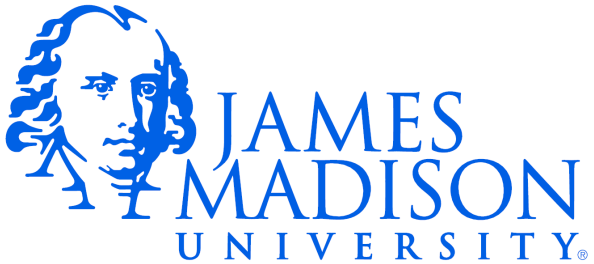
CS354



Nodes/Topics/Messages

- A ROS application is a collection of processes that communicate in a peer-to-peer computation graph
  - Nodes — A process that perform computation
  - Topics — Named buses over which nodes exchange messages
  - Messages — Simple data structures, comprised of typed fields

http://wiki.ros.org/ROS/Concepts

Investigating ROS Topics

```
# View a graphical representation of active nodes
rqt_graph

# List all topics, inspect the list for the topic of interest:
rostopic list

# Get some basic information about the topic of interest:
rostopic info your_topic

# Investigate the topic's message type:
rosmsg show message_package/MessageType

# (If desired) Display messages to the terminal:
rostopic echo your_topic

# (If desired) Publish a message from the terminal:
rostopic pub your_topic message_package/MessageType "message"
```

Listing Topics

```
$ rostopic list
/camera/depth/camera_info
/camera/depth/image_raw
/camera/depth/points
/camera/parameter_descriptions
...
/mobile_base/events/cliff
...
```

There may be too many topics to conveniently display.

You can pipe the output to `less`:

```
$ rostopic list | less
```

Topic Information

```
$ rostopic info /mobile_base/events/cliff

Type: kobuki_msgs/CliffEvent

Publishers:
 * /gazebo (http://cimorene:58845/)

Subscribers: None
```

Message Information

```
$ rosmsg show kobuki_msgs/CliffEvent
uint8 LEFT=0
uint8 CENTER=1
uint8 RIGHT=2
uint8 FLOOR=0
uint8 CLIFF=1
uint8 sensor
uint8 state
uint16 bottom
```

All-caps fields are named constants. Lower-case fields contain data.

More Message Information

The -r or --raw flags show the entire message definition, including comments.

```
$ rosmsg show --raw kobuki_msgs/CliffEvent
# Provides a cliff sensor event.
# This message is generated whenever a particular cliff sensor signals that the
# robot approaches or moves away from a cliff.
# Note that, despite cliff field on SensorState messages, state field is not a
# bitmask, but the new state of a single sensor.

# cliff sensor
uint8 LEFT   = 0
uint8 CENTER = 1
uint8 RIGHT  = 2

# cliff sensor state
uint8 FLOOR = 0
uint8 CLIFF = 1

uint8 sensor
uint8 state

# distance to floor when cliff was detected
uint16 bottom
```

Listening in on a Topic

```
$ rostopic echo /mobile_base/events/cliff
sensor: 0
state: 1
bottom: 42647
---
sensor: 2
state: 1
bottom: 42647
---
sensor: 1
state: 1
bottom: 42647
---
```

Again, the output can be piped to `less`.

Publishing to a Topic

```
$ rostopic pub /mobile_base/events/cliff kobuki_msgs/CliffEvent "sensor: 0
state: 0
bottom: 0"

publishing and latching message. Press ctrl-C to terminate
```

This is an odd example.

Really, only the robot should publish to this topic, but ROS won't stop us.

rqt

- Most of the tasks above can be accomplished through the rqt GUI, but it is clunky.

Rviz

- rviz is the ROS 3D visualization tool.
- Let's take a look...

http://wiki.ros.org/rviz/UserGuide