Advanced robotics in the era of open-source software

Brian Gerkey
3 November 2016
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
We are the stewards of two large projects:
Part 1: Update on the state of the community
Community growth

http://answers.ros.org
Questions to date: 30,129
Answer rate: 73%

We estimate 100s of 1000s of users/developers
Community metrics (not counting mirrors)

Downloads (July 2016)
• Packages: 8,441,279 (+158% over 2015)
• IP addresses: 113,345 (+143% over 2015)

By ROS distro:
• Groovy and earlier: 0.46%
• Hydro: 1.92%
• Indigo: 72.84%
• Jade: 3.96%
• Kinetic: 11.02%

Research use (cumulative)
• Citation count of “ROS: an Open-Source Robot Operating System” (Quigley et al., 2009): 2683 (+46% over 2015)

wiki.ros.org visitors

1. United States 85,023 (23.06%)
2. China 62,933 (17.07%)
3. Japan 30,745 (8.34%)
4. Germany 28,022 (7.60%)
5. India 12,918 (3.50%)
6. Taiwan 11,301 (3.06%)
7. South Korea 9,990 (2.71%)
8. Singapore 9,015 (2.44%)
9. Canada 8,928 (2.42%)
10. Spain 8,517 (2.31%)

http://wiki.ros.org/Metrics
Community service: ROS build farm

- 3 ROS distros supported at a time
- Currently building for 8 binary targets
- >40,000 build jobs total
- The build farm code is open and reusable
Estimating Codebase Size

ros_comm ("core")
100 KLOC

desktop-full ("core+tools")
400 KLOC

all build farm ("universe")
4000 KLOC

only includes publicly released code!
Attendees by geographic region

- Middle East and North Africa: 0.5%
- Europe and Central Asia: 17.8%
- Other / unspecified: 22.9%
- Americas: 15.0%
- Asia-Pacific: 43.9%

Total: 450-460
(+20% from 2015, sold out 2nd year in a row)

Attendees by registration type

- Student: 38%
- Non-student: 62%
+44% in number of sponsors
+60% in sponsorship funding
Part 2: Industry use cases
Use case: driving and mapping vehicles
Use case: agriculture
Use case: monitoring & measurement

The physical robot controls the simulated robot, and the CI system checks whether the task is correctly achieved. Such simulation-based CI testing is now widely used in robotics companies that are developing ROS-based products and services.

4. Development Team

The authors represent a premier team of ROS developers and users who are well positioned to bring ROS to bear in DOE-EM applications:

- The Open Source Robotics Foundation (OSRF), developers and maintainers of the ROS core, located in Mountain View, California. For more about OSRF, refer to Appendix A.1.
- Southwest Research Institute (SwRI), developers and maintainers of ROS-I software for factory automation, and ROS software for UGVs, located in San Antonio, Texas. For more about SwRI, refer to Appendix A.2.
- The University of Texas at Austin Nuclear and Applied Robotics Group (UT NRG), a DOE-sponsored lab that uses ROS to develop robotic solutions for nuclear material handling and monitoring at Los Alamos National Labs (LANL). For more about UT NRG, refer to Appendix A.3.

5. DOE-EM Robotic Mission Overview

Robotic systems are needed to provide access and remote entry into areas that are inaccessible or prohibited by human works due to:

- Unsafe, unstable, or unknown conditions of the environment
- Presence of radiological or other hazards resulting in unacceptably high exposures
- Extreme operating environments (high temperatures, high pressures, poor air quality, presence of explosives, limited visibility, etc.)

Recent efforts—and the lessons learned from them—lend credence to the merit of this effort. For example, on June 24, 2011, the first mission of a customized Quince mobile platform was conducted in the reactor building of Unit 2 at the Fukushima Daiichi Nuclear Power Plant. After a six-week period of training, TEPCO operators attempted to complete six variously complex missions. After completing the sixth mission, the robot was unable to return. System developers documented the critical elements lacking during the project:

- Field knowledge for researchers
- Precise communication between researchers and users
- Education of users

The lessons learned were not technical, but programmatic. The first lesson was magnified by the nature of the tragedy and the severe time constraints imposed on the team. In one case, critical information describing the dimensions of the staircase had been washed away by the tsunami.


Use case: material delivery
Use case: drones
Part 3: Current areas of development at OSRF

(note that the ROS community is much much much much much larger than OSRF)
Turtlebots!

Turtlebot 1 (2010)
- iRobot Create
- x86 Atom netbook
- Microsoft Kinect

Turtlebot 2 (2013)
- Yujin Kobuki
- x86 Atom netbook
- Asus Xtion Pro

Turtlebot 3 (2016)
- ROBOTIS chassis
- Any SBC
- Any sensor
- Shipping 2017 Q1

2500+ sold worldwide to academia, industry, and hobbyists
ROS on SBCs
“Classic” ROS is not secure

Front door is wide open
  ○ fully discoverable
  ○ no authentication
  ○ no encryption
  ○ no access control
  ○ this was by design

ROS assumes a secure network! If that assumption is not true, arbitrarily bad things can happen
How ROS talks

Dashed = ephemeral XML-RPC
Solid = persistent TCP
Add some security

All traffic using Transport Layer Security (TLS/SSL) for encryption and authentication
When nodes get owned

When evildoers compromise a node, in classic ROS they can subscribe to all traffic, spoof other nodes, shut down nodes, …
Graph memorization: access control

SROS records the “intended” graph topology, which is later enforced by roscore and other nodes. Owned node can’t affect entire system.
Process-level Security

When nodes get owned, we want to limit what evildoers can do in the kernel after compromise. SROS provides common AppArmor profiles for this.

- http://wiki.ros.org/sros
- Work in progress; Python-only for now
- Adversarial analysis is key
Historical: custom messaging

We made all this stuff up
Data Distribution Service (DDS)

- an industry-standard middleware specification
- many implementations exist
- **proven** in massive critical applications
  - 2014 report: $1T of systems run DDS (!)

- Big huge stuff!
- 10M publish-subscribe pairs!
- Spaceships! (at least, launch pads)
DDS/RTPS networking

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."

Open Source Robotics Foundation
Client API Upgrades

– To support real-time, we needed to change the API. And Python3 and C++11 make for nicer code.

– “Classic” ROS API
  – roscpp + nodelet + actionlib + dynamic_reconfigure

– “Modern” ROS API
  – All comes “for free” with the new C++11/Python3 API
“Classic” vs “Modern” API

### “Classic” API stack

- user code
  - actionlib
  - dynamic parameters
  - nodelets
- C++ client library (roscpp)

### “Modern” API stack

- user code
  - C++ client library
  - ROS client library (rcl)
  - ROS middleware layer
  - Connext DDS
  - FastRTPS
  - OpenSplice DDS

- OS
  - Linux
  - OS X
  - Windows

---

Open Source Robotics Foundation
What about “classic” ROS?

- “Classic” ROS API and networking will work
  - Millions of lines of code use it
  - Everything will interoperate

- For the foreseeable future, we expect that “hybrid” systems will exist:
  - global bridge
  - library shim
  - local bridges
Compatibility: Global Bridge

“Classic” ROS API and networking

“Next-gen” ROS API over DDS

Dynamic Global Bridge
Compatibility: Library Shim

Classic ROS API

Networking via DDS
Compatibility: Local Bridges

Concept: “next-gen” ROS nodes can use both DDS and “classic” networking.
Compatibility: Summary

- **We care deeply about compatibility**
  - Huge codebase uses “classic” API and networking
  - Code migration will take many years
  - Hybrid systems will exist for the foreseeable future
ROS 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

Beta 1 to be released by end of 2016
We’re hiring!

We’re looking for more industry partners!

http://openrobotics.org