Robot Operating System

Robots and Complexity

A Robot is an amalgam of sensors and actuators, each of which needs to be interpreted/controlled in synchronicity with the others so that the whole produces desirable behaviours. Since microprocessors are typically used to drive robots, the following can be said about the software system:

- Very complex,
- Hard to design,
- Difficult to achieve,
- Hard to maintain,
- Contain lots of bugs, and
- Hard to evolve.

Modular Approach

Using a modular approach, the complexity of the robotic system is abstracted into a graph based hierarchy of conceptually simpler tasks. The tasks then need to be integrated with each other.

- Sensor and actuator units with built in communication protocols
- CPU intensive tasks need remote processing
- Finite State Machines (FSM) model non-linear behaviours
- Robot as a Service (RaaS) providing computing resources for common but complex tasks

A robust and versatile communication framework (middleware) is needed to bring these together.

- ROS
- guWhiteboard
- Others (ppPDC, CARMEN)

Installing ROS

- Linux (Ubuntu 13.10 and 14.04)
  - Setup your `sources.list` and get your apt-key
  - Install using apt-get
- Mac OSX
  - Install Homebrew
  - Follow the ROS installation guide

- ROS is available in three styles:
  - With everything : "desktop-full"
  - Without Simulators and Navigation/Perception : "desktop"
  - Without GUI tools : "ros-base"

(see slide notes for more info, or visit http://wiki.ros.org/ROS/Installation)

ROS Command Line Tools

ROS creates an environment within your command line shell. As such, it has an extensive list of commands that help you:

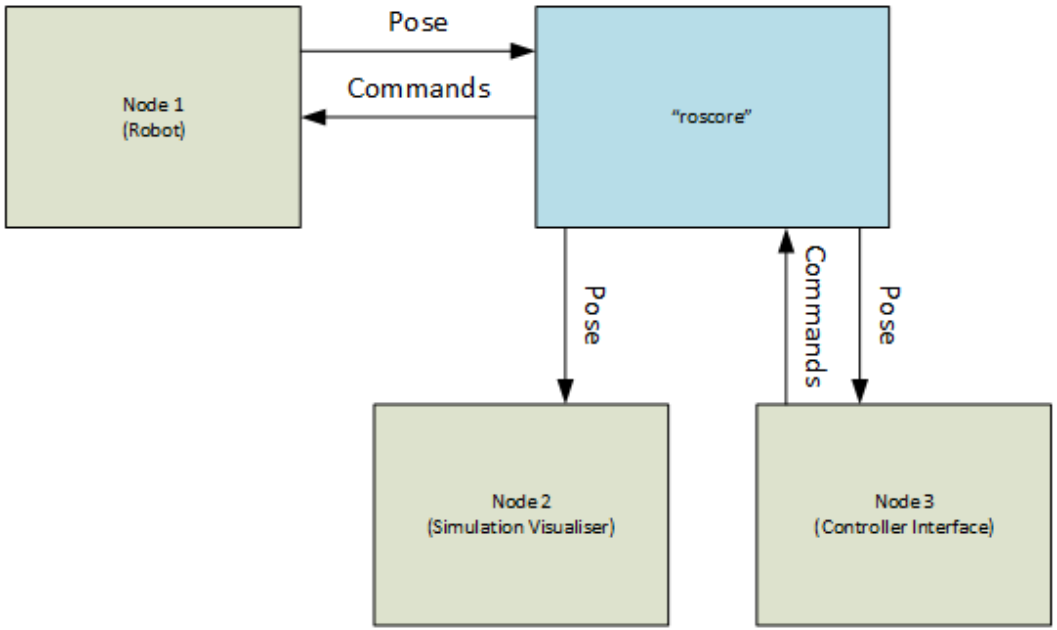
- Navigate through packages,
- Create/Build your packages,
- Monitor/Debug/Log Traffic Flows,
- Visualise data and simulation environments

(I have compiled a document that outlines some of these commands)

Other ROS Concepts

- ROS makes use of the CMake system, which ROS refers to as ‘catkin’.
- ROS has a large community who have contributed much code in the form of “Packages”
- ROS can be programmed in C++, Python, and Lisp. And the ‘catkin’ philosophy supports the use of all of them in a single package.
- ROS has a wiki site where API documentation can be found (though sometimes difficult), and the community has a ‘StackOverFlow’ style forum.

ROS Overview



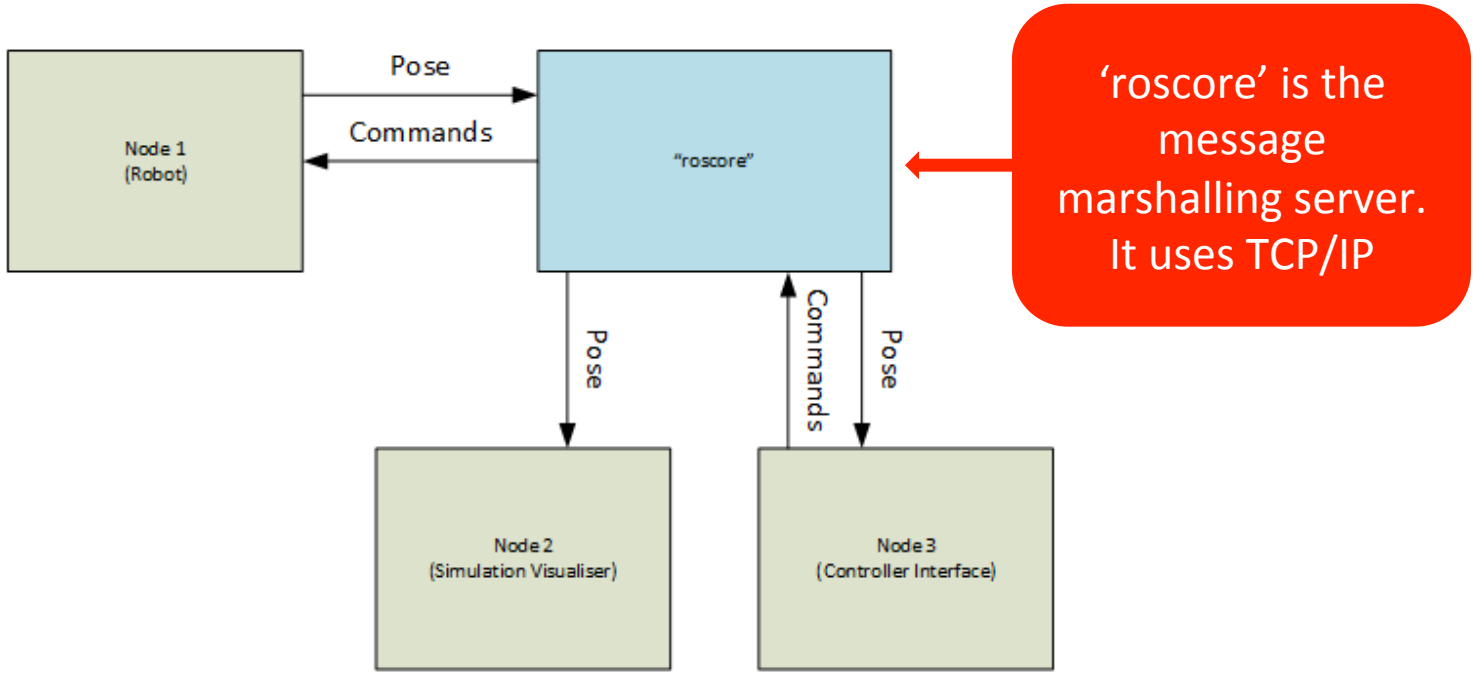
Arrow heads depict direction of data flow

- Away from node = Publisher
- Into node = Subscriber

Line Label denotes the name of the Topic

Topic Name implies a defined message structure/type

ROS Overview



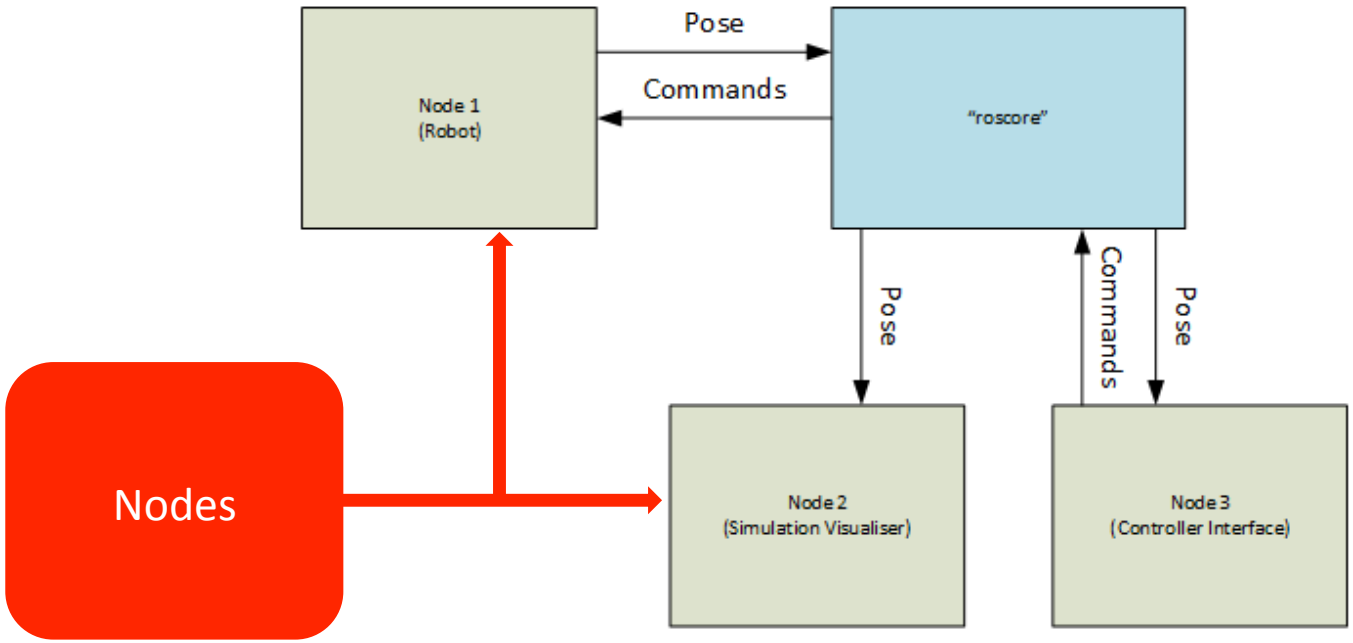
Arrow heads depict direction of data flow

- Away from node = Publisher
- Into node = Subscriber

Line Label denotes the name of the Topic

Topic Name implies a defined message structure/type

ROS Overview



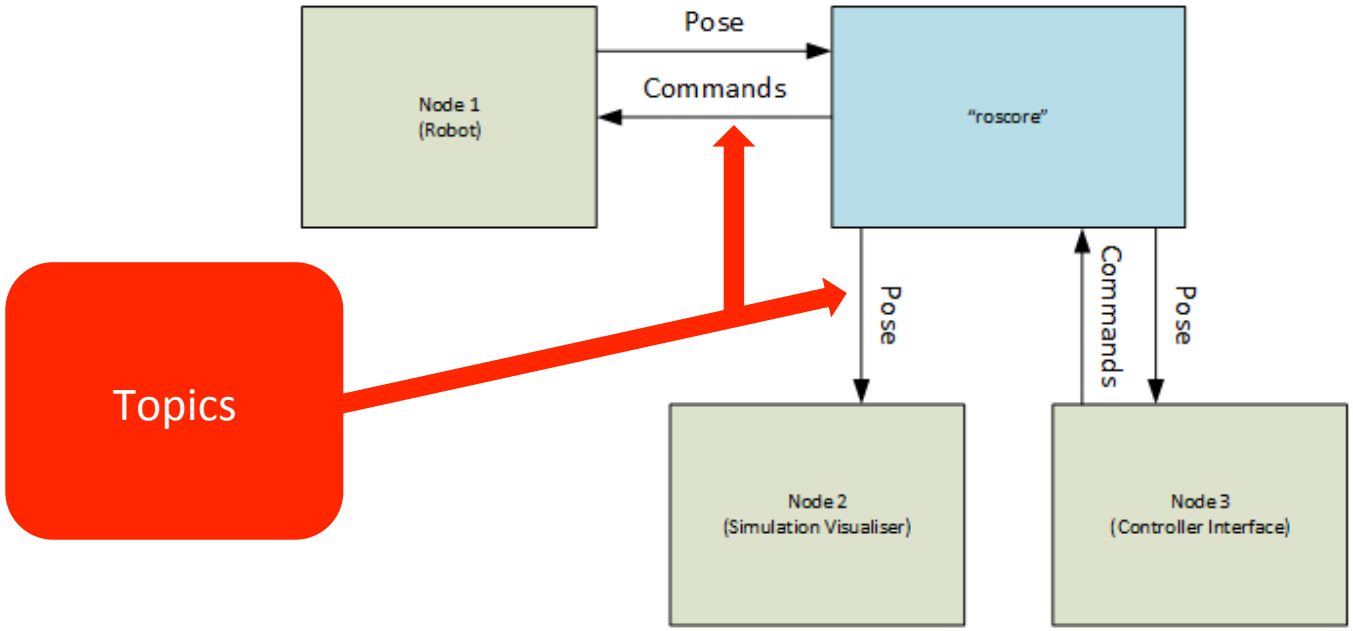
Arrow heads depict direction of data flow

- Away from node = Publisher
- Into node = Subscriber

Line Label denotes the name of the Topic

Topic Name implies a defined message structure/type

ROS Overview



Arrow heads depict direction of data flow

- Away from node = Publisher
- Into node = Subscriber

Line Label denotes the name of the Topic

Topic Name implies a defined message structure/type

MiPal Example Code

Some code you can have a look at to see how ROS is put to use:

- `src/MiPal/GUNao/webots/catkin_ws_mipal_webots/src/webots_mipal_ros_bridge`
- `src/MiPal/GUNao/webots/catkin_ws_mipal_webots/src/demo_webots_ros_driver`
- `src/MiPal/GUNao/webots/EpuckFollowsLineROSController`
- `src/MiPal/GUNao/posix/guWhiteboardROSbridgeTester`
- `src/MiPal/GUNao/posix/gusimplewhiteboard/wbperf/ros_publish_test.h`
- `src/MiPal/GUNao/posix/guWhiteboardROSbridge`

And

- `GettingStarted_Webots_MiPal_ROS_Bridge.pdf` in the MiPal Docs folder.

You will notice that the ROS related code in the MiPal folder doesn't follow the 'catkin workspace' layout. This is because we want the codebase to be consistent with the remainder of the MiPal work. We use a make target to transform this code into the correct layout. 'make catkin' will copy the files into `${MIPAL_DIR}/ros/catkin_ws` and place them in the right places.