Advanced robotics in the era of open-source software

Brian Gerkey and Morgan Quigley
We are the stewards of two large projects:
Mission statement: “...to support the development, distribution, and adoption of open source software for use in robotics research, education, and product development.”

http://osrfoundation.org
OSRF Sponsors

YOUR ORGANIZATION NAME HERE
Part I: The past and present: ROS 1
Part II: The future: ROS 2
The Problem: Robot Software is Hard
Our Approach: Collaboration, Modularity, and Simulation
First, make it work in simulation for testing and debugging
The Goal: Robust, Portable, Tested Robot Software
Movie: "8 Years of ROS"

(click here to watch)
ROS plumbing: dynamic messaging graph
ROS: drivers for common robot hardware

- 2d/3d cameras
- laser scanners
- robot actuators
- inertial units
- audio
- GPS
- joysticks
- etc.
ROS tool example: rviz

- rviz: plugin-based visualizer
- render common data streams
- live 3D visualizations
- many other CLI and GUI tools!
ROS capability example

2D navigation system
used on many robots

http://wiki.ros.org/navigation
32,000 build jobs in cloud
Estimating Community Size via Server Logs

Month-long snapshots of unique wiki user-counts

>=10 wiki mirrors (not included in plot)

http://answers.ros.org
Questions to date: **28,169**
Answer rate: **73%**
## ROS.org

### Unique wiki visitors Mar 2015 - Feb 2016

<table>
<thead>
<tr>
<th>Country</th>
<th>Sessions</th>
<th>% Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>819,907</td>
<td>22.57%</td>
</tr>
<tr>
<td>China</td>
<td>356,821</td>
<td>9.82%</td>
</tr>
<tr>
<td>Germany</td>
<td>349,593</td>
<td>9.62%</td>
</tr>
<tr>
<td>Japan</td>
<td>280,419</td>
<td>7.72%</td>
</tr>
<tr>
<td>France</td>
<td>134,301</td>
<td>3.70%</td>
</tr>
<tr>
<td>India</td>
<td>128,497</td>
<td>3.54%</td>
</tr>
<tr>
<td>Spain</td>
<td>117,212</td>
<td>3.23%</td>
</tr>
<tr>
<td>Canada</td>
<td>104,231</td>
<td>2.87%</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>100,309</td>
<td>2.76%</td>
</tr>
<tr>
<td>Italy</td>
<td>87,228</td>
<td>2.40%</td>
</tr>
</tbody>
</table>
Citations of "ROS: an open-source Robot Operating System"

Total citations through May 2016: 2370
ROS ecosystem: ROSCon
A fun book for the whole family

Also useful for treating insomnia
An open world offers many options
Use case: DARPA Robotics Challenge (‘12-'15)

Of the 23 teams in the DRC Finals:

18 teams ran **ROS**
14 teams used **GAZEBO**

Cost in real-life: $1M
Cost in simulation: $0
Use case: driving and mapping vehicles
Use case: service robots
Use case: smarter factory automation
Use case: drones
Use case: NASA Robonaut
Part I: The past and present: ROS 1
Part II: The future: ROS 2
Data Distribution Service (DDS)

- an industry-standard communication system
  \[ \text{OMG DDS} = \text{discovery} + \text{serialization} + \text{transport} \]
- Wire protocol (RTPS over UDP) separable from API (DDS)
- commercial as well as open source implementations
- **proven** in massive critical applications
  - 2014 report by RTI: $1T of systems run DDS
- some are NASA- / DOD-verified
- some small / embedded solutions

10M publish-subscribe pairs!

Spaceships! (at least, launch pads)
ROS2: plumbing upgrades

- DDS + ROS usability

Plumbing

- less time spent here

Means

- more time to spend here

Tools

Capabilities

Ecosystem
# ROS2 software stack

<table>
<thead>
<tr>
<th>Userland code</th>
<th>C++ client library</th>
<th>C client library</th>
<th>Python client library</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ROS client library (rcl)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ROS abstract middleware layer (RMW) (C API)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenSplice DDS</td>
<td>RTI Connext DDS</td>
<td>Twin Oaks CoreDX DDS</td>
<td>eProsima FastRTPS</td>
</tr>
<tr>
<td><strong>OS</strong></td>
<td><strong>Linux</strong></td>
<td><strong>OS X</strong></td>
<td><strong>Windows</strong></td>
</tr>
</tbody>
</table>
ROS1 plumbing details: anonymous pub/sub

- **Publisher**: TCP socket in UNIX process (typically C++ or Python)
- **Subscriber**: TCP socket in UNIX process (typically C++ or Python)
- **Name Service**: XML-RPC server in Python
  - URI known to all participants
  - Propagation via name service

- **advertise()**: Publisher
  - Connect subscribers to publishers
  - Registrations propagate via name service

- **subscribe()**: Subscriber
  - Listen for messages
  - Publish() sends messages to all current subscribers
ROS2 plumbing details: DDS/RTPS over UDP

- **Publisher**
  - Publish() sends messages to all current subscribers via mcast or unicast UDP

- **Subscriber**
  - Participants broadcast existence via mcast UDP
  - Participants negotiate topic connection parameters
ROS2: DDS/RTPS spans networking spectrum

UDP (best-effort)

TCP (reliable)

lots of steel+concrete = super bad comms
"packet drop? oh well."

"packet drop? retry a few times, then move on."
Small embedded systems are everywhere!

The trend towards "microcontrollers everywhere" continues to accelerate. ROS2 goal: run on all classes of systems.
ROS2 can fit in sensors!

- UDP-based middleware: easier to embed, no deps.
- Eliminate custom drivers
- Real-time is easier to guarantee on small systems
ROS 1: hard partition between realtime and non-realtime
ROS 2: mix realtime and non-realtime nodes
ROS2: connecting embedded to Linux

STM32F4 dev. board

<table>
<thead>
<tr>
<th>Timer interrupt</th>
<th>MEMS accelerometer</th>
<th>IMU sender demo app</th>
<th>FreeRTPS</th>
<th>Ethernet PHY</th>
</tr>
</thead>
</table>

Measured jitter on the wire: ~50ns std.dev., ~500ns max
Measured jitter in Linux RT-PREEMPT userspace: <50us typical, 250us max
ROS 2 Feature Status: May 2016

- Publish / subscribe, services (RPC), parameters
- Code generation for messages using existing ROS IDL
- DDS/RTPS implementations: RTI Connext, PrismTech OpenSplice, eProsima FastRTPS, OSRF FreeRTPS (partial)
  - Including cross-implementation testing
  - ROS middleware (rmw) abstraction handles differences
- Continuous integration on Ubuntu Trusty, OS X 10.10 and Windows 8.1
- Partial support for bare-metal ARM Cortex-M processors
- Client libraries: C++, C (partial), Python (partial), Java (preliminary)
- Multiple nodes per process (no more node vs. nodelet)
- Real-time support via pluggable memory allocation strategies (C++)
- ROS 1 <-> ROS 2 bridge to route messages between
- ROS 1 -> ROS 2 C++/Python library shims (partial)
- Preliminary support for Android
- TurtleBot demo in progress

First useable beta release planned for later in 2016
ROS 2 Development Process

Open and collaborative design process:  
http://design.ros2.org

Core code kept in the GitHub ros2 org:  
https://github.com/ros2

Emphasis on automated testing via Jenkins:  
http://ci.ros2.org/

Documentation coming together in the ros2/ros2 wiki:  
https://github.com/ros2/ros2/wiki

Alpha releases made on a 6-week cadence:  
https://github.com/ros2/ros2/wiki/Releases

Fun! Free! Open!
Join us at ROSCon in Seoul!

Oct 8-9, 2016
(just before IROS)