ROS Crash-Course, Part I
Introduction to ROS distribution, build system and infrastructure

Jonathan Bohren
With some information and figures adapted from http://www.ros.org
and the COTESYS-ROS School 2010 presentation given by Radu Rusu
Outline

1. Introduction
   - High-Level
     - The ROS Ecosystem
     - ROS Community

2. ROS as a Communication Platform
   - The ROS Network Graph
   - Running and Connecting Nodes
   - Analyzing the System at Runtime

3. ROS as a Build Platform
   - Distribution & Package Management System
   - Build System
Outline (revisted)

1. Introduction
   - High-Level
     - The ROS Ecosystem
     - ROS Community

2. ROS as a Communication Platform
   - The ROS Network Graph
   - Running and Connecting Nodes
   - Analyzing the System at Runtime

3. ROS as a Build Platform
   - Distribution & Package Management System
   - Build System
What is ROS?

More than just middleware

- A “meta” operating system for robots
- A collection of packaging, software building tools
- An architecture for distributed* inter-process/inter-machine communication and configuration
- Development tools for system runtime and data analysis
- Open-source under permissive BSD licenses (*ros core libraries*)
- A language-independent architecture (c++, python, lisp, java, and more)
- A scalable platform (ARM CPUS to Xeon Clusters)

With the intent to enable researchers to rapidly develop new robotic systems without having to “reinvent the wheel” through use of standard tools and interfaces.
What is ROS not?

No confusion

- An actual operating system
- A programming language
- A programming environment / IDE
- A hard real-time architecture*
What does ROS get you?

All levels of development

Developed and maintained by the international ROS community

general tools for distributed computing

universe

algorithms frameworks hardware drivers "robotic apps"

main

Maintained by Willow Garage, inc and some external developers
What does ROS get you?

All levels of development

- packaging & build tools
- communication infrastructure
- ROS API language bindings
- introspection tools

...fetching beer, scraping the seafloor...

- tf, opencv, pcl, kdl, cisst
- simulation, drivers...

...grasping, control, execution, navigation...

Jonathan Bohren (JHU LCSR)
ROS Crash-Course, Part I: Introduction
Outline (revisted)

1. Introduction
   - High-Level
   - The ROS Ecosystem
   - ROS Community

2. ROS as a Communication Platform
   - The ROS Network Graph
   - Running and Connecting Nodes
   - Analyzing the System at Runtime

3. ROS as a Build Platform
   - Distribution & Package Management System
   - Build System
**ROS Core**

Where it all comes together

The ROS core is a set of the only three programs that are necessary for the ROS runtime. They include:

- **ROS Master**
  - A centralized XML-RPC server
  - Negotiates communication connections
  - Registers and looks up names for ROS graph resources

- **Parameter Server**
  Stores persistent configuration parameters and other arbitrary data

- **roslout**
  Essentially a network-based stdout for human-readable messages
ROS “Graph” Abstraction

Named network resources

The ROS runtime designates several classes of named ROS *graph resources*. These resources can exist in namespaces to reduce naming collisions, and fall into the following categories:

- **nodes**
  Represent processes distributed across the ROS network. A ROS node is a source and sink for data that is sent over the ROS network.

- **parameters**
  Persistent (while the core is running) data such as configuration & initialization settings, stored on the parameter server.

- **topics**
  Asynchronous many-to-many communication streams.

- **services**
  Synchronous one-to-many network-based functions.
ROS “Graph”

rxgraph: communication network visualization
Creating and Running ROS Nodes

Distributing computation with ROS

ROS provides a mechanism for simultaneously configuring and launching collections of ROS nodes. This is done with lightweight xml files and the roslaunch program.

Launch files enable users to:

- 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
- Change node names, namespaces, topics, and other resource names without recompiling
- Easily distribute nodes across multiple machines
- Attach gdb to a series of nodes
ROS Communication Protocols
Connecting nodes over the network

ROS supports a growing number of communication capabilities that enable distributing computation in a robotic system. These capabilities are currently built entirely on two high-level communication APIs:

- **ROS Topics**
  - Asynchronous “stream-like” communication
  - TCP/IP or UDP Transport
  - Strongly-typed (ROS .msg spec)
  - Can have one or more *publishers*
  - Can have one or more *subscribers*
ROS supports a growing number of communication capabilities that enable distributing computation in a robotic system. These capabilities are currently built entirely on two high-level communication APIs:

- **ROS Topics**
  - Asynchronous “stream-like” communication
  - TCP/IP or UDP Transport
  - Strongly-typed (ROS .msg spec)
  - Can have one or more *publishers*
  - Can have one or more *subscribers*

- **ROS Services**
  - Synchronous “function-call-like” communication
  - TCP/IP or UDP Transport
  - Strongly-typed (ROS .srv spec)
  - Can have only one *server*
  - Can have one or more *clients*
Asynchronous Distributed* Communication

ROS TCP Topics

```
ros
"master"
camera
```

```
viewer
```

Jonathan Bohren  (JHU LCSR)  ROS Crash-Course, Part I: Introduction
Asynchronous Distributed Communication

ROS TCP Topics

advertise("images")

ros "master"

camera

viewer
Asynchronous Distributed* Communication

ROS TCP Topics

- `ros
  "master"
  topic:images`

- `camera`
- `viewer`
Asynchronous Distributed* Communication

ROS TCP Topics

camera

ros
"master"

subscribe("images")

topic:images

viewer

Jonathan Bohren (JHU LCSR) ROS Crash-Course, Part I: Introduction
Asynchronous Distributed Communication

ROS TCP Topics

```python
ros
"master"

subscribe("images")
```
Asynchronous Distributed Communication

ROS TCP Topics

camera

images(tcp)

viewer

ros
"master"

topic:images

Jonathan Bohren (JHU LCSR) ROS Crash-Course, Part I: Introduction
Asynchronous Distributed* Communication

ROS TCP Topics

```
ros
"master"

topic:images

images(tcp)

camera

view

publish(img)
```
**Asynchronous Distributed** Communication

ROS TCP Topics

```
ros
"master"
```

```
topic:images
```

```
images(tcp)
publish(img)
camera
```

```
viewer
```
Asynchronous Distributed* Communication

ROS TCP Topics

```
ros
"master"
topic:images
```

```
images(tcp)
camera	publish(img)
```

```
viewer
viewer_too
```
Asynchronous Distributed Communication

ROS TCP Topics

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"

subscribe("images")
```

```
ros
"master"
```

```
Asynchronous Distributed Communication

ROS TCP Topics

camera

images(tcp)

publish(img)

viewer

viewer_too

images(tcp)

ros
"master"

topic:images
rosout Messaging

stdout on steroids

ROS provides mechanisms in all languages for specifying different \textit{levels} of human-readable log messages. The five default levels are:

- fatal
- error
- warn
- info
- debug

These enable a user to “add printf’s” to their code for debugging, and selectively enable and disable them \textit{at runtime} without a large performance hit. For example, useful debug messages that might be necessary to diagnose a problem could be left in the code and re-enabled at a critical time.
ROS Graph Introspection

No more wireshark

ROS provides several tools for analyzing the data flowing over ROS communication resources:

- **rosnodes**: Gives a user information about a node: publications, subscriptions, etc.
- **rostopic**: Gives datarate, actual data, publishers, subscribers
- **rosservice**: Enables a user to call a ROS Service from the command line
- **roswtf (wire trouble finder)**: Diagnoses problems with a ROS network
**ROS GUI Tools**

There are lots...
ROS GUI Tools

There are lots...
ROS GUI Tools

There are lots...
ROS GUI Tools

There are lots...
ROS GUI Tools

There are lots...
ROS GUI Tools

There are lots...
rviz - 3D Visualization

Modular state and sensor visualization
Outline (revisted)

1 Introduction
   High-Level
   The ROS Ecosystem
   ROS Community

2 ROS as a Communication Platform
   The ROS Network Graph
   Running and Connecting Nodes
   Analyzing the System at Runtime

3 ROS as a Build Platform
   Distribution & Package Management System
   Build System
ROS Stacks & Packages

How to organize code in a ROS ecosystem

ROS code is grouped at two different levels:

- **Packages**
  A named collection of software that is built and treated as an atomic dependency in the ROS build system.

- **Stacks**
  A named collection of packages for distribution.
ROS Distributions

Delivering ROS packages to the masses

- source code
- header declarations
- scripts
- message definitions
- service definitions
- configuration files
- launch files
- metadata
  ...
ROS Distributions

Delivering ROS packages to the masses

source code
header declarations
scripts
message definitions
service definitions
configuration files
launch files
metadata
...

"package"

package_one
ROS Distributions
Delivering ROS packages to the masses

"package"

source code
header declarations
scripts
message definitions
service definitions
configuration files
launch files
metadata
...

package_one

package_two
ROS Distributions
Delivering ROS packages to the masses

"package"

source code
header declarations
scripts
message definitions
service definitions
configuration files
launch files
metadata
...

package_one
package_two
package_n
ROS Distributions

Delivering ROS packages to the masses

"package"

source code
header declarations
scripts
message definitions
service definitions
configuration files
launch files
metadata
...

"stack"

package_one
package_two
package_n
ROS Distributions

Delivering ROS packages to the masses
ROS Distributions

Delivering ROS packages to the masses

stack_a-0.4.0
**ROS Distributions**

Delivering ROS packages to the masses

stack_a-0.4.0  stack_b-1.0.2
ROS Distributions

Delivering ROS packages to the masses
ROS Distributions
Delivering ROS packages to the masses

"distribution"
ROS Distributions

Delivering ROS packages to the masses

"distribution"

ROS "Box Turtle"
March 2, 2010

ROS "C-Turtle"
August 2, 2010

ROS "Diamondback"
March 2, 2011
ROS Distributions

Delivering ROS packages to the masses

stack_a-0.4.0  stack_b-1.0.2  stack_c-0.2.1

"distribution"

ROS "Box Turtle"  ROS "C-Turtle"  ROS "Diamondback"
March 2, 2010  August 2, 2010  March 2, 2011

Future: ROS "Electric Emys"
August, 2011
ROS Meta-Filesystem
Increasing codebase flexibility

The minimal representation of a ROS package is a directory in the $ROS_PACKAGE_PATH which contains a single file:

- `manifest.xml`
The minimal representation of a ROS package is a directory in the $ROS_PACKAGE_PATH which contains a single file:

- **manifest.xml**
  - Contains package metadata (author, license, url, etc)
The minimal representation of a ROS package is a directory in the `$ROS_PACKAGE_PATH` which contains a single file:

- **manifest.xml**
  - Contains package metadata (author, license, url, etc)
  - Specifies `system` and `package` dependencies
ROS Meta-Filesystem

Increasing codebase flexibility

The minimal representation of a ROS package is a directory in the $ROS_PACKAGE_PATH which contains a single file:

- **manifest.xml**
  - Contains package metadata (author, license, url, etc)
  - Specifies system and package dependencies
  - Specifies language-specific export flags
ROS Meta-Filesystem
Increasing codebase flexibility

The minimal representation of a ROS package is a directory in the
$ROS_PACKAGE_PATH which contains a single file:

- **manifest.xml**
  - Contains package metadata (author, license, url, etc)
  - Specifies system and package dependencies
  - Specifies language-specific export flags

- **CMakeLists.txt**
  Contains ROS build rules (executables, libraries, custom build flags, etc)
ROS Meta-Filesystem
Increasing codebase flexibility

The minimal representation of a ROS package is a directory in the $ROS_PACKAGE_PATH which contains a single file:

- **manifest.xml**
  - Contains package metadata (author, license, url, etc)
  - Specifies system and package dependencies
  - Specifies language-specific export flags

- **CMakeLists.txt**
  Contains ROS build rules (executables, libraries, custom build flags, etc)

- **Makefile**
  Just a proxy to build this package
ROS Meta-Filesystem

This meta-filesystem allows ROS (rospack, specifically) to locate any package in the designated path, be it at compile time or runtime.
This meta-filesystem allows ROS (rospack, specifically) to locate any package in the designated path, be it at compile time or runtime.

Since ROS can find any package at any time, it enables packages to be moved around in the actual filesystem and greater codebase flexibility.
While ROS uses CMake (www.cmake.org) internally to compile and link code, the ROS build system it adds several useful features that make it easier to build ROS code.
While ROS uses CMake (www.cmake.org) internally to compile and link code, the ROS build system it adds several useful features that make it easier to build ROS code.

- rosbuild pulls compile and linker flags out of ROS package manifests, so compiling against other ROS code is as easy as specifying the package name.
While ROS uses CMake (www.cmake.org) internally to compile and link code, the ROS build system it adds several useful features that make it easier to build ROS code.

- **rosmake** pulls compile and linker flags out of ROS package manifests, so compiling against other ROS code is as easy as specifying the package name.
- **rosdep** can be used to install system dependencies specified in a package's manifest.xml.
While ROS uses CMake ([www.cmake.org](http://www.cmake.org)) internally to compile and link code, the ROS build system it adds several useful features that make it easier to build ROS code.

- **robuild** pulls compile and linker flags out of ROS package manifests, so compiling against other ROS code is as easy as specifying the package name.
- **rosdep** can be used to install system dependencies specified in a package’s `manifest.xml`.
- **ROS CMake macros** enable rapid building of executables, libraries, and automated regression tests.
Thank you!

Now proceed to the ROS Beginner Tutorials!

http://www.ros.org/wiki/ROS/Tutorials
Thank you!
Now proceed to the ROS Beginner Tutorials!
http://www.ros.org/wiki/ROS/Tutorials
The ROS Community

Researchers using common tools to enable collaboration

79 Institutional ROS Repositories, all over the world (July, 2011)
(jhu-lcsr-ros-pkg would make 80)
www.ros.org - The ROS Hub
A centralized location for ROS users and developers
answers.ros.org - ROS Questions & Answers

Community-supported help for ROS users
Introduction ROS Community

code.ros.org - Willow Garage, Inc Code
Hosting and project management for “official” packages

code.ros.org is a development site for ROS-related software, including the ros-plg and willow-plg projects. ros-plg provides general robot software capabilities, while willow-plg is a repository for Willow Garage-related software, including the PR2 robot.

Repositories

The following ROS-related repositories are hosted on code.ros.org:
- ros: ROS core software platform
- ros-plg: general robot software libraries for ROS, including navigation, 3-D visualization, coordinate transforms, drivers, and more.
- willow-plg: PR2 software, Willow Garage research, and other Willow Garage-related software.
- openCV: OpenCV computer vision library
- xortl: Karto mapping library from KHR International

Other Repositories

There are many repositories of ROS-related software available as open source. Please see http://ros.org/wiki/Repositories for the most up-to-date list.

Bug Reports/Feature Requests

Please see the Tickets page on the ROS wiki.

Mailing List Archives

Please see the list archives.

Recent News

No news items found.
ros mailing lists
Getting in touch with the developer community

- ROS Users - *for general ROS-related discussions*
  https://code.ros.org/mailman/listinfo/ros-users

- ROS Developers - *for ROS core development*
  https://code.ros.org/mailman/listinfo/ros-developers

- Other Lists & List Archives
  http://code.ros.org/lurker