ROS Workshop

Lorenz Mösenlechner
Technische Universität München

July 18th, 2012

Outline

Outline

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*)

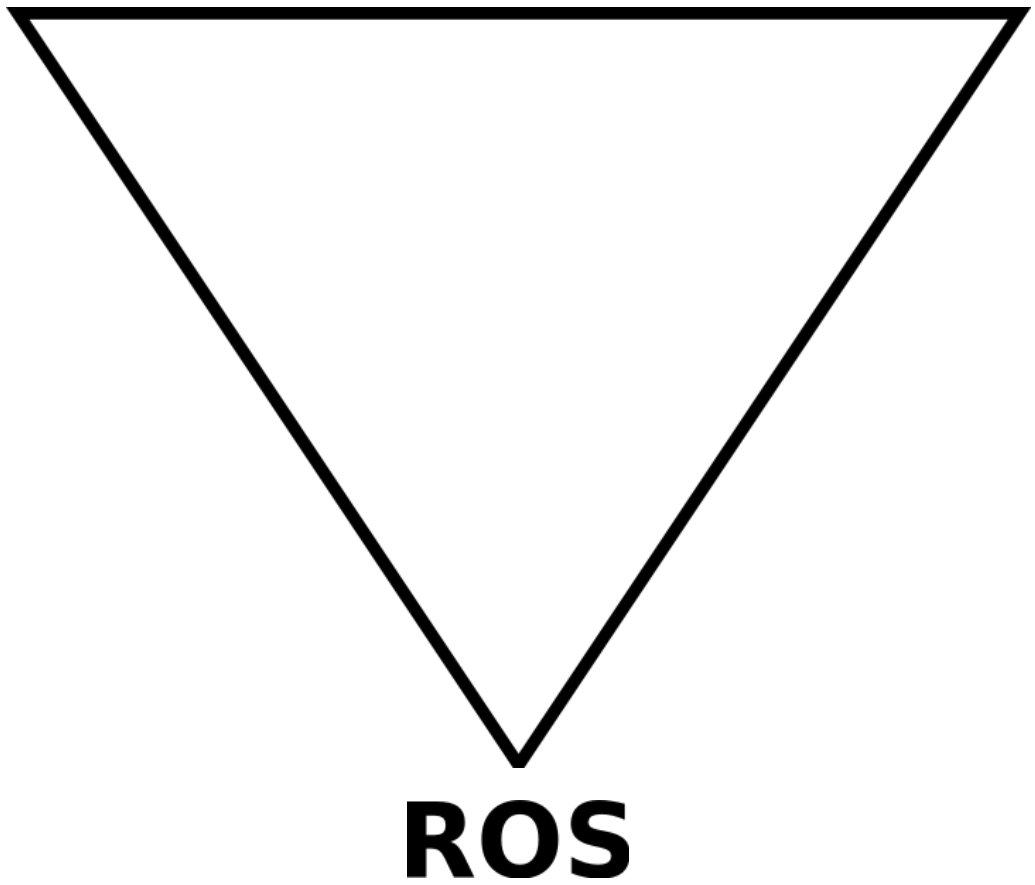
What is ROS not?

No confusion

- An *actual* operating system
- A programming language
- A programming environment / IDE
- A hard real-time architecture

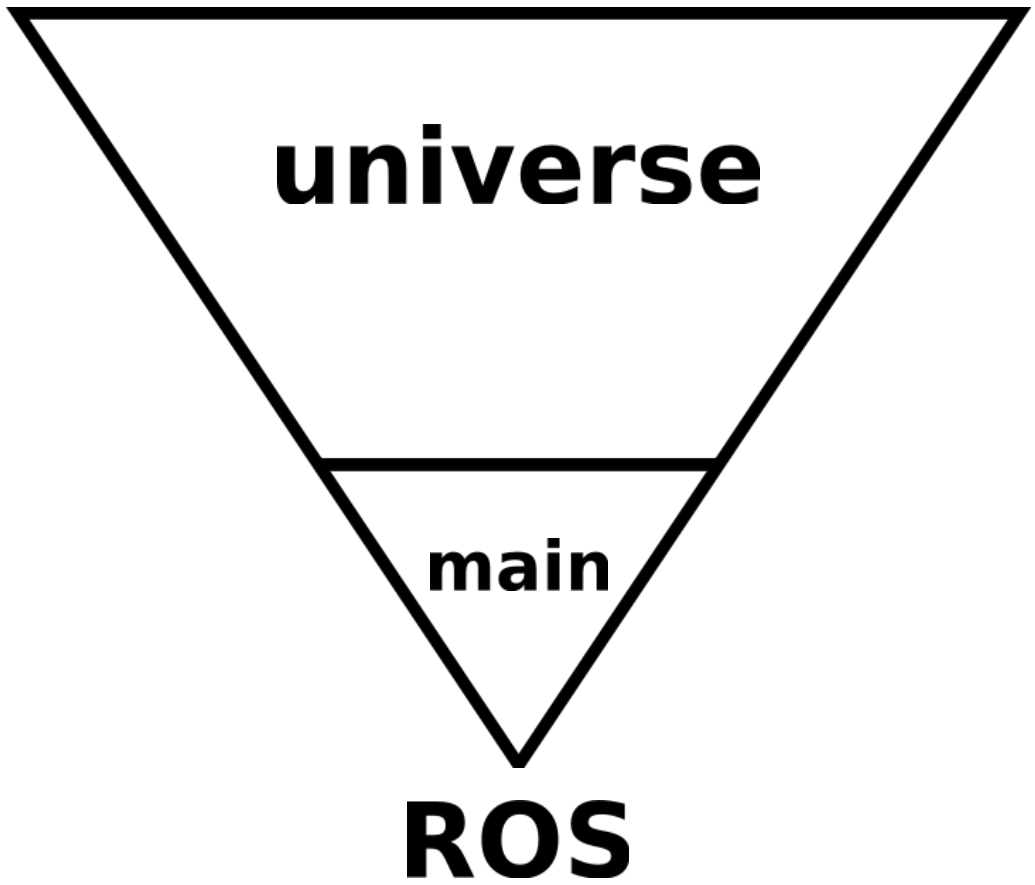
What does ROS get you?

All levels of development



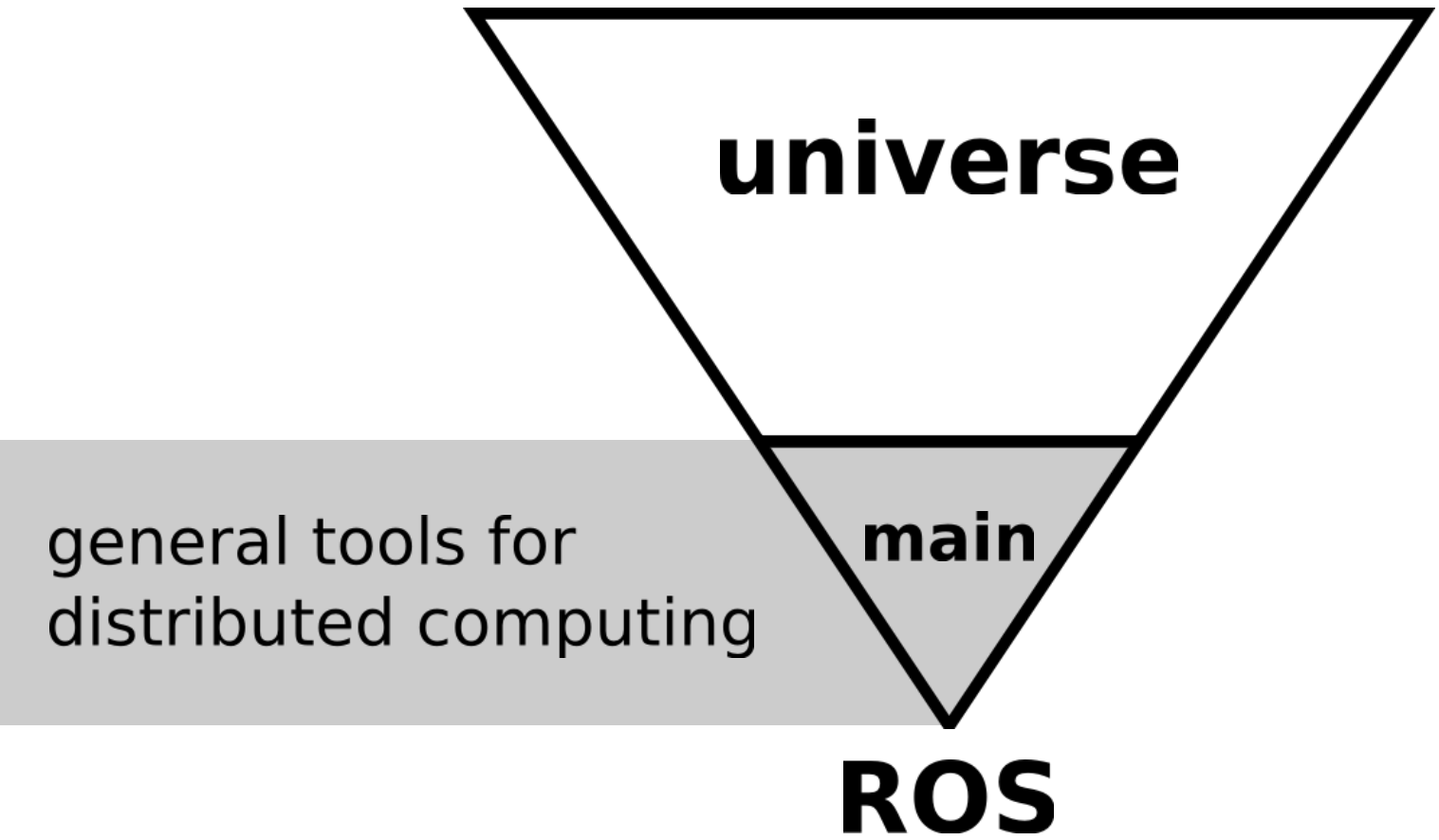
What does ROS get you?

All levels of development



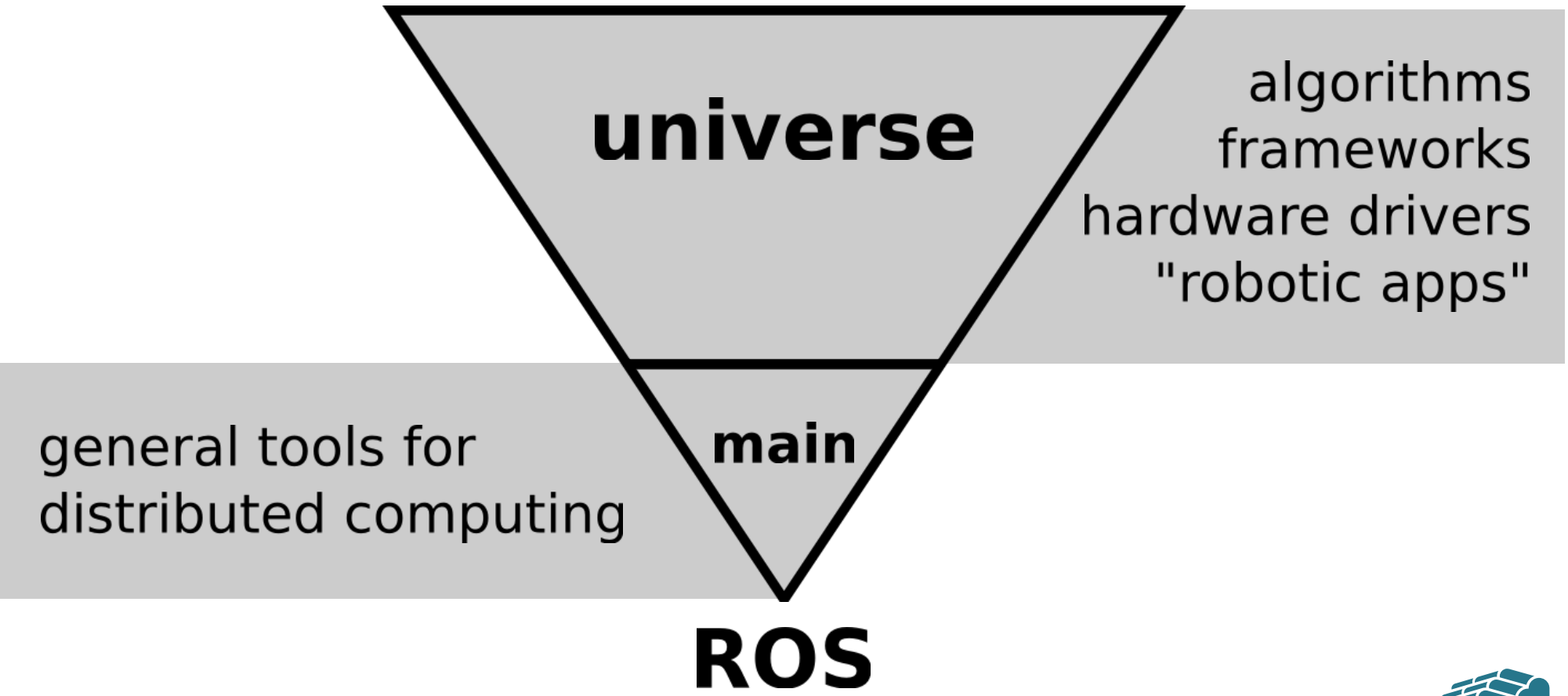
What does ROS get you?

All levels of development



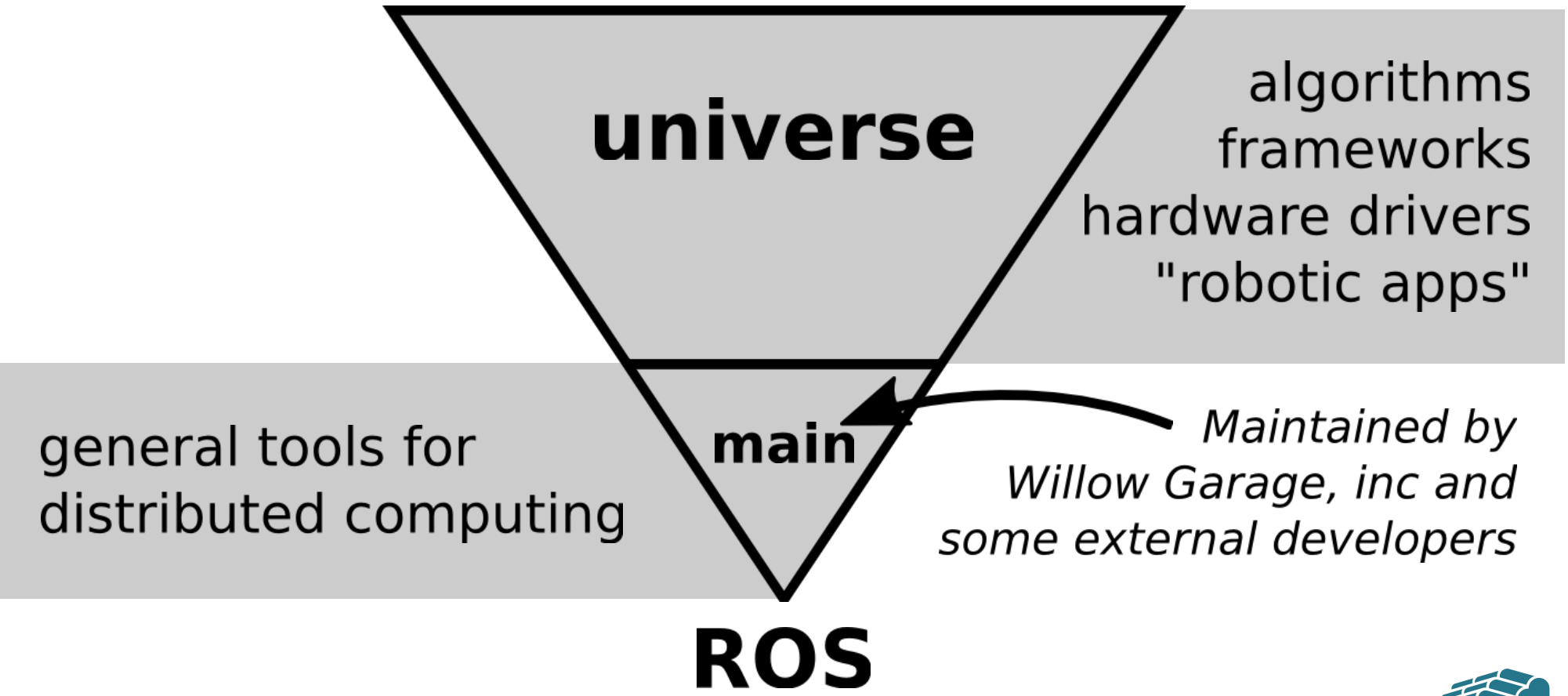
What does ROS get you?

All levels of development



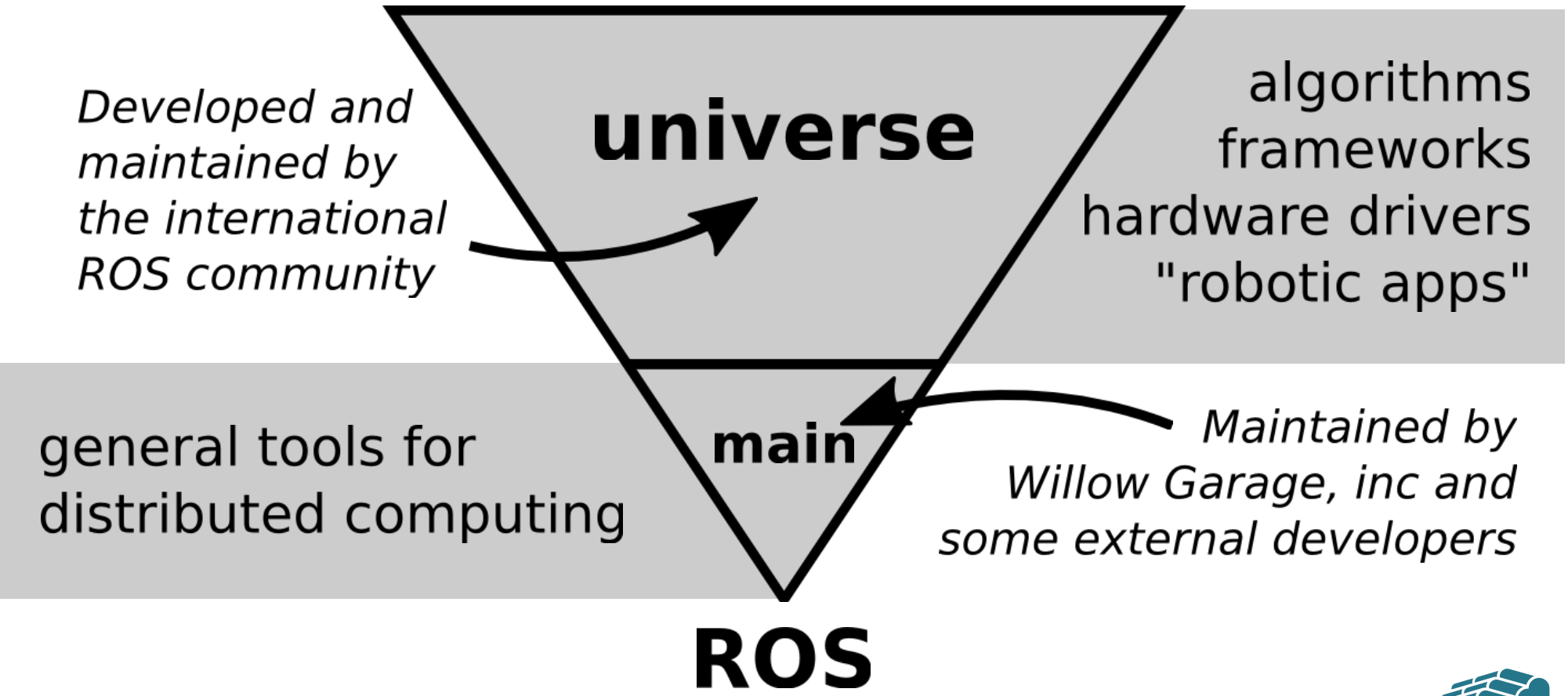
What does ROS get you?

All levels of development



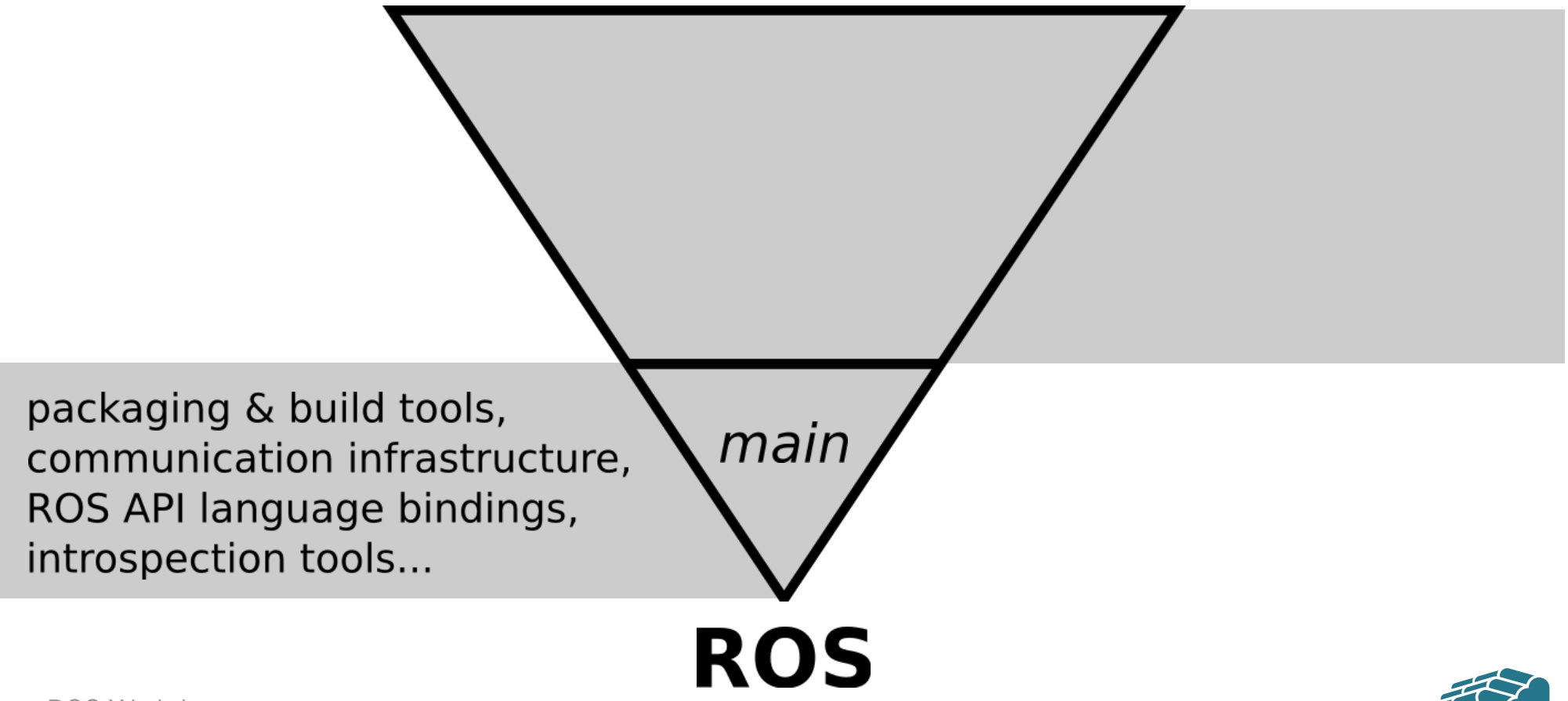
What does ROS get you?

All levels of development



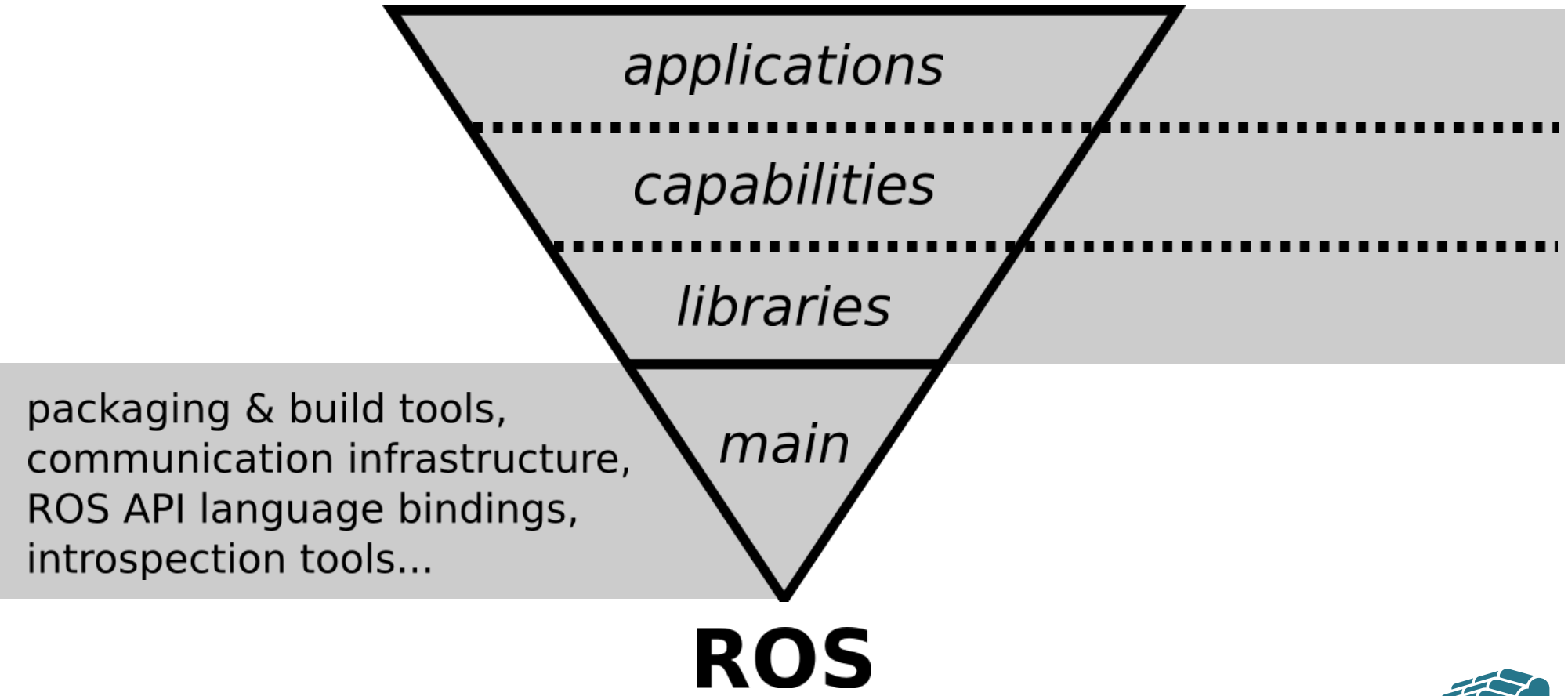
What does ROS get you?

All levels of development



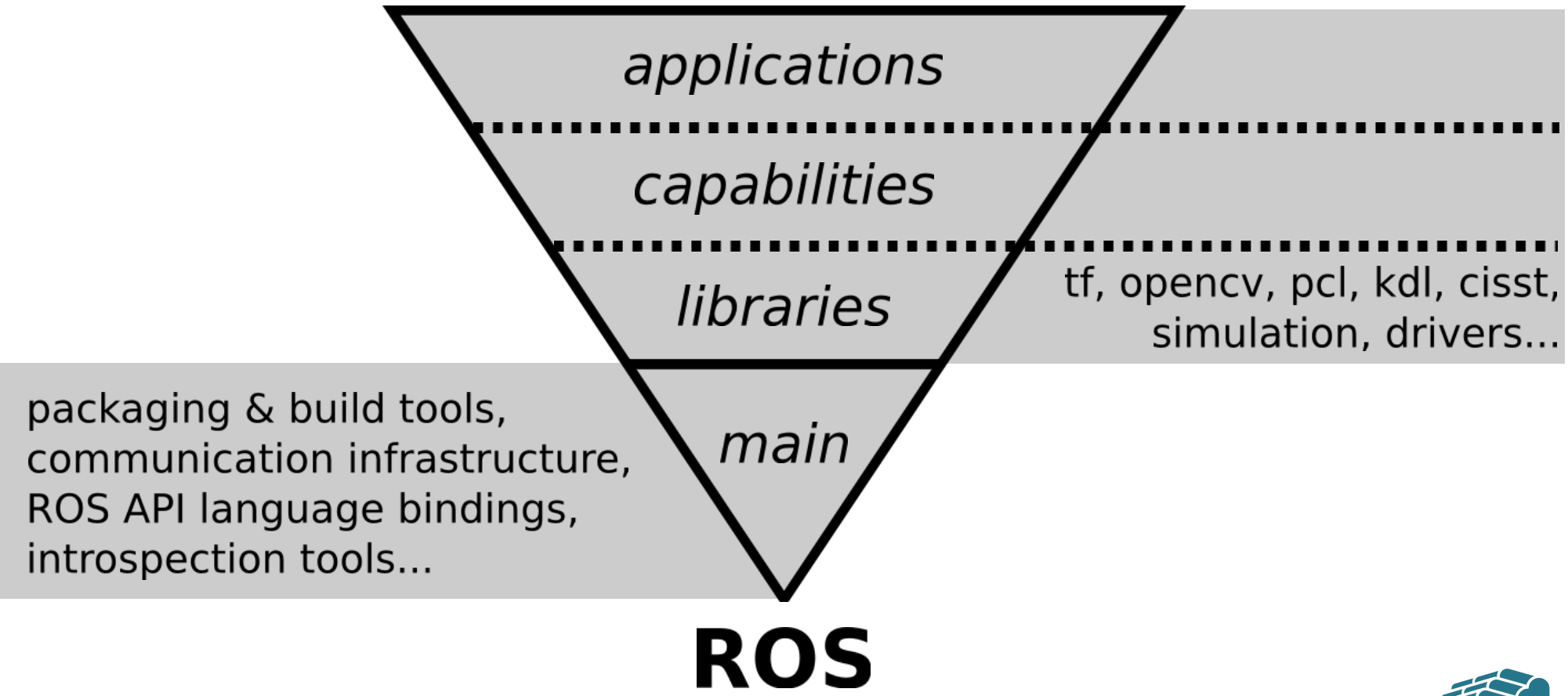
What does ROS get you?

All levels of development



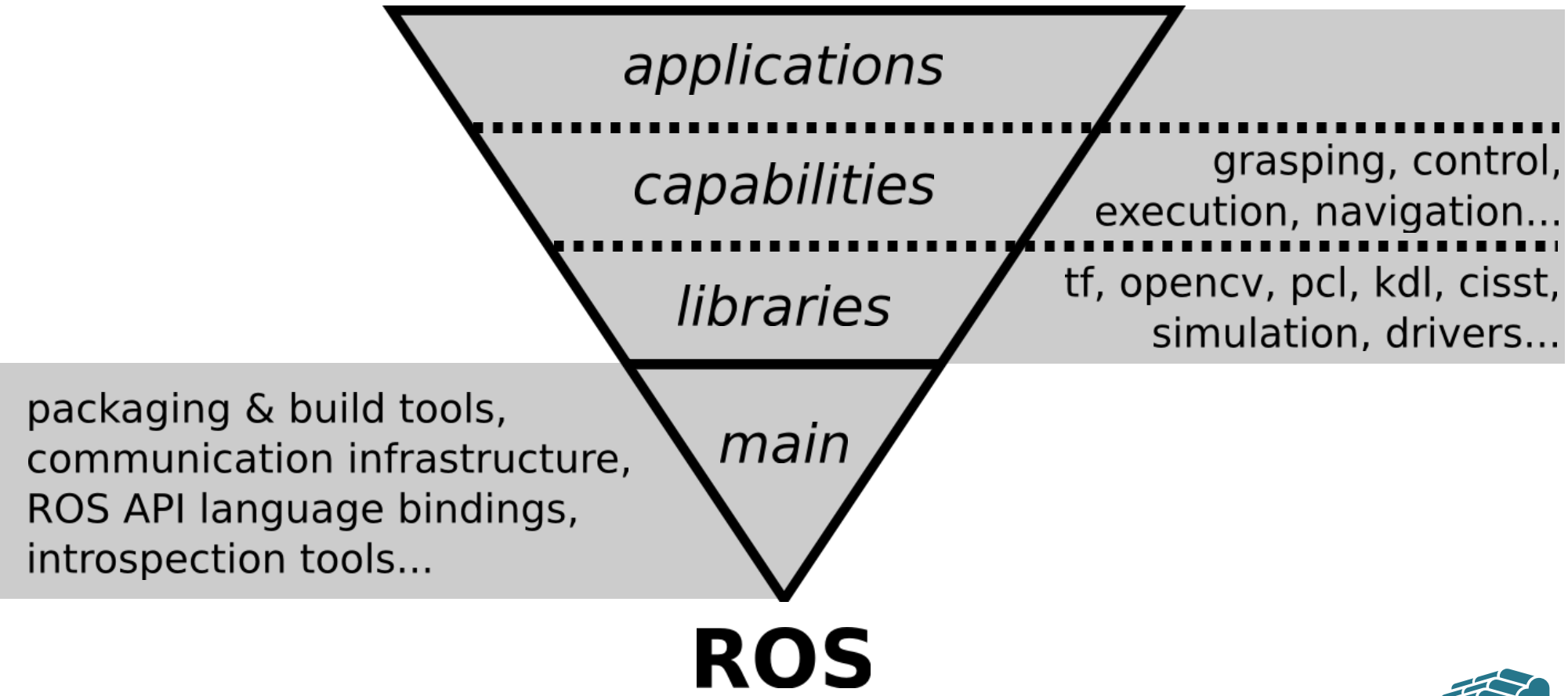
What does ROS get you?

All levels of development



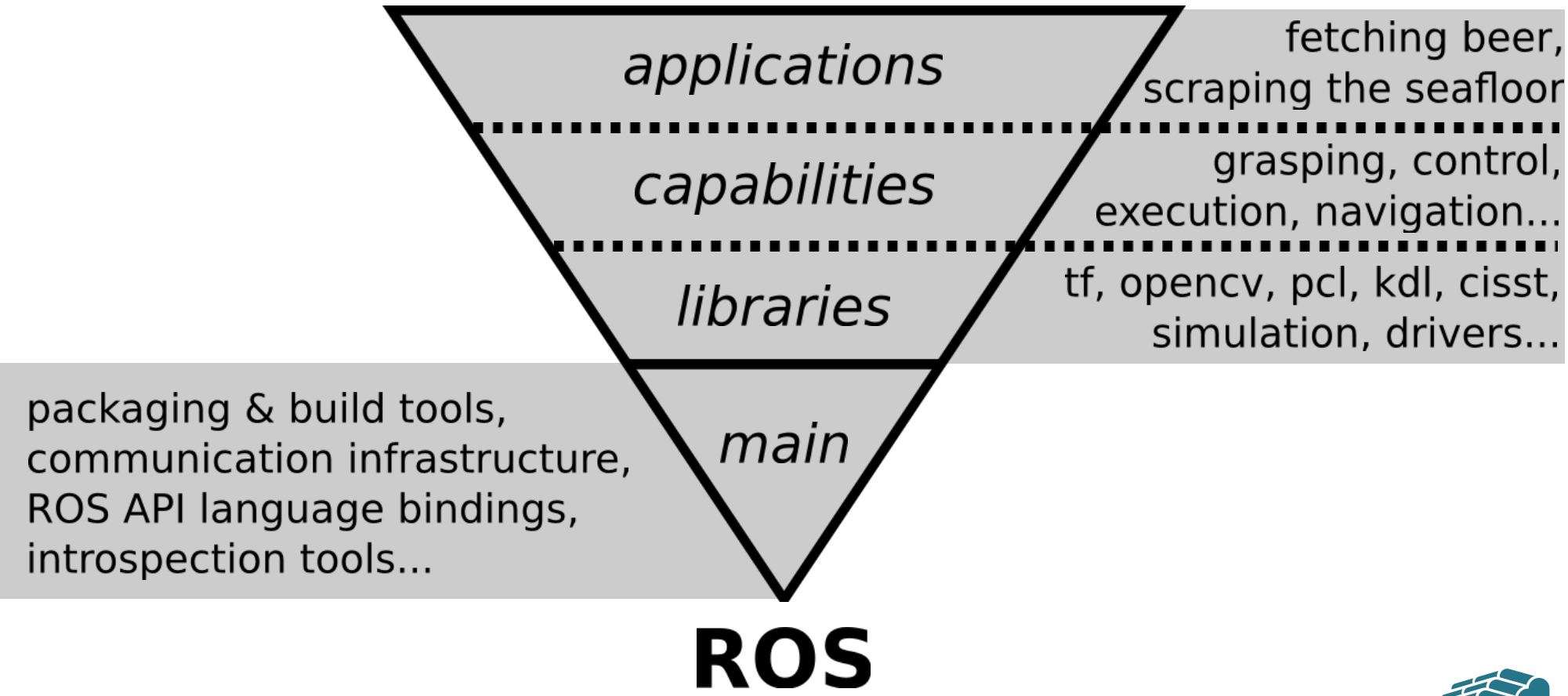
What does ROS get you?

All levels of development



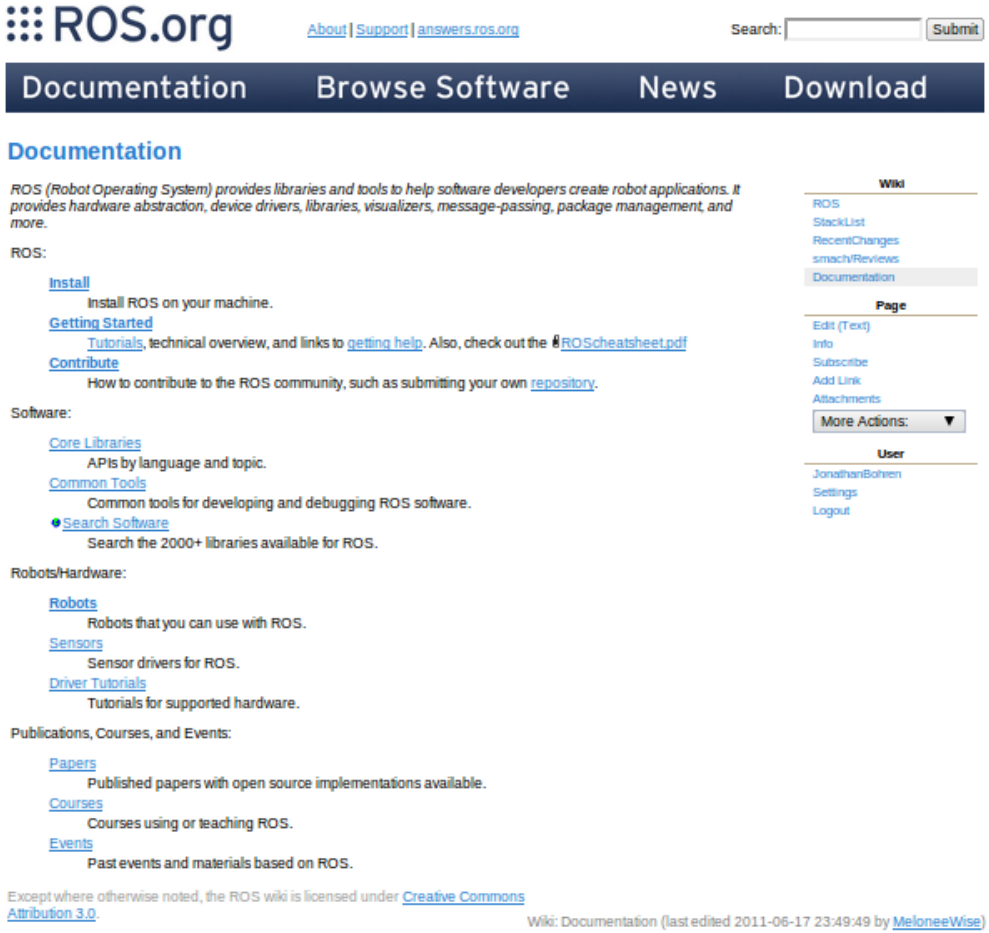
What does ROS get you?

All levels of development



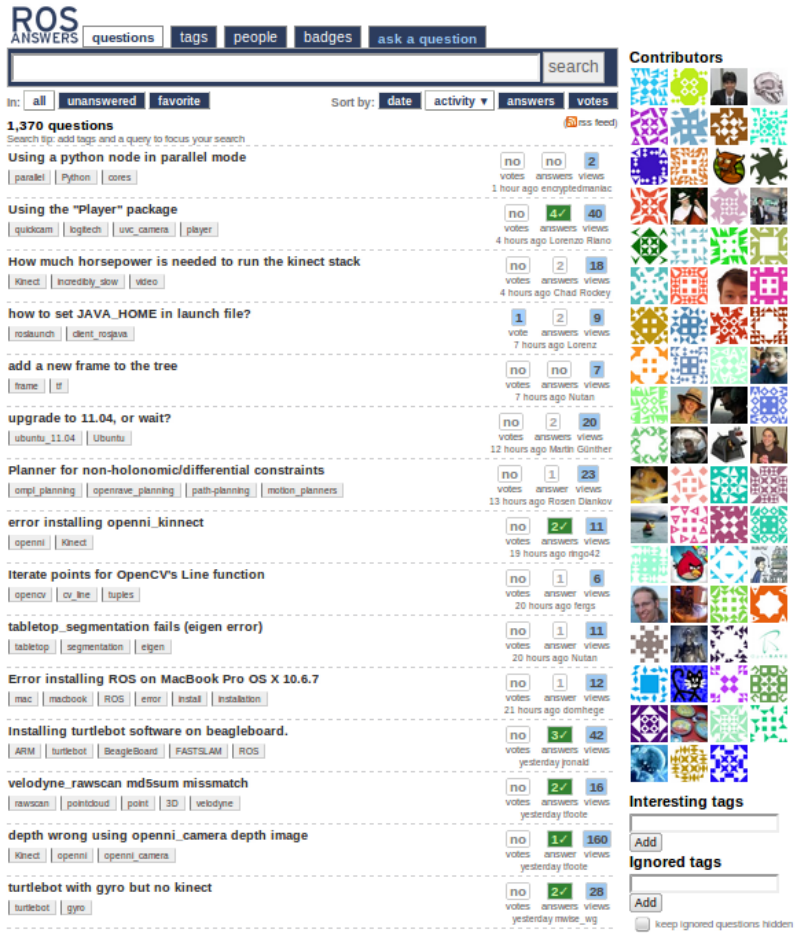
www.ros.org - The ROS Hub

A centralized location for ROS users and developers



answers.ros.org - ROS Questions & Answers

Community-supported help for ROS users



Outline

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
- rosout

  Essentially a network-based `stdout` for human-readable messages

ROS "Graph" Abstraction

Named network resources

ROS graph resources:

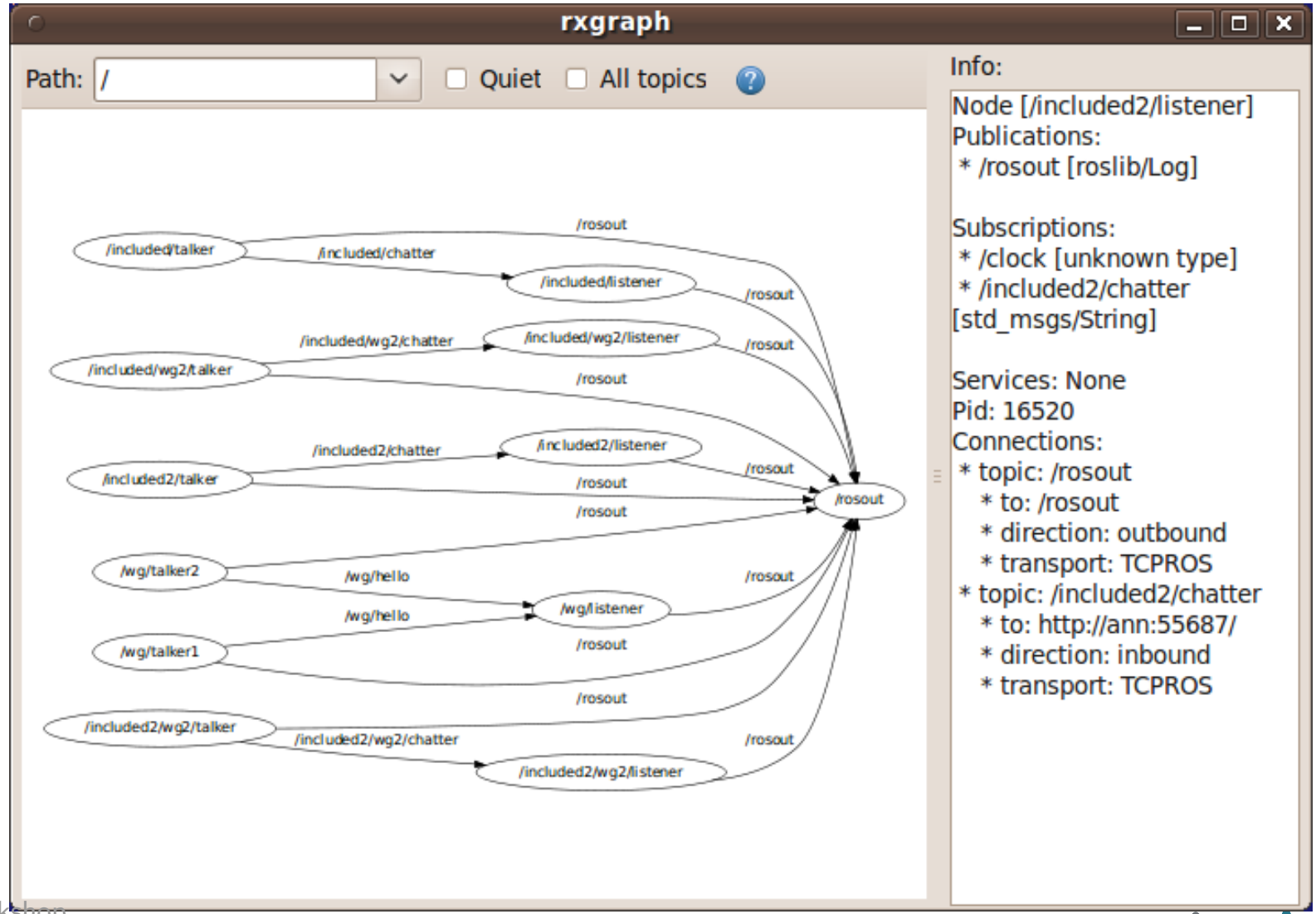
- nodes
  - processes
  - produce and consume data
- parameters
  - persistent data storage
  - configuration, initialization settings
  - stored on parameter server
- topics

  Asynchronous many-to-many communication streams.
- services

  Synchronous one-to-many network-based functions.

ROS "Graph"

rxgraph: communication network visualization



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

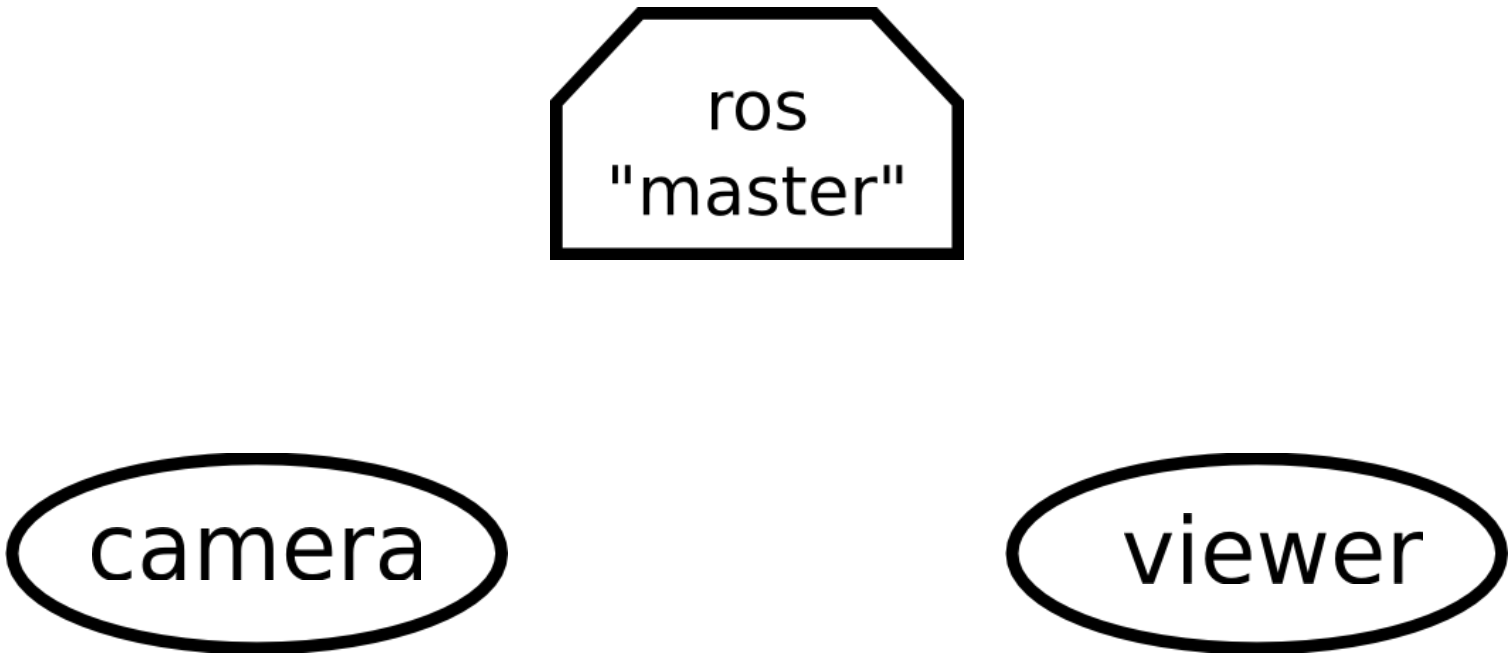
ROS Communication Protocols

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

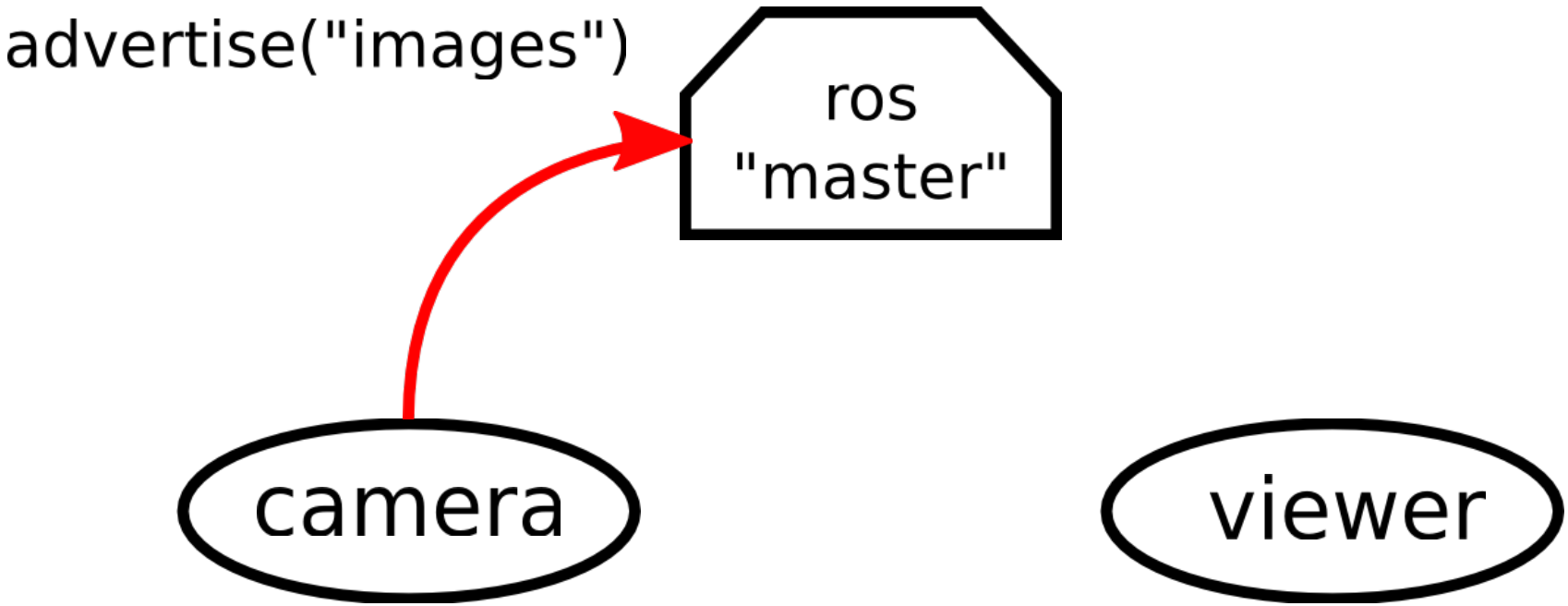
Asynchronous Distributed Communication

ROS TCP Topics



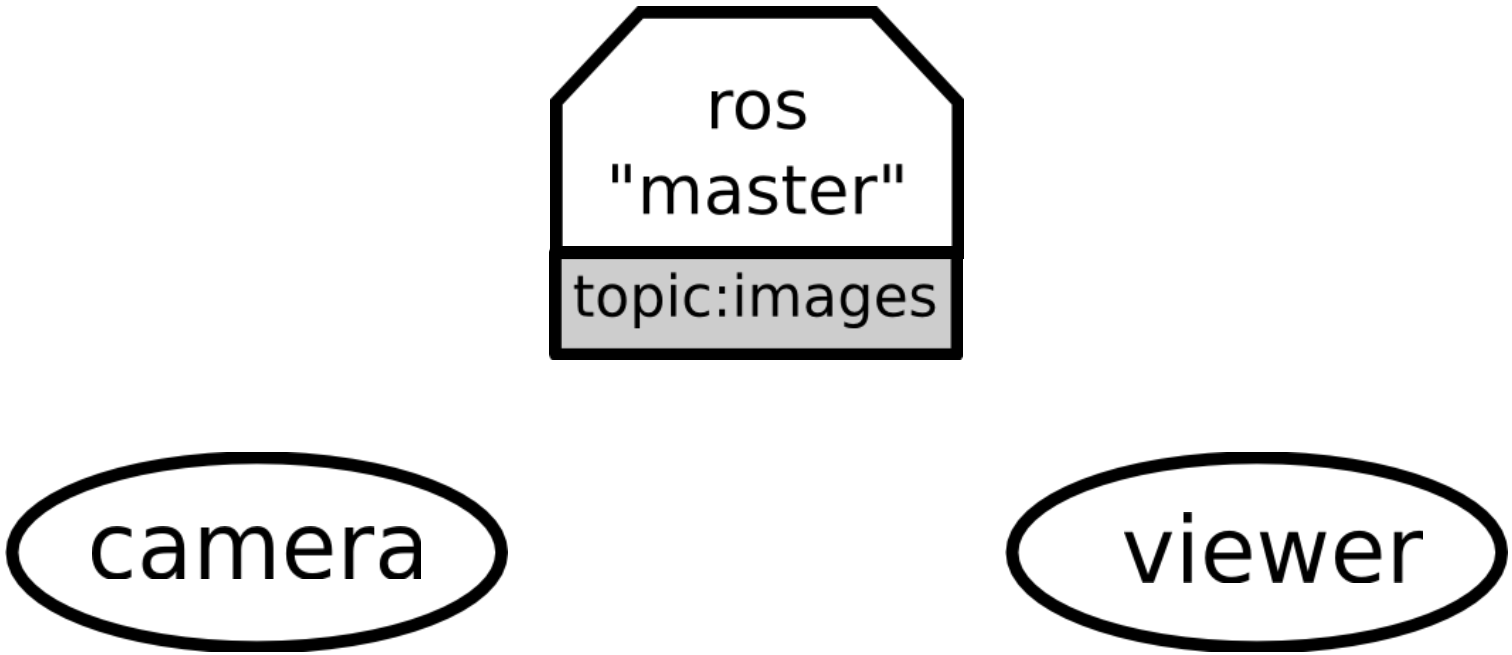
Asynchronous Distributed Communication

ROS TCP Topics



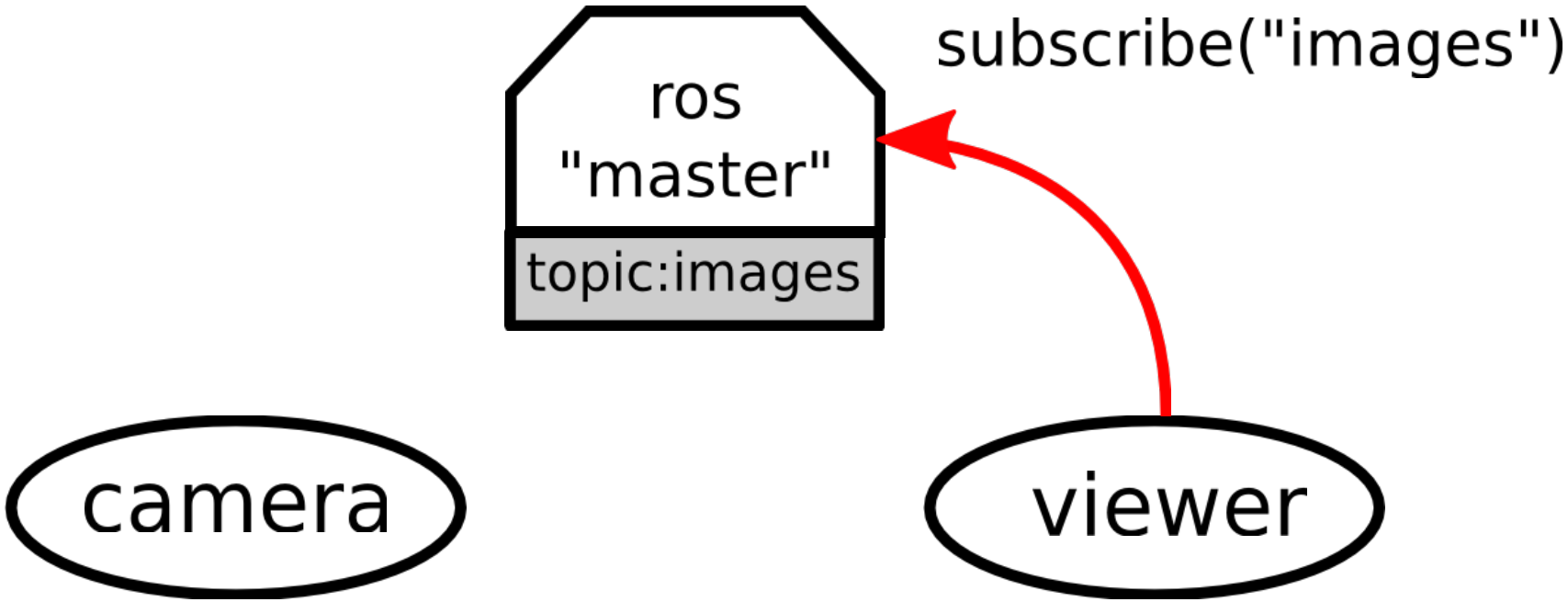
Asynchronous Distributed Communication

ROS TCP Topics



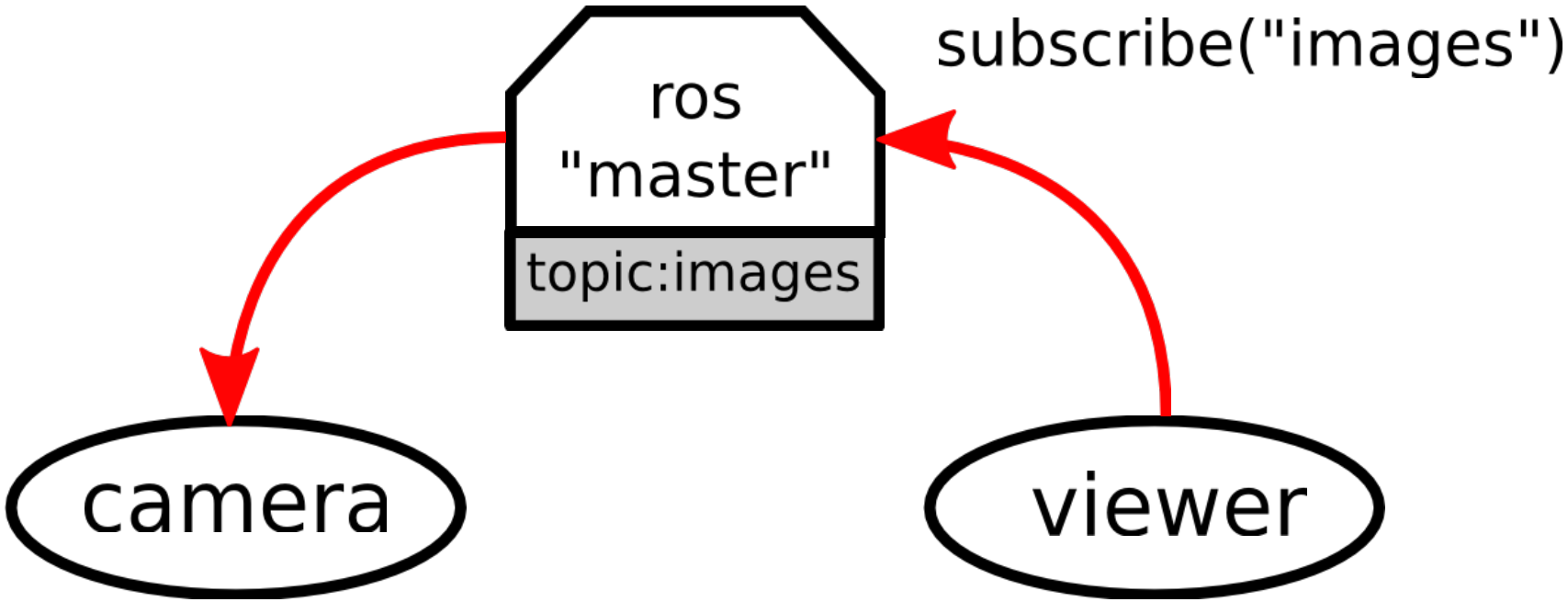
Asynchronous Distributed Communication

ROS TCP Topics



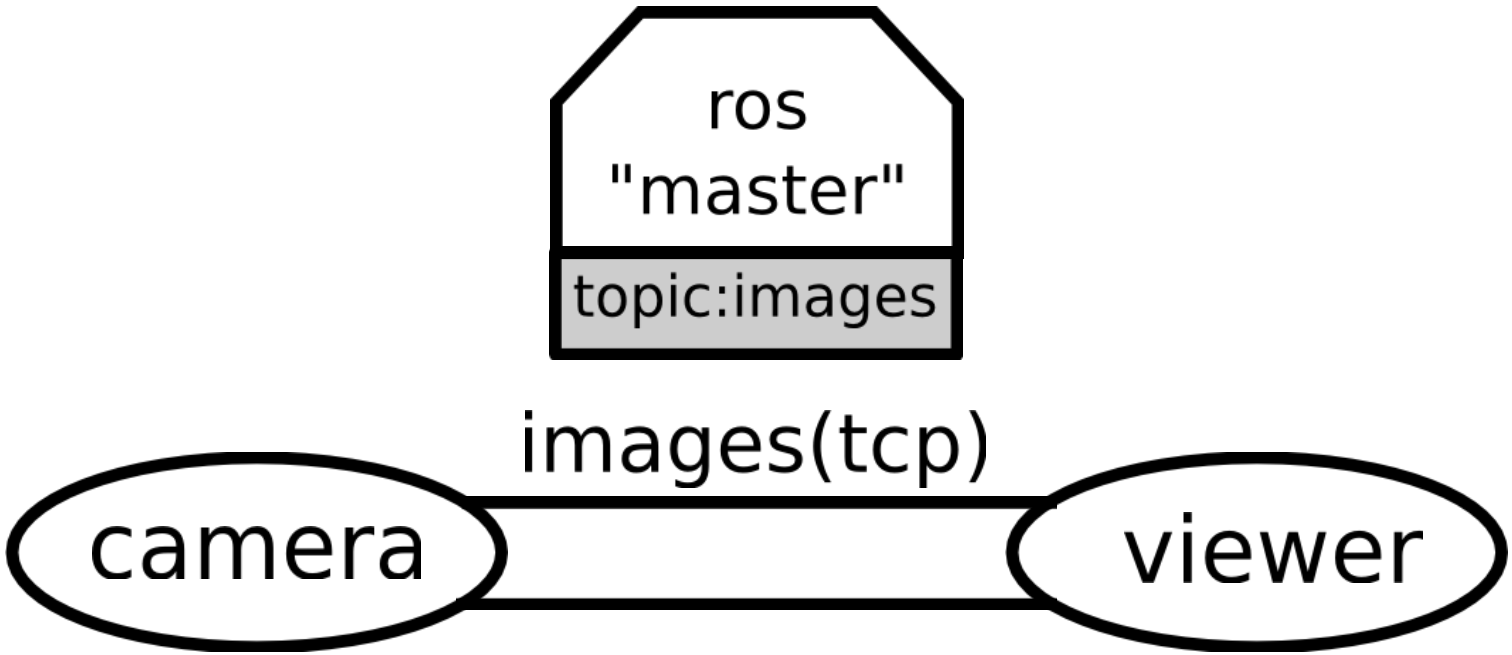
Asynchronous Distributed Communication

ROS TCP Topics



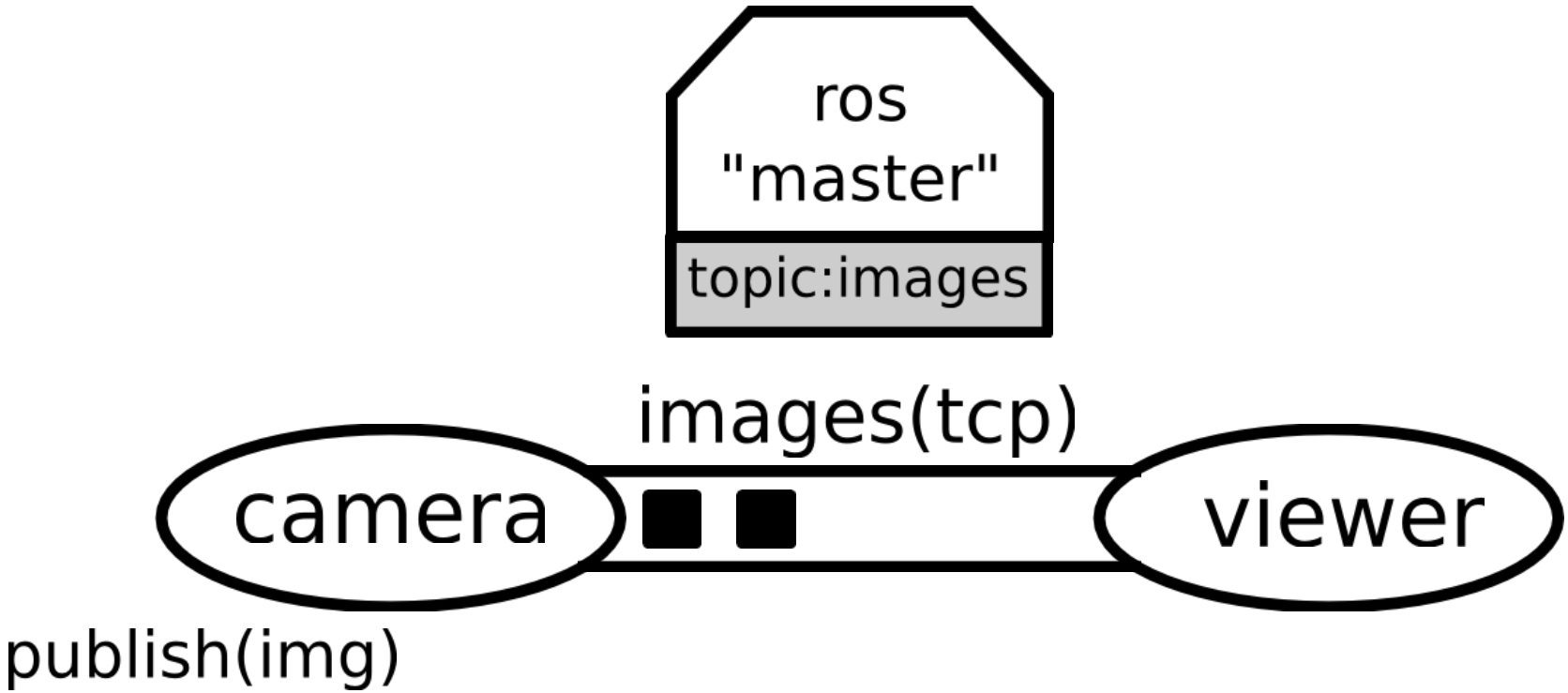
Asynchronous Distributed Communication

ROS TCP Topics



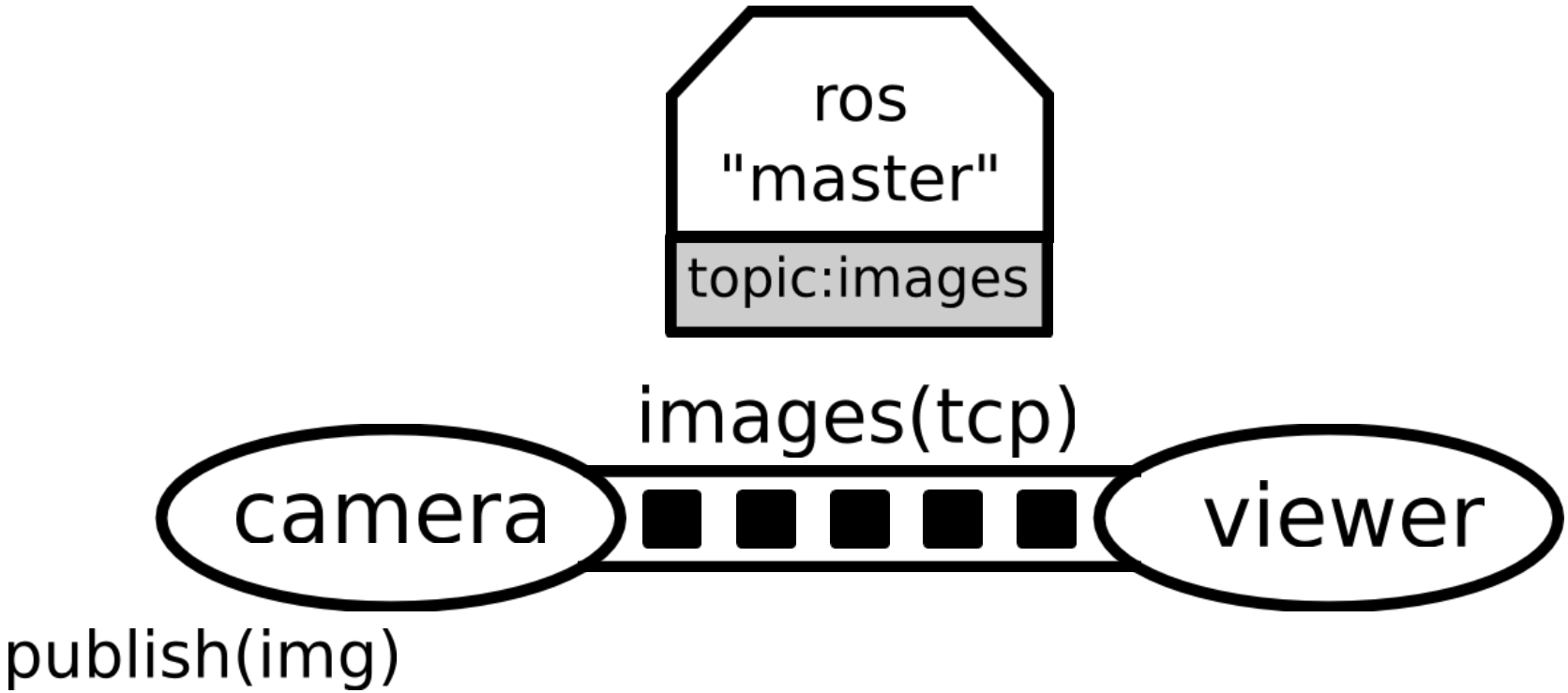
Asynchronous Distributed Communication

ROS TCP Topics



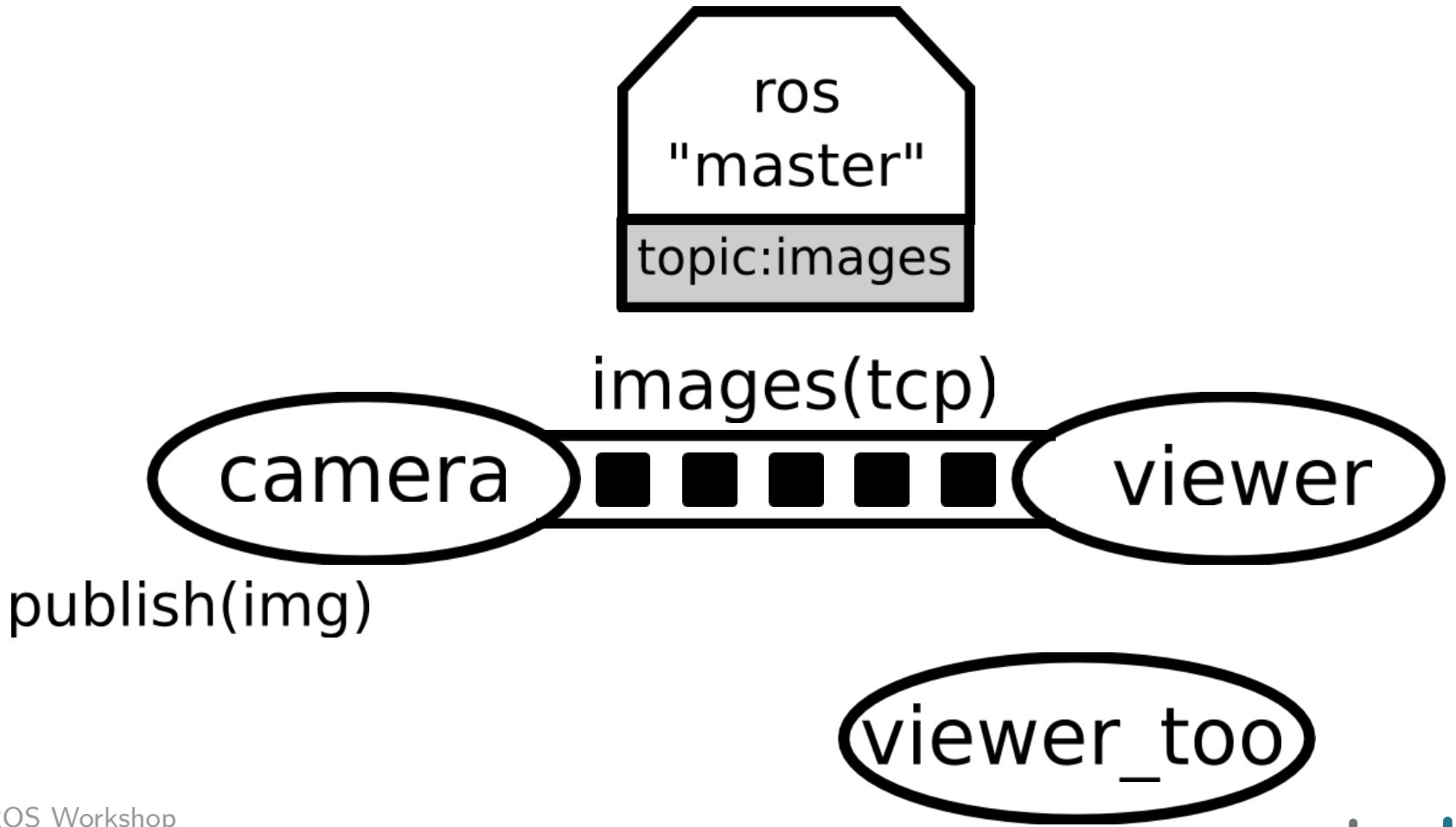
Asynchronous Distributed Communication

ROS TCP Topics



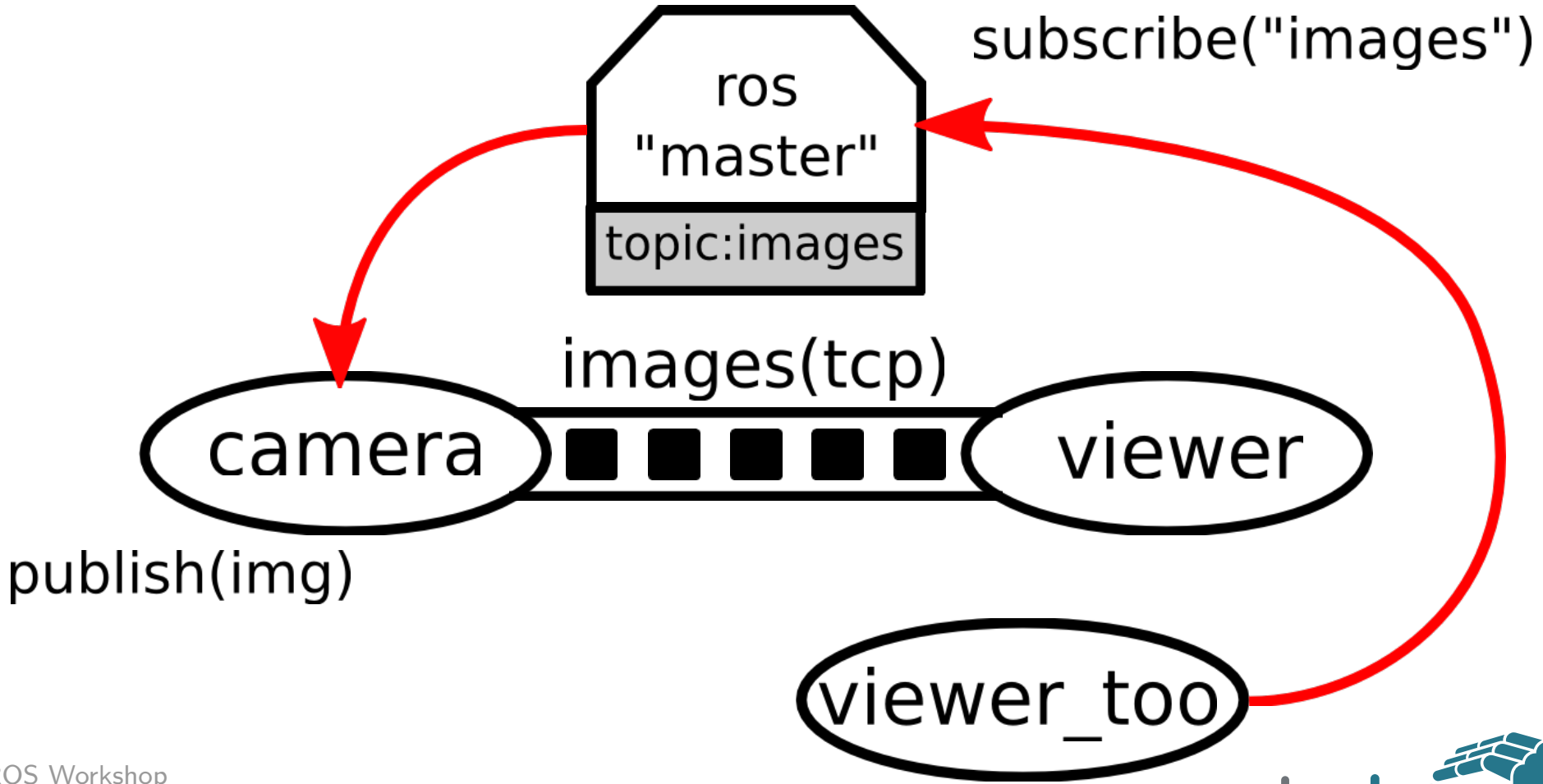
Asynchronous Distributed Communication

ROS TCP Topics



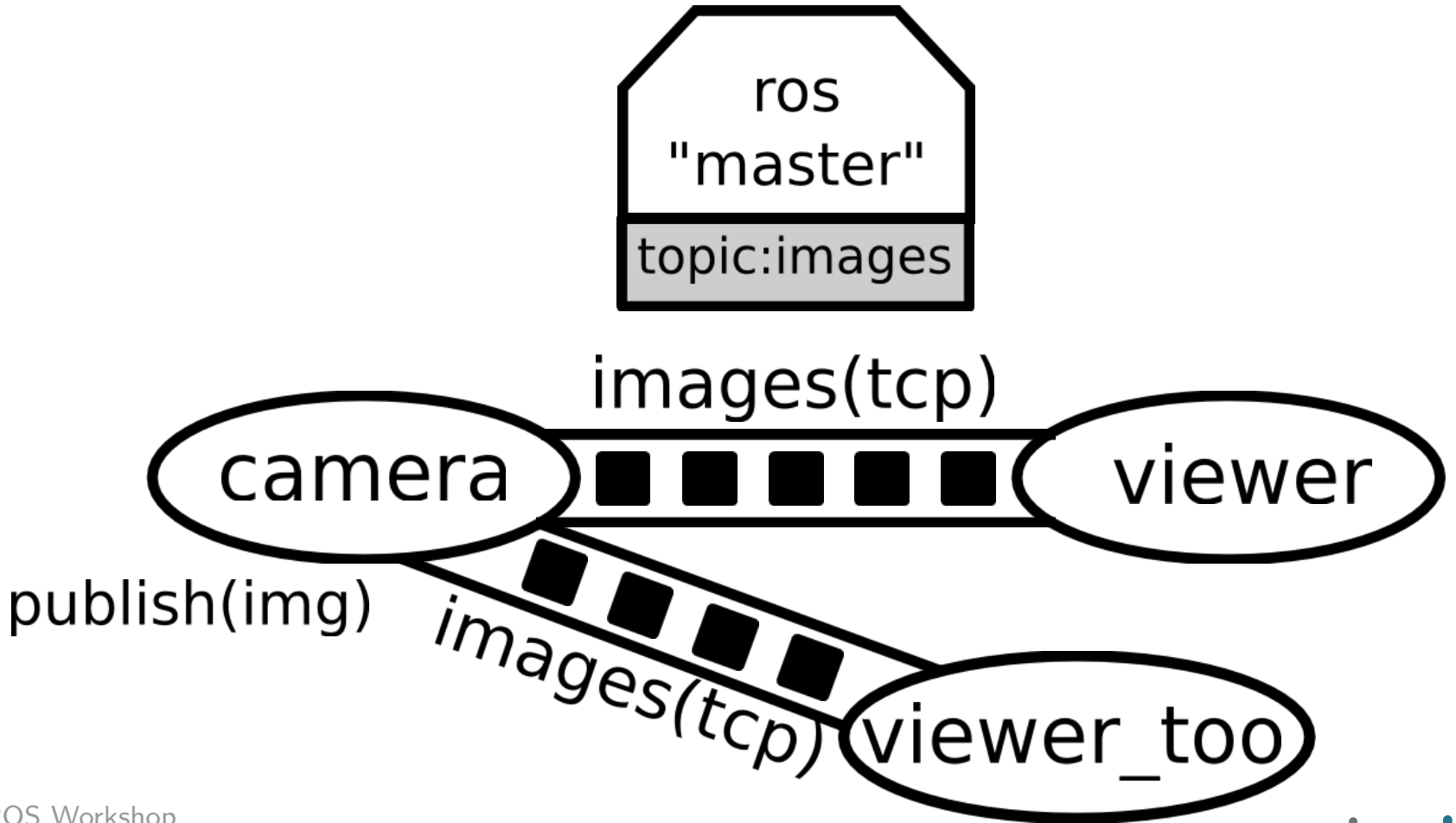
Asynchronous Distributed Communication

ROS TCP Topics



Asynchronous Distributed Communication

ROS TCP Topics



Debugging

rosout

ROS provides mechanisms in all languages for specifying different *levels* of human-readable log messages.

The five default levels are:

- fatal
- error
- warn
- info
- debug

Corresponding logging commands (C++):

- `ROS_FATAL(...)`
- `ROS_ERROR(...)`
- `ROS_WARN(...)`
- `ROS_INFO(...)`
- `ROS_DEBUG(...)`

ROS Graph Introspection

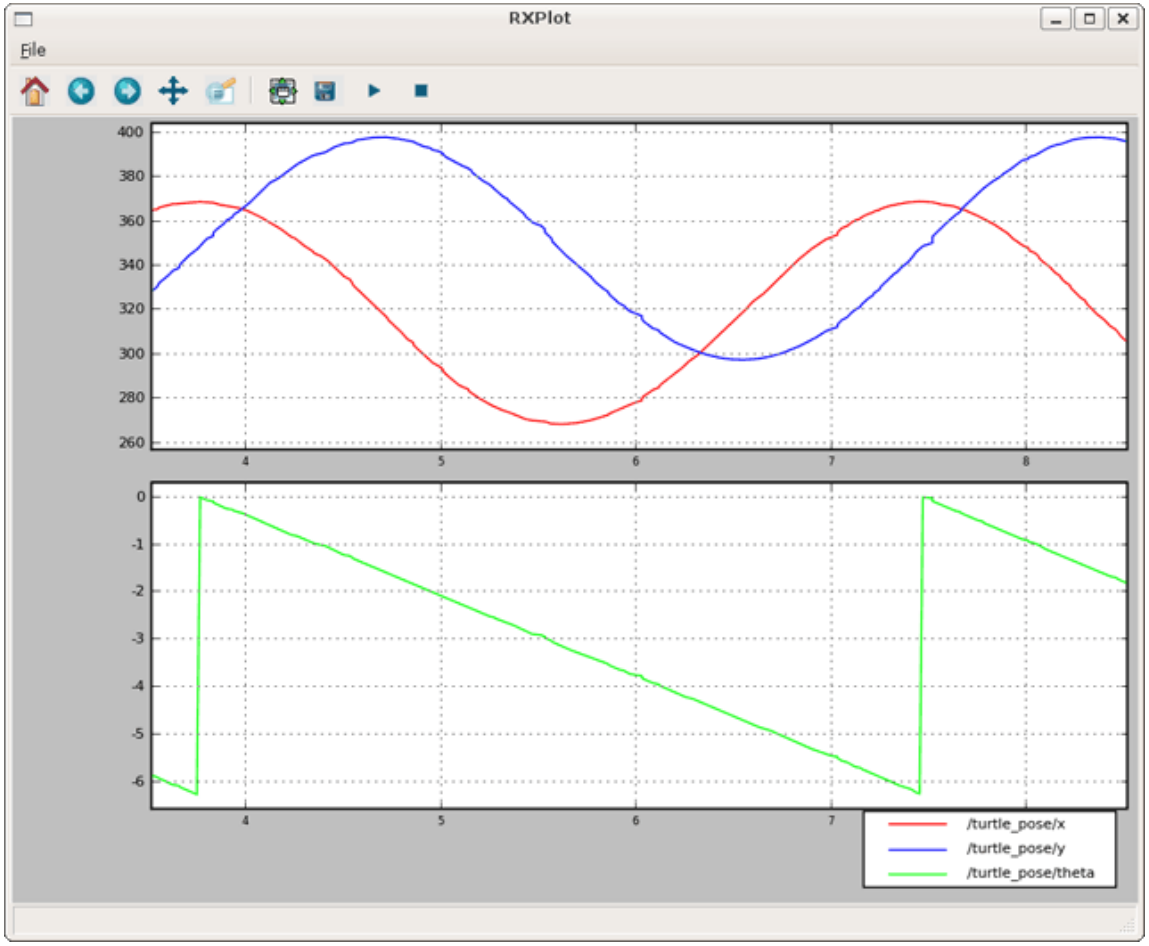
No more wireshark

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

- rosnode
  Gives a user infomation 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

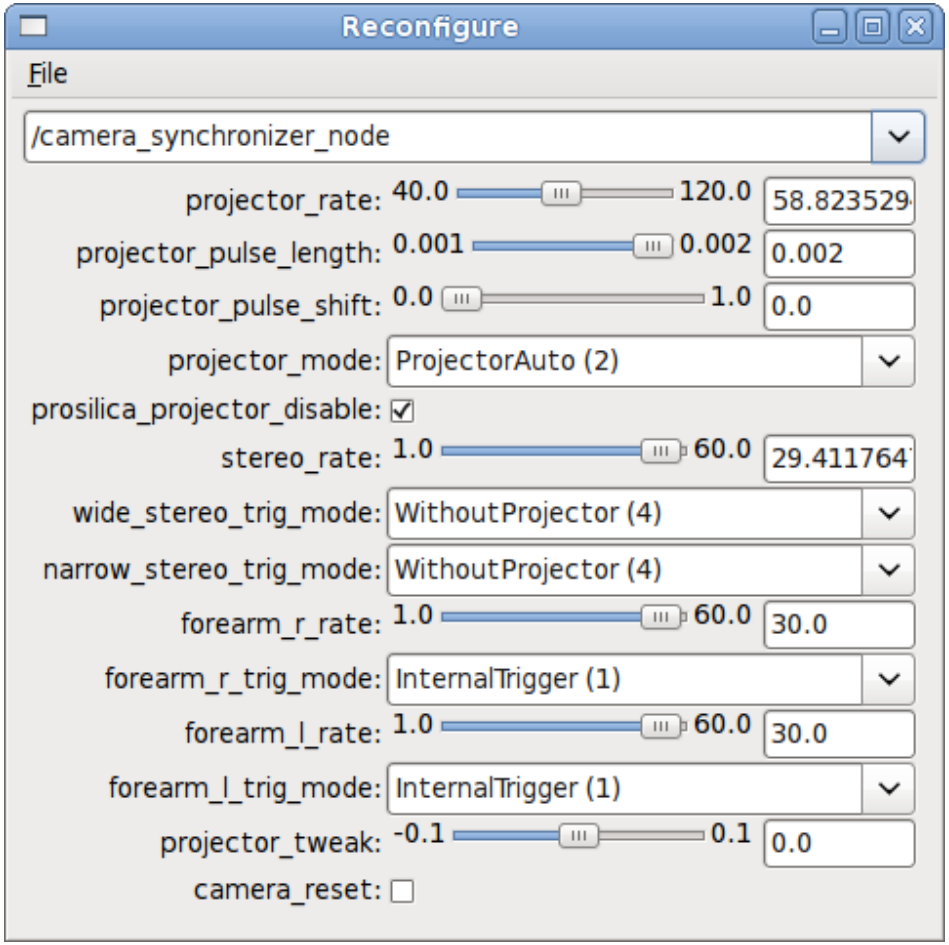
ROS GUI Tools

There are lots...



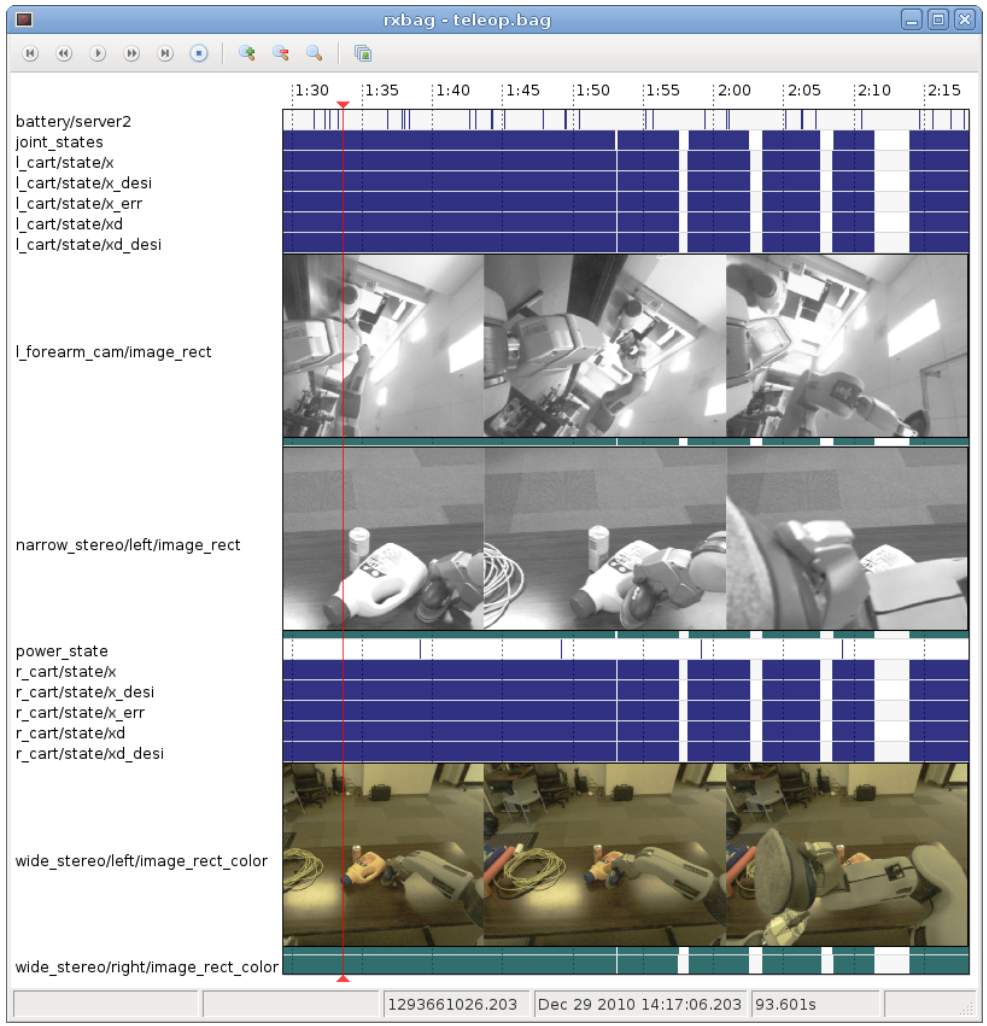
ROS GUI Tools

There are lots...



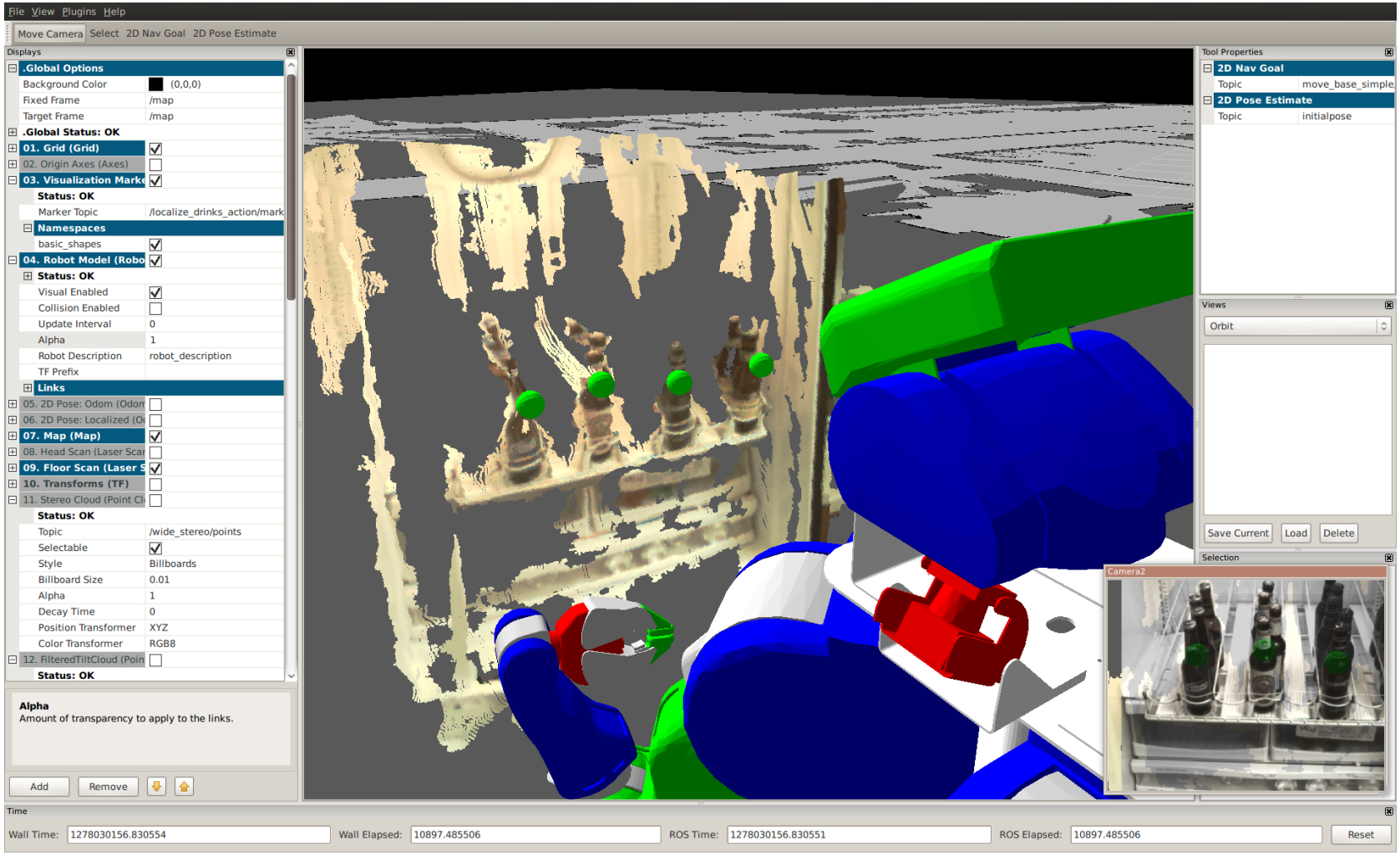
ROS GUI Tools

There are lots...



rviz - 3D Visualization

Modular state and sensor visualization



ROS Distributions

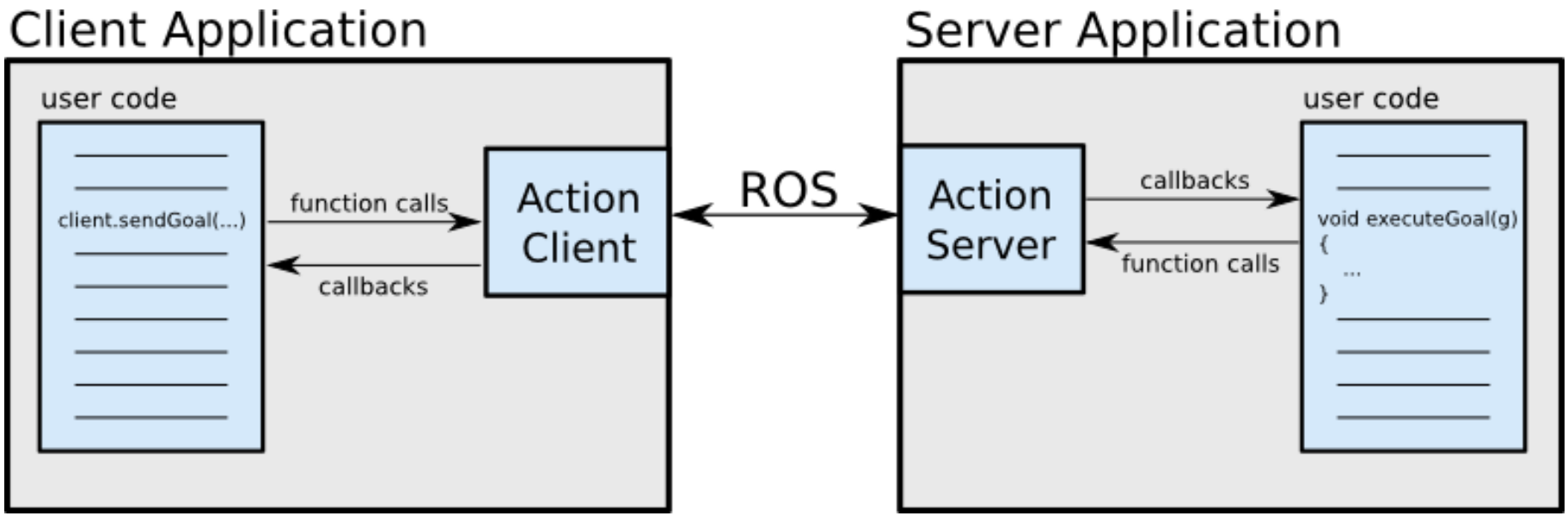
Delivering ROS packages to the masses

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

Actions

Using function calls and callbacks

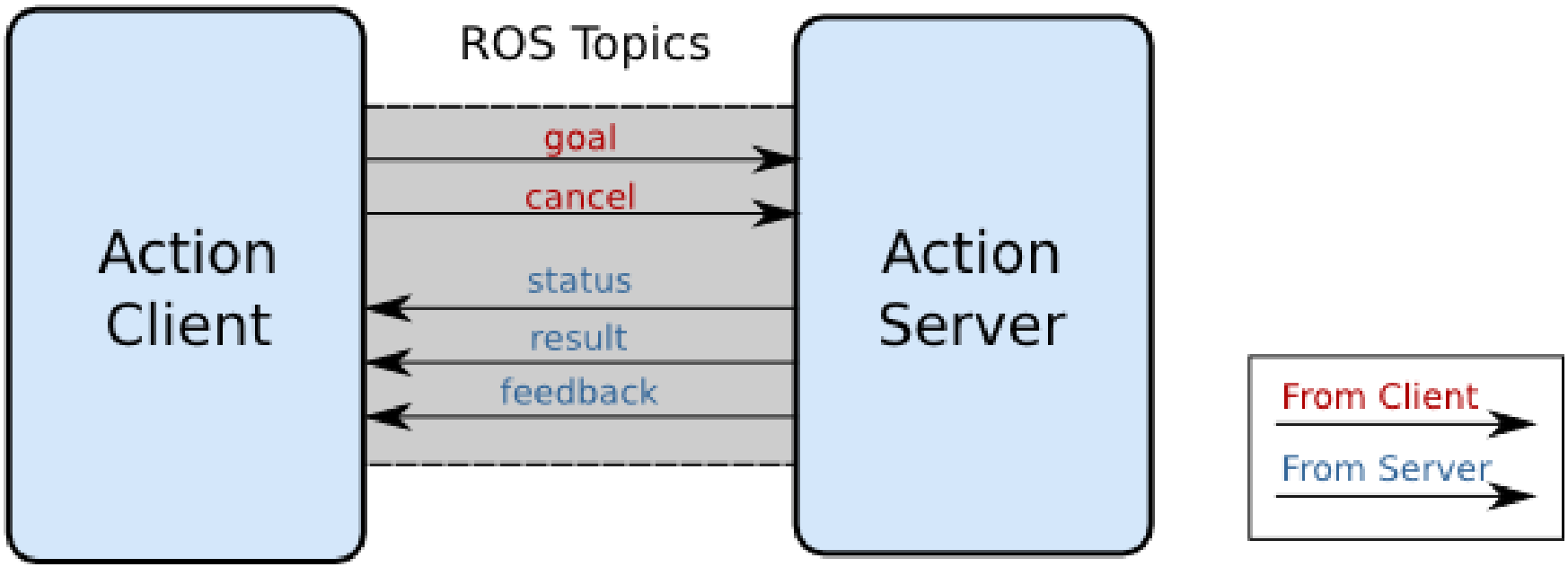
- request goals (client side)
- execute goals (server side)



Actions

- action protocol relies on ROS topics to transport messages

Action Interface



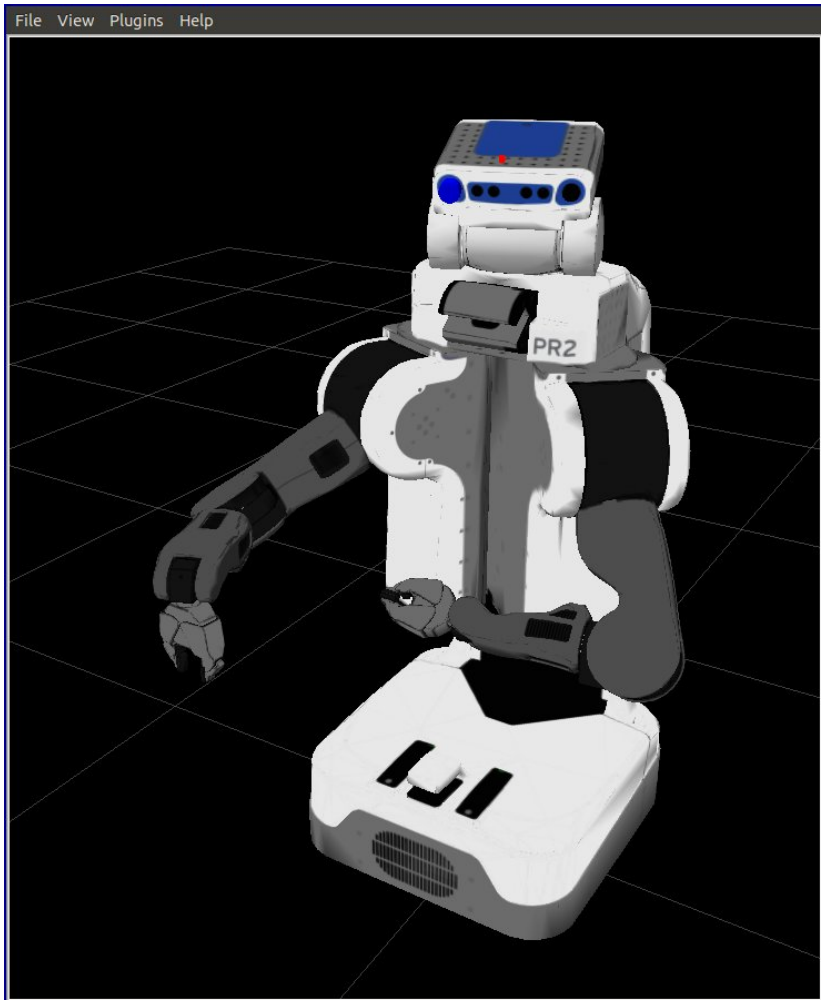
Action Definitions

- Similar to messages and services.
- Definition: Request + result + feedback
- Defined in ros-package/action/*.action
- Generated by CMake macro genaction().
- Example: actionlib_tutorials/Fibonacci.action

```
#goal definition
int32 order
---
#result definition
int32[] sequence
---
#feedback
int32[] sequence
```

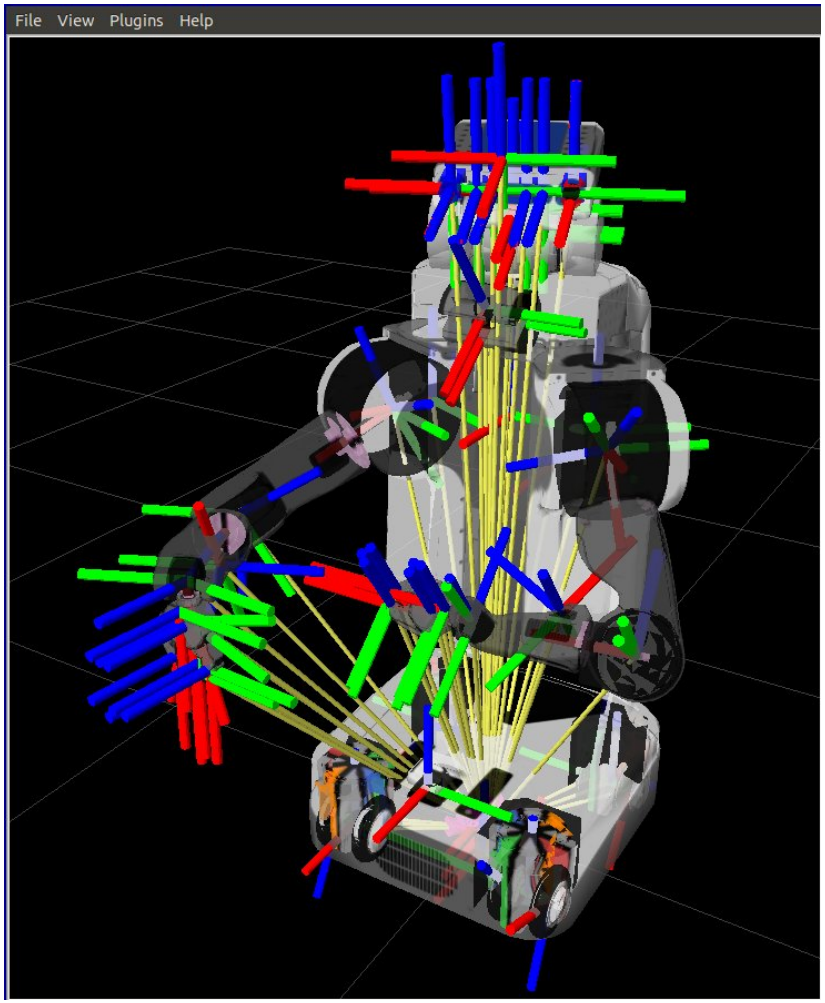
Outline

Coordinate frames



- robots consist of many links
- every link describes its own coordinate system
- sensor measurements are local to the corresponding link
- links change their position over time

Coordinate frames



- robots consist of many links
- every link describes its own coordinate system
- sensor measurements are local to the corresponding link
- links change their position over time

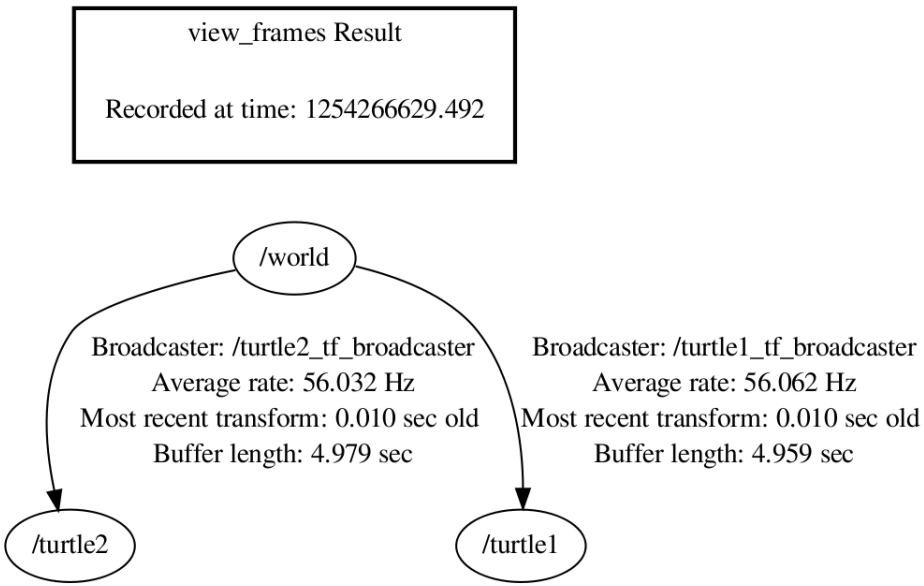
Transforms are distributed

- Transforms are produced by different nodes:
  - Localization in map (AMCL, gmapping)
  - Odometry (base controller)
  - Joint positions (robot controllers and robot_state_publisher)
- Many publishers, many consumers
- Distributed system, redundancy issues, ...

What is TF?

- decentralized: many publishers, many subscribers
- A coordinate frame tracking system
  - standardized protocol for publishing transforms
  - Classes and methods for looking up, calculating and sending transforms
  - transforms are published on the /tf topic
- No central instance managing the tree of transforms
- Command line tools

The transform tree



- Consists of frames (links) and the transforms between them.
- Each link is cached (10 secs default caching time)
- Works with multiple disconnected trees
- Transforms must form a proper tree (no cycles)

TF data types

- Transforms and poses
- stamped data types (via ROS header)
- Header contains time stamp and frame names
- StampedTransform and PoseStamped
- StampedTransform: frame name and child frame name

TF and time

- TF buffers transforms for 10 seconds
- query transforms in the past
- TF interpolates frames
- fixed frame: frame that doesn't move (reference)