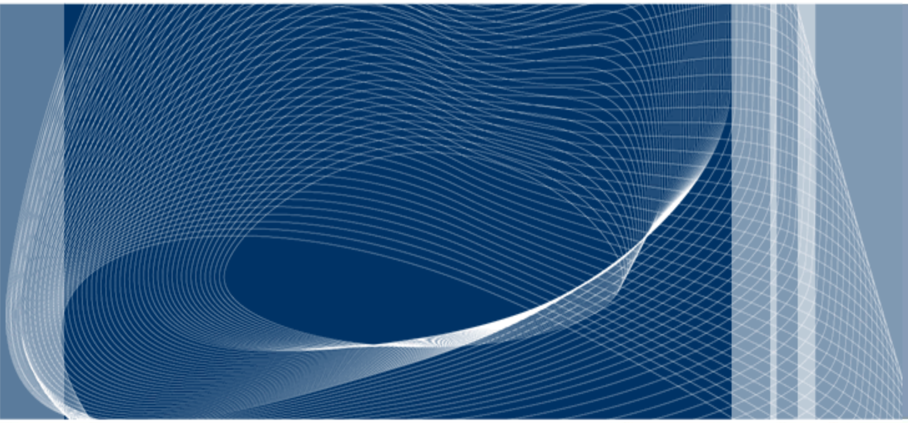


Cognitive Robotics – ROS Introduction

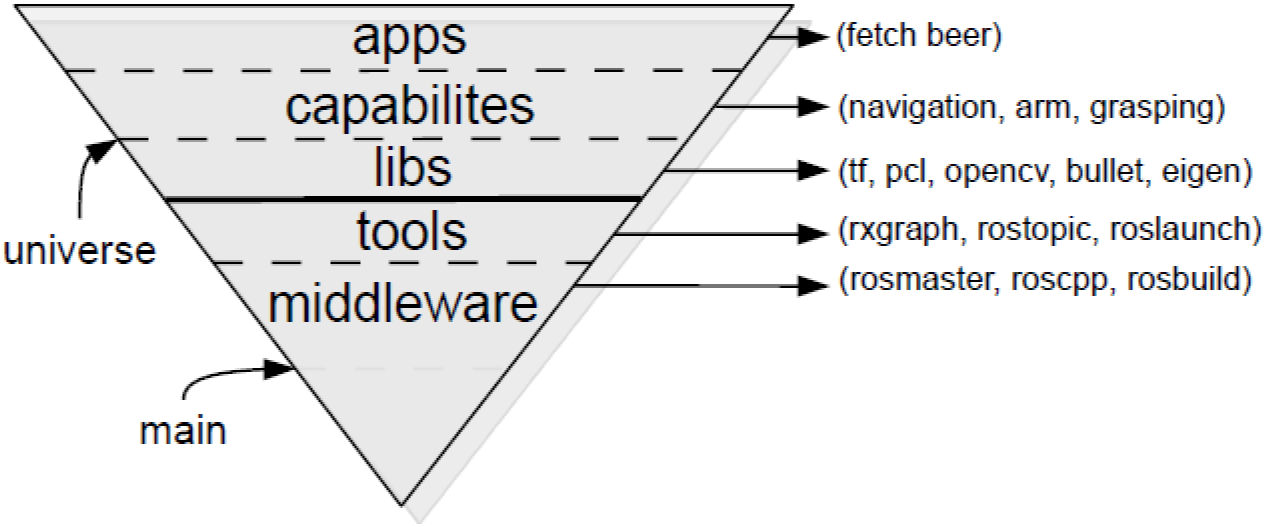
Matteo Matteucci – matteo.matteucci@polimi.it

POLITECNICO DI MILANO

ROS: Robot Operating System

Presented in 2009 by Willow Garage is a meta-operating system for robotics with a rich ecosystem of tools and programs

:::ROS
=
Plumbing
+
Tools
+
Capabilities
+
Ecosystem



apps → (fetch beer)

capabilites → (navigation, arm, grasping)

libs → (tf, pcl, opencv, bullet, eigen)

tools → (rxgraph, rostopic, roslaunch)

middleware → (rosmaster, roscpp, rosbuild)

universe

main

ROS: Robot Operating System

ROS main features:

- Distributed framework
- Reuse code
- Language independent
- Easy testing on Real Robot & Simulation
- Scaling.

ROS Components

- Filesystem tools
- Building tools
- Packages
- Monitoring and GUIs
- Data Logging

ROS.org

ROS: Filesystem Tools

Change directory in the ROS filesystem

- **roscd** [locationname[/subdir]]

Examples:

- roscd roscpp && pwd   /opt/ros/hydro/share/roscpp
- roscd roscpp/src   /opt/ros/hydro/share/roscpp/src
- …

ROS Filesystem Tools

Getting information about installed packages

- **rospack** <command> [options] [package]

Allowed commands (among the others)

help [subcommand]	help menu
depends1 [package]	package dependencies
find [package]	find package directory
List	list available packages

Examples:

• rospack find roscpp	/opt/ros/hydro/share/roscpp
• rospack list	<several packages>
• …	

ROS: Package Creation

Command to create a new package

- **catkin_create_pkg** [package_name] [depend1] [depend2] [depend3]

Example

- catkin_create_pkg beginner_tutorials std_msgs rospy roscpp

Important Notes

- Since Groovy catkin has become the default building tool
- roscpp and rospy are client libraries to use C++ and Python
- Before being able to do that you should have creates a ros_workspace

```
echo $ROS_PACKAGE_PATH
```

Overview of ROS architecture

Nodes: executables that uses ROS middleware to communicate with other nodes, they are processes and communication happens by publish/subscribe

Topics: nodes can publish messages to a topic or subscribe to a topic to receive messages; a topic is a typed communication channel

Messages: data type for the Topics

Master: Name service for ROS

rosout: standard output and standard error for ROS

roscore: Master + rosout + parameter server

ROS Core

The ROS core is a set of the only three programs that are necessary for the ROS runtime.

They include:

- ROS Master
  - A centralized XML-RPC server
  - Negotiates communication connections
  - Registers and looks up names for ROS graph resources

- Parameter Server
  Stores persistent configuration parameters and other arbitrary data

- rosout
  A network-based stdout for human-readable messages

Starting ROS middleware

To start the ROS middleware just type in a terminal

- **roscore**

Now it is possible to display intormation about the nodes currently running

- **rosnode** list

Retrieve information about a specific node

- **rosnode** info /rosout

Note: commands should be executed on a new shell …

ROS Nodes

The basic elements of a ROS architecture are nodes

- Nodes use a client library to communicate with other nodes
- Nodes can publish/subscribe to a Topic
- Nodes can use a Service
- Nodes are implemented using client libraries
    - rospy: Python library
    - roscpp: C++ library
    - rosjava: java library (for android)
    - …

The rosnode command can be used to get information about nodes

Getting Information About ROS Nodes

Getting information about installed packages

- **rosnode** <command>

Allowed commands (among the others)

rosnode ping	*test connectivity to node*
rosnode list	*list active nodes*
rosnode info	*print information about node*
rosnode kill	*kill a running node*
rosnode cleanup	*purge registration information of unreachable nodes*

Examples:

- rosnode list
- rosnode info /rosout

ROS “Graph” Abstraction

The ROS runtime designates several named ROS graph resources

- Nodes: represent processes distributed across the ROS network. A ROS node is a source and sink for data that is sent over ROS network.
- Parameters: Persistent (while the core is running) data such as configuration & initialization settings, stored on the parameter server.
- ROS Topics
  - Asynchronous “stream-like” communication
  - TCP/IP or UDP Transport
  - Strongly-typed (ROS .msg spec)
  - Can have one or more publishers / subscribers
- ROS Services
  - Synchronous “function-call-like” communication
  - TCP/IP or UDP Transport
  - Strongly-typed (ROS .srv spec)
  - Can have only one server, but several clients

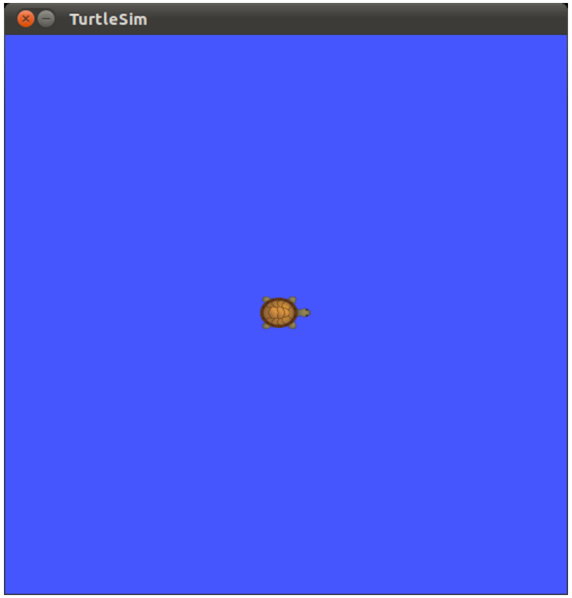
Starting ROS Nodes Execution

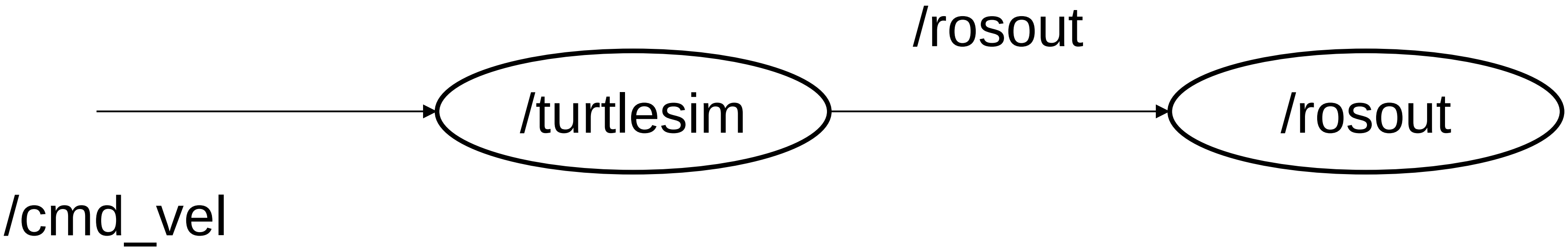
To start a ROS node type in a terminal

- **rosrun** [package_name] [node_name]

Examples:

- rosrun turtlesim turtlesim_node
- rosnode ping turtlesim
- rosnode info turtlesim





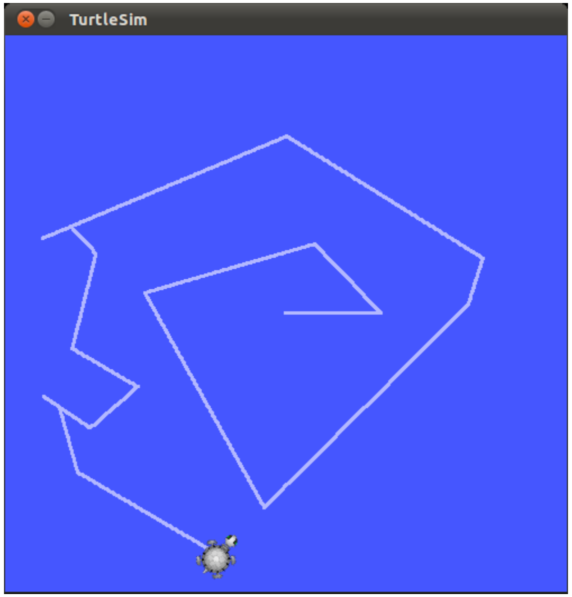
Sending Commands to the Turtle

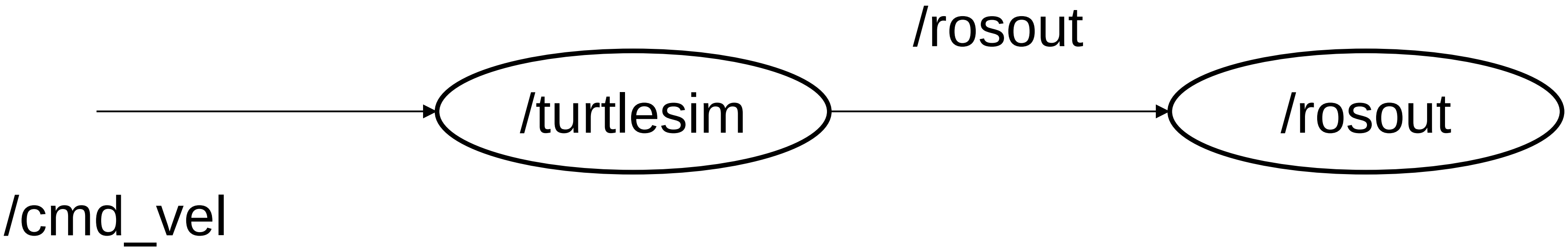
In a new terminal

- rosrun turtlesim turtle_teleop_key

Notes:

- turtle_teleop_key is publishing the key strokes on a topic
- turtlesim subscribes to the same topic to receive the key strokes





Dealing with Running Nodes

To show the running node type in a terminal

- **rosrun** rqt_graph rqt_graph

To monitor the current topic type in a terminal

- **rosrun** rqt_topic rqt_topic

To plot published data on a topic

- **rosrun** rqt_plot rqt_plot
  - /turtle1/pose/x
  - /turtle1/pose/y
  - /turtle1/pose/theta

To monitor a topic on a terminal type

- **rostopic** echo /turtle1/cmd_vel

ROS Topic Command (I)

Getting information about ROS topics

- **rostopic** <command> [options]

Allowed commands (among the others)

rostopic bw	*display bandwidth used by topic*
rostopic echo	*print messages to screen*
rostopic find	*find topics by type*
rostopic hz	*display publishing rate of topic*
rostopic info	*print information about active topic*
rostopic list	*list active topics*
rostopic pub	*publish data to topic*
rostopic type	*print topic type*

Type rostopic <command> -h for more detailed usage, e.g. 'rostopic echo -h'

ROS Topic Command (II)

Getting information about ROS topics

- **rostopic** type [message]

Examples:

- rostopic type /turtle1/cmd_vel
- rosmsg show turtlesim/Pose

Publishing ROS topics

- **rostopic** pub [topic] [msg type] [args]

Example:

- rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'

ROS “Hello World” Nodes

To see how two nodes using topics work check

- talker.cpp
- listener.cpp

To see how two nodes using service

- add_two_ints_server.cpp
- add_two ints_client.cpp

For more in depth examples please refer to beginners tutorials on

- wiki.ros.org

POLITECNICO DI MILANO

Cognitive Robotics – ROS Introduction

Matteo Matteucci – matteo.matteucci@polimi.it