Introduction to ROS

Kurbakov Dmytro, Slonina Michal

11.10.2019

All slides are available here:
https://github.com/project-omicron/robocar/

History of ROS

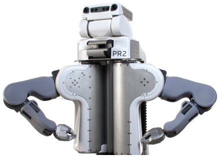
Early years of ROS, ... - 2007

No single solution how to program robots

Eric Berger and Keenan Wyrobek, PhD students at Stanford, build PR1 (Personal Robot One) and began to work on software from it, borrowing the best practices from other early open source robotic software frameworks

Early funding of US$50,000 was provided by Joanna Hoffman and Alain Rossmann, which supported the development of the PR1

Early years of ROS, 2007 - 2013



PR2 was introduced

Introduction of many packages

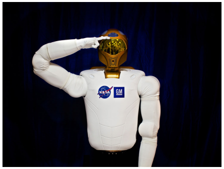
First RVIZ documentation, first paper on ROS

Initiation of the ROS.org website

Release of ROS 1.0, in January 2010

Creating the Open Source Robotics Foundation

Early years of ROS, 2013 - now



A new version of ROS every year

ROSCons have occurred every year since 2012

Robotnaut 2: forst ROS based robot in space

ROS2 was released

Structure of ROS

What is a workspace in ROS

Workspace is the folder inside which you are going to be actively developing. Keeping things in a folder with connected development helps keep separation of development models

Catkin is a low-level build system macros and infrastructure for ROS

Catkin workspace is a folder where you modify, build, and install catkin packages

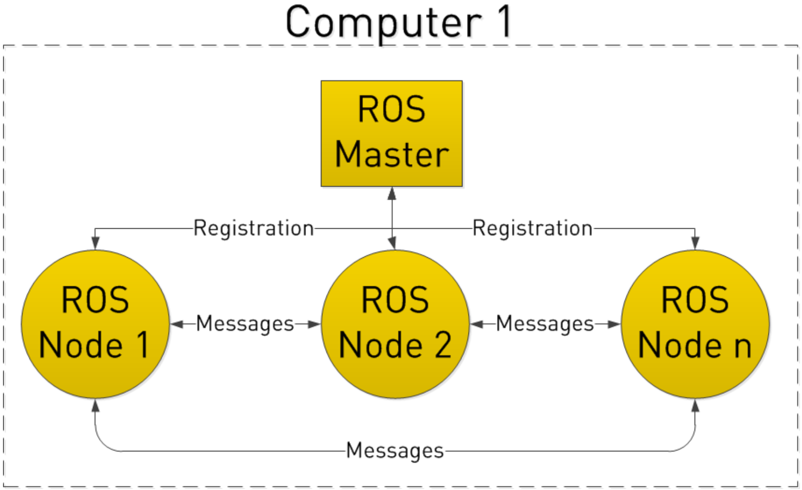
For more information, check: ROS catkin documantation

Introduction to catkin_init_workspace and catkin_make

catkin_init_workspace initializes a catkin workspace by creating a top level CMakeLists.txt

catkin_make is a convenience tool for building code in a catkin workspace. catkin_make follows the standard layout of a catkin workspace

Robot architecture in ROS, example



ROS Master

The ROS Master provides naming and registration services to the rest of the nodes in the ROS system.

- Tracks publishers and subscribers nodes
- Tracks topics
- Tracks services
- Provide parameter server
  Shared, multi-variate dictionary, accessible via network APIs.
  Nodes use this server to store and retrieve parameters at runtime.

Publisher vs. Subscriber

Publisher: Node that puts information to the topic

Subscriber: Node that checkes if the information arrives to the topic. Once the information arrived, it can react correspondingly

In ROS, every note can be a publisher, subscriber or both

For more info, check here

What is the topic in ROS

Topic is a named buse over which nodes exchange messages

Each topic is strongly typed by the ROS message type

ROS currently supports TCP/IP-based and UDP-based message transport

1. TCPROS is the default transport used in ROS
2. UDP-based transportis currently only supported in roscpp

Introduction to the command rostopic

rostopic is a command-line tool for interacting with ROS topics

Once you run the robot, you can see all available topics
rostopic list

Current value in the topic
rostopic echo /topic_name

Information about the frequenze of the topic
rostopic hz /topic_name

More information: ROS Topic

What is the service in ROS

Request/reply is done via a Service, which is defined by a pair of messages: one for the request and one for the reply.

Services are defined using srv files.

Generally saying: service is the RPC in ROS.

Introduction to the command rosservice

rosservice contains the rosservice command-line tool for listing and querying ROS Services

List all the services that are currently available
rosservice list

Print information about specified service *rosservice info /rosout*

Call a service from the command line
rosservice call /service_name service-args

For more information: ROS Service

MUX

mux is a ROS node that subscribes to a set of incoming topics and republishes incoming data from one of them to another topic

Example:
We use mux to switch from AI controller to human controller.

More information and examples: ROS mux

roscore and rosrun

roscore is a collection of nodes and programs that are pre-requisites of a ROS-based system. You **must** have a roscore running in order for ROS nodes to communicate.

rosrun allows you to run an executable in an arbitrary package from anywhere without having to give its full path or cd/roscd there first.

Example:
roscore
rosrun package node _parameter:=value

For more information, see ROS rosrun, ROS roscore

roslaunch

roslaunch is a tool for easily launching multiple ROS nodes locally and remotely via SSH, as well as setting parameters on the Parameter Server.

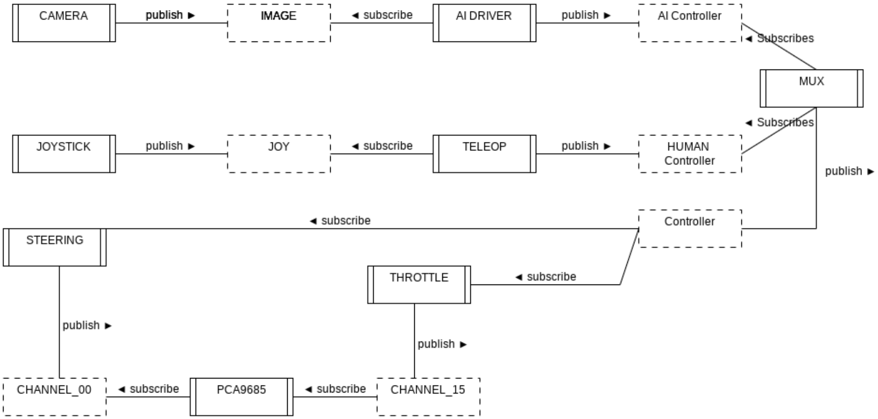
roslaunch takes in one or more XML configuration files (with the .launch extension) that specify the parameters to set and nodes to launch, as well as the machines that they should be run on.

Example:
roslaunch package_name file.launch

For more information, see ROS roslaunch

Project OMICRON

High level description of nodes



Q & A

Contacts

Kurbakov Dmytro
Email: dmytro.kurbakov@tomtom.com
LinkedIn: https://linkedin.com/in/dmytro-kurbakov/

Slonina Michal
Email: michal.slonina@tomtom.com
LinkedIn: https://www.linkedin.com/in/michalslonina/

What to know more? Useful links

The Origin Story of ROS, the Linux of Robotics
ROS History, ROS.org
ROS, wiki page
ROS Industrial ROS Tutorial for publisher and subscriber