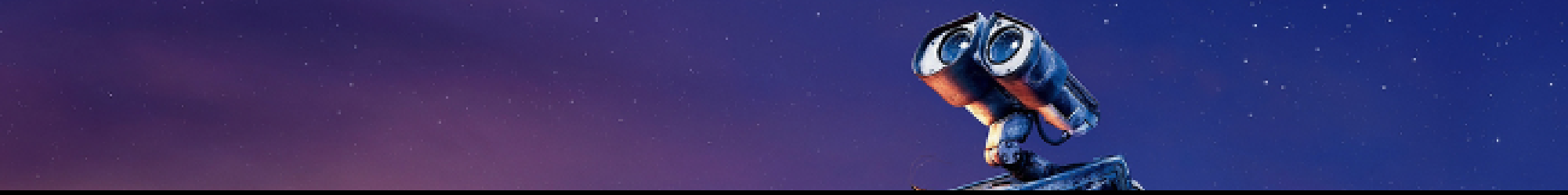


UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Introduction

RO51 - Introduction to Mobile Robotics

Zhi Yan

March 13, 2024

https://yzrobot.github.io/

www.utbm.fr

utbm
université de technologie
Belfort-Montbéliard

Lecturers

Zhi Yan
Assistant Professor
Module Coordinator

Iaroslav Okunevich
Ph.D. Student
Teaching Assistant

Goal of this course:

- Provide an overview of problems and approaches in mobile robotics
- Hands-on experience

About the course

- Created in 2021, covering my 14+ years of experience in mobile robotics:
  - Lecture 1: Introduction
  - Lecture 2: Locomotion
  - Lecture 3: Kinematics
  - Lecture 4: Perception
  - Lecture 5: Learning
  - Lecture 6: Decision-making
  - Lecture 7: Planning
  - Lecture 8: Control
  - Lecture 9: Navigation (part I)
  - Lecture 10: Navigation (part II)
  - Lecture 11: System Integration
  - Lecture 12: Multi-robot Systems
- The courseware is open to the public:

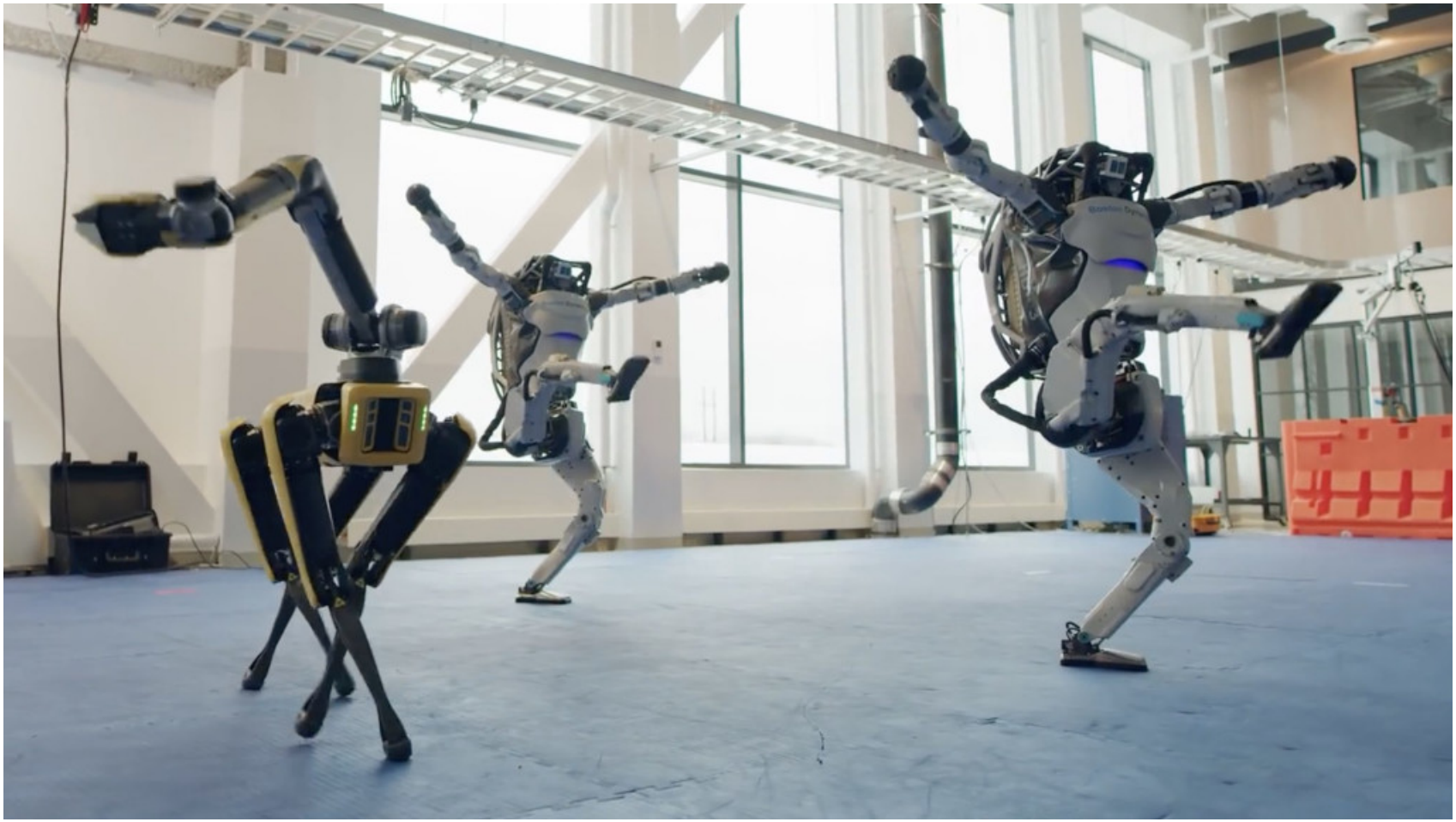
  `https://yzrobot.github.io/introduction_to_mobile_robotics/`

About the course

- CM, TD, TP: face-to-face, in English
  - CM: fundamentals
  - TD: getting familiar with ROS
  - TP: shared with RO50
- Attendance confirmation: No (study for yourself)
- Following the course requires some math and programming foundation
- Mode of examination: midterm exam (handwriting or programming) + final project (shared with RO50)

Mobile Robotics Today

Represented by YouTube celebrity Boston Dynamics:



What is a robot?

Origin of Words

The word **robot** was first used in 1921 by Czech author Karel Capek in his science fiction play R.U.R. (Rossum's Universal Robots). It comes from the Czech word "robota", which means "forced labor".

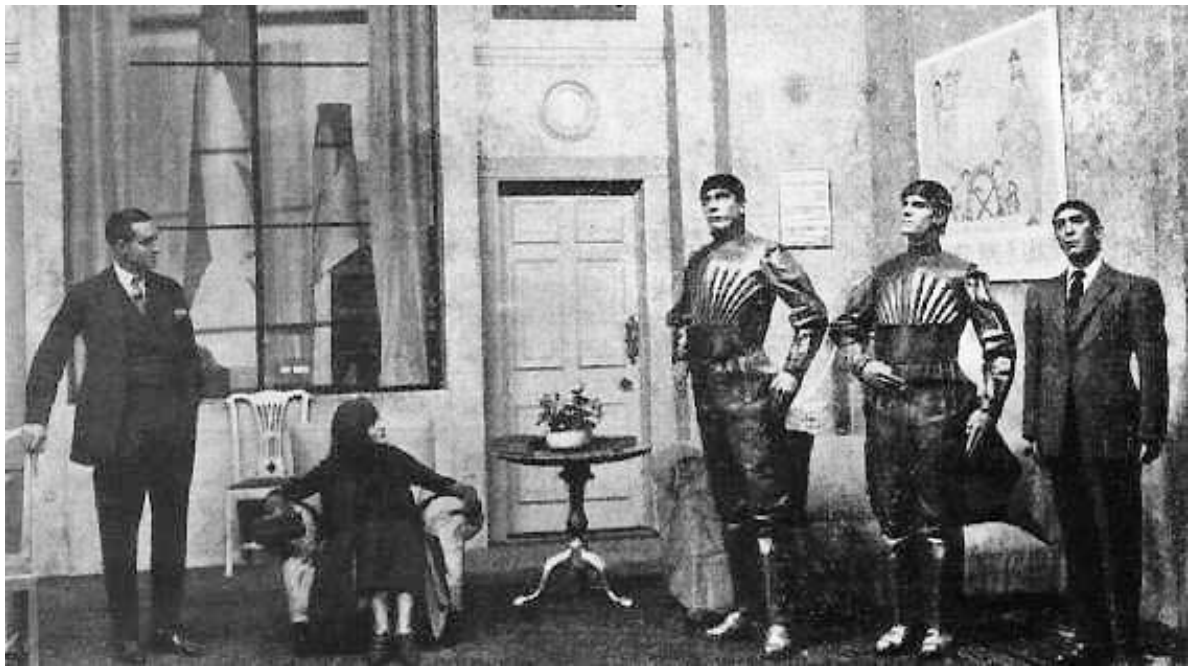


Figure: A scene from the play, showing three robots.

What is a robot?

Origin of Words

According to the Oxford Dictionary, a robot is a machine that can perform a complicated series of tasks by itself.

- Definition reflecting to some extent people's expectations for robot autonomy.
- Existing robots have a wider range of manifestations, including but not limited to, fully autonomous, semi-autonomous, and remote controlled.
- The impact of its introduction into the human society on the labor market: a supplement while at the same time generating competition.
- The ethical issues: compliant human ethics, military use, etc.

Industrial and service robots

According to the International Federation of Robotics (IFR), existing robots can be divided into two categories:

- **Industrial robot**: an automatically controlled reprogrammable multipurpose manipulator, programmable in three or more axes.
- **Service robot**: a robot that performs useful tasks for humans or equipment excluding industrial automation applications.

Industrial and service robots

Figure: Two different types of robots. Left: industrial robots, where the upper picture shows the first autonomous and programmable modern robot, the Unimate in 1961, and the lower picture shows the KUKA robot cluster deployed on the automobile production line in 1983. Right: HSR service robot manufactured by Toyota in 2017.

Industrial and service robots

- The first modern robot can be traced back to 1961: Unimate (an industrial robotic arm).
- Industrial robots have been deployed on a larger scale and more widely used than service robots.
- The situation is related to the complexity of the problem the robot addresses.
  - **Robotic arms**: work in constrained workspaces, have absolute measurements of position, not really need to perceive the world around them.
  - **service robots**: operate in unconstrained environments, need external detection to determine position and avoid obstacles.

Industrial and service robots

- Industrial robotic arms:
  - Save humans from heavy and repetitive tasks.
  - Improve production efficiency and quality.
- Service robots (are placed more expectations):
  - Highly repetitive, risky or unpleasant tasks in a variety of sectors: agriculture, construction, transport, healthcare, firefighting or cleaning services, etc.
  - Something to think about and discuss: *should robots in the future be (over) anthropomorphic (whether in terms of hardware or software design)?*

Asimov's Three Laws

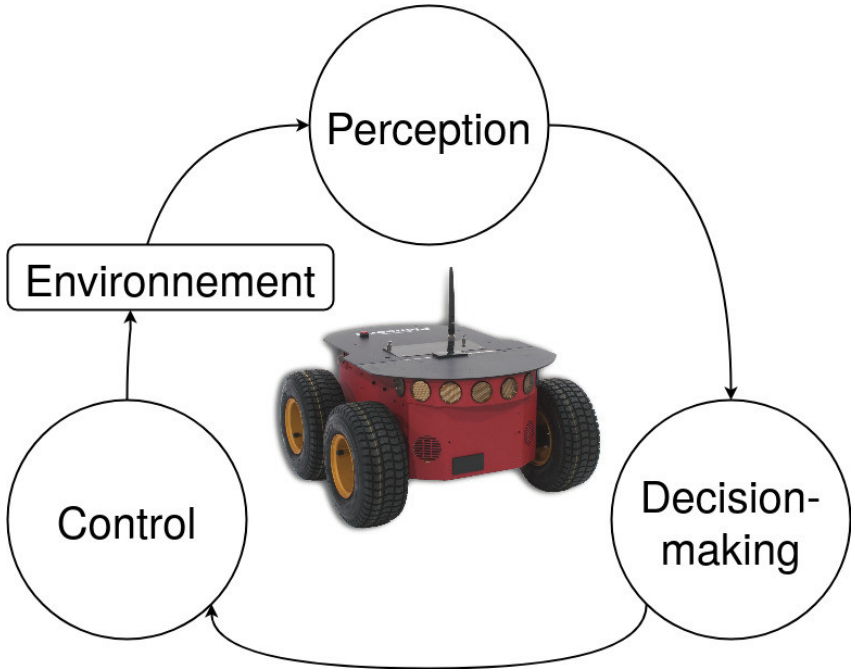
1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

— *short story "Runaround", 1942, by Isaac Asimov*

Service robots

Operational definition

- The "sense-think-act" paradigm: perception, decision-making, and control.
- Interaction with the environment in this paradigm can be regarded as an abstract imitation of human activities (I feel (see/smell/listen/touch/etc.), I think, and I react).



Service robots

Operational definition

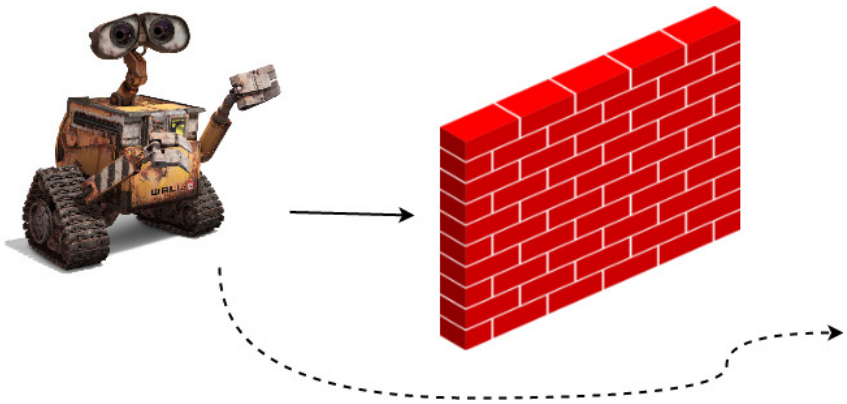
- **Perception**: what looks like around me?
- **Decision-making**: what should I do?
- **Control**: how do I achieve the action?

Service robots

An example

When a robot encounters an obstacle while moving autonomously, it:

- **senses (perception)**: there is an obstacle (i.e. a wall) in front of me;
- **thinks (decision-making)**: turn slightly to the right to avoid and go around the obstacle; and ultimately,
- **acts (control)**: adjust the speed of the wheels.



Service robots

Current trends

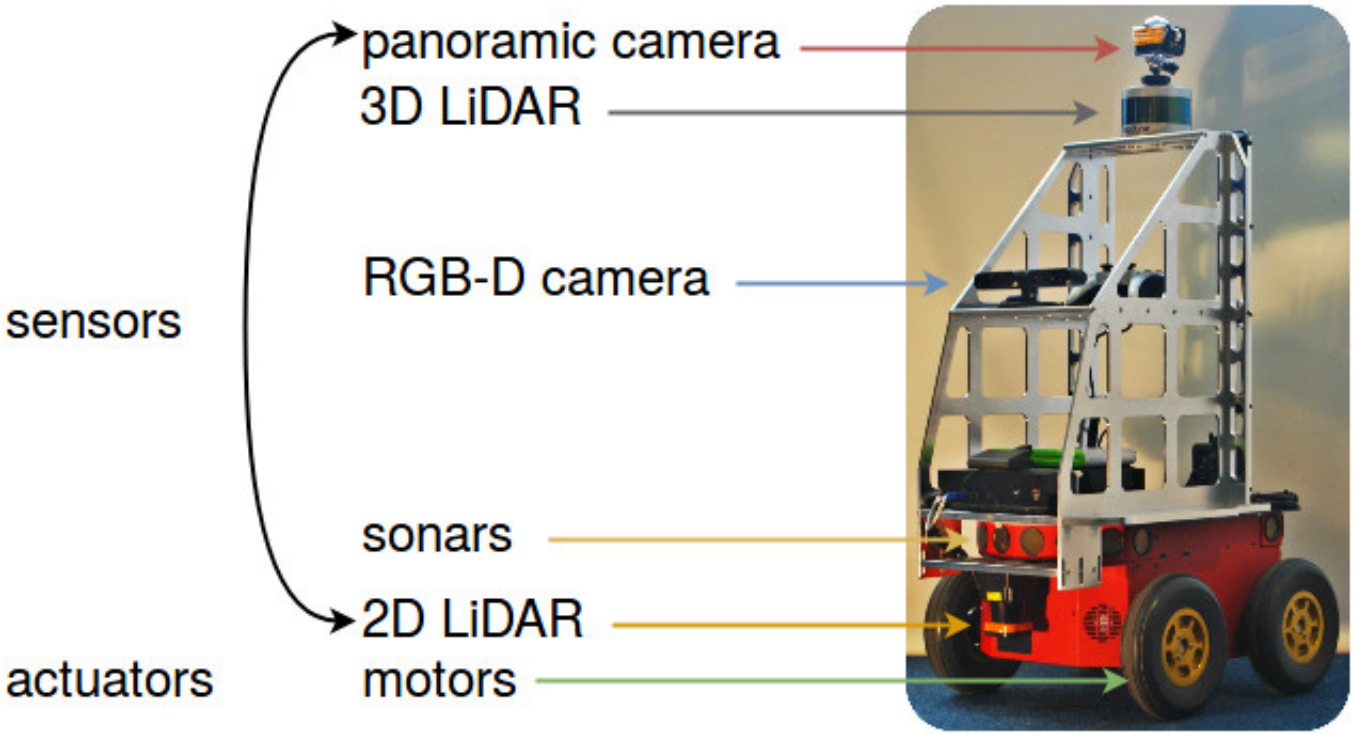
- Entertainment, toys
- Personal services
- Medical, surgery
- Industrial automation
- Hazardous environments
- Self-driving cars
- etc.

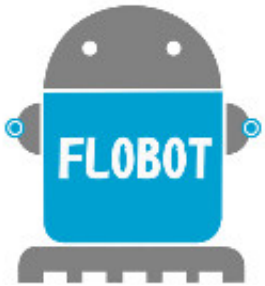
Mobile robotics

- Mobile robotics is concerned with a fundamental question of service robots, namely, how to move in the physical world.
- Mobile robots typically have to answer three fundamental questions:
  - Where am I?
  - Where am I going?
  - How to get there?
- To answer these questions the robot must:
  - take measurements,
  - model the environment,
  - locate itself,
  - and plan a path to its goal location.

Mobile robotics

A typical modern mobile robot





ROS

Since our TD and TP will be based on ROS, we need to briefly introduce some of its basic knowledge first.

- ROS: **R**obot **O**perating **S**ystem
- In 2010, ROS became the de facto standard for robotics software.

ROS

What is ROS?

- A meta-operating system for robots, something between the operating system and the middleware
- A distributed architecture for inter-process and inter-machine communication and configuration
- A collection of software packages and building tools
- A set of development tools for system execution and data analysis

What is ROS not?

- A real operating system
- A programming language
- A programming environment (like IDE)
- A hard real-time architecture (ROS2 will be)

History of ROS

- **In 2007**, ROS was initially developed by the *Stanford Artificial Intelligence Laboratory*.
- **From 2008 to 2013**, development was carried out mainly by the laboratory *Willow Garage*.
- **Since 2013**, managed by *Open Source Robotics Foundation (later became Open Robotics in 2017)*.

History of ROS

- 13 versions have been released since 2010.
- Naming convention: alphabetical (starting with B because there was an unofficial version of the initial release).
- Noetic Ninjemys: last ROS 1 release, May 2020 - May 2025.
- **Our working version** : Noetic with Ubuntu 20.04 (Focal)

ROS philosophy

- **Reuse**: The main goal of ROS is to support code reuse in robotics research and development.
- **Peer-to-peer**: Individual programs communicate over defined API (*Application Programming Interface, such as ROS messages, services, etc.*).
- **Distributed**: Programs can be run on multiple computers and communicate over the network.
- **Multi-language support**: ROS modules can be written in any programming language for which a client library exists (C++,. Python, MATLAB, Java, etc.).
- **Light-weight**: Stand-alone libraries are wrapped around with a thin ROS layer.
- **Free and open-source**: Most ROS software is open-source and free to use.

ROS concepts

The ROS architecture has been designed and divided into three sections or levels of concepts:

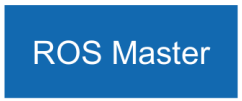
- **The Filesystem level**: A group of concepts are used to explain how ROS is internally formed, the folder structure, and the minimum number of files that it needs to work.
- **The Computation Graph level**: All the concepts and mechanisms that ROS has to set up systems, handle all the processes, and communicate with more than a single computer, etc.
- **The Community level**: A set of tools and concepts to share knowledge, algorithms, and code between developers.

ROS computation graph

ROS master[1]:

- Manage the communication between *nodes*.

Start a *master* with: `$ roscore`



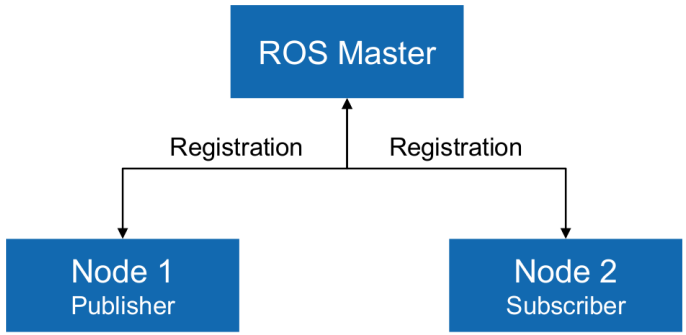
[1] There is no master/roscore in ROS 2.

ROS computation graph

ROS nodes:

- Every *node* registers itself at startup with the *master*.
- Single-purpose, executable program.
- Individually compiled, executed and managed, organized in *packages*.

Launch a *node* with: `$ rosrun package_name node_name`

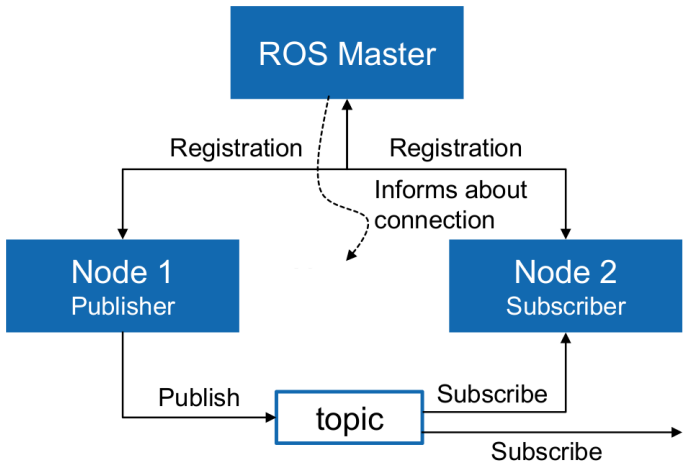


ROS computation graph

ROS topics:

- The *topic* is a name used to identify the content of *messages*.
- *Nodes* communicate with each other over *topics*.
- A *node* can publish or subscribe to *topics*.

List currently active *topics* with: `$ rostopic list`

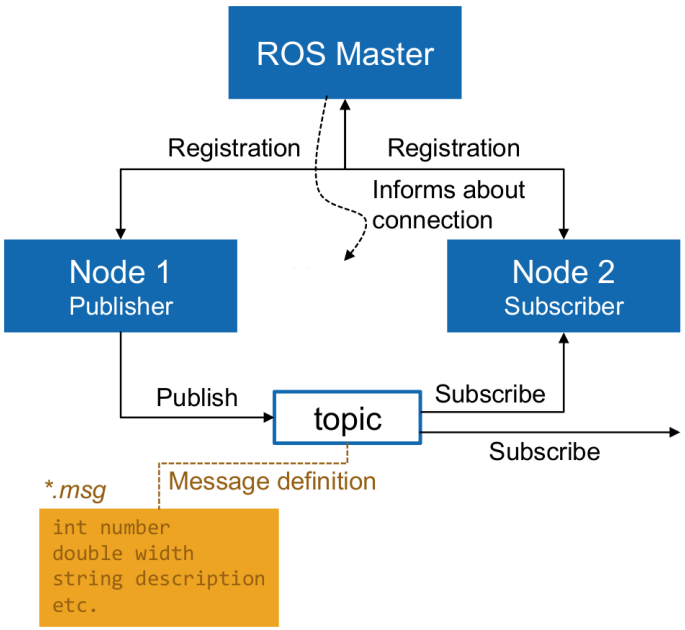


ROS computation graph

ROS messages:

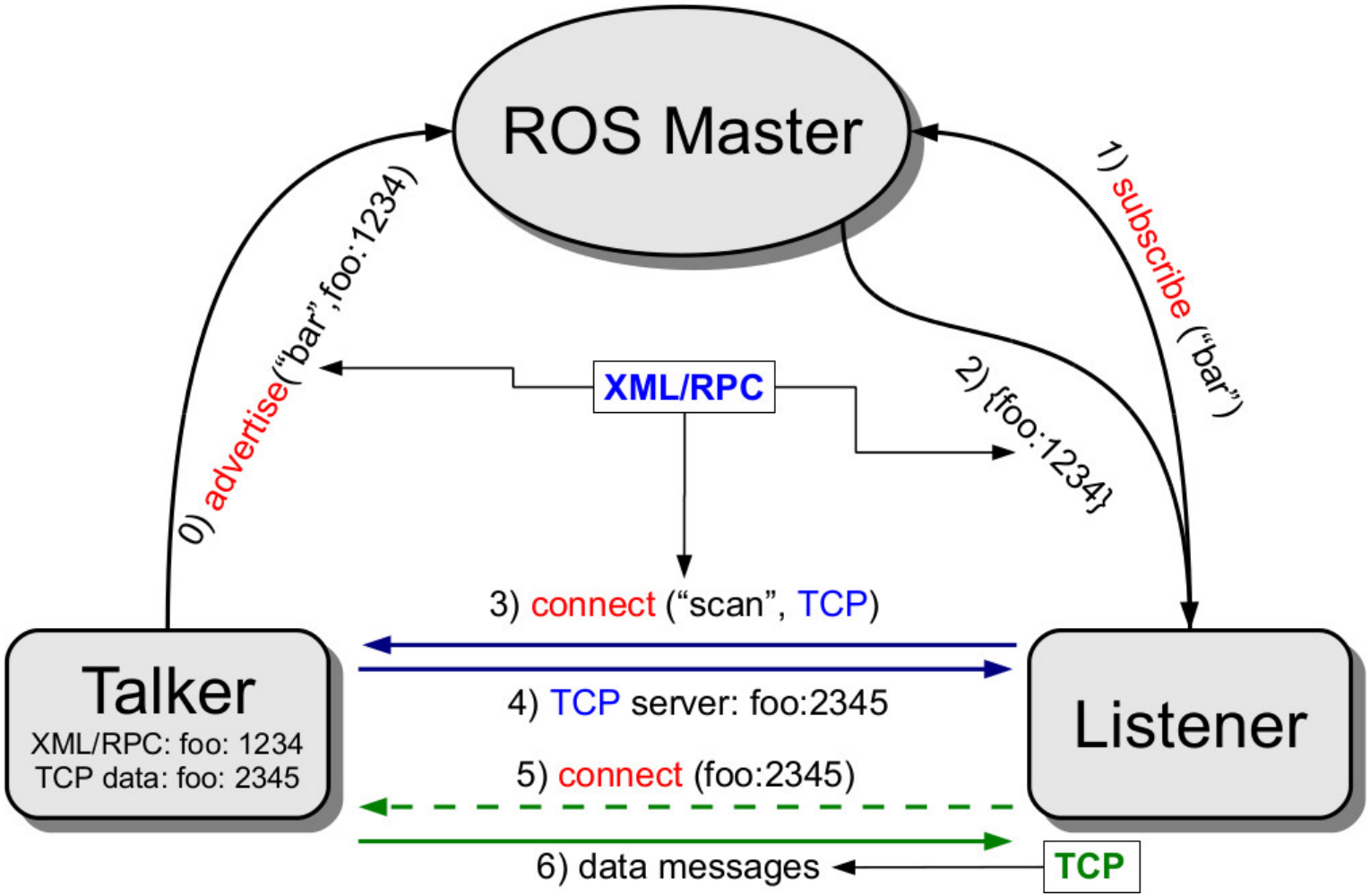
- A simple data structure, comprising typed fields.
- Standard primitive types (integer, float, etc.) are supported.
- Can include arbitrarily nested structures and arrays.

Show *message* content with: `$ rostopic echo topic_name`



ROS computation graph

A closer look at ROS topics (with communication details):

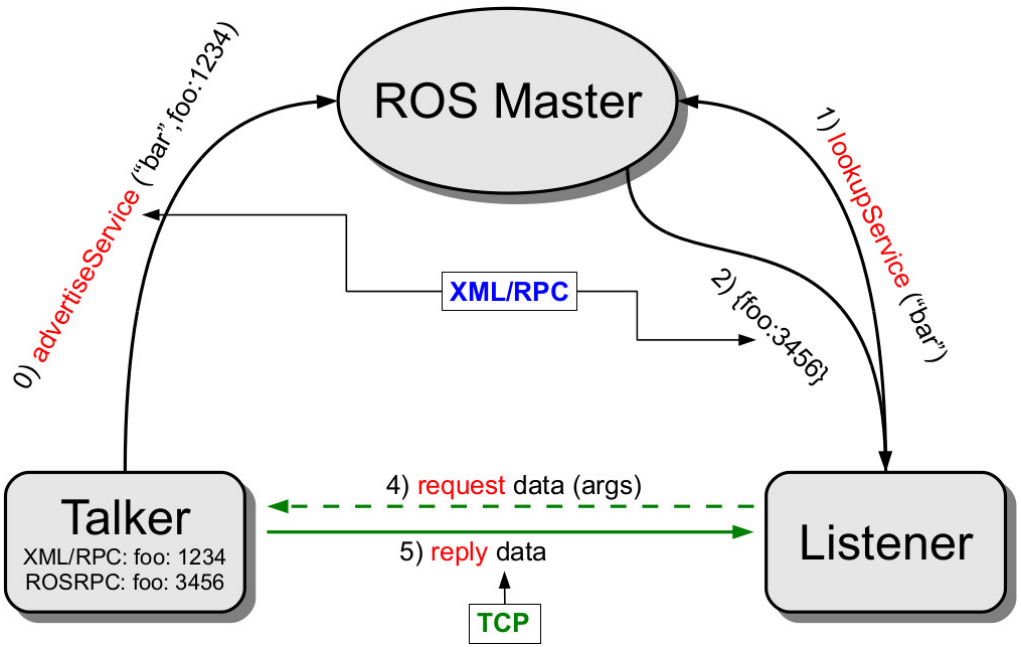


ROS computation graph

ROS services:

- Designed for RPC request / reply interactions.
- Implemented by a pair of *messages*: one for the request and one for the reply.

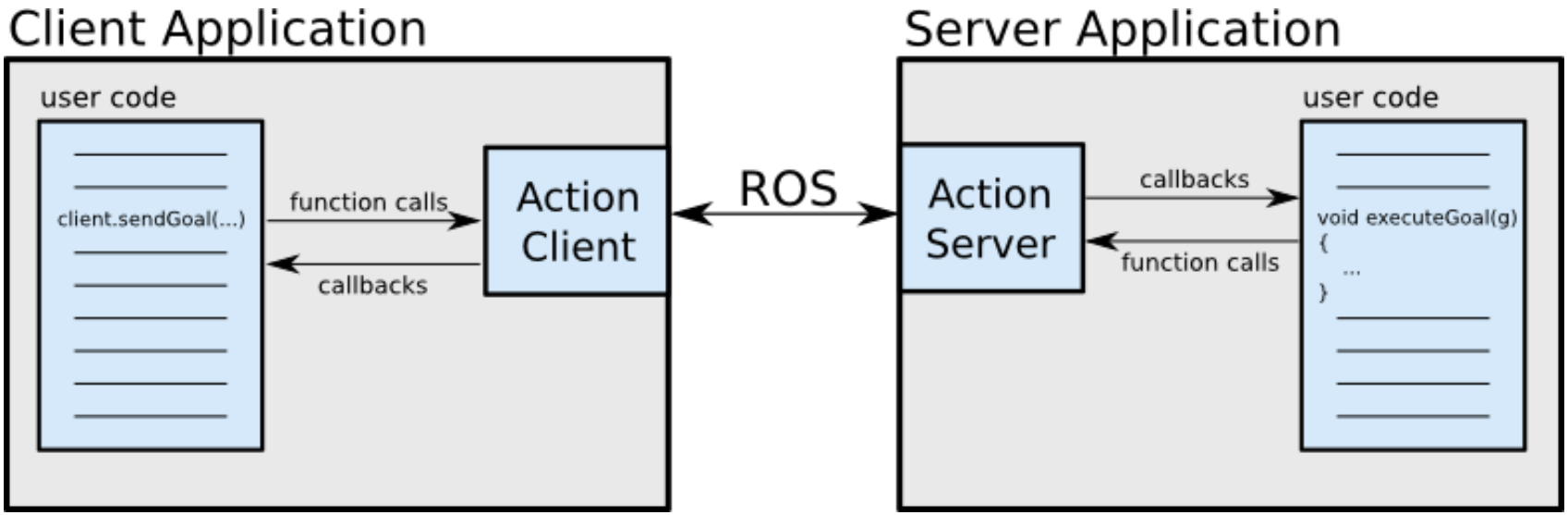
List currently active *services* with: `$ rosservice list`



ROS computation graph

ROS actions (i.e. *actionlib*):

- For services that take a long time to execute, *actionlib* allows the user to cancel the request during execution or get periodic feedback on the progress of the request.
- Different from getting "quick" response through ROS *services*.

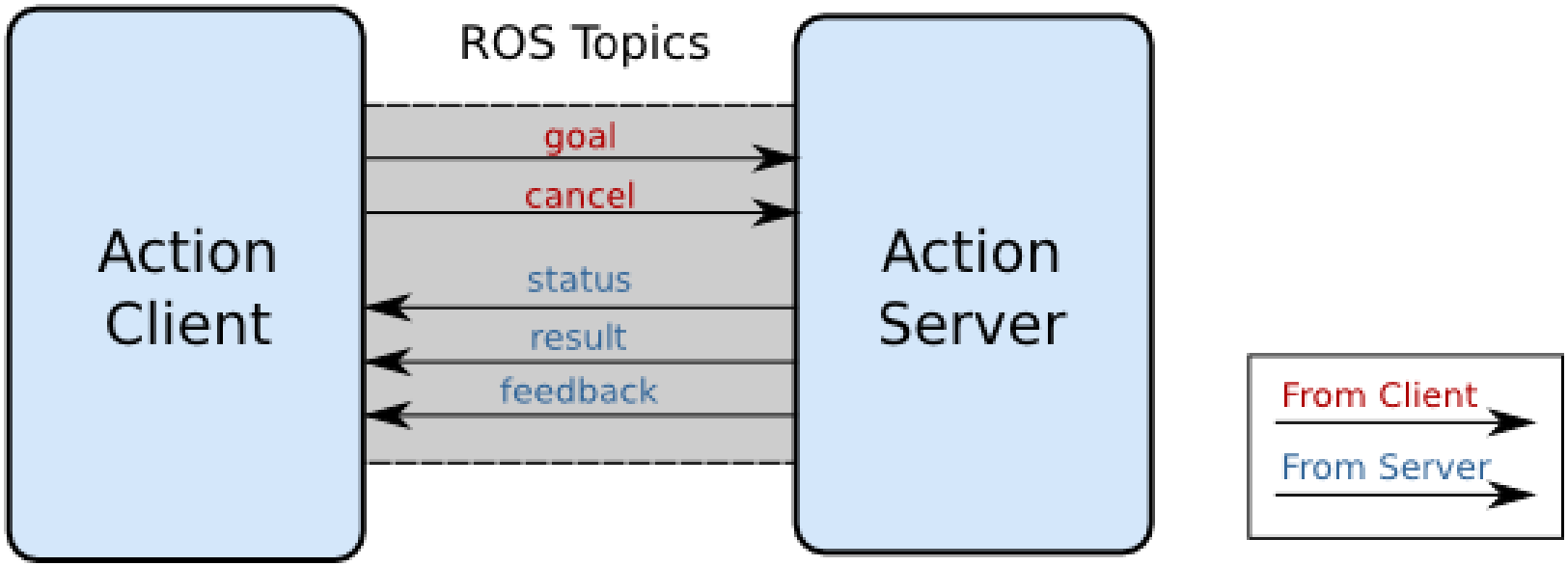


ROS computation graph

ROS actions (i.e. *actionlib*):

- *Action protocol* relies on ROS topics to transport messages.
- Default defined message types: *Goal*, *Feedback*, and *Result*.

Action Interface



ROS2

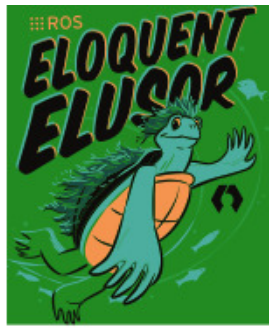
- As for now ROS is not very popular in the industry, and lacks some of the most important requirements, such as real-time, safety, certification, security.
- One of the goals for ROS2 is to make it compatible with industrial applications.

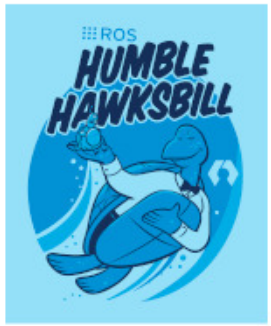
ROS2

- Active years: 2014 - present
- Under very active development, release cycle: twice a year (unlike every 1 to 2 years for mature ROS 1)
- Does not break ROS, nor does it rollout into ROS
- Breaking API with ROS, but conceptually very similar
- Building on DDS (Data Distribution Service (for Real-Time Systems))
- Interoperating with ROS (by *ros1_bridge*)

ROS2

- 13 versions (4 betas, 9 releases) have been released since 2015.
- Naming convention: same as ROS
- Iron Irwini: the latest version (for Ubuntu 22.04 (Jammy), Windows 10 (Visual Studio 2019))





YZRobot

- https://github.com/yzrobot
- https://www.youtube.com/@yzrobot

The end

Thank you for your attention!

Any questions?