Motivation

Today's robotic systems are complex.
Many sensors.
Highly distributed, many processes, many computers.
Teams of engineers.

⇒ ROS — The Robot Operating System.
Outline

Overview

ROS Communication Layer

ROS Build System

Programming with ROS

The TF Library

What is ROS?

More than just a 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
- A language-independent architecture (c++, python, lisp, java, and more)

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
What does ROS get you?
All levels of development

universe

main

ROS

algorithms frameworks
hardware drivers
"robotic apps"

general tools for
distributed computing

Maintained by
Willow Garage, inc and
some external developers

Overview ROS Communication Layer ROS Build System Programming with ROS The TF Library
Lorenz Mösenlechner Introduction to ROS

What does ROS get you?
All levels of development

- Developed and maintained by the international ROS community
- universe
- algorithms, frameworks, hardware drivers, "robotic apps"
- general tools for distributed computing

Maintained by Willow Garage, Inc and some external developers

### Overview
- ROS Communication Layer
- ROS Build System
- Programming with ROS
- The TF Library

---

**Introduction to ROS**

Lorenz Mösenlechner
What does ROS get you?
All levels of development

The ROS Community
Researchers using common tools to enable collaboration

www.ros.org - The ROS Hub
A centralized location for ROS users and developers

79 Institutional ROS Repositories, all over the world (July, 2011)
Intelligent Autonomous Systems

answers.ros.org - ROS Questions & Answers
Community-supported help for ROS users

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

Outline

Overview
ROS Communication Layer
ROS Build System
Programming with ROS
The TF Library

ROS Core
Where it all comes together

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

*Named network resources*

ROS graph resources:

- **nodes**
  - processes
  - produce and consume data

- **parameters**
  - persistent data storage
  - configuration, initialization settings
  - stored on parameter server

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

- **services**
  Synchronous one-to-many network-based functions.

---

**Creating and Running ROS Nodes**

Distributing computation with ROS

Launch files

- XML files for launching nodes
- 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

---

**Example Launch File**

*Example launch file*

```xml
<launch>
  <node name="my_node" pkg="foo" type="bar">
    <remap from="/base_laser/scan" to="scan" />
    <rosparam>
      use_foo: True
      frame_id: base_laser
    </rosparam>
  </node>
</launch>
```

- Launch with `roslaunch package foo.launch`
**ROS Communication Protocols**

Connecting nodes over the network

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

- **ROS Services**
  - Synchronous “function-call-like” communication
  - Strongly-typed (ROS .srv spec)
  - Can have only one server
  - Can have one or more clients

- **Actions**
  - Built on top of topics
  - Long running processes
  - Cancellation

---

**Asynchronous Distributed Communication**

**ROS TCP Topics**

advertise("images")

```python
ros
"master"
```

- camera
- viewer

---
Asynchronous Distributed Communication

ROS TCP Topics

```
ros
"master"

subscribe("images")
```

camera

```
ros
"master"

topic:images
```

viewer

```
images(tcp)
```

```
publish(img)
```
Asynchronous Distributed Communication
ROS TCP Topics

Overview
ROS Communication Layer
ROS Build System
Programming with ROS
The TF Library

Lorenz Mösenlechner
Introduction to ROS
37

Asynchronous Distributed Communication
ROS TCP Topics

Overview
ROS Communication Layer
ROS Build System
Programming with ROS
The TF Library

Lorenz Mösenlechner
Introduction to ROS
38

Asynchronous Distributed Communication
ROS TCP Topics

Overview
ROS Communication Layer
ROS Build System
Programming with ROS
The TF Library

Lorenz Mösenlechner
Introduction to ROS
39

Asynchronous Distributed Communication
ROS TCP Topics

Overview
ROS Communication Layer
ROS Build System
Programming with ROS
The TF Library

Lorenz Mösenlechner
Introduction to ROS
40
ROS Graph Introspection
No more wireshark

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

- `rosnode`
  Gives a user information about a node: publications, subscriptions, etc
- `rostopic`
  Gives datarate, actual data, publishers, subscribes
- `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...
rviz - 3D Visualization
Modular state and sensor visualization