Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

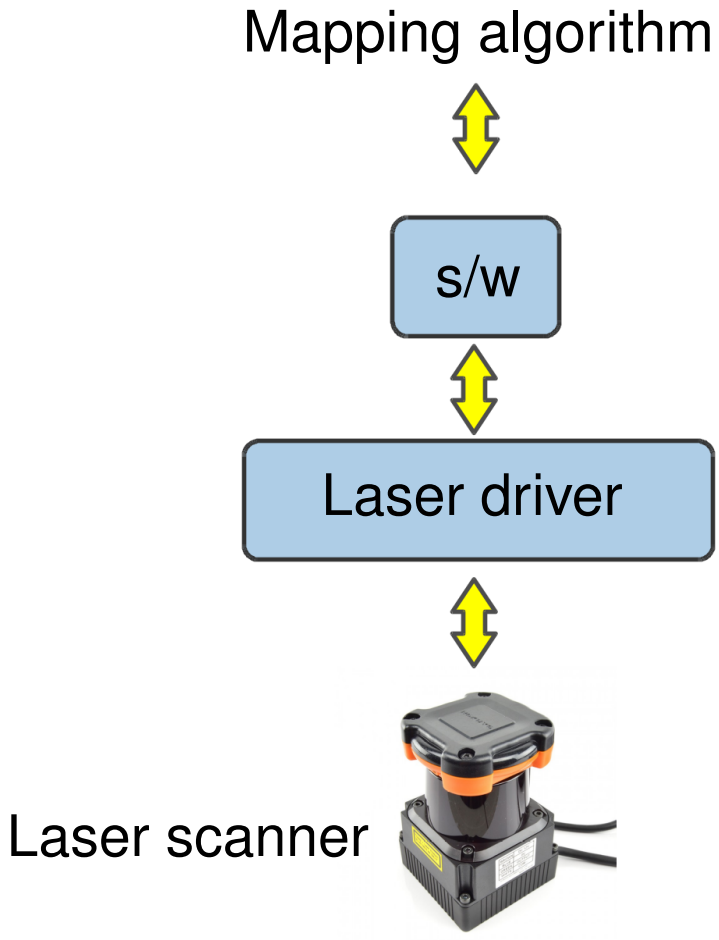
Introduction to Robot Operating System (ROS)

Foundation course

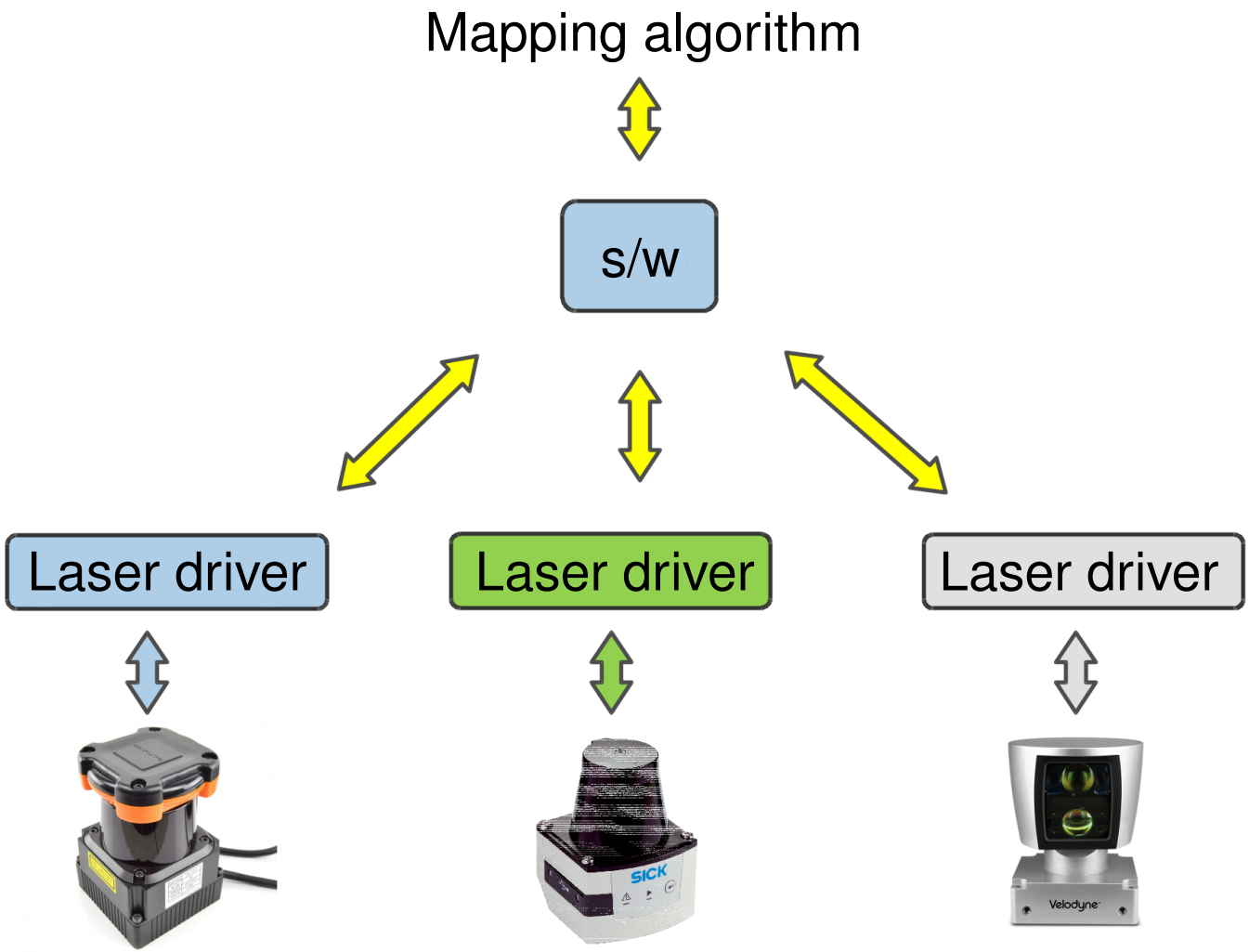
December 6, 2022

Author: Hassan Umari
Edited by Vamsi Krishna & Hamsa Datta

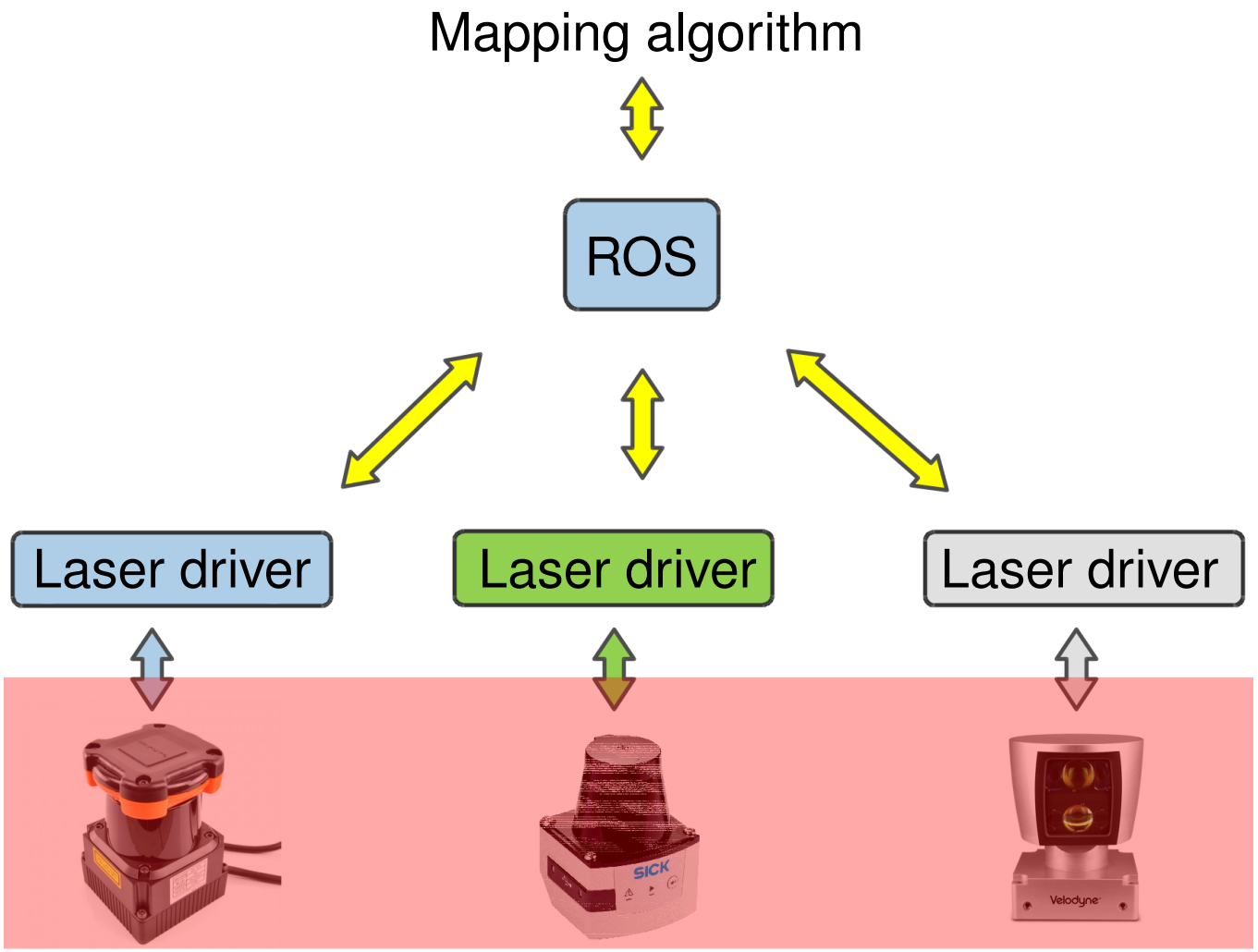
Scenario



Scenario

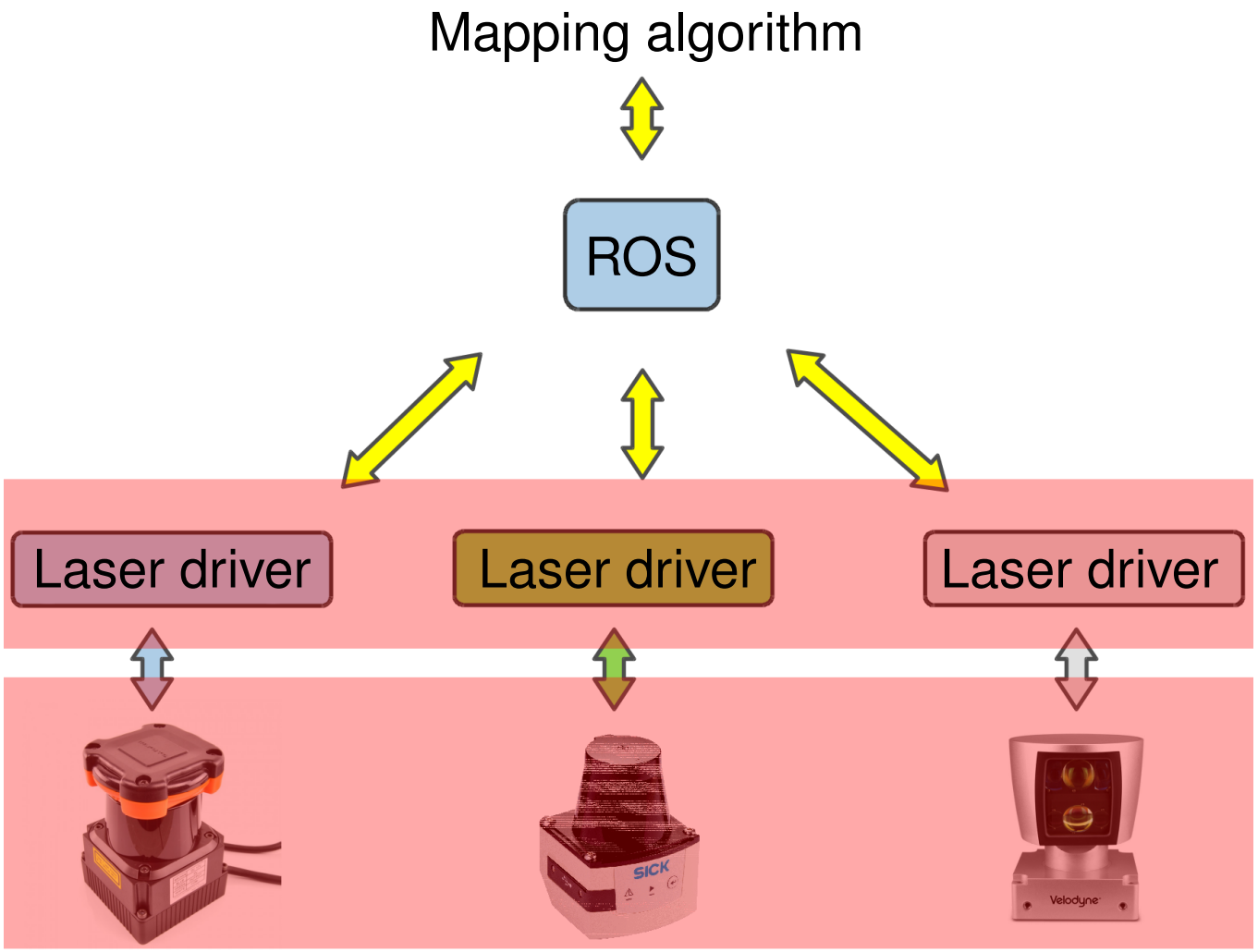

Mapping algorithm
s/w
Laser driver
Laser driver
Laser driver
SICK
Velodyne

Scenario

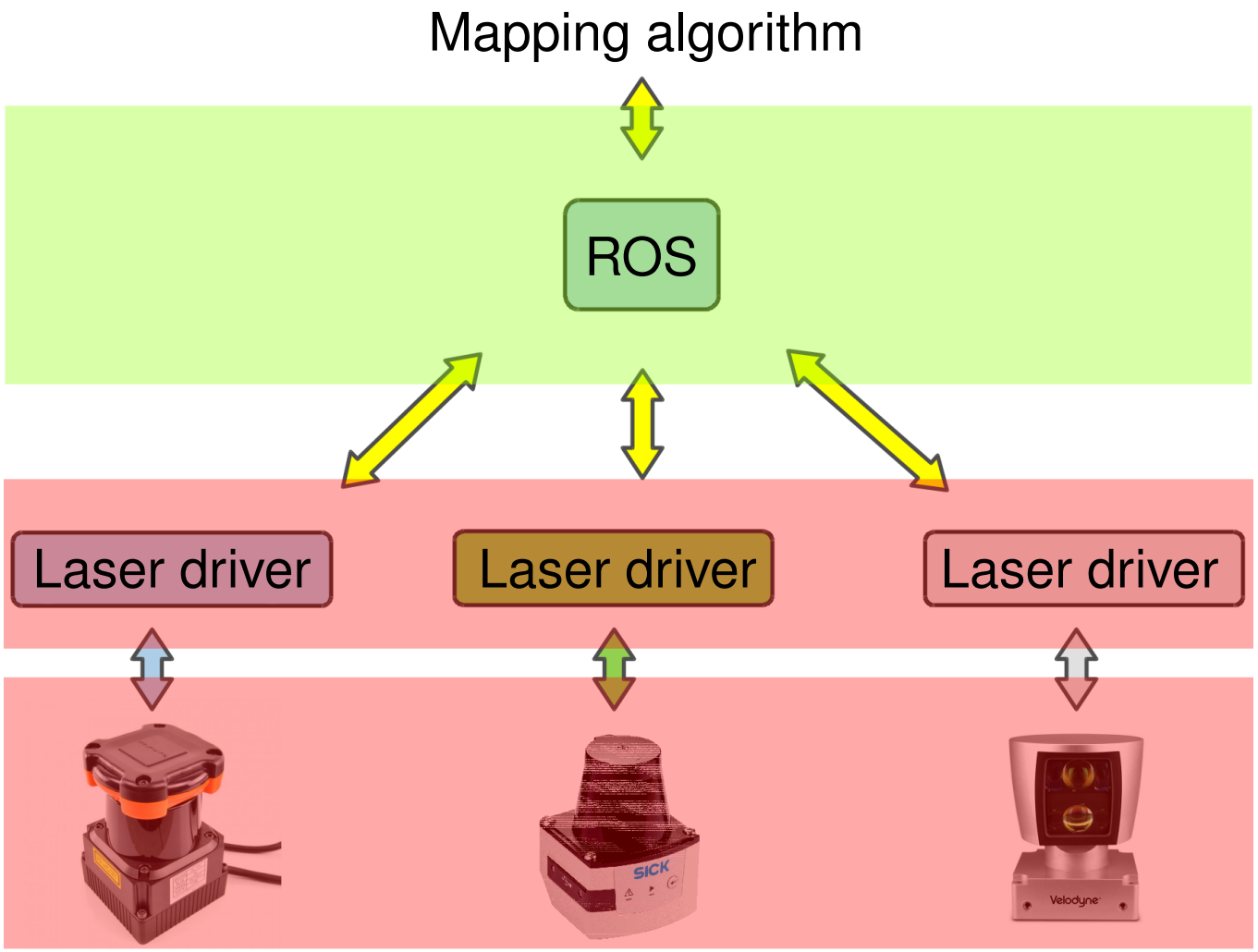


Different hardware

Scenario


Mapping algorithm
ROS
Laser driver
Laser driver
Laser driver
SICK
Velodyne
Different drivers
Different hardware

Scenario

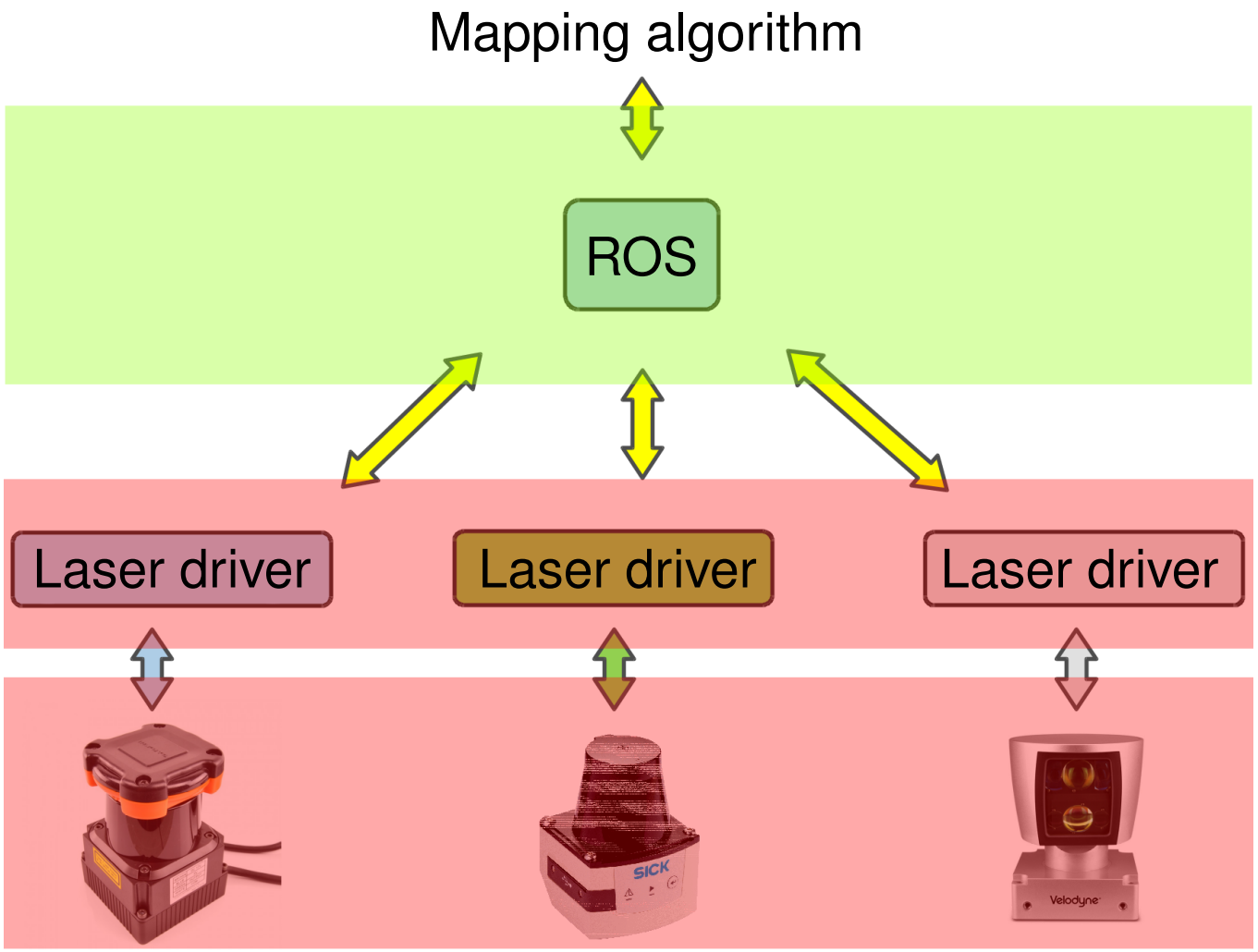


Data received in the same message format

Different drivers

Different hardware

Scenario

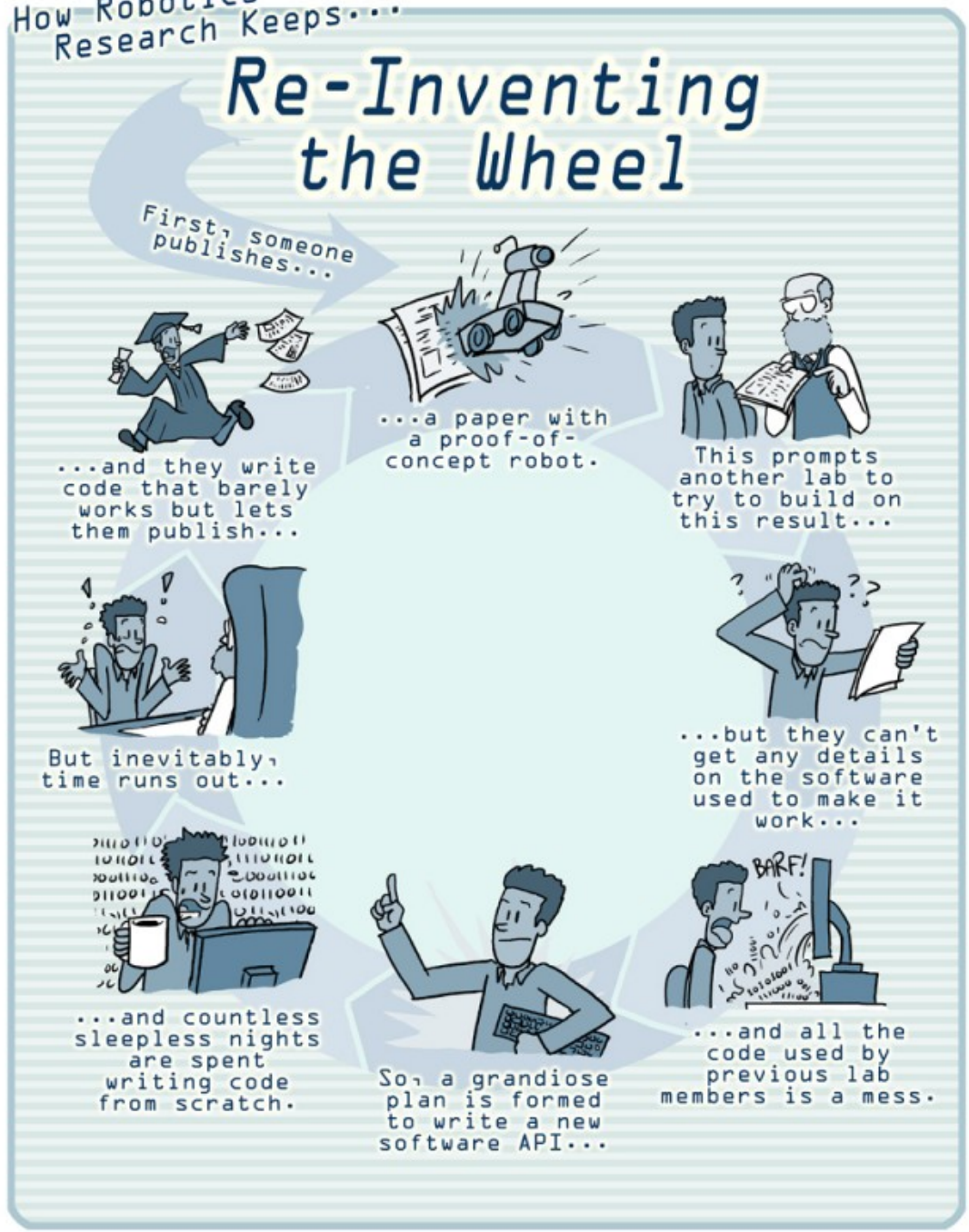


No change

Data received in the same message format

Different drivers

Different hardware



http://www.willowgarage.com/blog/2010/04/27/reinventing-wheel

What ROS is

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

What ROS is

- A way to standardize writing software for robots.
- It enhances code reusability  .
- ROS is open-source  .
- It is a meta-operating system.
- ROS is installed on top of Linux/ Windows/ Mac.

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

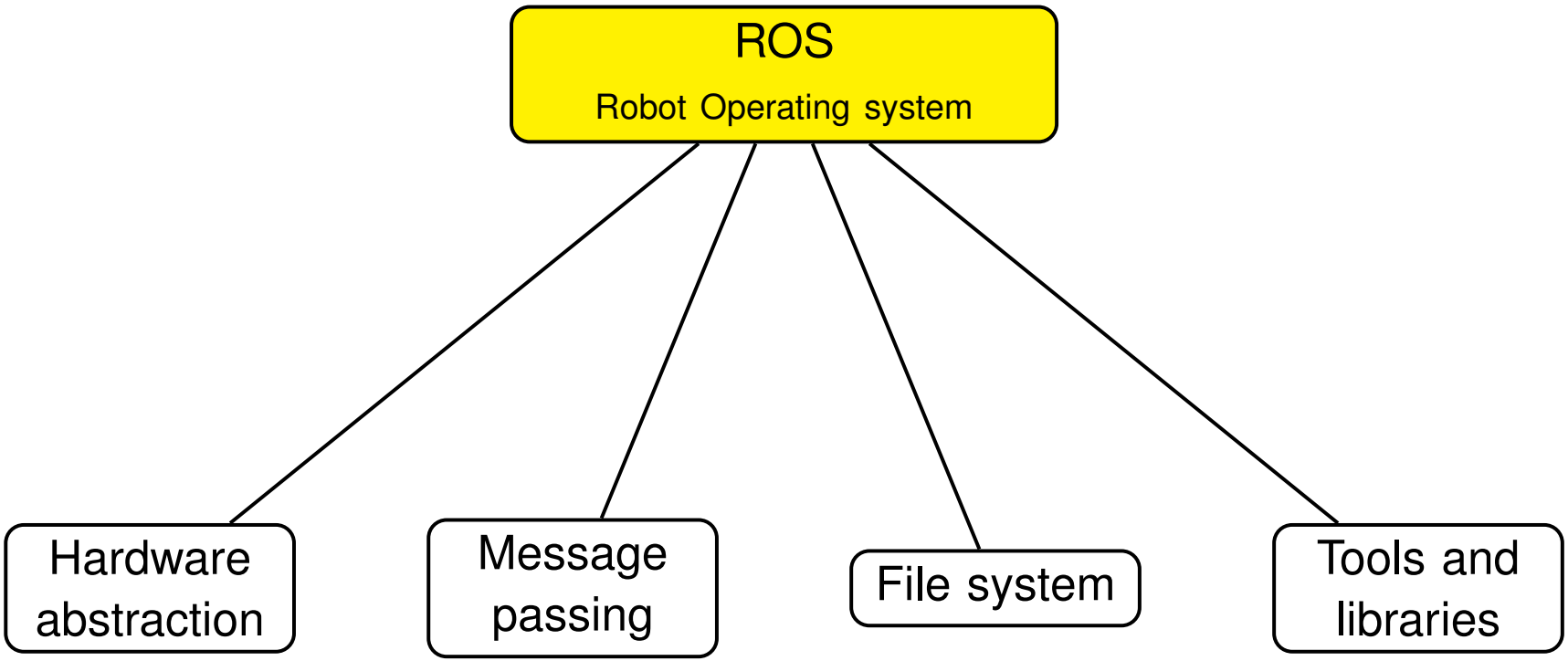
Mapping | Navigation | Pick & place

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

Features of ROS

- ROS functionalities are implemented as a library in different programming languages (Python, C++, ...).
- These libraries are referred to as ROS client libraries.

Language independent

Features of ROS

ROS client libraries

- ROS Supported Client libraries:
  - Cpp, Python
- ROS Community-maintained client libraries:
  - C (micro-ROS)
  - Java and Android
  - Node.js
  - Rust
- ROS support on MATLAB:
  - Robotics System Toolbox

Distributed and modular

Features of ROS

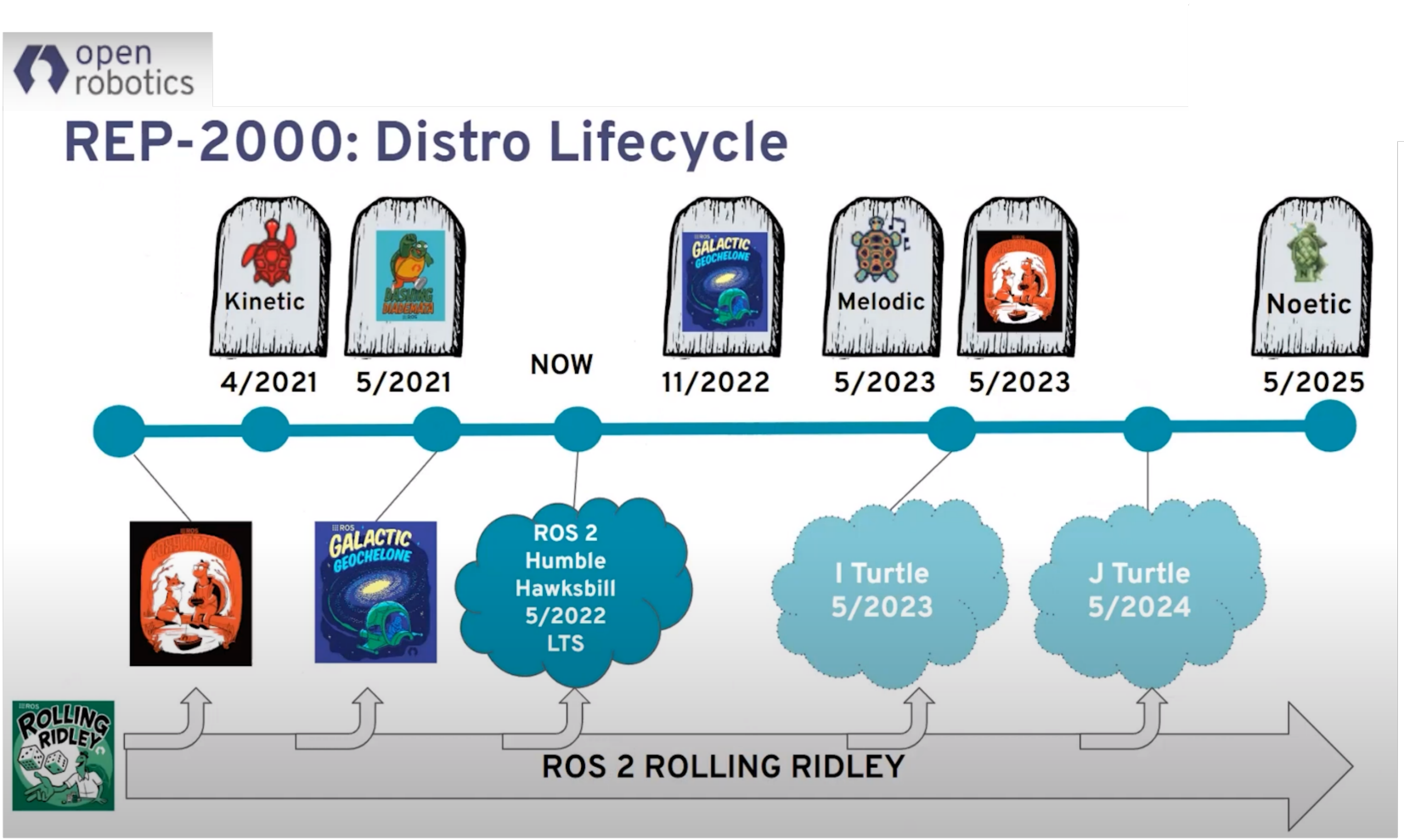
- ROS supports running processes on multiple computers connected together through a LAN.
- In a system running ROS, there will be multiple of processes where each process can do certain task. A process can be changed without altering the remaining processes.

A lot of libraries and tools

Features of ROS

- Examples of libraries:
  - Navigation stack
  - SLAM (gmapping, slam_toolbox, etc.)
  - Localization (amcl, etc.)
  - Motion planning for manipulators (MoveIt)
  - Support for popular libraries (OpenCV, PCL)
- Examples of tools:
  - rviz (3D visualization)
  - ros bag (recording/playing back sensor data)
  - catkin, colcon (build system)
  - Command line tools (topic, param, etc.)

ROS distributions

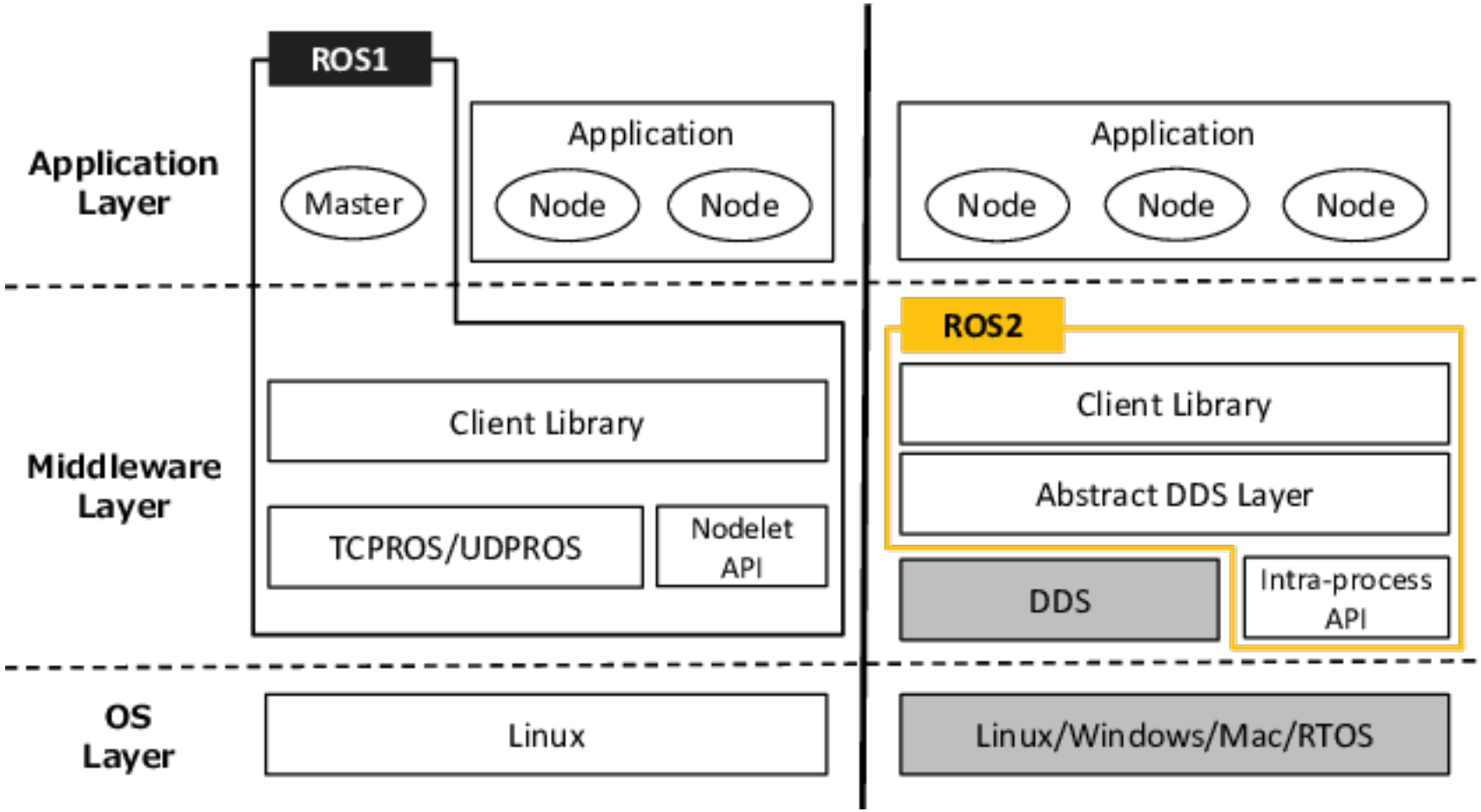


Source: The Latest Advances In ROS 2, Kat Scott, Open Robotics.

Drawbacks of ROS1

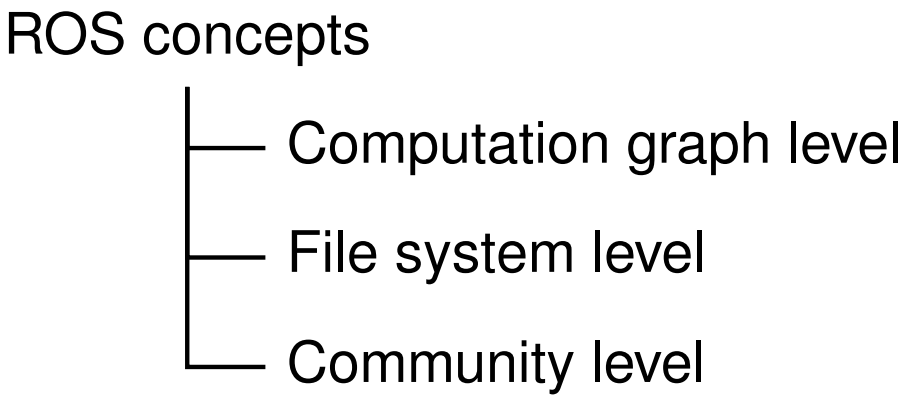
- It needs a PC. Does not work on a microcontroller at all (we can only set up a ROS node on the microcontroller)!
- Officially supported only on Ubuntu (but there are also experimental versions for Windows, OS X and more).
- No standard approach for multi-robot systems.
- Not a real distributed system, due to the ROS master.
- For ROS1 versions older than Noetic we are forced to "live in the past". ROS1 python client library (rospy) used python 2 instead of 3.

ROS1 vs ROS2 architecture



Source: Maruyama, Yuya, Shinpei Kato, and Takuya Azumi. "Exploring the performance of ROS2." Proceedings of the 13th International Conference on Embedded Software. 2016.

ROS concepts



Computation graph level

ROS concepts

- In an application that uses ROS, the computations are executed by a collection of elements processing data together.
- It comprises of executables and connections between them.

Computation graph level

ROS concepts

Concepts related to ROS computation graph:

1. Nodes
2. Topics
3. Messages
4. Services
5. Actions
6. Bags

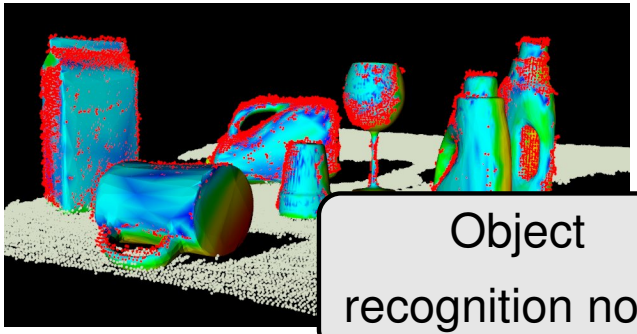
Computation graph level

ROS concepts

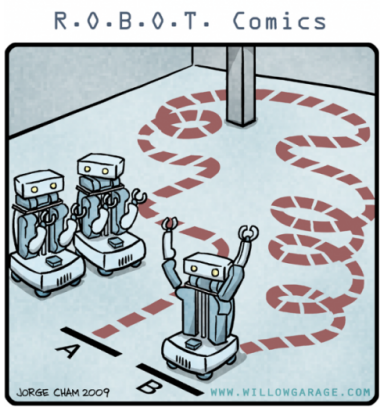
Nodes:

- It is the basic functional block of code that performs a function eg publish to robot velocity(cmdvel), receive laser scan data(scan)
- A ROS node is a process that exchanges data with other processes through ROS network.
- It may be a python script, a C++ written process, or even a MATLAB script.
- The advantage of ROS is that you can write your nodes in different languages and can still make them work together
- They can be run on single or multiple computers and are connected together in a peer-to-peer network.

Camera driver node



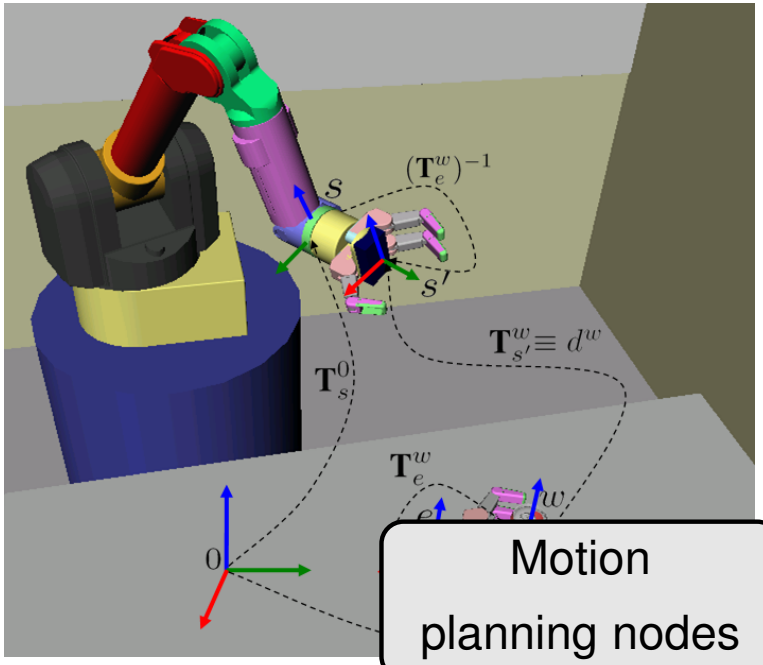
Object recognition node



Navigation node

Manipulator driver node

Robot driver node



Motion planning nodes

Computation graph level

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.

Computation graph level

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.
- Topic have a defined type eg: topic cmd_vel has a type of \geometry_msgs\Twist

Example (TurtleSim)

Computation graph level

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 **call-and-response** model.

Computation graph level

ROS concepts

Services:

- It happens between two nodes, the service **server** node, and the service **client** node.
- A Client node calls a service with a request, a node serving this service responds, and the communication is over.
- A client node call can be synchronous or asynchronous.
- it is a one-to-one, one-to-many, two-way, one-time communication.

Example
(TurtleSim again)

Computation graph level

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 can be canceled at any time, and also provides a steady feedback.
- In ROS actions, client calls are asynchronous.

Computation graph level

ROS concepts

Actions:

- Action client can request for feedback which the action server provides during execution.
- The action nodes can perform other processes depending on this feedback eg: Send a goal to robot to move from point A to B but closer the robot gets to point B the robot can slow down
- Once the server finishes executing the task, it sends a result message to the client.

Computation graph level

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.
- This is very useful for finding problems in the robot

Example
(TurtleSim one last time)

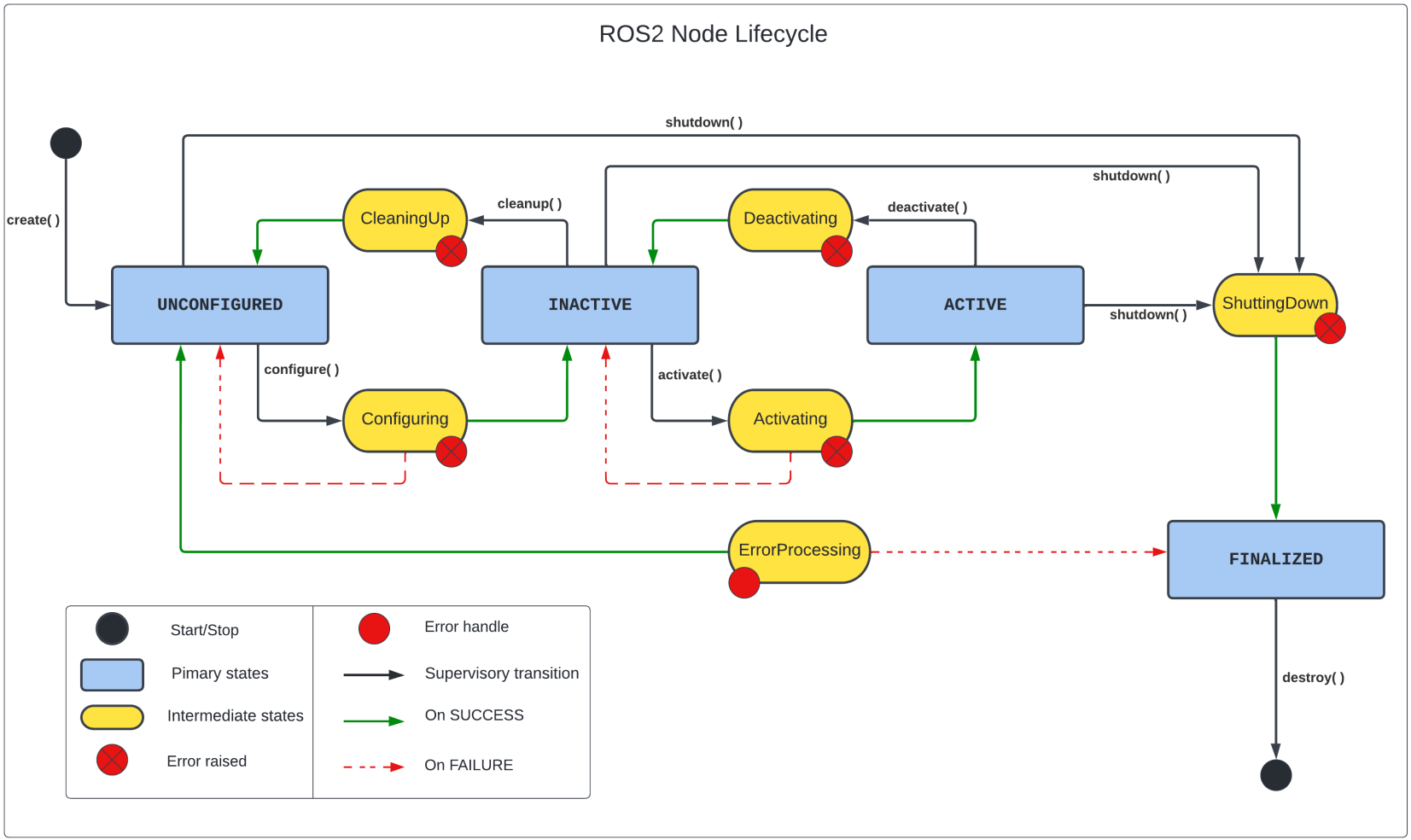
ROS1 and ROS2 differences

ROS concepts

- No more ROS Master
- DDS as middleware
- One base library - **rcl**
- Structure of nodes
- Components
- Lifecycled nodes
- No global parameters
- QoS - Quality of Service

ROS2 node lifecyle

ROS2 concepts



Source: Geoffrey Biggs, Tully Foote, Managed nodes,ROS2 design.

DDS

ROS2 concepts

"Data Distribution Service is a middleware protocol and API standard for data-centric connectivity. It is Pub/Sub technology for real-time, dependable and high performance data exchanges."

DDS is Pub/Sub on Steroids

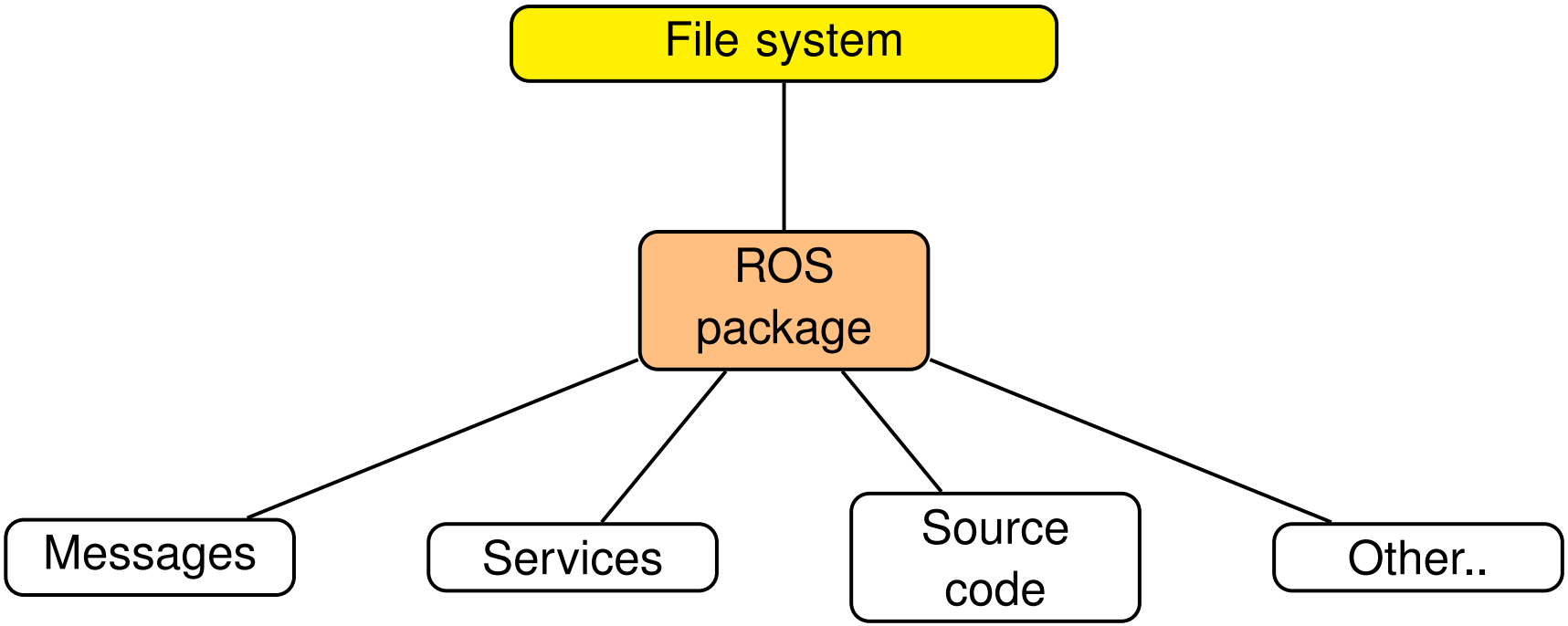
key challenges addressed by DDS:

- **Real-time:** the delivery of the appropriate information at the appropriate location and time on a continuous basis.
- **Dependable:** assuring reliability, safety, and integrity despite software and hardware failures.
- **High-Performance:** allowing for the rapid distribution of huge amounts of data.

Source: The DDS Tutorial by Angelo Corsaro.

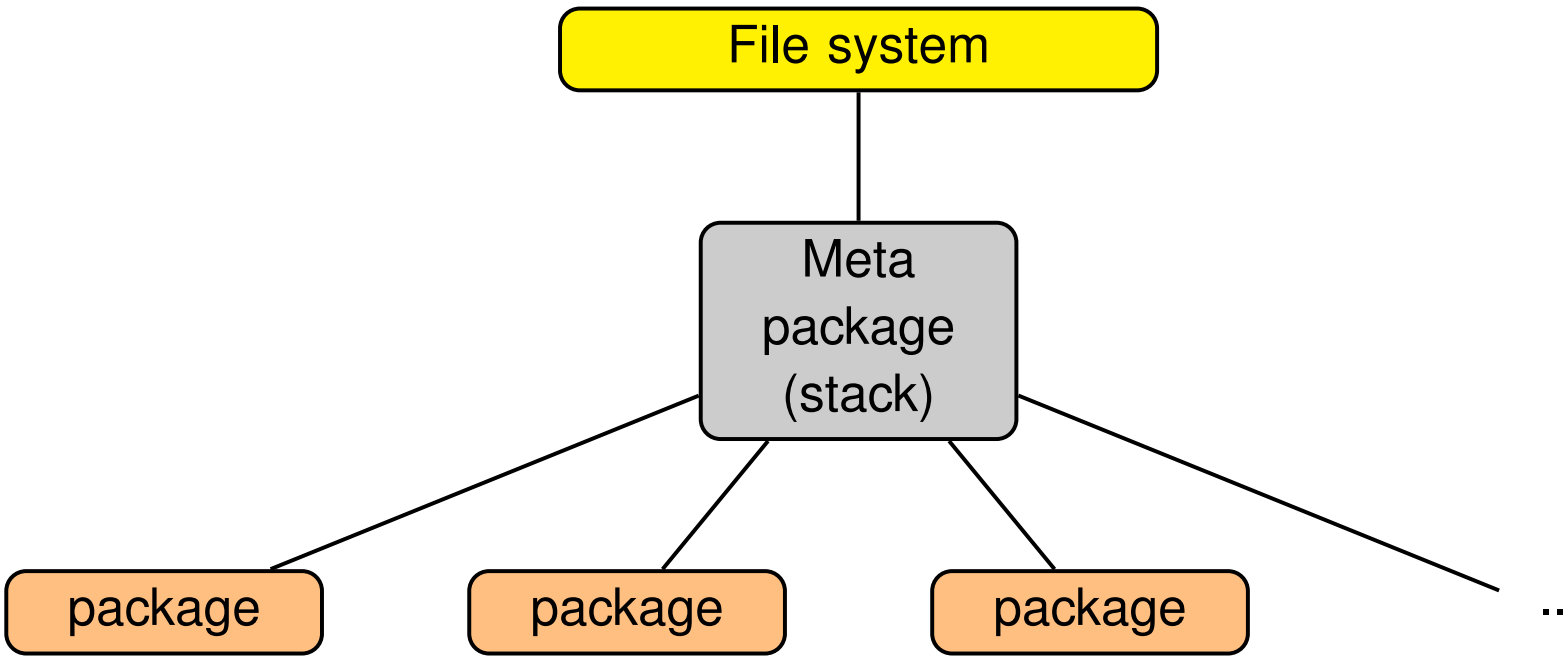
File system level

ROS concepts



File system level

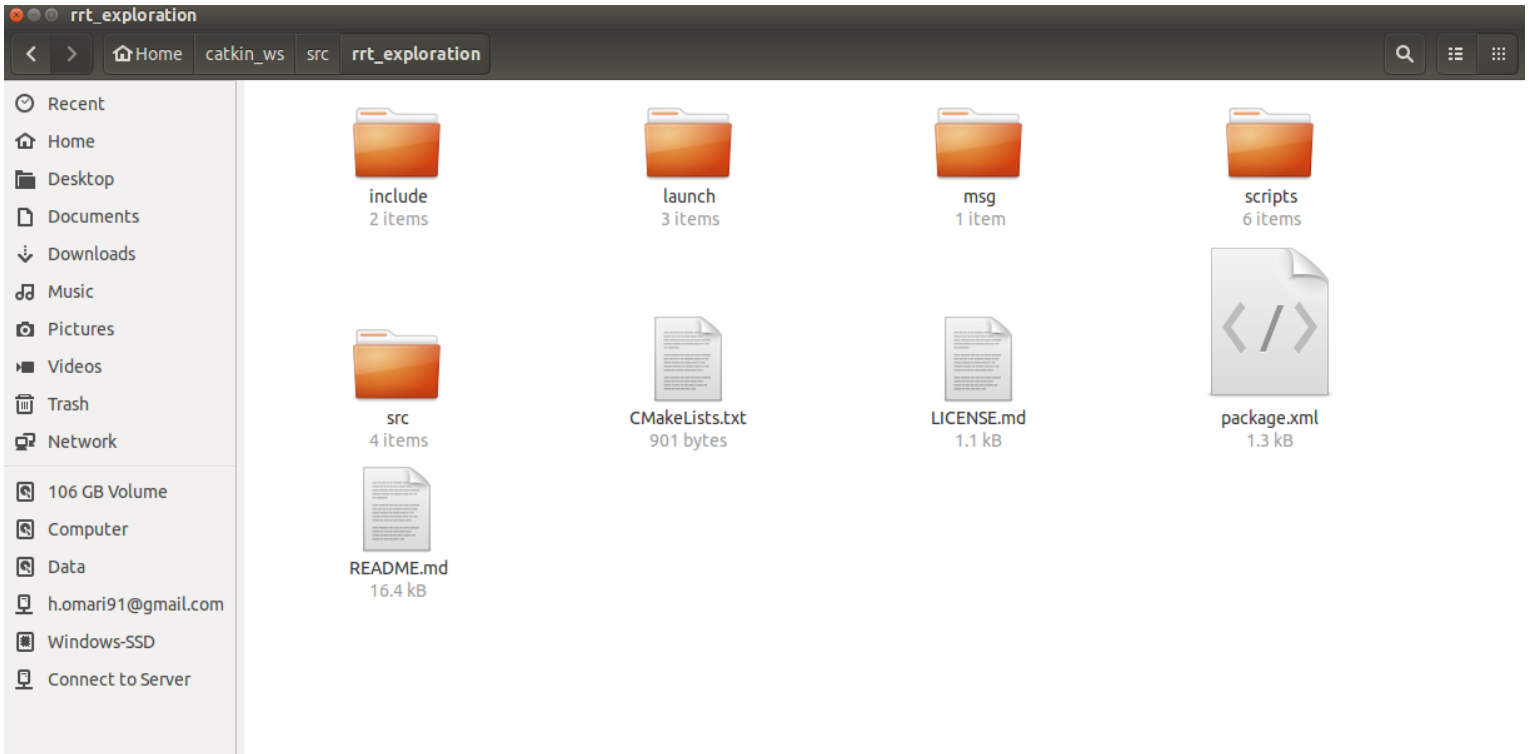
ROS concepts



File system level

ROS concepts

Inside a ROS package:



Community level

ROS concepts

Concepts related to ROS2 development process and it's community:

1. The ROS Wiki.
2. Repositories.
3. Mailing Lists.
4. ROS Answers.
5. ROS Distributions.

References

1. ROS Wiki.
2. ROS2 Wiki, on the drawbacks of ROS.
3. Overview on ROS services.
4. Overview on ROS actions.
5. ROS introduction slides by Rada.
6. ROS Client Libraries ROS1, ROS2.
7. Previous material of the foundation_course.

Thank you

Any questions?