Introduction ROS

Master Robotique

Laurent Lequièvre
Juan Antonio Corrales Ramon

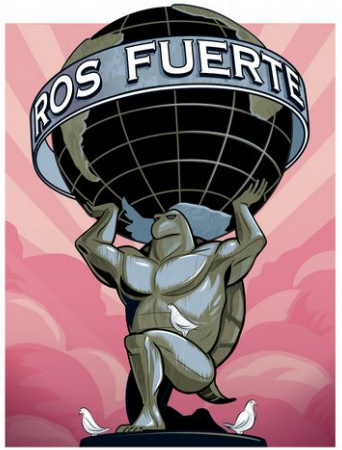
Index

INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

INTRODUCTION

Definition of ROS

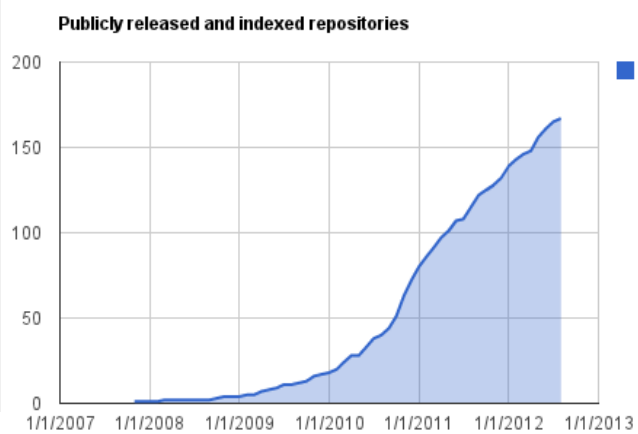
- ROS (Robot Operating System) is a **software platform** that is able to build and execute code between several computers and several robots.
- It provides services similar to an **operating system** for working with robots:
  - Hardware abstraction
  - Device low level control
  - Message-based communication
  - Commands and utilities
  - Package management



INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

History of ROS

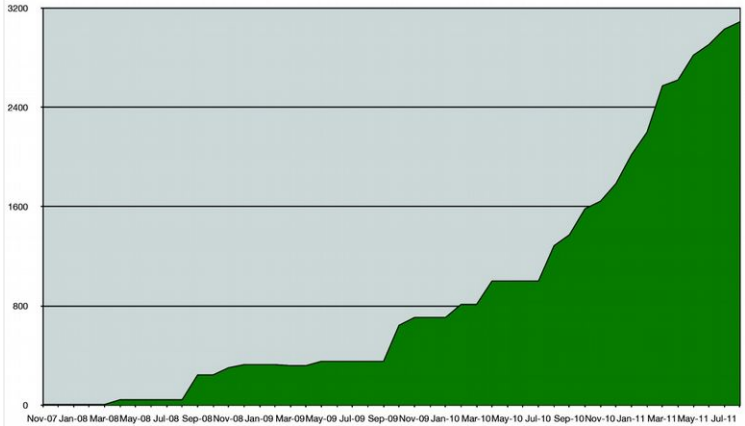
- Initially developed by **Standford University** in the "STAIR" project in 2007.
- From 2008 to 2013, its development continued mainly due to the contributions of **Willow Garage** for their humanoid robot PR2 and the scientific community.
- From 2013, ROS depends on the **Open Source Robotics Foundation** and its development continues.
- Exponential increase in the number of repositories (more than 170 in 2013).



INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Robots in ROS

- Exponential increase in the number of packages (more than 3500 in 2012).



- Use in more than 50 robots: http://wiki.ros.org/Robots

https://www.youtube.com/watch?v=PGaXiLZD2KQ


Fraunhofer IPA Care-O-bot


Videre Erratic


TurtleBot


Aldebaran Nao


Lego NXT


Shadow Robot


Willow Garage PR2


iRobot Roomba

Robotnik Guardian


Merlin miabotPro


AscTec Quadrotor


CoroWare Corobot


Clearpath Robotics Husky


Clearpath Robotics Kingfisher


Festo Didactic Robotino


Gostai Jazz


Neobotix mp-500


Neobotix mpo-500


Neobotix mpo-700


ROS-Industrial

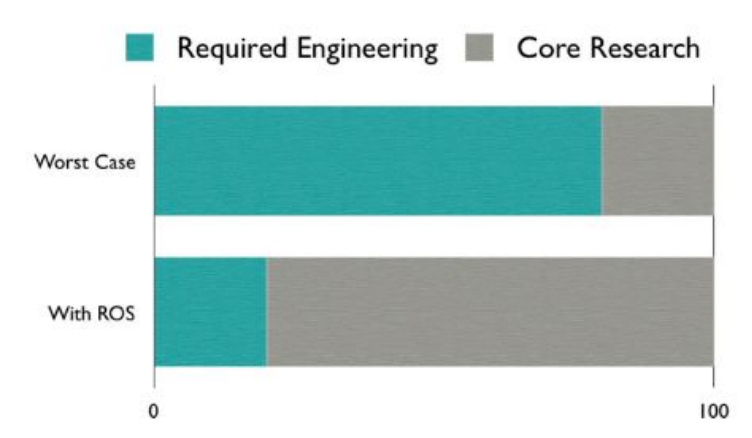
INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

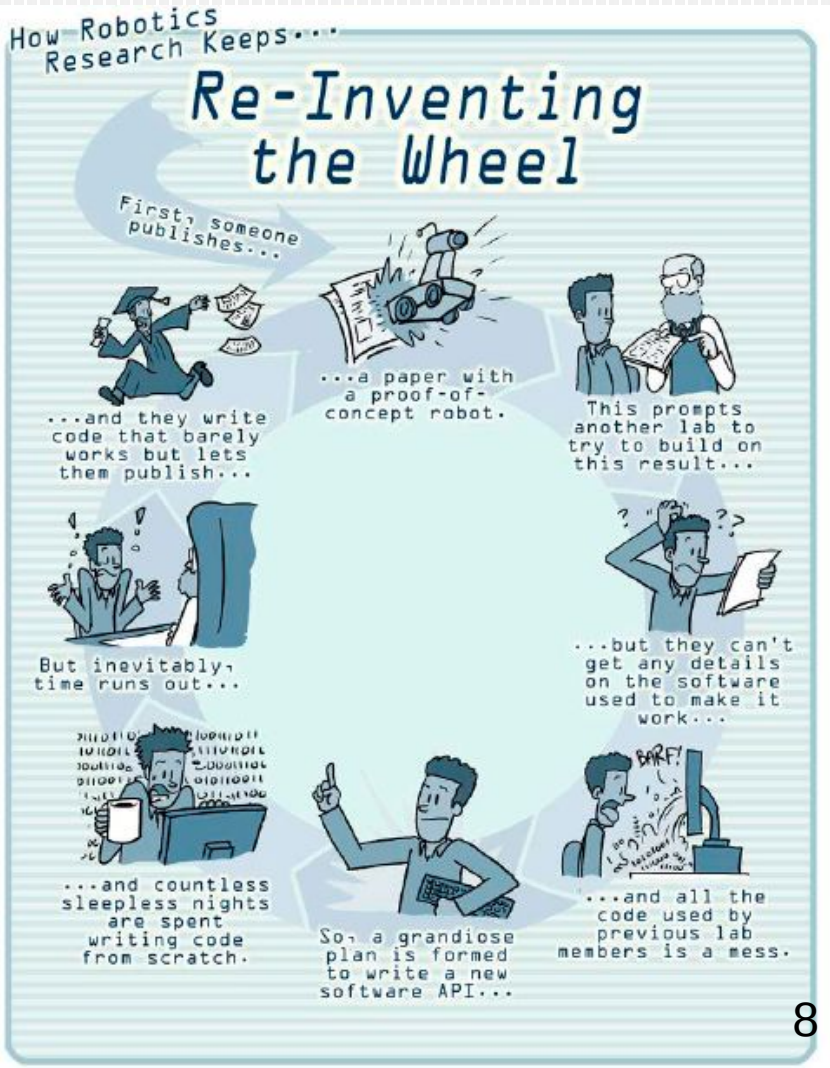
Advantages

- The main features of ROS are :
  - **Peer-to-peer:** It consists of a group of processes, that can be in different systems, are connected between them in a peer-to-peer topology.
  - **Multi-language:** ROS packages can be implemented in different programming languages (C++, Python, Octave y LISP). ROS messages are defined in a neutral lenguage (IDL) that enables the automatic generation of code in different languages.
  - **Tools-Based**: Micro-kernel design with a high number of small tools. Bigger stability and better complexity management are possible.
  - **Light:** The ROS system is built in a modular way. All complexity is moved towards external libraries: easy importation and exportation.
  - **Open-source:** All source is available and most packages have BSD licence (commercial and non-commercial projects).

Advantages

- Main goal of ROS:
  - Reusing code and comparing results
  - Having more time for research
  - Less time for re-implementation
  - Current Platform: > 150 People-Year.





INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Applications

- Main applications of ROS in robotics:
  - Visualization and simulation: **rviz**, stage (2D), **gazebo** (3D).
  - Drivers: camera_drivers, laser_drivers, imu_drivers.
  - 3D processing: perception_pcl (**PCL**), laser_pipeline.
  - Image processing (2D): vision_opencv (**OpenCV**), visp.
  - Transformations: tf, tf_conversions.
  - Navigation (odometry, ego-motion, SLAM): navigation.
  - Controllers (position, force, speed, transmissions): ros_control.
  - Robot modelling: urdf (XML description of robots).
  - Motion planning: **MoveIt!** (library OMPL).
  - Grasping and manipulation: GraspIt!, OpenRAVE.

INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Index

- Introduction
  - Definition
  - History
  - Robots
  - Advantages
  - Applications
- Basic components
  - Nodes
  - Topics
  - Services

INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

BASIC COMPONENTS

ROS Levels

- The basic concepts of ROS can be analyzed from 3 different levels:
  - **File-system level:** Elements that are in the hard-drive
    - Packages, manifests, meta-packages (stacks), message types (msg) and service types (srv).
  - **Execution level:** ROS process that treat data in a peer-to-peer architecture
    - Nodes, master, parameter server, messages, topics, services and bags.
  - **Community level:** Resources for ROS (software and documentation) that are shared between different groups of users.
    - Distributions, repositories, ROS Wiki, mailing lists, ROS Answers and ROS Blog.

Packages

- Packages are the **basic unit** that organize software in ROS.
- They can contain: processes (nodes), libraries, data, configuration files, etc.
- It is a folder inside ROS_ROOT or **ROS_PACKAGE_PATH** that contains a file **package.xml**
- Packages usually have the same structure:
  include/package_name: headers for libraries C++.
  msg/: Types of messages (msg).
  src/package_name/: Source code.
  srv/: Types of services (srv).
  scripts/: executable scripts (normally in Python).
  CMakeLists.txt: Cmake file, necessary for compilation of the package.
  package.xml: File with XML specification of package meta-data.
  - Package description, author and licence.
  - <build_depend>/<run_depend>: Dependencies of other packages (in compilation/execution).

- Command **rospack** (options 'find, depends, list') and command **roscd**.

INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Workspace

- It is a folder for working with packages: modify, compile and install them.
- It is composed by 4 spaces:
  - **src:** source code files (cpp/py) of packages
  - **build:** intermediate files of CMake
  - **devel:** output files obtained from compilation (executable, libraries...)
  - **install:** installation of generated files
- Automatic generation of code: **catkin**
- Creation of catkin workspace:
  1. mkdir -p ~/catkin_ws/src
  2. cd ~/catkin_ws/src
  3. catkin_init_workspace 4. cd ~/catkin_ws/
  5. catkin_make 6. source devel/setup.bash

```
workspace_folder/        -- CATKIN WORKSPACE
  src/                   -- SOURCE SPACE
    CMakeLists.txt       -- The 'toplevel' CMake file
    package_1/
      CmakeLists.txt     -- CMake file for package_1
      package.xml
      ...
    package_n/
      CMakeLists.txt
      package.xml
      ...
  build/                 -- BUILD SPACE
  devel/                 -- DEVELOPMENT SPACE
    bin/
    etc/
    include/
    lib/
    share/
    setup.bash
    ...
  install/               -- INSTALL SPACE
```

INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Types of Messages

- ROS uses a text file in the IDL language for describing **data types** (messages) that are publish by ROS nodes.
- This description is stored in **.msg files** inside the **msg/ subfolder** of a ROS package.
- There are two parts in a .msg file:

  Fields (Data types that are sent inside the message):
  - Field: data type + name. Example: int32 x
  - Data types:
    - Basic: bool, int32, float32, float 64, string, time, duration, etc.
    - Arrays.
    - Other messages. Example: geometry_msgs/PoseStamped.
    - Header: ID, timestamp, frame ID.

  Constants (Values for interpreting the fields):
  - Constant: Type_constant name_constant= constant_value. Example: int32 X=123.
- Automatic generation of message through CmakeLists.txt and package.xml

INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Types of Services

- ROS uses one text file in a descriptive language for indicating data types of the **request/response** of a service.

- This description is stored in **ficheros .srv files** inside the **srv/ subfolder** of a ROS package.
- There are two parts in a .srv file, separated by a line containing "---":
  - Request
  - Response

```
#request
int8 foobar
another_pkg/AnotherMessage
---
#response
another_pkg/YetAnotherMessage
val uint32 an_integer
```

- Automatic generation of services through CmakeLists.txt and package.xml

INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Nodes

- A node is a **process** that executes a computation task.
- Nodes **communicate between them** through: services, topics and parameters.
- ROS is composed in the execution level by a set of nodes that communicate between them and that can be distributed in different machines.
- Nodes are programmed through client libraries: roscpp, rospy y roslisp.
- Command **rosnode** (options 'list, info, kill, ping') and command **rosrun** (ex: turtlesim).

Master

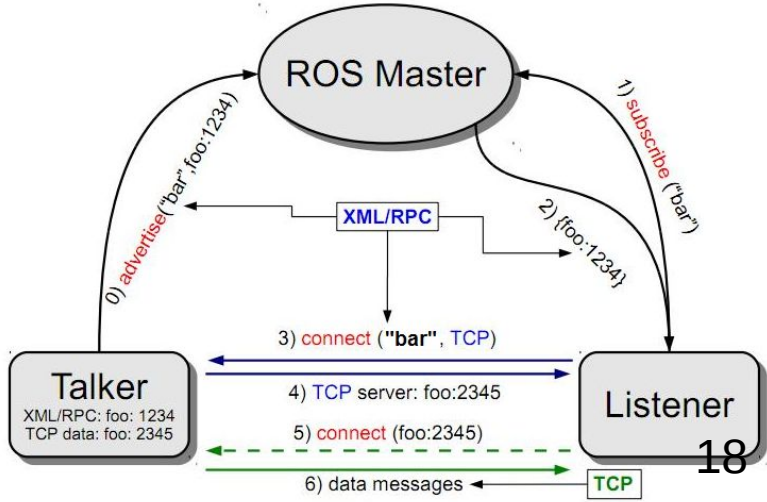
- Program that enables localization between ROS nodes (similar to DNS).
- Once nodes are localized, the master does not participate and there is peer-to-peer communication between them.
- **It registers topics and services** of every node.
- It is initialized by the command **roscore**.

INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Topics

- A topic is a **bus with a name** through which nodes interchange messages.
- It is a method of **asynchronous communication** between the nodes:
  - **Publishers**: Nodes that publish messages in a topic.
  - **Subscribers**: Nodes that receive messages through a topic.
  - There can be several publishers and subscribers for a same topic.

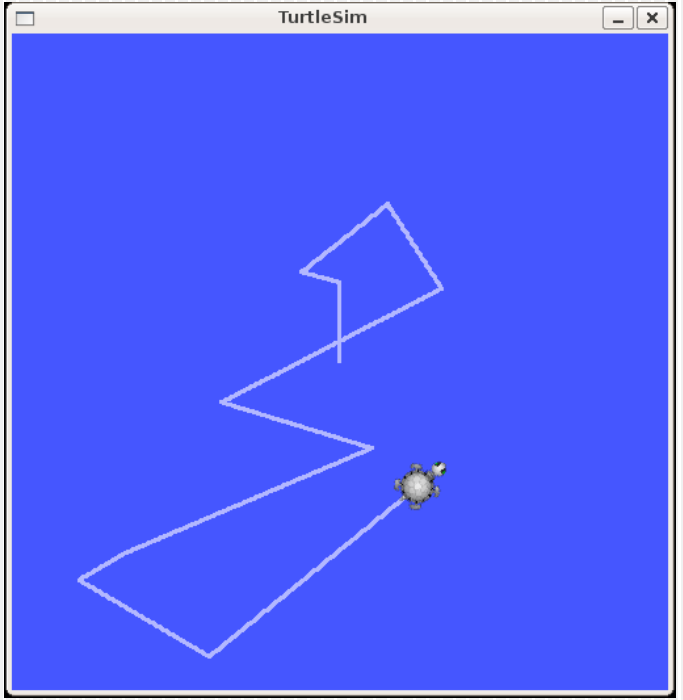
- The initial connection between subscribers and publishers is done through the Master. Later, their communication is peer-to-peer through TCP:
  1. The subscriber registers the topic in the Master.
  2. The publisher asks the topic to the Master.
  3. The master gives the URI of the publisher to the subscriber.
  4. The subscriber asks the publisher for a topic connection.
  5. The publisher informs about the TCP setup.
  6. The subscriber connects to the TCP data port.
  7. Bidirectional communication of data through TCP is done.



INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Topics

- Example turtlesim:
  - Execute publisher: rosrun turtlesim turtle_teleop_key
  - Execute subscriber: rosrun turtlesim turtlesim_node
  - Communication through topic /turtle1/command_velocity
  - Command **rqt_graph** to see execution graph (nodes+topics):



  - Command **rostopic** (options: list, echo, type, pub):
    Examples: **rostopic echo /turtle1/cmd_vel ; rostopic type /turtle1/cmd_vel**
    rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
    rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'

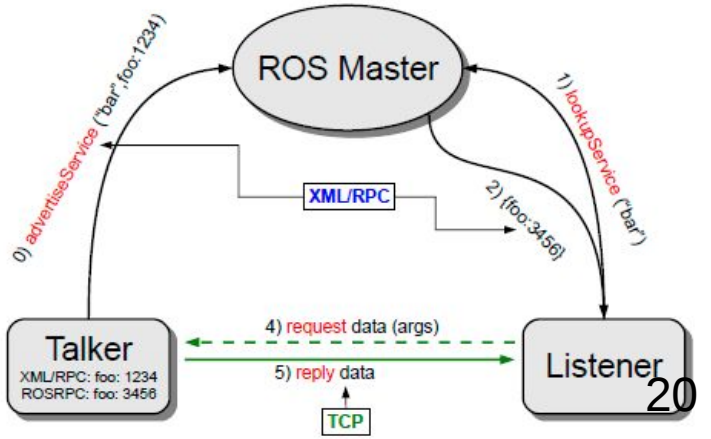
INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Services

- A service enables a node to send a request (message **request**) and receive a response (message **response**) from other node.

- It s a method of **synchronous communication** between the nodes (similar to RPC):
  - **Provider**: Node that provides the service (it receives request and sends response).
  - **Client**: Node that asks for the service (it sends request and receives response).

- The initial connection between providers and clients is done through the Master. Later, the communication is peer-to-peer through TCP:
  1. The provider registers the service in the Master.
  2. The client asks the Master for the service.
  3. The master informs the client about the TCP setup.
  4. The client sends the request message to the provider.
  5. The provider sends the response message to the client.



Services

- Command **rosservice** (Examples with turtlesim):
  - list: List of active services.
  - call: Execute the servie with the indicated parameters.
  - type: Print the type of service (types request/response).
  - find: Find the service.
  - Examples: **rosservice call clear ; rosservice call spawn 2 2 0.2 ""**

- Command **rossrv** (Examples with turtlesim):
  - show: Show the types of the request y response messages of the service.
  - package: List the types of services defined in the srv folder of the package.
  - Examples: **rosservice type spawn | rossrv show**

- Command **rosmsg** (Examples with turtlesim):
  - show: Show the fields of the message.
  - package: List the types of the messages defined in the msg folder of the package.

INSTITUT PASCAL
sciences de l'ingénierie et des systèmes

Concepts of ROS community

- **Distribution:** Group of packages of a version. It is similar to Linux distributions and makes easy software integration (compatible libraries) and system stability (bugs management).
- **Repository:** Web server with ROS packages generated by the same institution. They are organised in a network where each institution keeps its own repository.
- **ROS Wiki:** Website with documentation and tutorials. It is the main source of ROS information (http://www.ros.org/wiki/).
- **Mailing lists:** Mails with news (https://code.ros.org/mailman/listinfo/ros-users).
- **ROS Answers:** Forum where users ask and answer questions (http://answers.ros.org).

INSTITUT PASCAL
sciences de l'ingénierie et des systèmes