition node

Manipulator driver node

Robot driver node



Motion planning nodes

ROS Concepts

Topics and Messages:

- Nodes send data by publishing messages on a named topic.
- Nodes receive data by subscribing to a topic.
- Multiple nodes can publish/subscribe to the same topic.

ROS Concepts

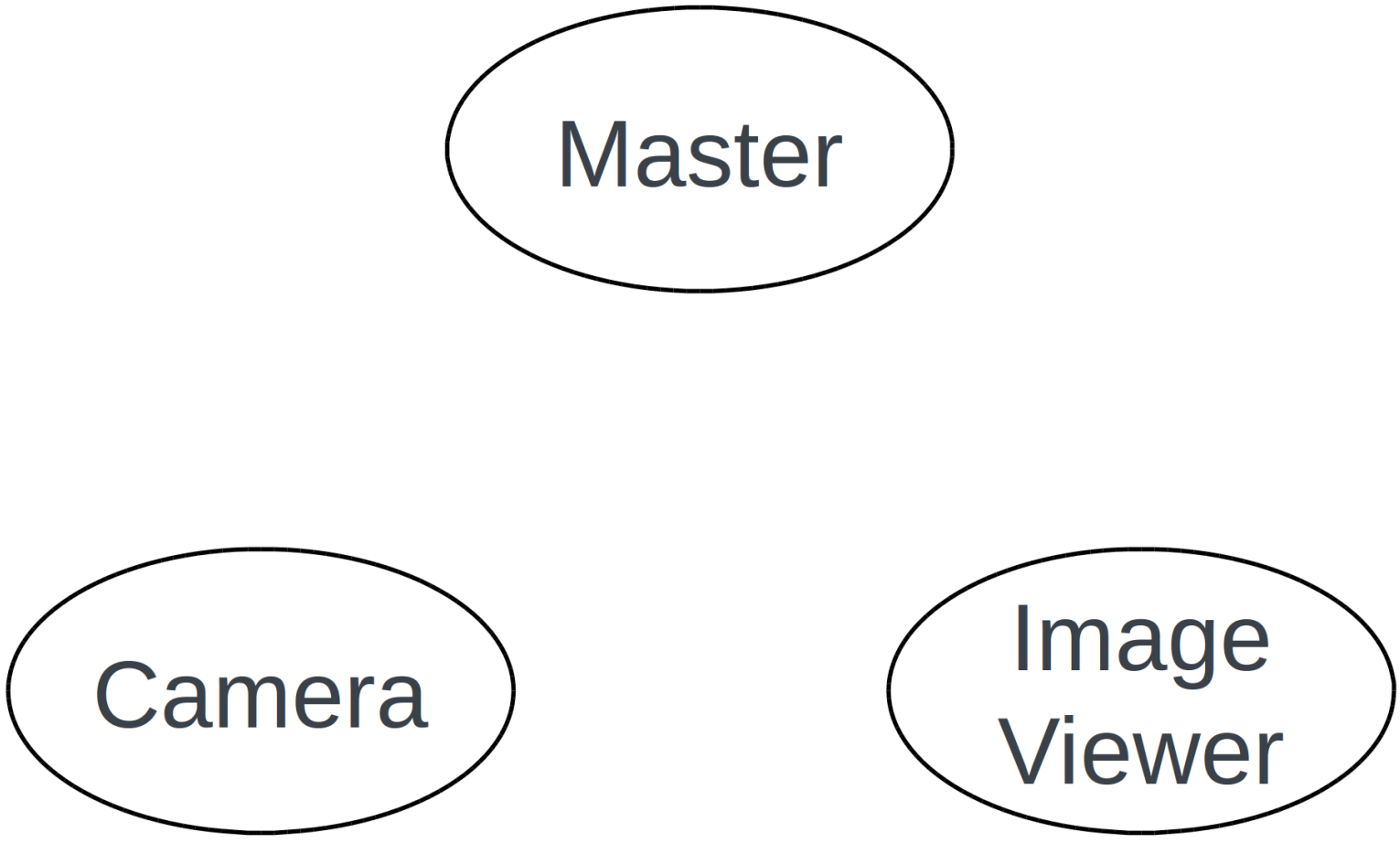
Topics and Messages:

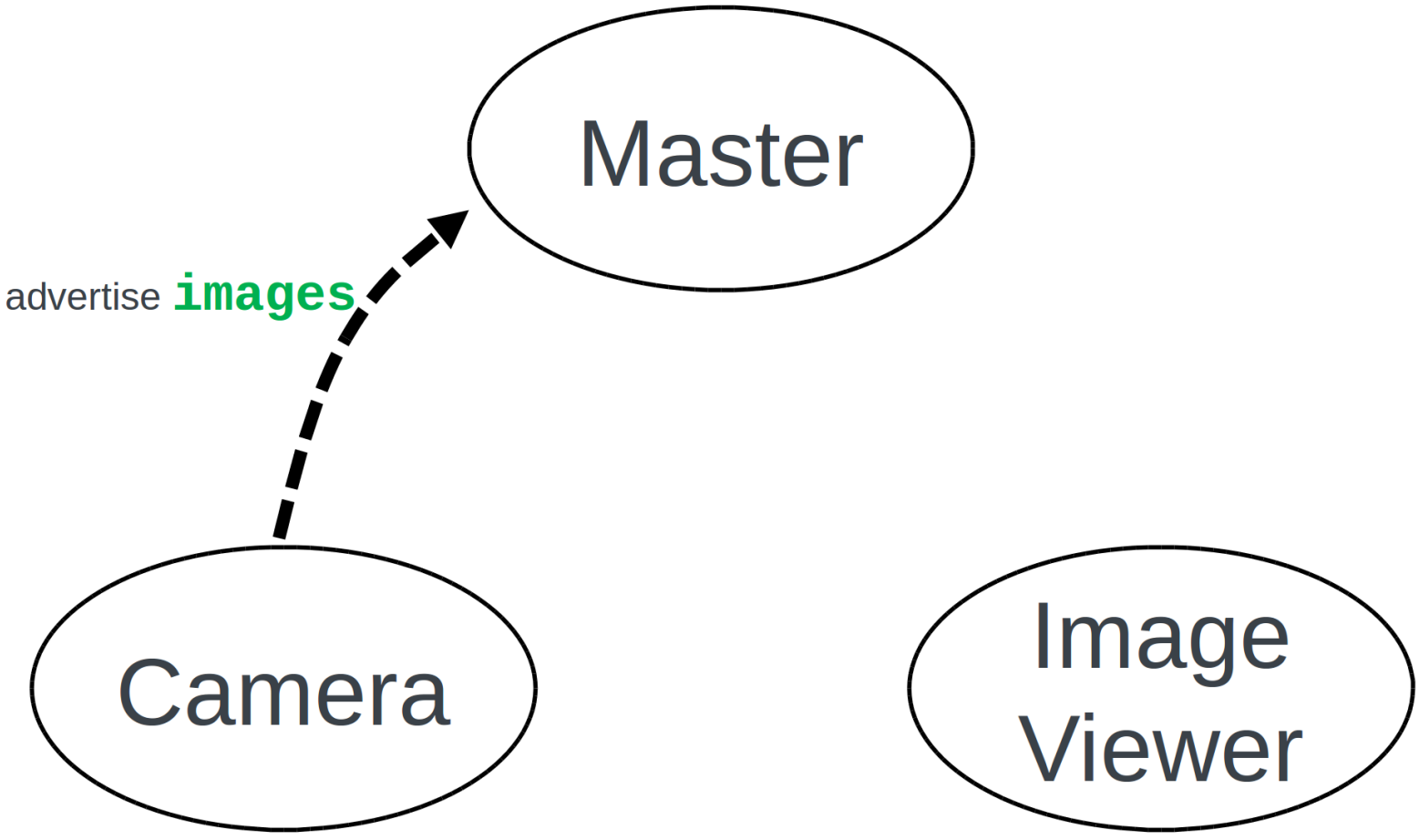
- Publisher node publishes the messages on a topic at a chosen frequency.
- This **publish/subscribe** communication paradigm is a many-to-many one-way transport mechanism of data.
- The publishing node and subscribing node are not aware of each other's existence.

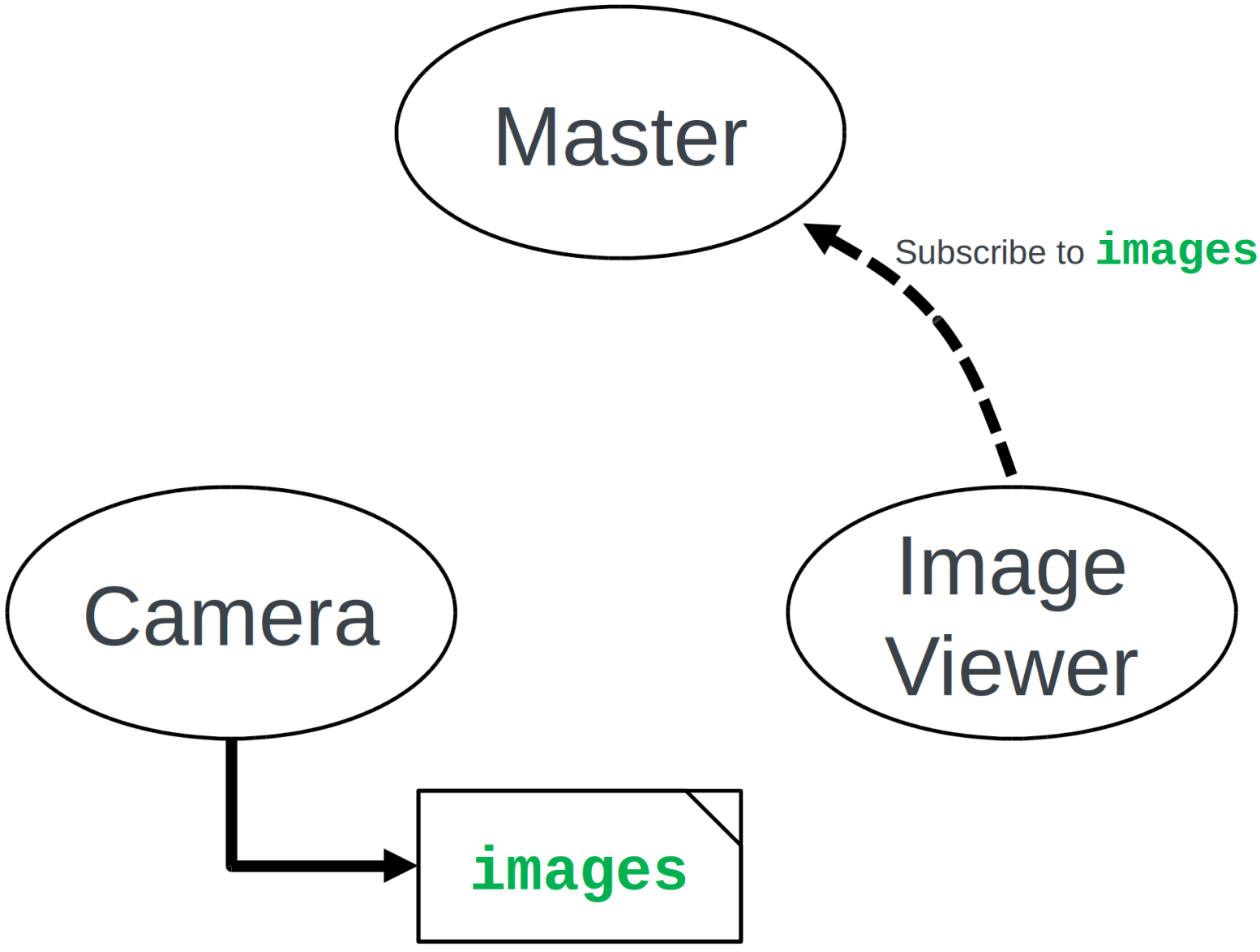
ROS Concepts

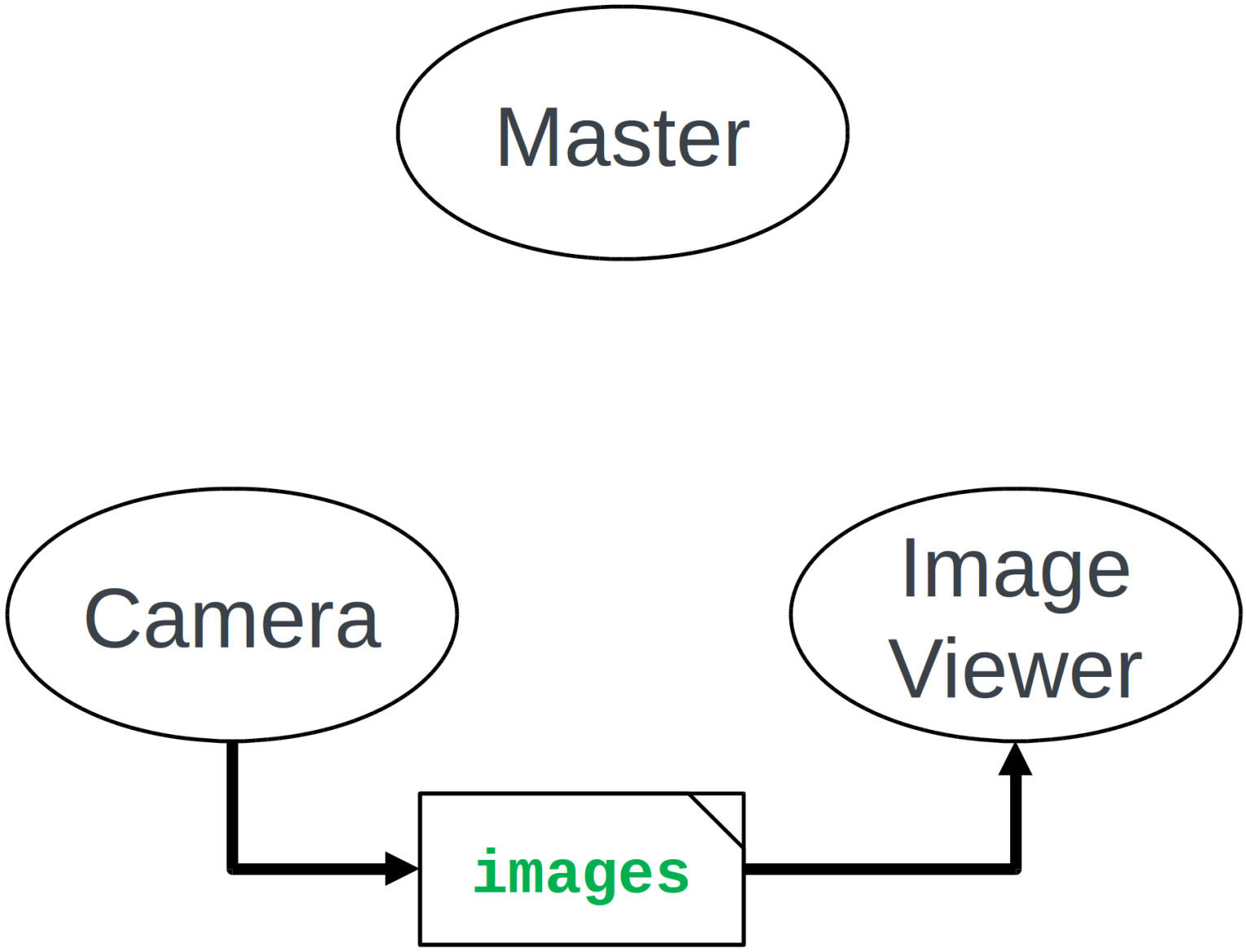
Master:

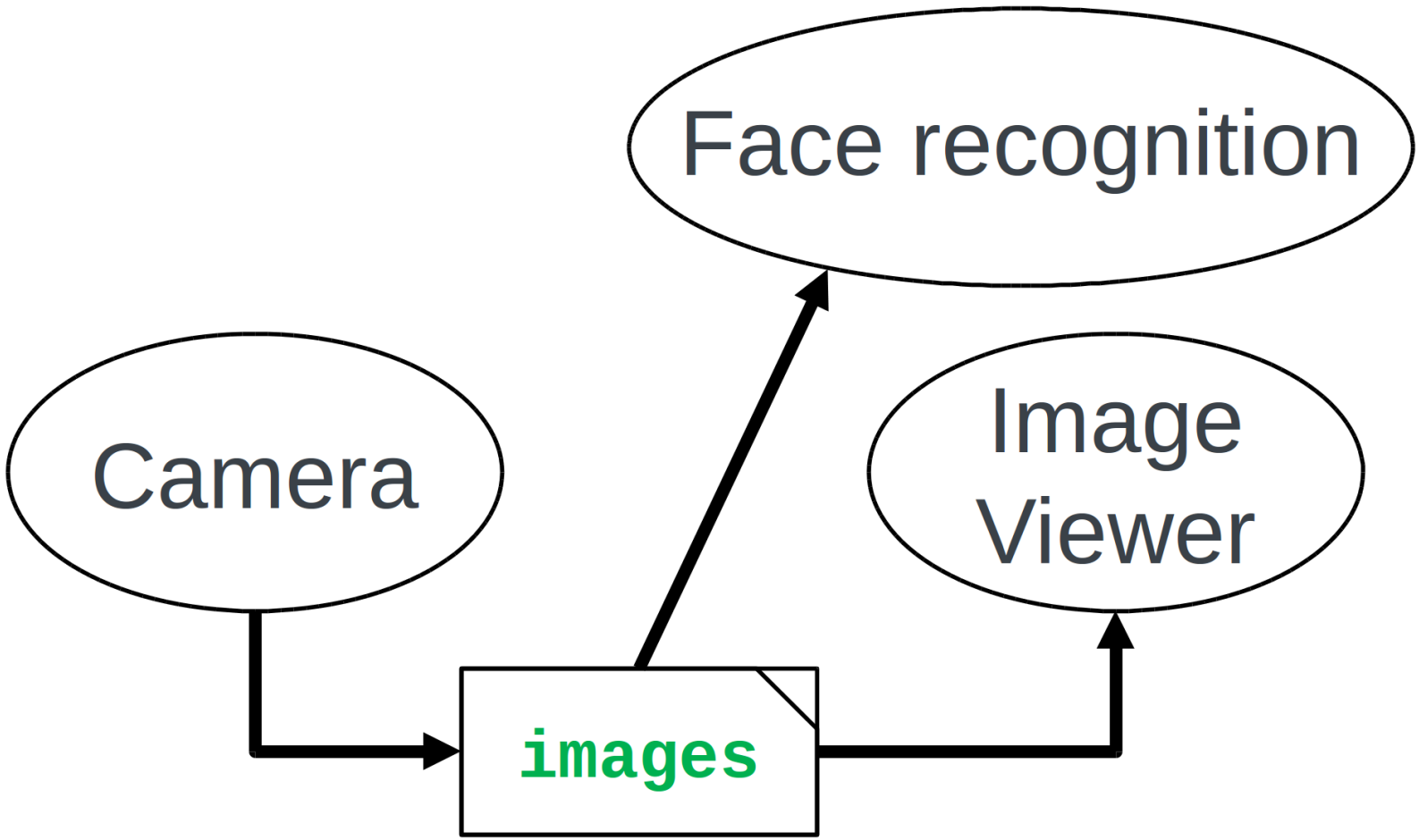
- The first process to run in an application that uses ROS, is the Master.
- The ROS Master provides name registration and lookup to the rest of the nodes.
- In a distributed system, we should run the master on one computer, and other remote nodes can find each other by communicating with this master.


Master
Camera
Image
Viewer


Master
advertise images
Camera
Image
Viewer


Master
Subscribe to images
Camera
Image Viewer
images

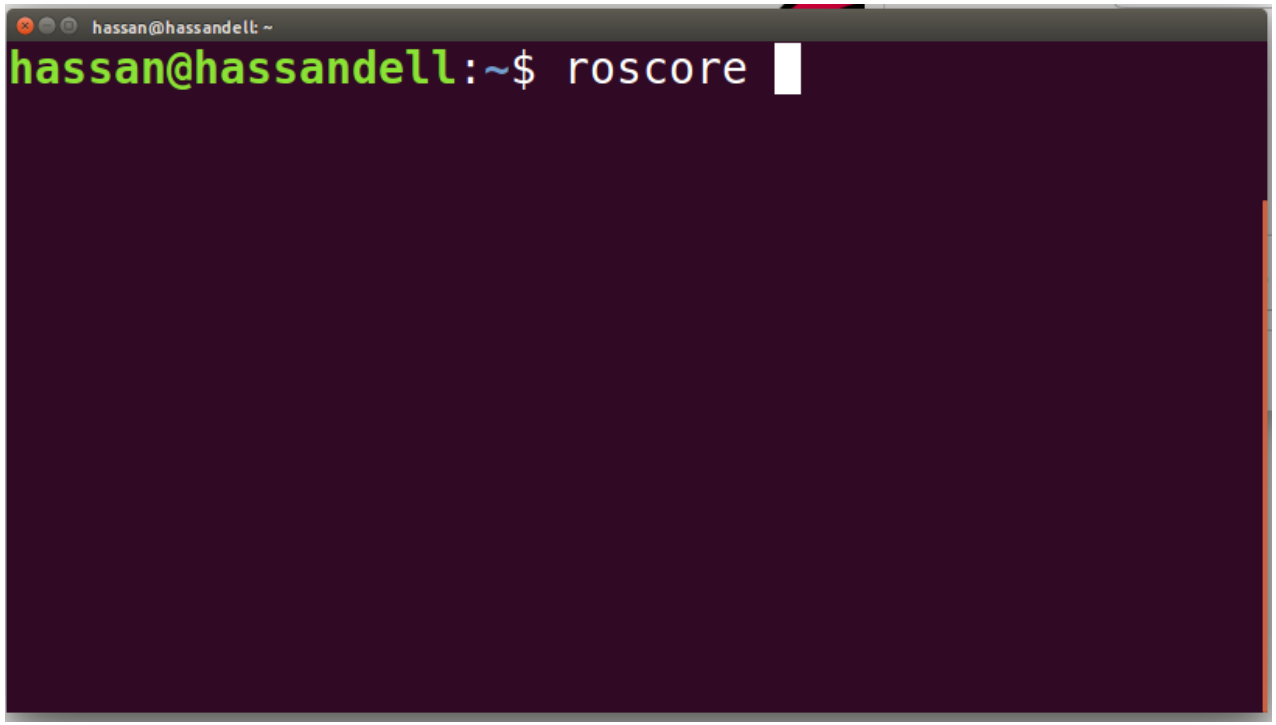

Master
Camera
Image
Viewer
images


Face recognition
Camera
Image
Viewer
images

ROS Concepts

Master:

- ROS master is invoked by this command:



Example

ROS Concepts

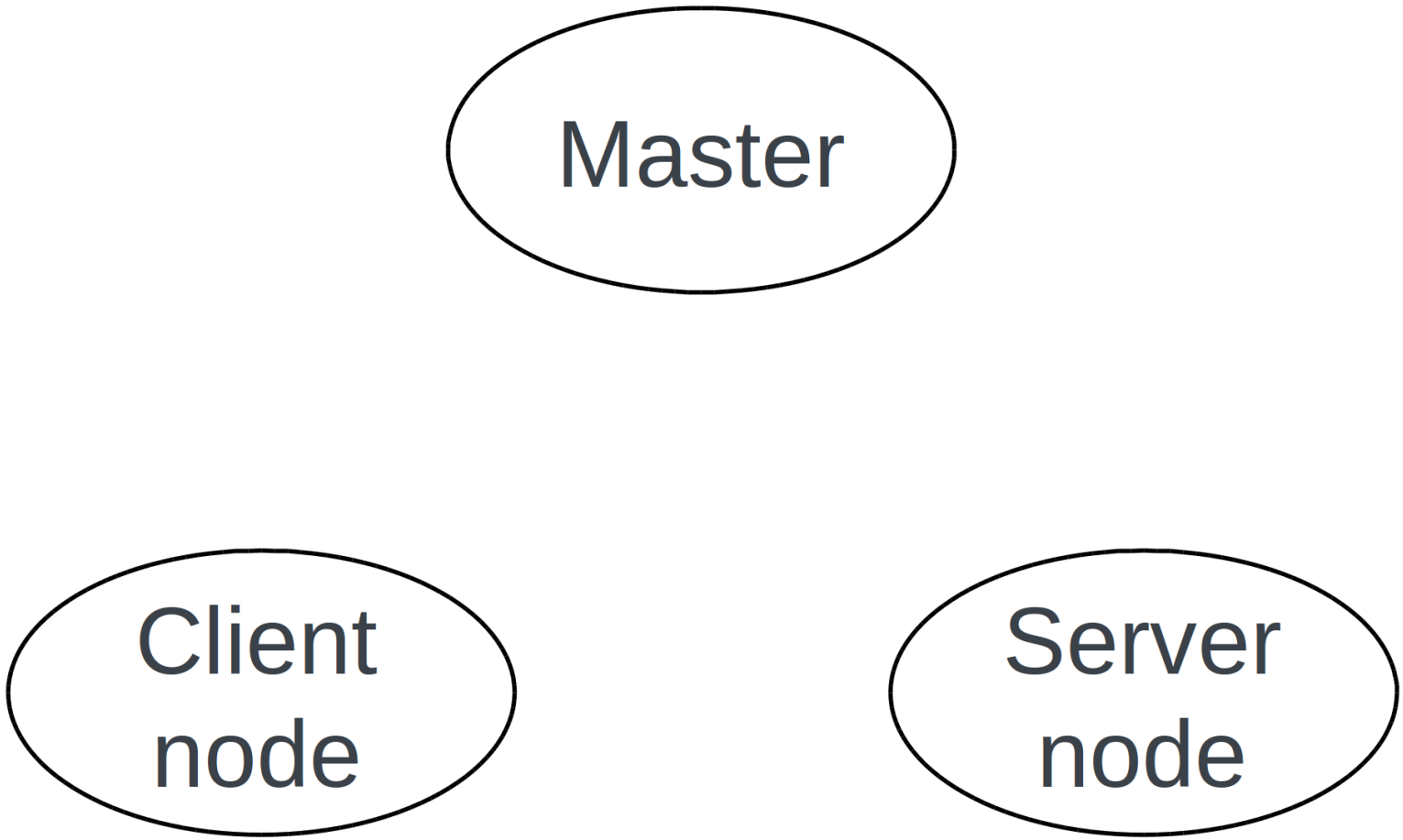
Services:

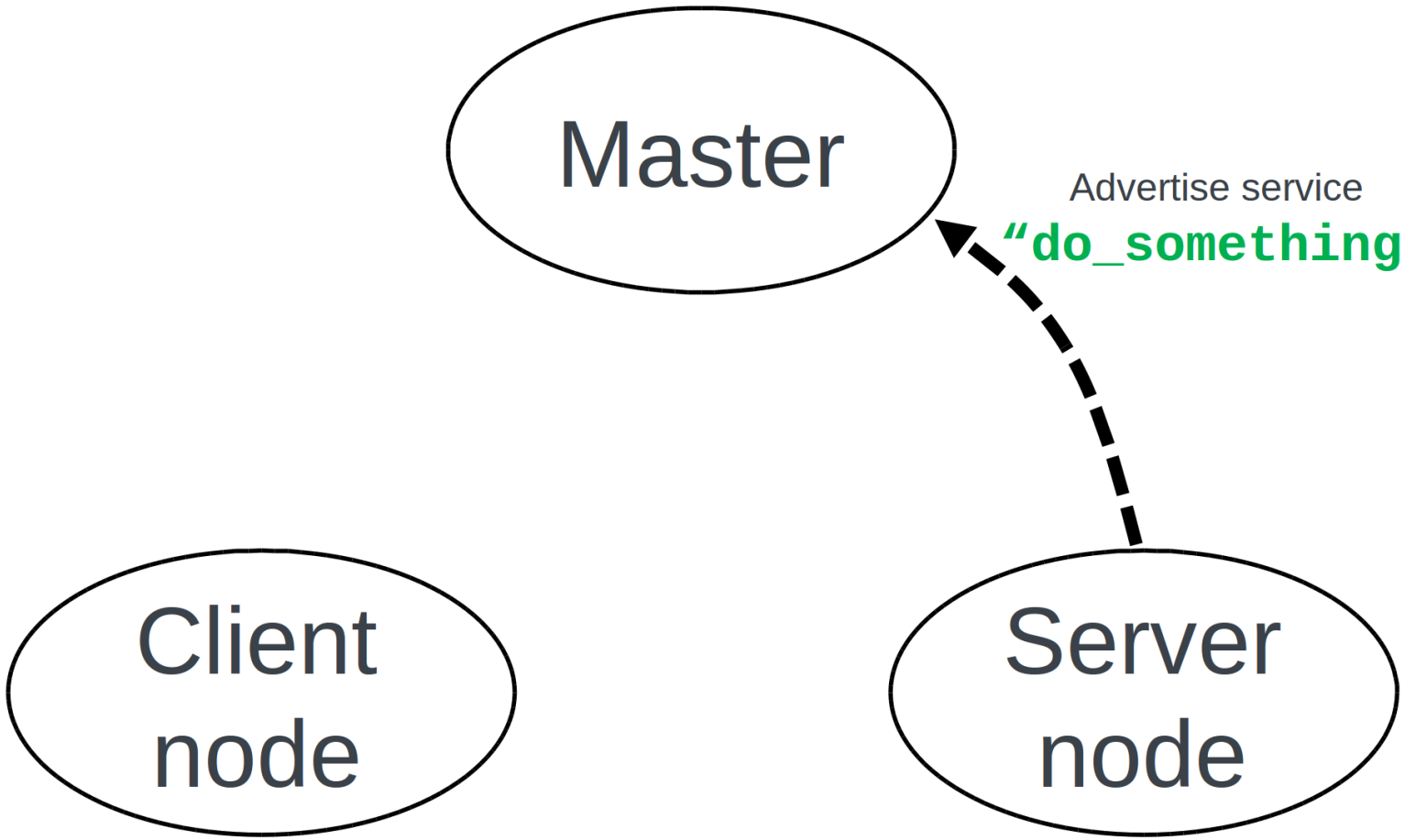
- In many scenarios a publish/subscribe model is not enough, it's a one-way communication.
- Example scenario: plan a path service.
- ROS Services provide an additional way of communication between nodes, a **request** / **reply** interaction.

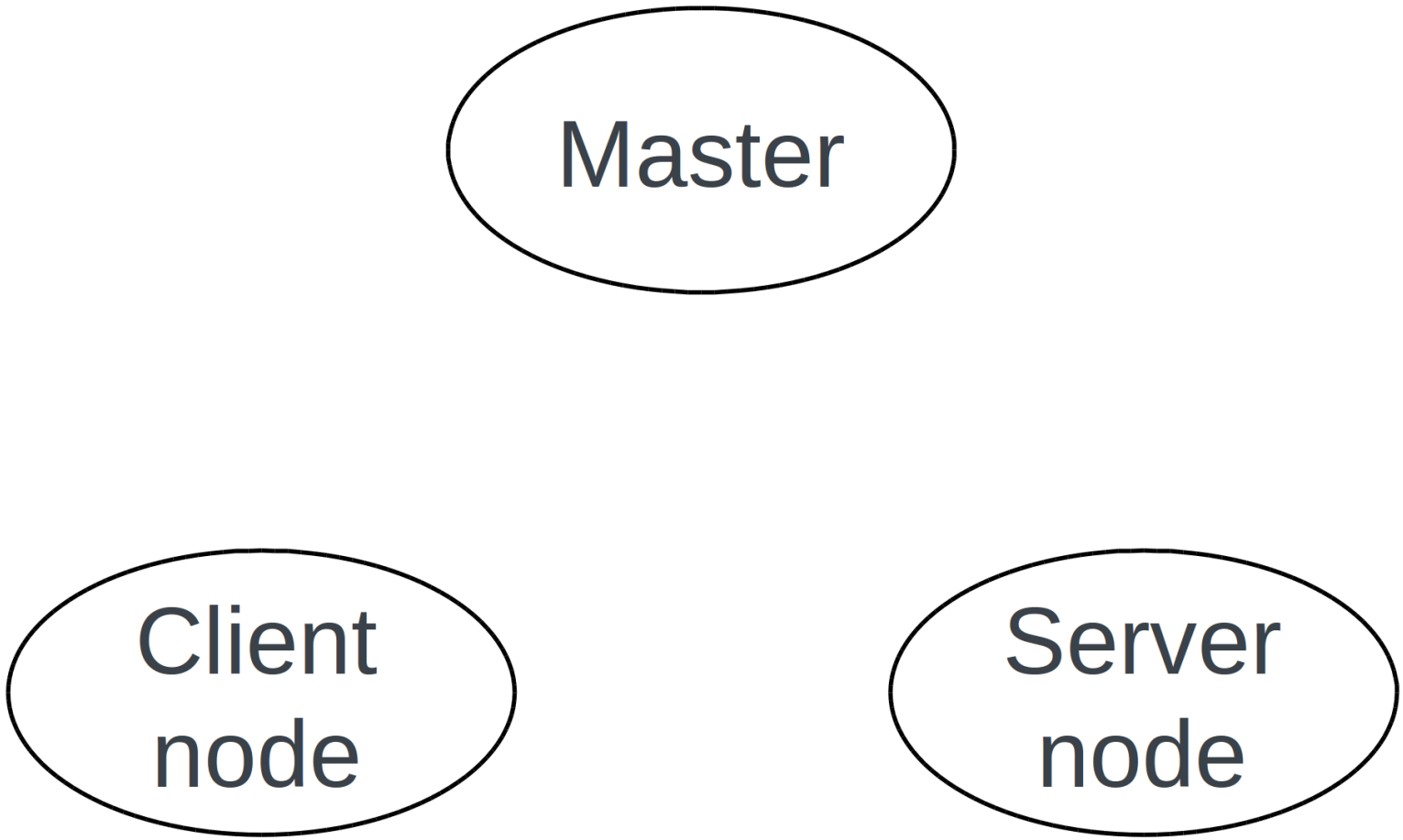
ROS Concepts

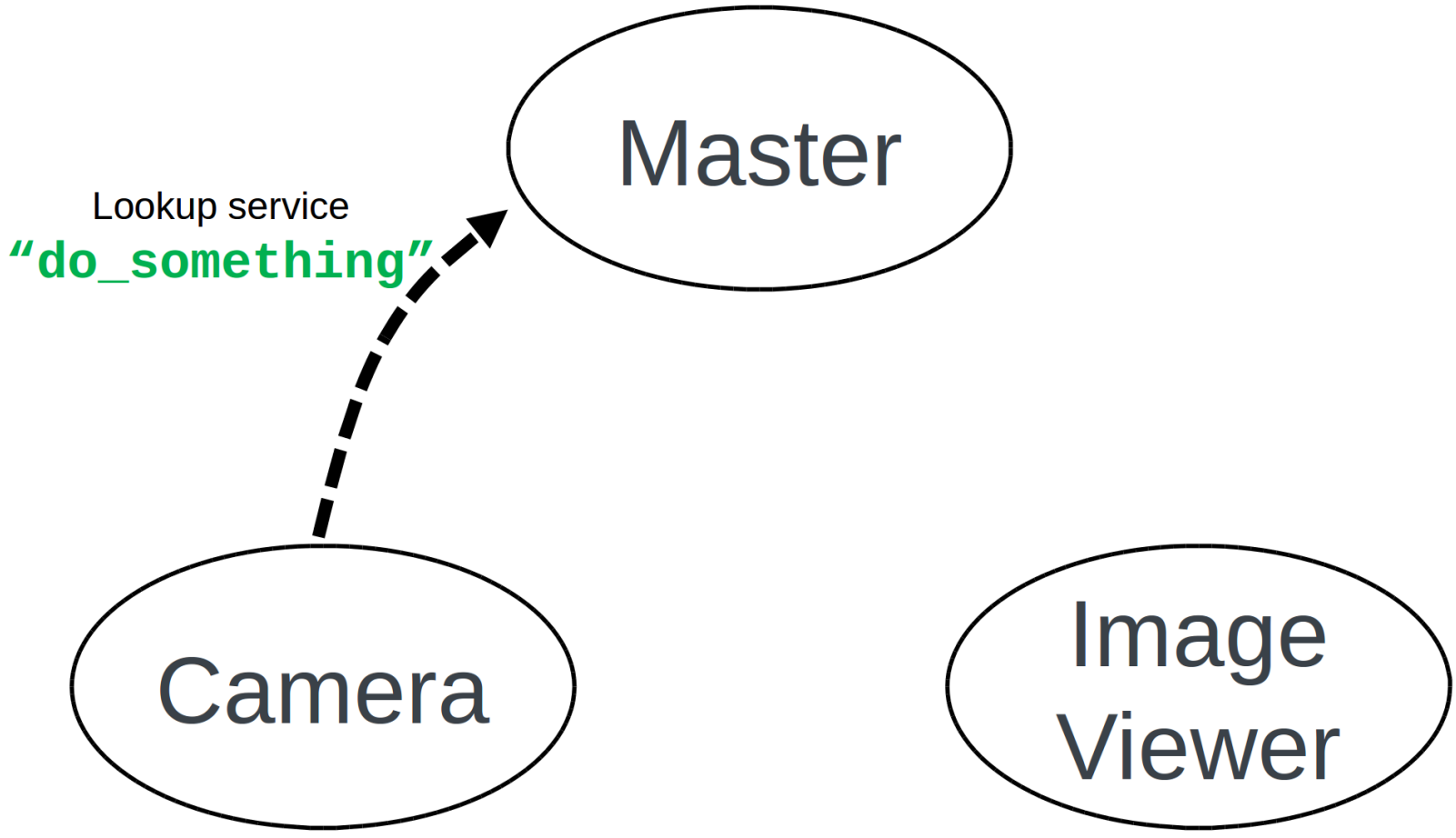
Services:

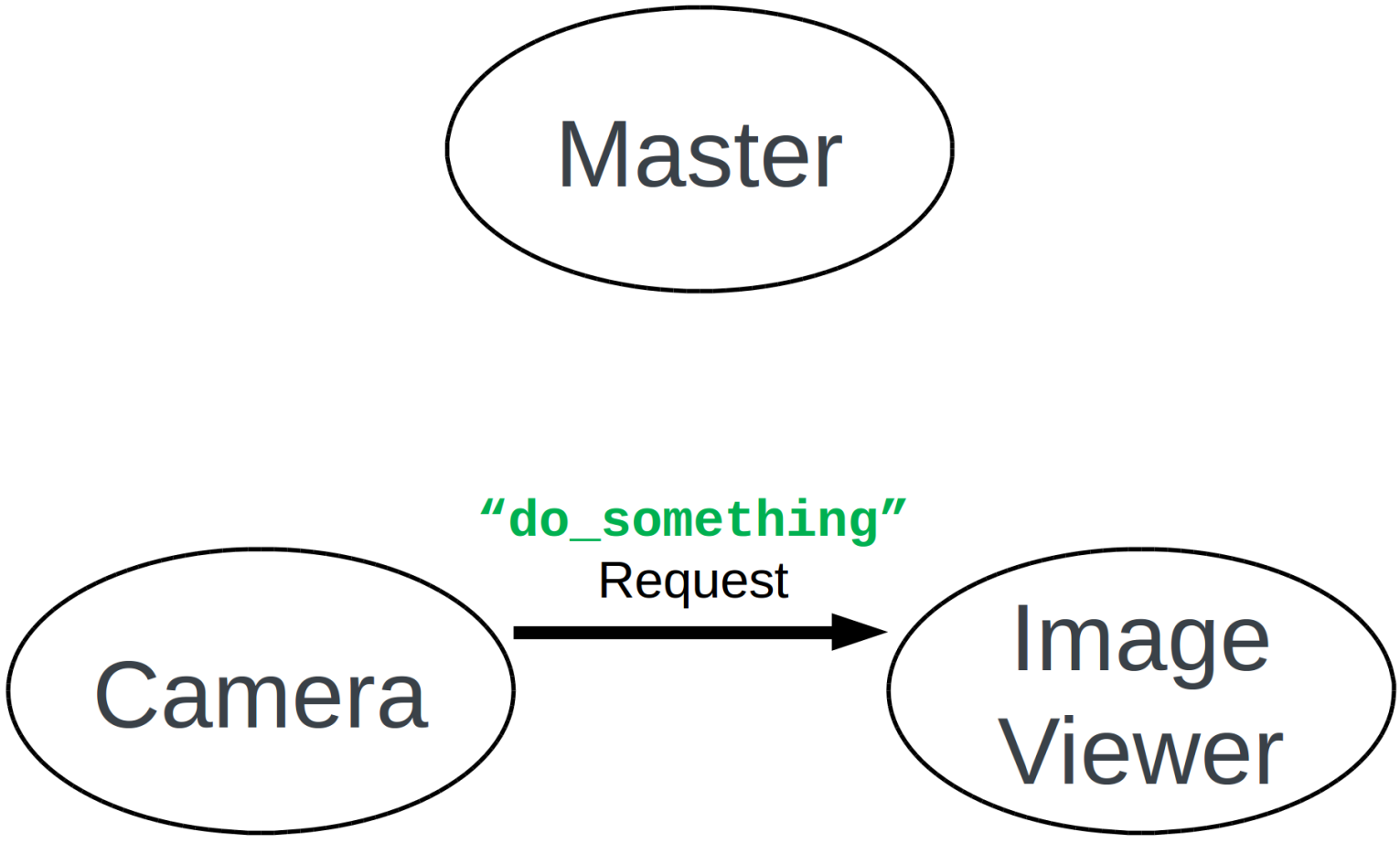
- It happens between two nodes, the service **server** node, and the service **client** node.
- A Client node sends a request for a named service and waits for the response, a node serving this service responds, and the communication is over.
- it is a one-to-one, two-way, one-time communication.

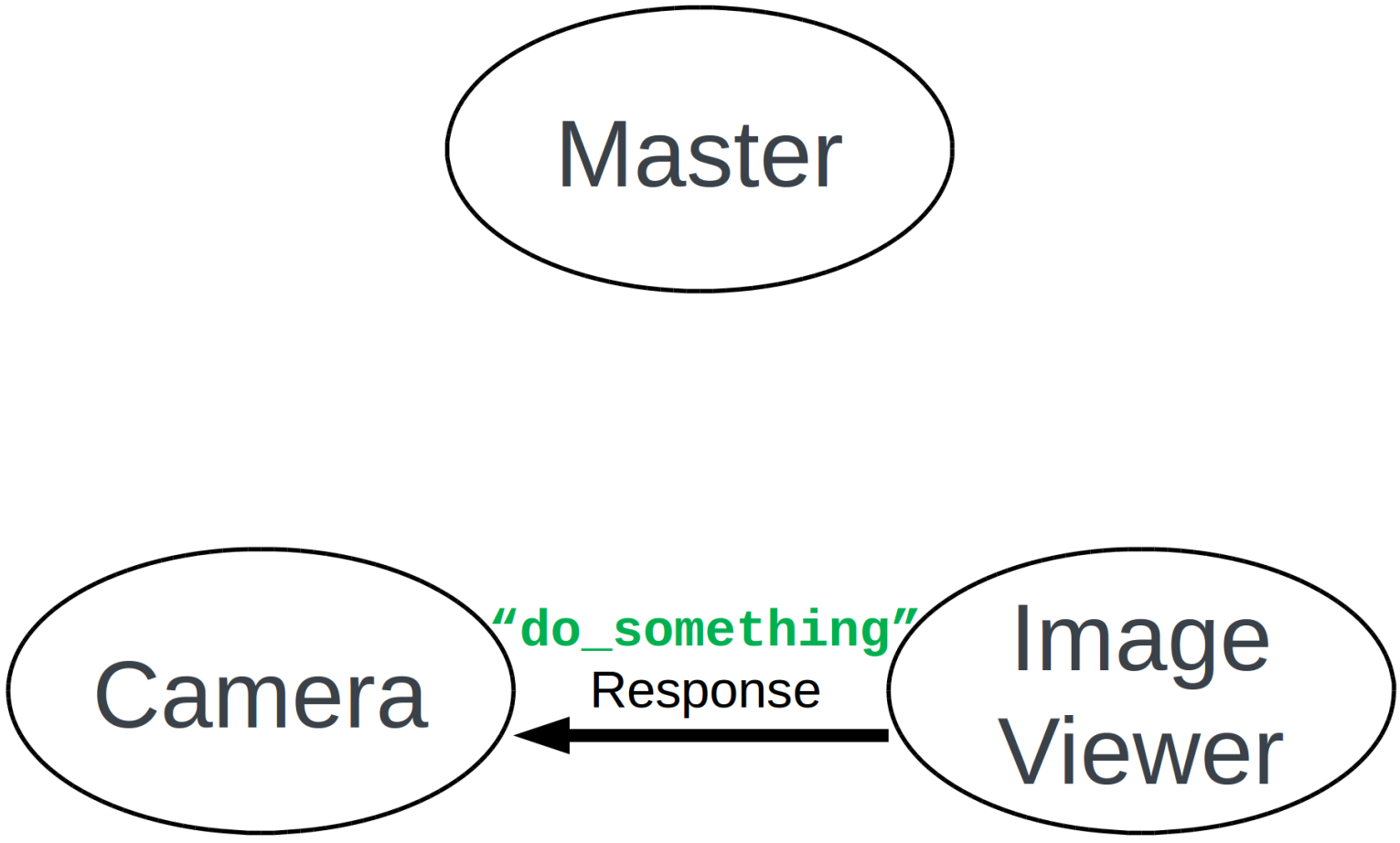

Master
Client node
Server node

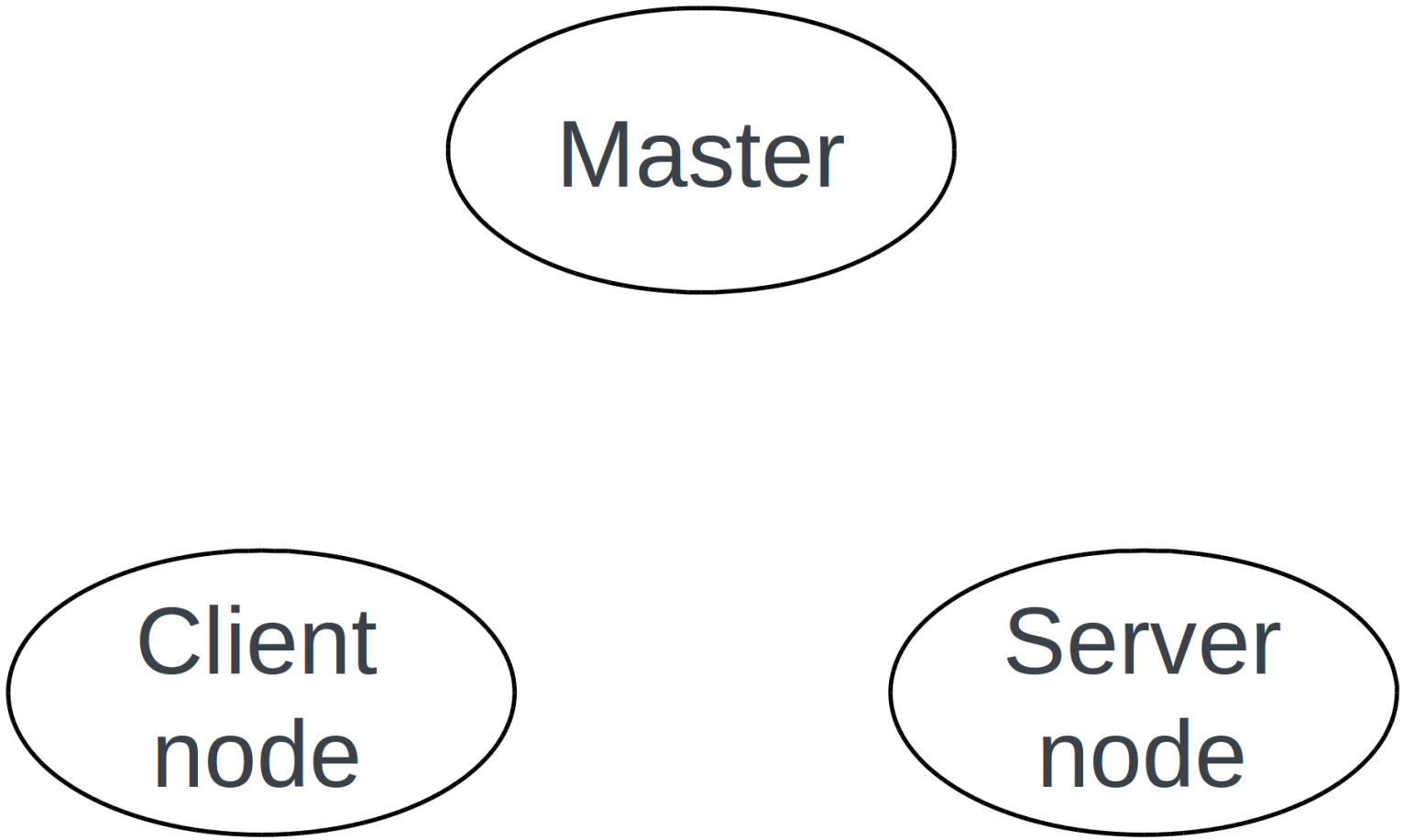

Master
Advertise service
“do_something”
Client node
Server node


Master
Client node
Server node


Master
Lookup service
“do_something”
Camera
Image
Viewer


Master
“do_something”
Request
Camera
Image
Viewer


Master
Camera
“do_something”
Response
Image
Viewer


Master
Client node
Server node

Example

ROS Concepts

Actions:

- ROS services are not suitable for long-term tasks, a client that have sent a service request keeps on waiting for the response from the server. ROS actions solves this.
- ROS actions are also useful for preemptable tasks, i.e. tasks capable of being interrupted with the option of resuming the task at a later time.
- In ROS actions, an action client sends a request to the server, the client doesn't have to wait for the response.

ROS Concepts

Actions:

- Action client can request for feedback which the action server provides during execution.
- Once the server finishes executing the task, it sends a result message to the client.

ROS Concepts

Parameter Server:

- A network-shared dictionary accessible to all nodes.
- Typically used to store static data, like parameters and configurations.
- A central location to store static values.
- All nodes can access and modify those values.
- Parameter server is a part of ROS Master.

Demo

ROS Concepts

Bags:

- ROS bag is a mechanism for recording data for later playback.
- You can record a complete session, with all the topics and messages being exchanged along with their time stamping.

References

1. ROS Wiki.
2. ROS2 Wiki, on the drawbacks of ROS:

   http://design.ros2.org/articles/why_ros2.html.
3. Overview on ROS services: https://www.youtube.com/watch?v=qhnlmrGQVvM.
4. Overview on ROS actions: https://www.youtube.com/watch?v=LoRXdNMuslQ.
5. ROS introduction slides by Rada:

   https://wiki.ros.org/Events/CoTeSys-ROS-School?action=AttachFile&do=get&target=ros_tutorial.pdf.