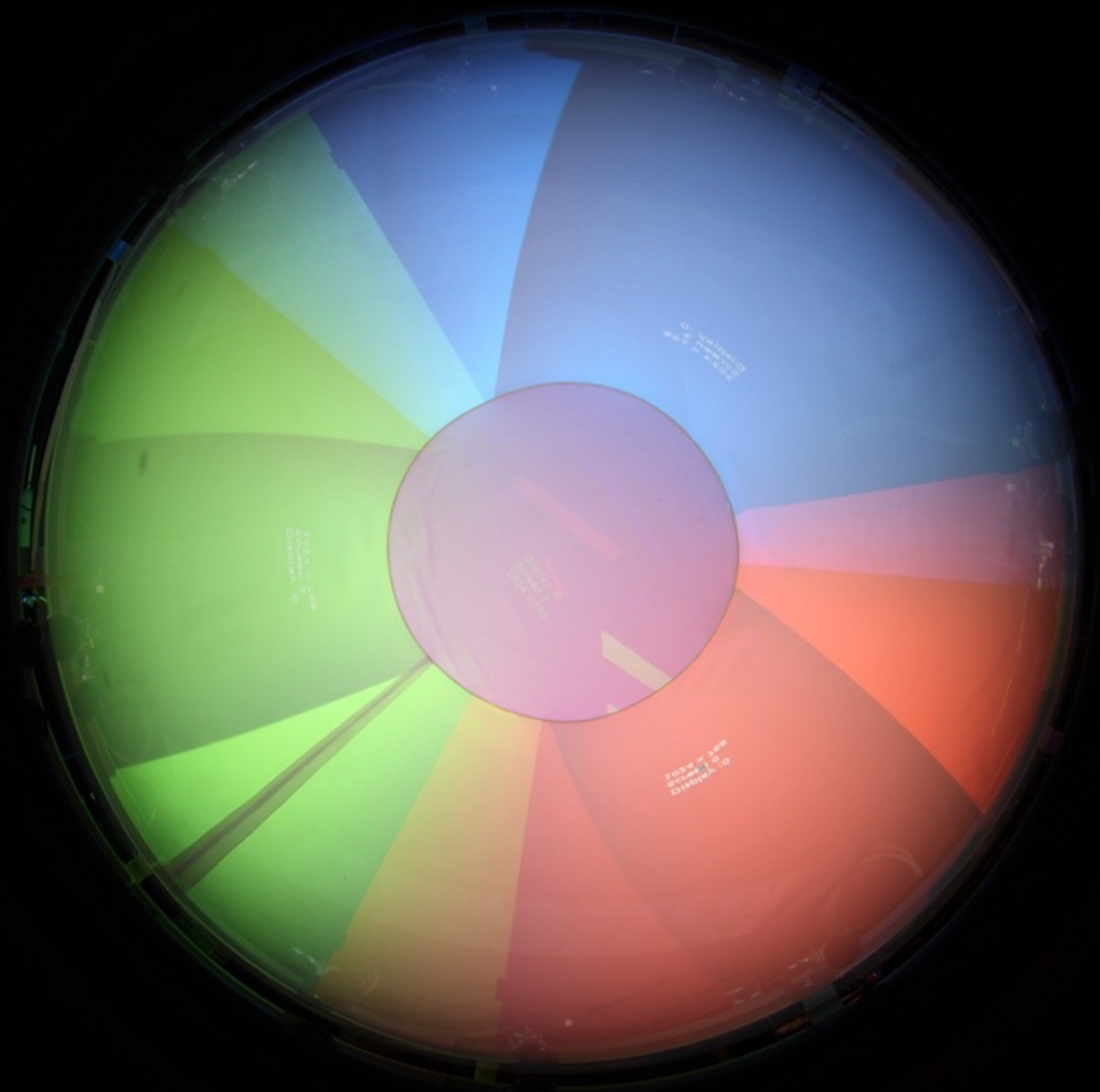
Managing Complex Experiments, Automation, and Analysis using Robot Operating System

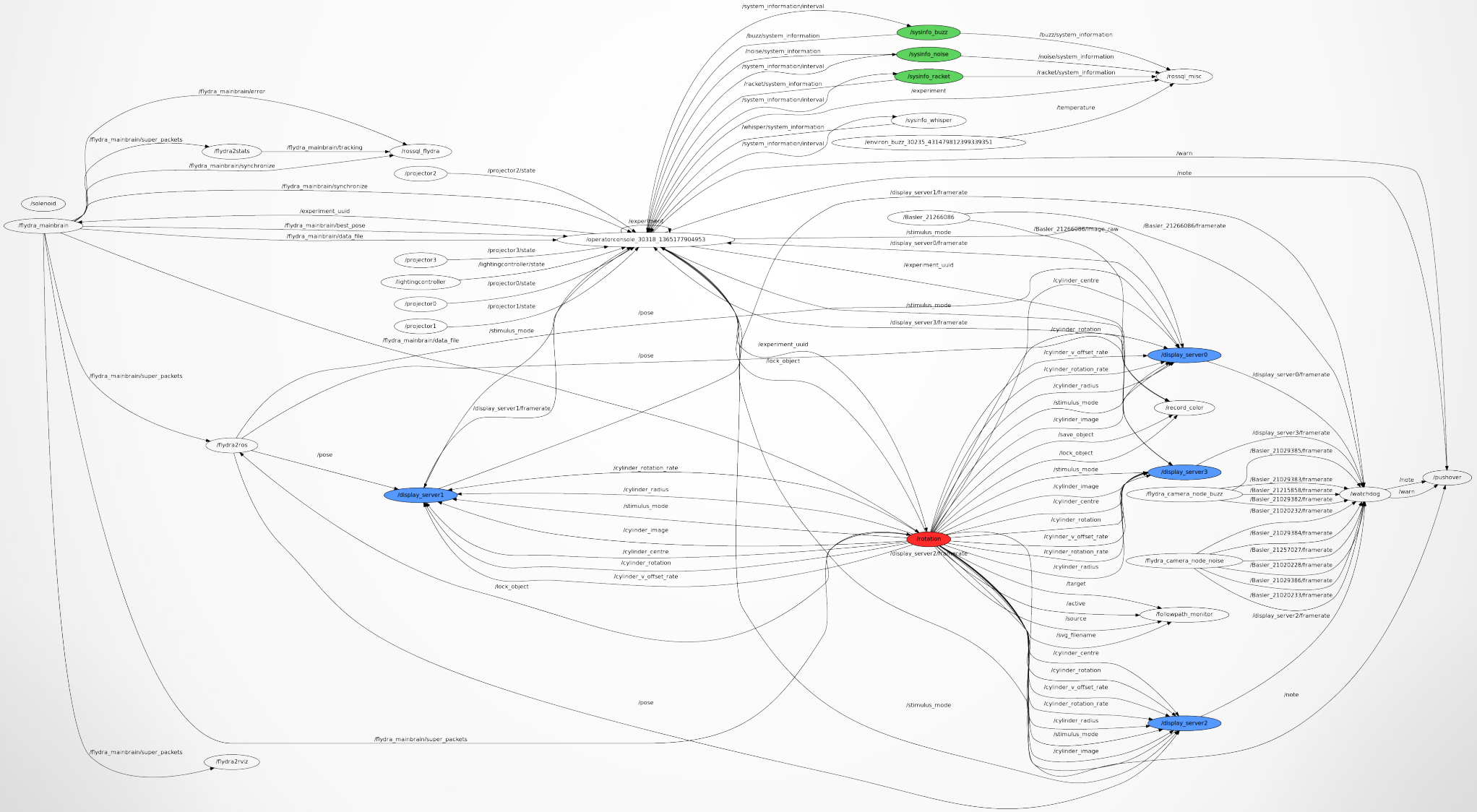


John Stowers (TU Wien) • John.stowers@gmail.com • June 26th 2013

Motivation

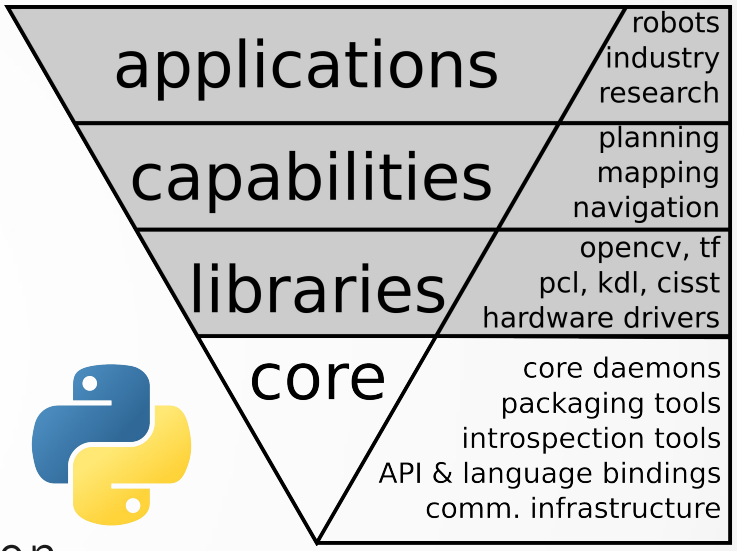
- Real-time closed-loop virtual reality control of Drosophila
- Experiment distributed over 4 computers
- A variety of data collected

Motivation


/system_information/interval
/buzz/system_information
/noise/system_information
/system_information/interval
/racket/system_information
/system_information/interval
/whisper/system_information
/system_information/interval
/sysinfo_buzz
/sysinfo_noise
/sysinfo_racket
/sysinfo_whisper
/environ_buzz_30235_431479812399339351
/buzz/system_information
/noise/system_information
/racket/system_information
/experiment
/temperature
/rossql_misc
/flydra_mainbrain/error
/flydra_mainbrain/super_packets
/flydra2stats
/flydra_mainbrain/tracking
/rossql_flydra
/flydra_mainbrain/synchronize
/projector2
/projector2/state
/flydra_mainbrain/synchronize
/solenoid
/flydra_mainbrain
/experiment_uuid
/flydra_mainbrain/best_pose
/flydra_mainbrain/data_file
/experiment
/operatorconsole_30318_1365177904953
/warn
/note
/display_server1/framerate
/Basier_21266086
/stimulus_mode
/display_server0/framerate
/experiment_uuid
/projector3
/projector3/state
/lightingcontroller
/lightingcontroller/state
/projector0
/projector0/state
/projector1
/projector1/state
/stimulus_mode
/flydra_mainbrain/data_file
/pose
/display_server3/framerate
/cylinder_centre
/cylinder_rotation
/cylinder_v_offset_rate
/cylinder_rotation_rate
/cylinder_radius
/stimulus_mode
/cylinder_image
/save_object
/lock_object
/experiment_uuid
/lock_object
/pose
/flydra_mainbrain/super_packets
/flydra2ros
/pose
/display_server1/framerate
/cylinder_rotation_rate
/cylinder_radius
/stimulus_mode
/cylinder_image
/cylinder_centre
/cylinder_rotation
/cylinder_v_offset_rate
/lock_object
/display_server1
/rotation
/display_server2/framerate
/display_server0
/record_color
/display_server3
/flydra_camera_node_buzz
/flydra_camera_node_noise
/Basier_21029385/framerate
/Basier_21215858/framerate
/Basier_21029382/framerate
/Basier_21020232/framerate
/Basier_21029384/framerate
/Basier_21257027/framerate
/Basier_21020228/framerate
/Basier_21029386/framerate
/Basier_21020233/framerate
/watchdog
/pushover
/note
/warn
/target
/active
/source
/svg_filename
/followpath_monitor
/cylinder_centre
/cylinder_rotation
/cylinder_v_offset_rate
/cylinder_rotation_rate
/cylinder_radius
/stimulus_mode
/cylinder_image
/display_server2
/pose
/stimulus_mode
/note
/flydra_mainbrain/super_packets
/flydra_mainbrain/super_packets
/flydra2rviz

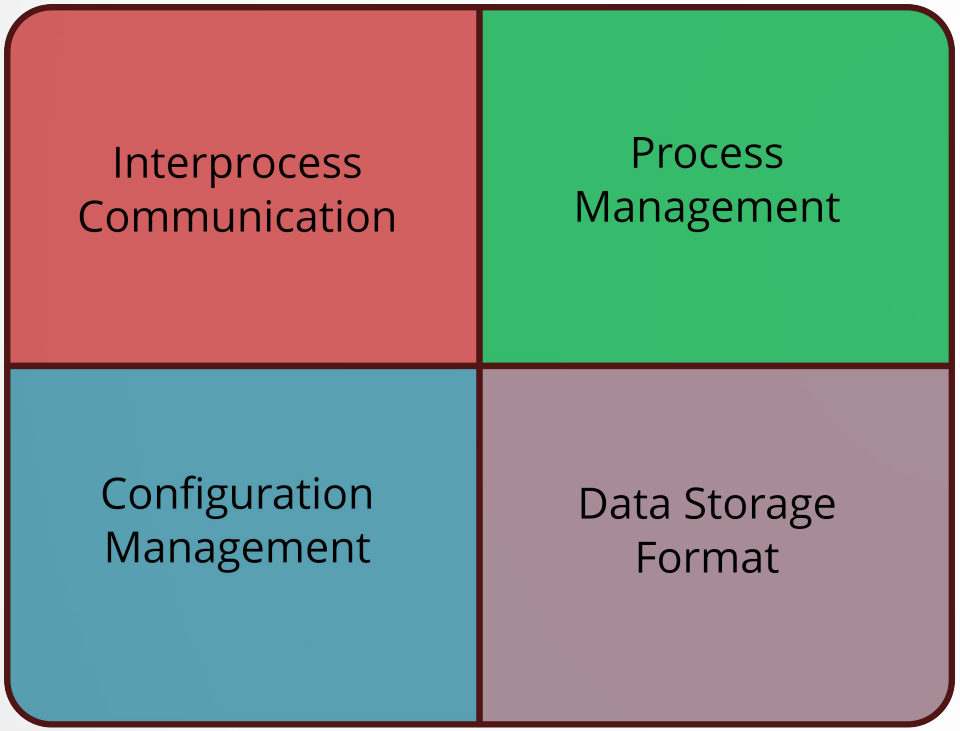
What Is Robot Operating System (ROS)

- A meta-operating system
- A collection of robotics related libraries
- A (bad) package management system
- An architecture for distributed inter-process inter-machine communication and configuration
- Development tools for system runtime and data analysis



The ROS core has nothing to do with robotics and has everything needed for reproducible Science

The Major Concepts of ROS



- ROS Python API
  - Exposes the ROS core
  - Used to interface with the ROS network
- All strongly typed

ROS IPC

- Nodes
  - Correspond to processes (ros mainloop)
- Topics
  - Asynchronous “stream-like” communication
  - Can have one or more publishers, and one or more subscribers
- Services
  - Synchronous, “function-call-like”
  - Can have only one server, and multiple clients

Event.msg

```
Header header
int32 reward
float32 speed

int32 REWARD_RED=1
int32 REWARD_GREEN=2
```

Command.srv

```
string command
---
string response
```

ROS Process and Configuration Management

- Launch files
  - XML files for launching multiple nodes across multiple machines
  - associate a set of parameters and nodes with a single file
  - hierarchically compose collections of other launch files
  - automatically re-spawn nodes if they crash
- Parameter Server
  - Provides parameter values to nodes
- Command line tools
  - for interacting with nodes
  - great for debugging
  - rosbag for recording messages on the network
- Gui tools
  - simple plotting
  - visualization of the ros graph
  - many vision/camera related tools

The ROS API(s)

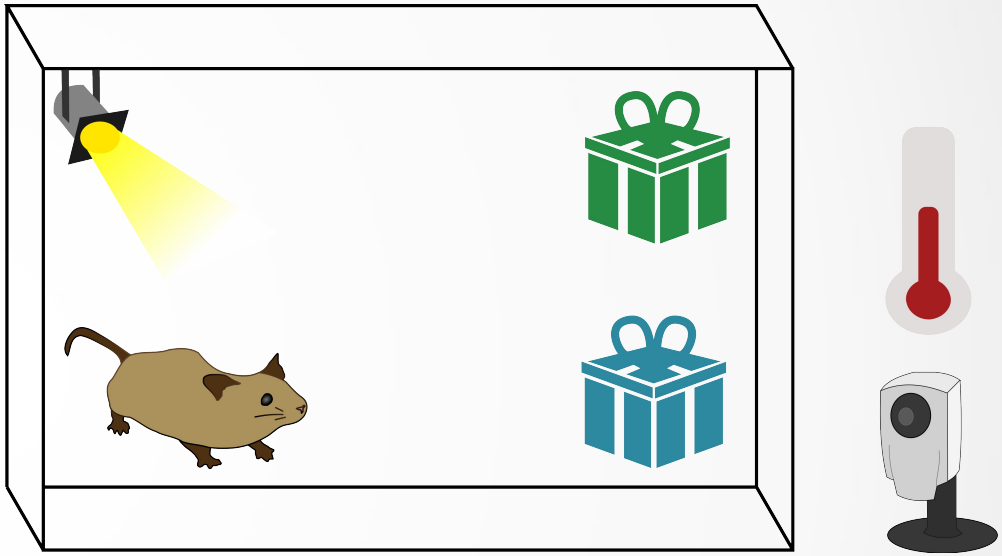
- The IPC is one API
  - Strongly typed, described by paths and msg / srv files
- Use the rospy module to
  - Communicate using this IPC
  - Get/Set parameters
  - Save data to bag files
  - Also provides timing primitives

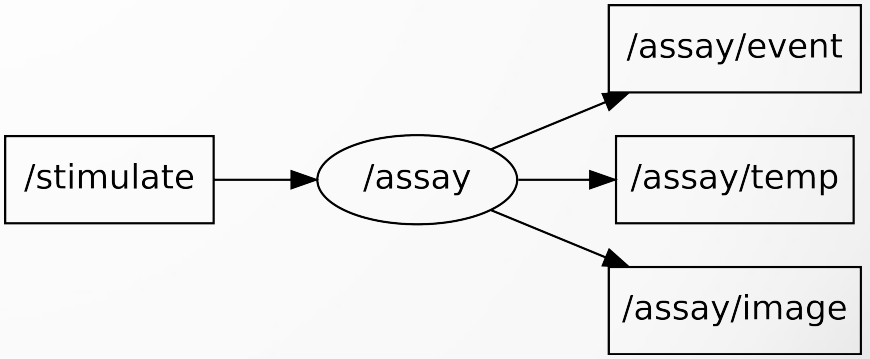
```
import roslib; roslib.load_manifest('scipy')
import rospy
from std_msgs.msg import Float32

rospy.init_node("answer")
pub = rospy.Publisher("info", Float32,
                      latch=True)
pub.publish(42)
rospy.spin()
```

```
import roslib; roslib.load_manifest('scipy')
import rospy
from std_msgs.msg import Float32

def on_msg(msg):
    print "the meaning of life is", msg.data

sub = rospy.Subscriber("info", Float32,
                       on_msg)

rospy.init_node("question")
rospy.spin()
```

A Real Life Example

- A conditioning assay
  - In response to stimulus (light) the mouse makes a choice
  - Record the choice
  - Record experiment data
    - Video
    - Temperature
- Use ROS to implement and parallelize





Live Coding
https://github.com/nzjrs/scipy2013-presentation

Best Practices for ROS Applications

- Prefer command line arguments to ROS remapping
- Use standard message types where possible
- Don't feel compelled to use ROS for everything
- Take advantage of subscriber callbacks and Latched messages
- Synchronizing Time
  - To correlation data collected on multiple machines
  - ROS does have a special notion of time, /clock
- But managing your own timebase is often better
  - PTPd maintains us synchronization between computer clocks
- Pandas indexing / interpolation = win

Interacting With External Partners

- Default data storage format is .bag files
  - Scientists should user inter-operable file formats
    - bag2hdf5 (export bag files to clean hdf5)
- ros_sql (persist bag messages an a sql database)
- ros_freeze
  - Repackage ROS nodes as standard python packages (including dependencies)
- http://www.github.com/strawlab/

Conclusion

- ROS is a healthy vibrant project with a strong future
- It's performance is sufficient for demanding real-time tasks
- Its graph/IPC model maps well to managing experiments
- The ROS core and API is pure python and easy to get to know

We are hiring • http://www.strawlab.org