ROS MAPE-K Framework Documentation

Andrea Cattaneo, Matteo Locatelli

Feb 23, 2019

Getting started:

ROS MAPE-K Framework is a simple framework that help developers build ROS applications based on the MAPE-K pattern.

CHAPTER 1

How to install

0. Install ROS. The full version is required.
1. Set up a catkin workspace (see this tutorial). Make sure that the workspace path does not contains any special characters.
2. Clone this repository into the src/ folder. It should look like */path/to/your/catkin_workspace/src/ros_mapek_framework*.
3. Activate your workspace.

```
cd /path/to/your/catkin_workspace
source devel/setup.bash
```

4. Build.

```
catkin_make
```

CHAPTER 2

Demo

The demo aims to emulate a sequence of light poles along a path. To save energy, the light poles are turned on in an alternate fashion (in this specific case one on and one off, but it is possible to change criterion). If, for some reasons, a light bulbs went out the system will be able to self-heal himself by adjusting the list of lights on to reduce at least the discomfort due to the breakage of the bulb.

2.1 Run the demo

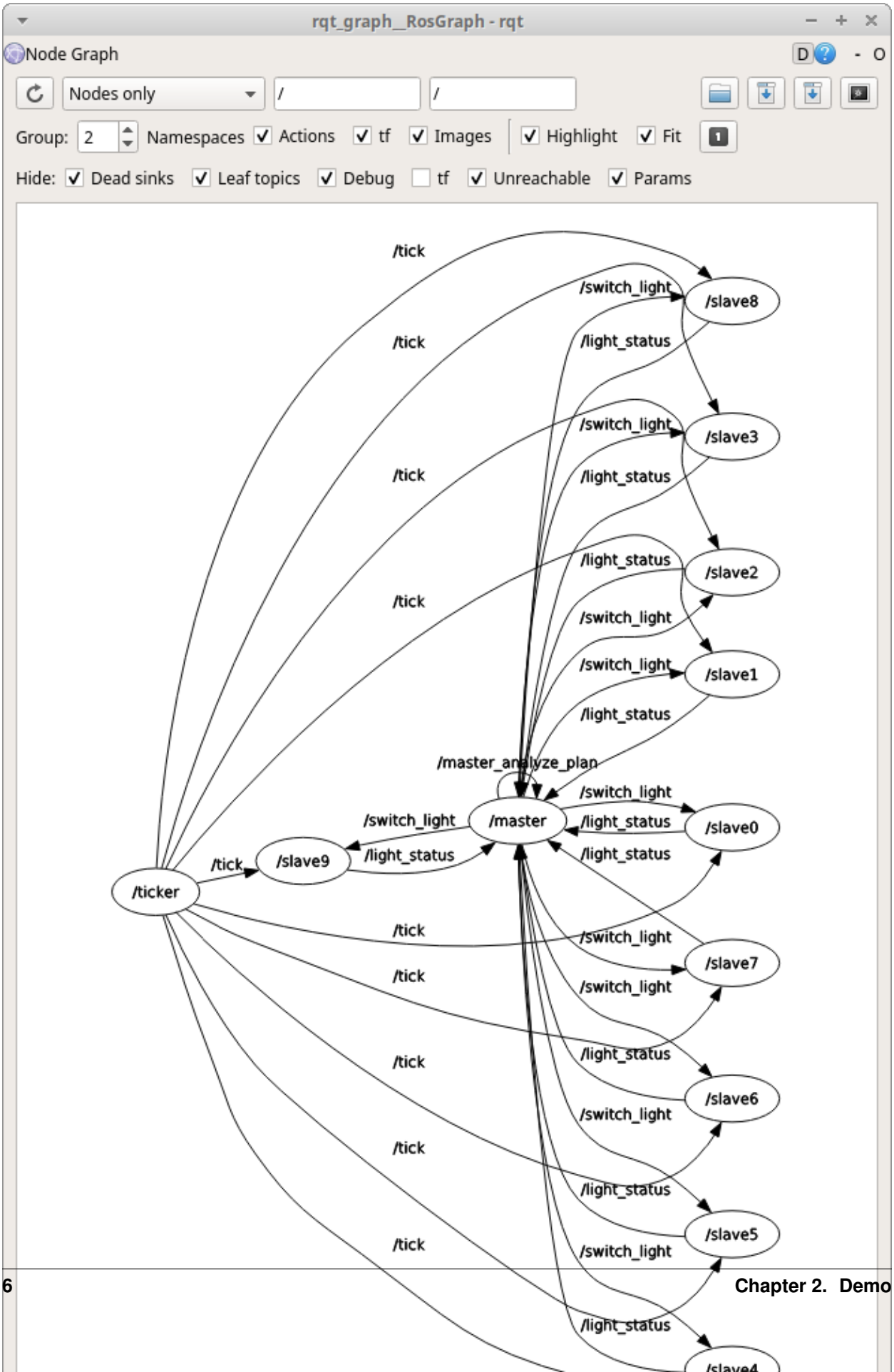
1. Activate your workspace.

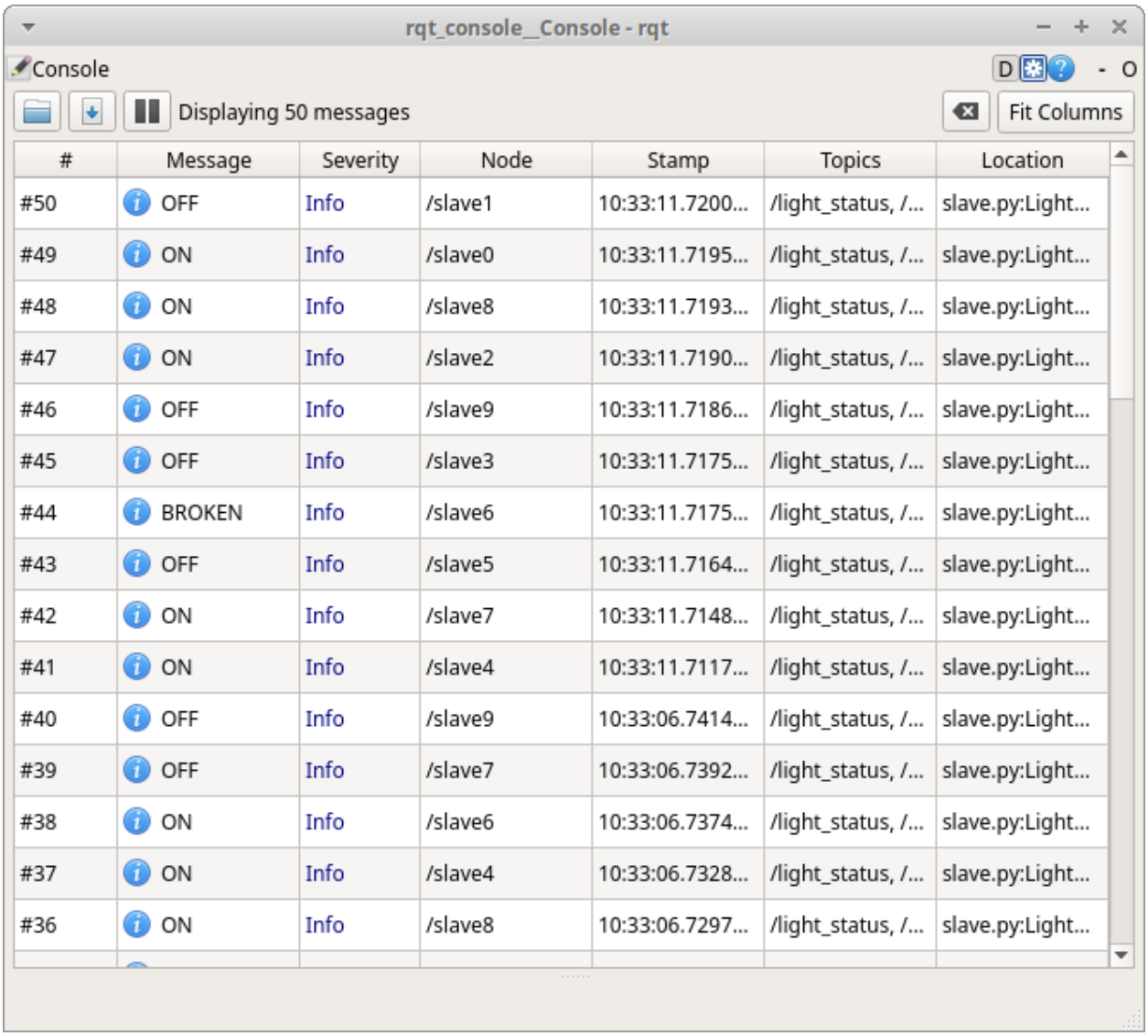
```
cd /path/to/your/catkin_workspace
source devel/setup.bash
```

2. Launch the demo using *roslaunch*.

```
roslaunch mapek_framework_demo demo.launch
```

3. (Optional) Inspect nodes using *rqt_graph* and *rosrun rqt_console rqt_console*.


rqt_graph__RosGraph - rqt
Node Graph
Nodes only
/
/
Group: 2
Namespaces
Actions
tf
Images
Highlight
Fit
Hide:
Dead sinks
Leaf topics
Debug
tf
Unreachable
Params
/tick
/switch_light
/light_status
/master_analyze_plan
/ticker
/master
/slave0
/slave1
/slave2
/slave3
/slave4
/slave5
/slave6
/slave7
/slave8
/slave9



#	Message	Severity	Node	Stamp	Topics	Location
#50	OFF	Info	/slave1	10:33:11.7200...	/light_status, /...	slave.py:Light...
#49	ON	Info	/slave0	10:33:11.7195...	/light_status, /...	slave.py:Light...
#48	ON	Info	/slave8	10:33:11.7193...	/light_status, /...	slave.py:Light...
#47	ON	Info	/slave2	10:33:11.7190...	/light_status, /...	slave.py:Light...
#46	OFF	Info	/slave9	10:33:11.7186...	/light_status, /...	slave.py:Light...
#45	OFF	Info	/slave3	10:33:11.7175...	/light_status, /...	slave.py:Light...
#44	BROKEN	Info	/slave6	10:33:11.7175...	/light_status, /...	slave.py:Light...
#43	OFF	Info	/slave5	10:33:11.7164...	/light_status, /...	slave.py:Light...
#42	ON	Info	/slave7	10:33:11.7148...	/light_status, /...	slave.py:Light...
#41	ON	Info	/slave4	10:33:11.7117...	/light_status, /...	slave.py:Light...
#40	OFF	Info	/slave9	10:33:06.7414...	/light_status, /...	slave.py:Light...
#39	OFF	Info	/slave7	10:33:06.7392...	/light_status, /...	slave.py:Light...
#38	ON	Info	/slave6	10:33:06.7374...	/light_status, /...	slave.py:Light...
#37	ON	Info	/slave4	10:33:06.7328...	/light_status, /...	slave.py:Light...
#36	ON	Info	/slave8	10:33:06.7297...	/light_status, /...	slave.py:Light...

CHAPTER 3

Docstrings

3.1 interaction.py

class mapek_framework.interaction.**Interaction**(*name*, *data_class*)
Class that rappresents an interaction between instances of mape elements.

Parameters

- **name** (*str*) – Name of the interaction.
- **data_class** – ROS message type associated with the interaction.

3.2 managed_system.py

class mapek_framework.managed_system.**ManagedSystem**
Abstract class supposed to be extended to implement a particular managed system.

3.3 group.py

class mapek_framework.group.**Group**(*node_name*, ***kwargs*)
Class that rappresents a group of mape elements.

Parameters

- **node_name** (*str*) – The desired ROS node name.
- ****kwargs** – Custom parameters to use to initialize the ROS node.

elements = []
Elements that are part of the group.

knowledge = {}
A dictionary-like object that rappresents the group knowledge.

managed_system = None
The managed system that interacts with the group.

spin()
Start the group.

Python Module Index

m

Index