Robotics Operating System

Ludovic Saint-Bauzel (saintbauzel@isir.upmc.fr)

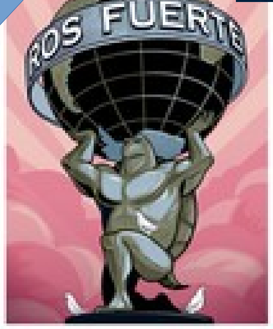
2021

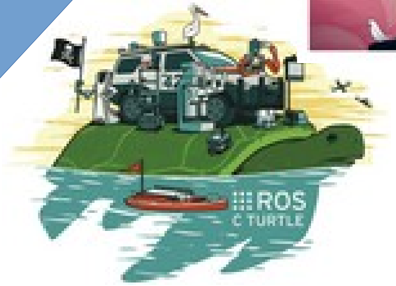
Summary

- Context
- Ros in a Nutshell
- Implementing a Robot

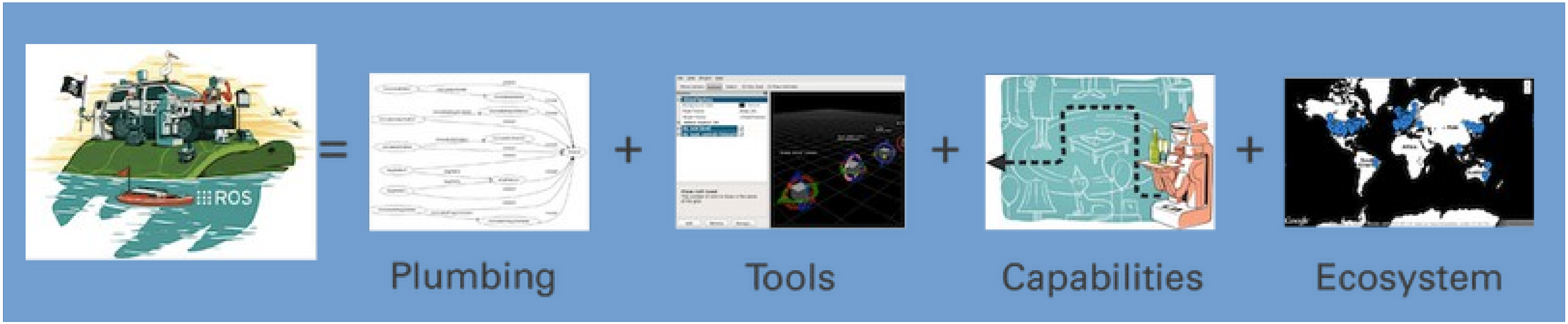
Timeline

- *2020 : **ROS 1.1 Noetic !!!*** **–––** ***ROS 2.0 Foxy***
- *2018 : ROS 1.1 Melodic Morenia* –– *ROS 2.0 AA*
- 2017 : ROS 1.1 Lunar Loggerhead
- 2016 : ROS 1.1 Kinetic Kame
- 2015 : ROS 1.1 Jade Turtle
- 2014 : ROS 1.1 Indigo Igloo
- 2014 : ROS Hydro Medusa
- 2012 : ROS Groovy Galapagos
- 2012 : ROS Fuerte
- 2011 : ROS Electric Emys
- 2011 : ROS Diamondback
- 2010 : ROS 1.0 : C Turtle
- 2009 : ROS 0.4
- 2007 : Beginning





Rationales



Community

- Distributions
- **Repositories**
- **ROS Wiki**
- Bug Tickets System
- Mailling Lists
- ROS Answers
- Blog

Robot-Specific Features

- Standard Message Definitions for Robots (poses,transforms, vectors, camera...)
- Robot Geometry Library (tf)
- Robot Description Language (KDL, URDF)
- Preemptable Remote Procedure Calls (actions)
- Diagnostics
- Pose Estimation (EKF)
- *Localization*
- *Mapping*
- *Navigation*

Integration

- GAZEBO
- OpenCV
- PCL
- MoveIt
- (Ros Industrial)

ROS1 → ROS2

- C++03, *C++11*
- Python 2
- catkin_make
- Roshell :

  roscd, rosls,

  rosmake…
- Custom Communication Framework

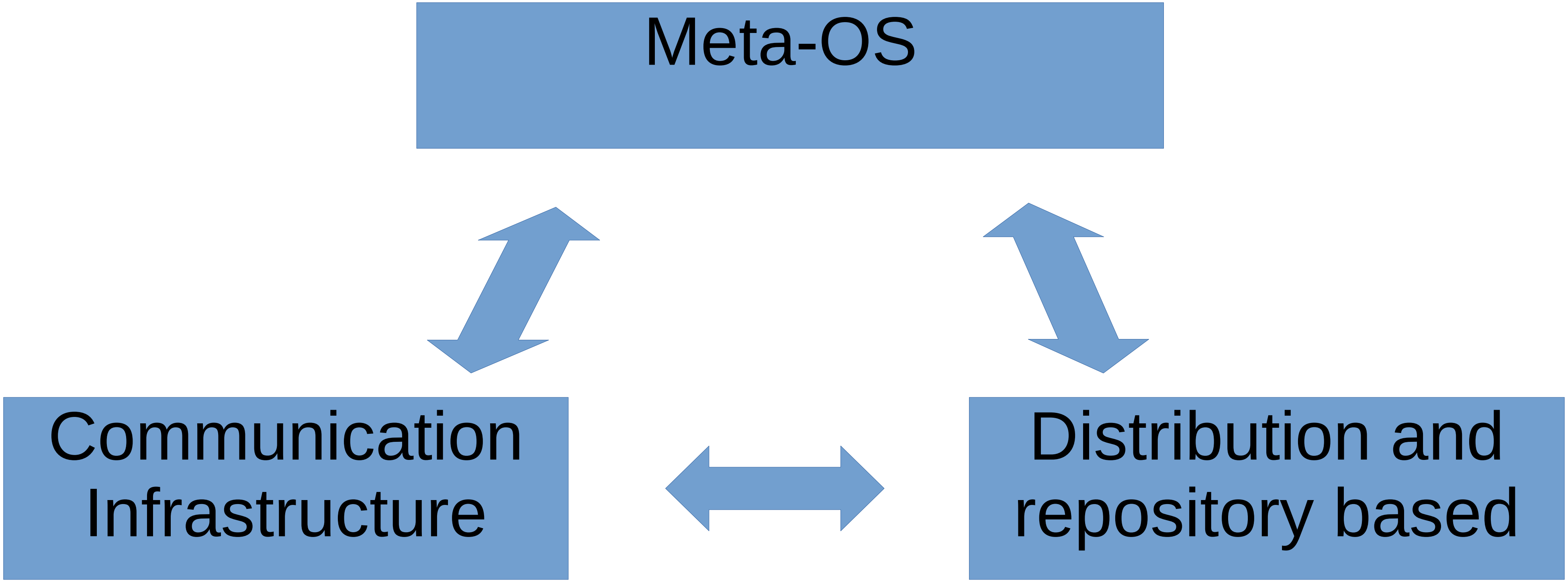
→

- **C++11, *C++14*, *C++17***
- Python **3**
- **ament**_make
- Roshell :

  ros2 cd, ros2 ls,

  ros2 build …
- **Industrial** Communication Framework

ROS In a Nutshell

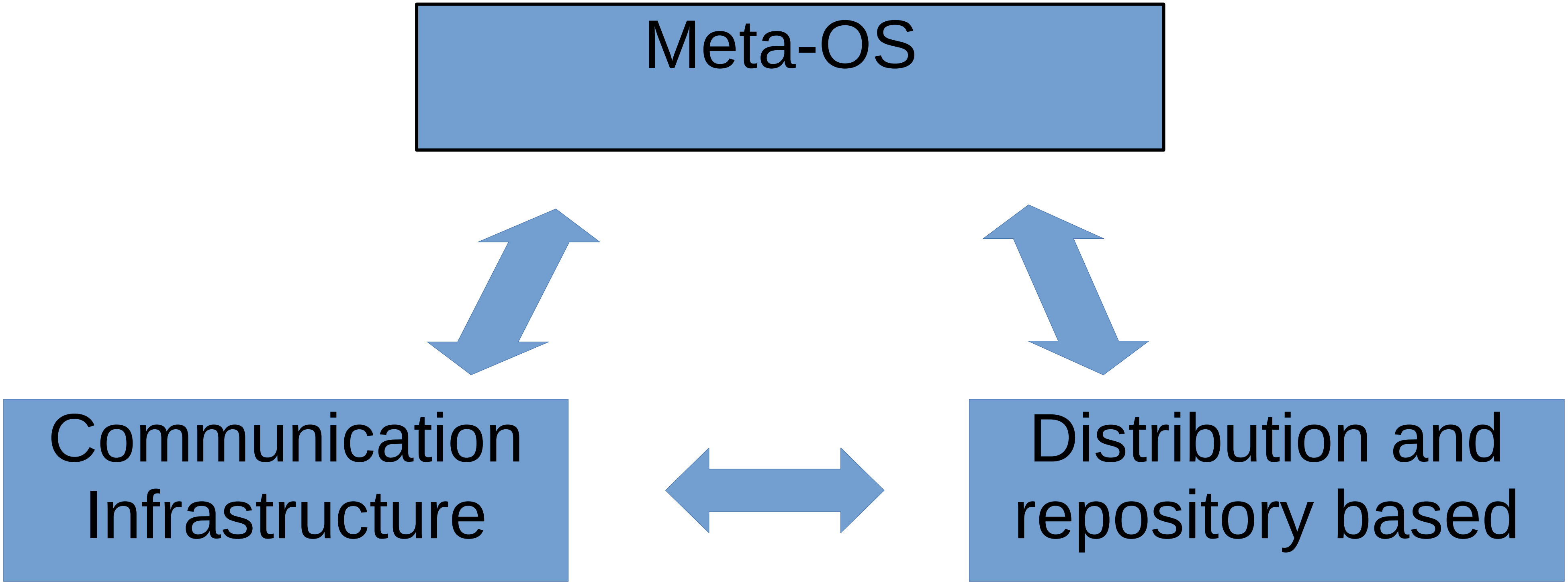


Meta-OS

Communication Infrastructure

Distribution and repository based

ROS In a Nutshell



Meta-OS

Communication Infrastructure

Distribution and repository based

ROS : Meta-Operating System

OS

- Shell
  - cd, ls
- Gestion de paquets
  - apt install, yum install
- Gestion compilation
  - make, cmake

Meta-OS

- ROS Shell
  - roscd, rosls, roslaunch, rosrun…
- Packages Management
  - rosinstall
  - rosdep
- Compilation
  - *rosmake*, catkin_make

ros+shell

- roscd package1 : ros+cd
- rosls package1 : ros+ls
- roscp package1 file path_dest : ros+cp
- rosed : ros+ed(itor) uses $EDITOR Global variable
- roscore : ros+core(server)
- rosgraph
- rosrun package program_node

rospack

- rospack list
- rospack find package1
- rospack depends1 package1
- rospack depends package1

rosparam

- rosparam list
- rosparam set /param1
- rosparam get /param1
- rosparam load file.yaml
- rosparam dump

rosnode

- rosnode list
- rosnode info node1
- rosnode ping node1
- rosnode kill node1
- rosnode machine

rostopic

- rostopic list
- rostopic info /topic1
- rostopic type /topic
- rostopic hz /topic1
- rostopic pub /topic1 std_msgs/string “Hello World!”
- rostopic echo /topic1

rosservice

- rosservice list
- rosservice type serv1
- rosservice args serv1
- rosservice call serv1 arg1:=val

rosmsg/rossrv

- rossrv show srv1

- rosmsg show msg1

roslaunch

- roslaunch package1 file.launch

```
<launch>
  <node name="listener" pkg="beginner_tutorial" type="listener.py" output="screen"/>

  <node name="talker" pkg="beginner_tutorial" type="talker" output="screen"/>

</launch>
```

rosbag

- rosbag record topic1 topic2

  -O filename

  --node=nodename

  -a (all)

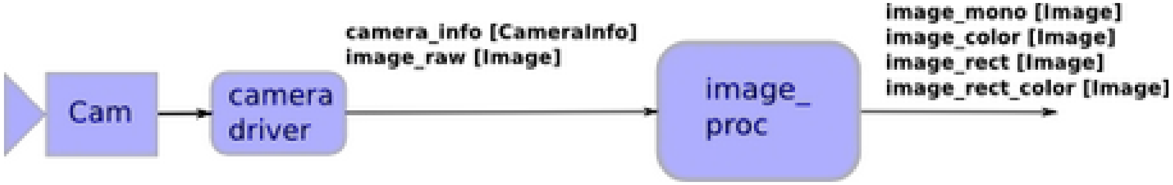
  -j (BZ2 compress)

  --split –duration=30/--size=1024
- rosbag play filename.bag
- rosbag check filename.bag
- rosbag fix filename.bag

Tools : CLI

- CLI : **every thing** can be command line
  - **roscd** : Change Directory in Package (roscd package/localpath)
  - **rosls** : List files in package (rosls package)
  - **rosrun** : execute programs in package (rosrun package program)
  - **roslaunch** : execute a deployment file
  - **catkin_make** : compile packages
  - **rostopic** : management of topics (list, publish, frequency...)
    - rostopic echo /topic; rostopic pub /topic type data
  - **rosnode** : management of topics (list,info...)
  - **rossrv** : management of services (list, info...)
  - **rosmsg** : management of messages (list, info…)
  - **rosbag :** management of topics recording and replying
    - rosbag record -O file -a; rosbag replay file.bag
  - ...

Use Case RosBag



- Nodes : (let run them with rosrun and have a look with rosnode, rostopic)
  - usb_cam/usb_cam_node
  - image_view/image_view image:=/usb_cam/image_raw
- *Node image_proc -> opencv*
- Rosbag usage :
  - rosbag record -O webcam -a
  - rosbag play webcam.bag

Graphical Tools : RQT

- Some usefull tools for data
  - **rqt_graph** : show the topology of nodes
  - **rqt_dep** : show the dependencies of nodes
  - **rqt_plot** : show the data in topics in a plot
  - **rqt_logger_level** : show the log information and manage levels
  - **rqt** : generic ros infrastructure visualisation (plugins)
  - ...

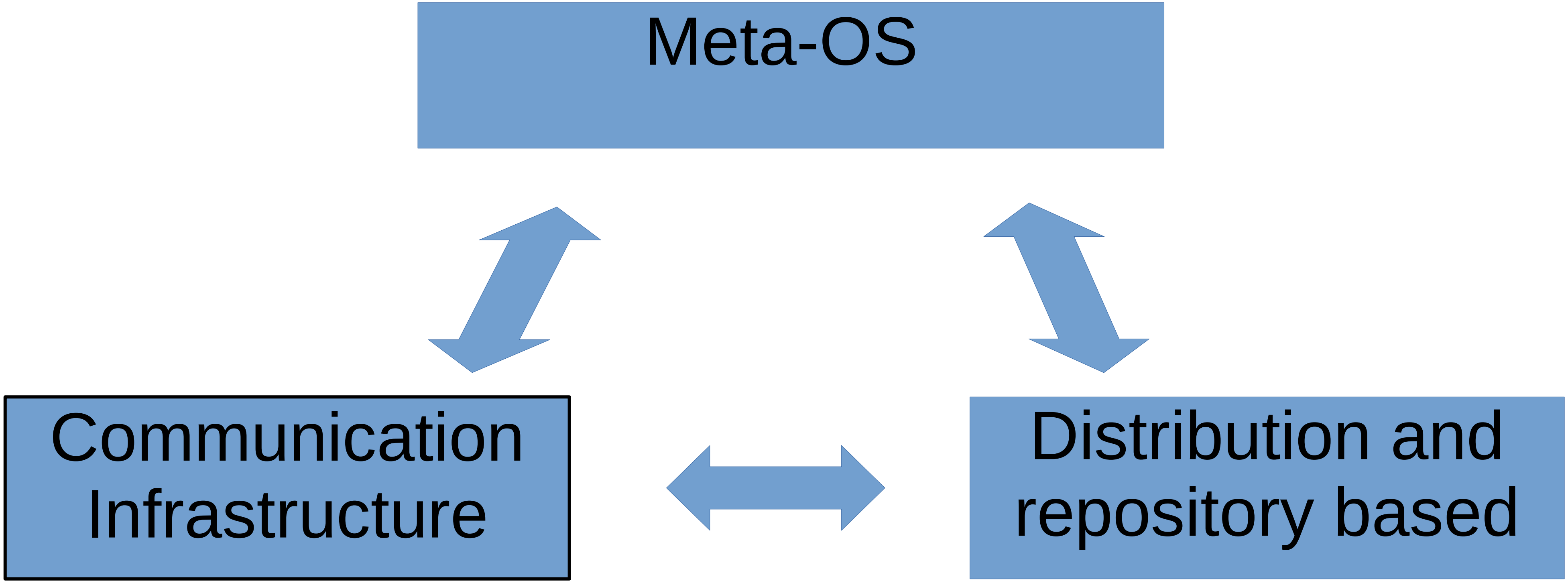
Use case ROS : turtlesim teleop

```
roscore
rosnode list
rostopic list
rosrun turtlesim turtlesim_node
rosnode list
rostopic list
rosrun turtlesim turtle_teleop
```

Communications Infrastructure

- publish/subscribe anonymous message passing
- recording and playback of messages
- request/response remote procedure calls
- distributed parameter system

ROS In a Nutshell



Meta-OS

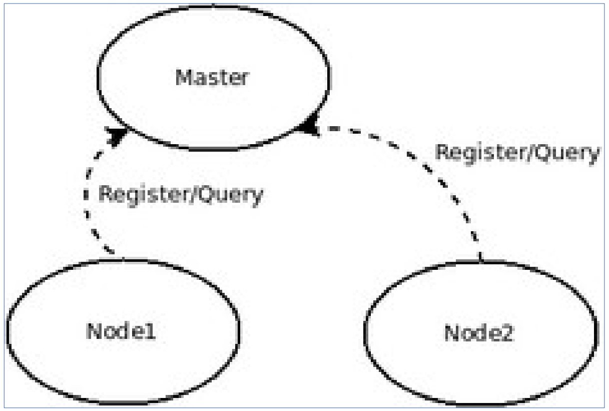
Communication Infrastructure

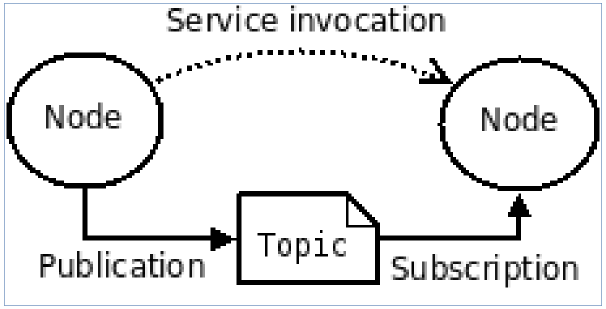
Distribution and repository based

Communications Infrastructure

- Master (XML RPC)
- Node
- Topics
- Messages (IDL,TCP, UDP,Serial)
- Services (RPC)
- Bags
- Parameter Server(Dictionnaries)

Master / Nodes / Topics



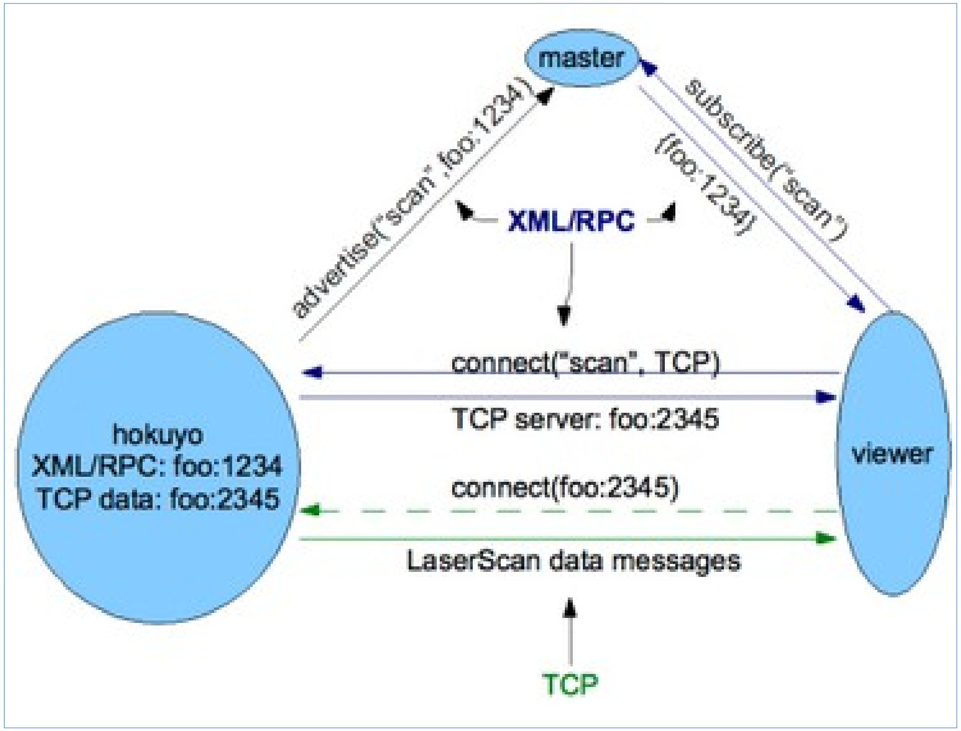


- Master : Information Server
  - roscore
  - Register, query : XMLRPC
- Node : Process
- Nodelet : Threads
- Topic : Transport “socket”
  - TCP, UDP, “*serial*”
  - Protocol and properties
- Service : RPC (xml rpc)

XML RPC

• Distributed / Local	HTTP/Shared Memory
• Parsing	XML
• Remote Procedure Call	Procedures, parameters
• Easy to implement	Perl, Python, Java, Frontier, C/C++, Lisp...

Master / Nodes / Topics : Example



http://wiki.ros.org/ROS/Technical Overview

Distributed Infrastructure

- Master URI
  - roscore
  - ROS_MASTER_URI

- Transport :
  - Supported TCP, UDP
  - Rosserial : rosserial_server → Serial (arduino...)

Graph Resource Names

Nodes, Param, Services …

- base
- relative/name
- /global/name
- ~private/name

Examples:

- / (the global namespace)
- /foo
- /stanford/robot/name
- /wg/node1

Messages

- std_msgs
  - int32, float64, string, float64multiarray…
- common_msgs
  - actionlib_msgs: messages for representing actions.
  - diagnostic_msgs: messages for sending diagnostic data.
  - geometry_msgs: messages for representing common geometric primitives.
  - nav_msgs: messages for navigation.
  - sensor_msgs: messages for representing sensor data.

http://wiki.ros.org/ROS/Higher-Level Concepts

Messages : Examples

- IDL Description

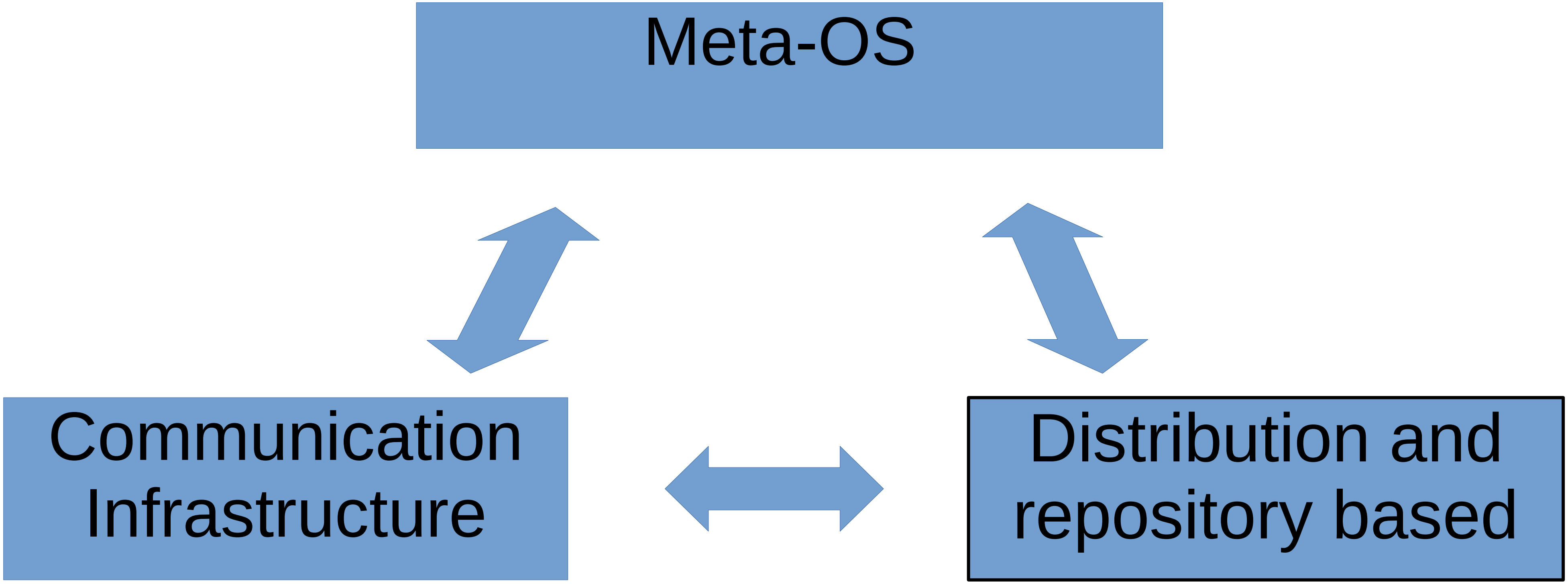
std_msgs/Float64MultiArray	geometry_msgs/Twist	geometry_msgs/Transform	sensor_msgs/Image
std_msgs/MultiArrayLayout layout float64[] data	geometry_msgs/Vector3 linear geometry_msgs/Vector3 angular	geometry_msgs/Vector3 translation geometry_msgs/Quaternion rotation	std_msgs/Header header uint32 height uint32 width string encoding uint8 is_bigendian uint32 step uint8[] data

http://wiki.ros.org/std_msgs
http://wiki.ros.org/common_msgs

Use case ROS : turtlesim

```
roscore

rosrun turtlesim turtlesim_node

rostopic info /turtle1/pose

rosmsg show /turtle1/pose

rosrun turtlesim turtle_teleop

rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist –
‘[1.0, 0.0,0.0]’ ‘[0.0, 0.0, 1.8]’

rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 –
‘[1.0, 0.0,0.0]’ ‘[0.0, 0.0, 1.8]’
```

ROS In a Nutshell



Meta-OS

Communication Infrastructure

Distribution and repository based

Installing and navigating

```
> source /opt/ros/noetic/setup.bash
    > mkdir -p ~/catkin_ws/src
    > cd ~/catkin_ws/src
> catkin_init_workspace
    > cd ../
> catkin_make
    >echo $ROS_PACKAGE_PATH
> source devel/setup.bash
    >echo $ROS_PACKAGE_PATH
> rospack find rospy
> rosls rospy
    >pwd
> roscd rospy
    > pwd
```

http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment
http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem

Package creation

- Package.xml
  - Author,
  - License,
  - Dependencies
- CMakeLists.txt
  - Compilation instructions
  - Depends

- *FileSystem Organisation*

```
src/
    CmakeLists.txt
    package_1/
    CMakeLists.txt
    package.xml
    …
    package_n/
    CMakeLists.txt
    package.xml
```

- *Tools*

  catkin_create_pkg package_1 roscpp rospy std_msgs

  rospack depends1 package_1

http://wiki.ros.org/ROS/Tutorials/CreatingPackage

Developing a Node

node_only.py

```
#!/usr/bin/env python
import rospy

if __name__ == '__main__':
    try:
        rospy.init_node('nodename', anonymous=True)
        rate = rospy.Rate(10) # 10hz
        while not rospy.is_shutdown():
                rate.sleep()

    except rospy.ROSInterruptException:
        pass
```

http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber

Developing a Topic Sub

talker.py

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

if __name__ == '__main__':
    try:
        pub = rospy.Publisher('chatter', String, queue_size=10)
        rospy.init_node('talker', anonymous=True)
        rate = rospy.Rate(10) # 10hz
        while not rospy.is_shutdown():
                hello_str = "hello world %s" % rospy.get_time()
                pub.publish(hello_str)
                rate.sleep()

    except rospy.ROSInterruptException:
        pass
```

http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber

Developing a Topic Sub

listener.py

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id()+ "I heard %s",data.data)

if __name__ == '__main__':
    try:
        rospy.init_node('listener', anonymous=True)
        rospy.Subscriber('chatter', String, callback)
        rospy.spin()

    except rospy.ROSInterruptException:
        pass
```

http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber

Developing a Topic Sub **C++**

talker.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"
int main(int argc, char **argv)
{
  ros::init(argc, argv, "talker");
  ros::NodeHandle n;

  ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);

  ros::Rate loop_rate(10);

  while(ros::ok()){
    msg.data = “hello world”;
    chatter_pub.publish(msg);
    ros::spinOnce();
    loop_rate.sleep();
  }
  return 0;
}
```

http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber c++

Custom Message

- IDL :
  Interface Description Language

```
msg/Num.msg

int64 num
```

- Compilation

```
CmakeLists.txt


add_message_files(
  FILES
  Num.msg
)
```

- Tools : rosmsg

```
$> rosmsg show beginner_tutorial/Num
```

Custom Service 1/3

- IDL :
  Interface Description Language

```
srv/AddTwoInts.srv

int64 a
Int64 b
---
Int64 sum
```

- Compilation

```
CMakeLists.txt

...
add_service_files(
  FILES
  AddTwoInts.srv
)
```

- Tools : rossrv

```
$> rossrv show beginner_tutorial/AddTwoInts
```

Custom Service 2/3

- **Server Implementation**

```
#!/usr/bin/env python

from beginner_tutorials.srv import AddTwoInts,AddTwoIntsResponse
import rospy

def handle_add_two_ints(req):
    print "Returning [%s + %s = %s]"%(req.a, req.b, (req.a + req.b))
    return AddTwoIntsResponse(req.a + req.b)

def add_two_ints_server():
    rospy.init_node('add_two_ints_server')
    s = rospy.Service('add_two_ints', AddTwoInts, handle_add_two_ints)
    print "Ready to add two ints."
    rospy.spin()

if __name__ == "__main__":
    add_two_ints_server()
```

Custom Service 3/3

- **Client Implementation**

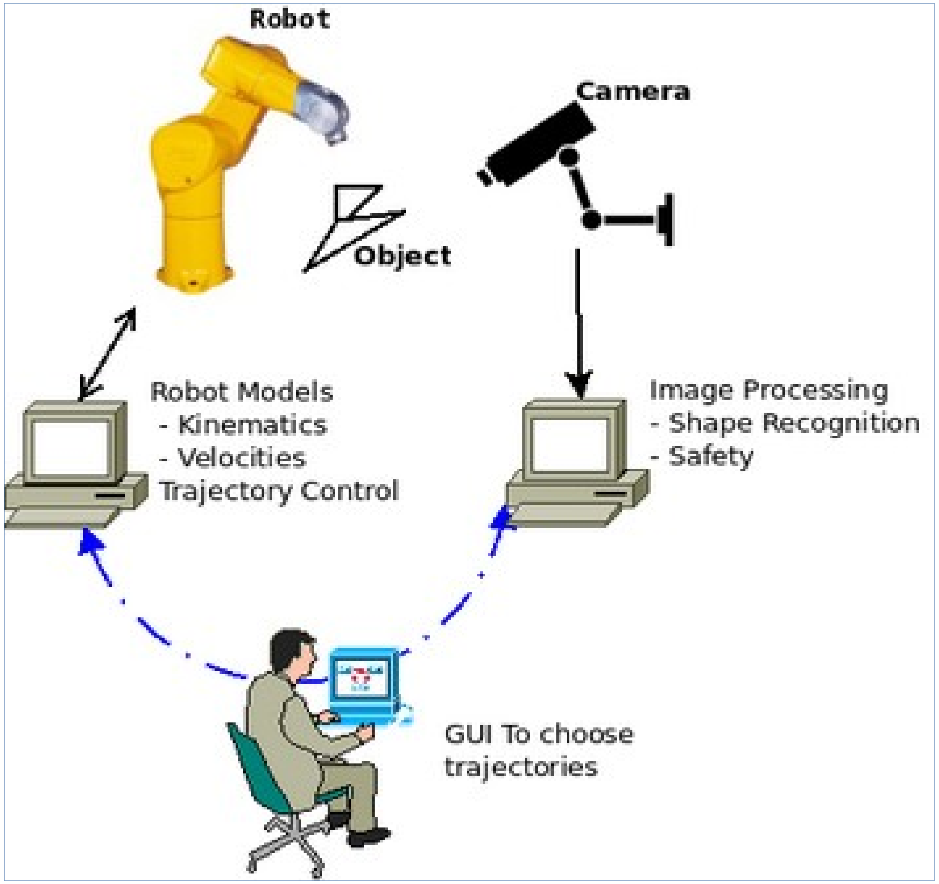
```
#!/usr/bin/env python
import sys
import rospy
from beginner_tutorials.srv import *
def add_two_ints_client(x, y):
    rospy.wait_for_service('add_two_ints')
    try:
        add_two_ints = rospy.ServiceProxy('add_two_ints', AddTwoInts)
        resp1 = add_two_ints(x, y)
        return resp1.sum
    except rospy.ServiceException, e:
        print "Service call failed: %s"%e
def usage():
    return "%s [x y]"%sys.argv[0]
if __name__ == "__main__":
    if len(sys.argv) == 3:
        x = int(sys.argv[1])
        y = int(sys.argv[2])
    else:
        print usage()
        sys.exit(1)
    print "Requesting %s+%s"%(x, y)
    print "%s + %s = %s"%(x, y, add_two_ints_client(x, y))
```

Designing a ROS Software Requirements

- Communication diagram
- Network RPC (Services, Actions)

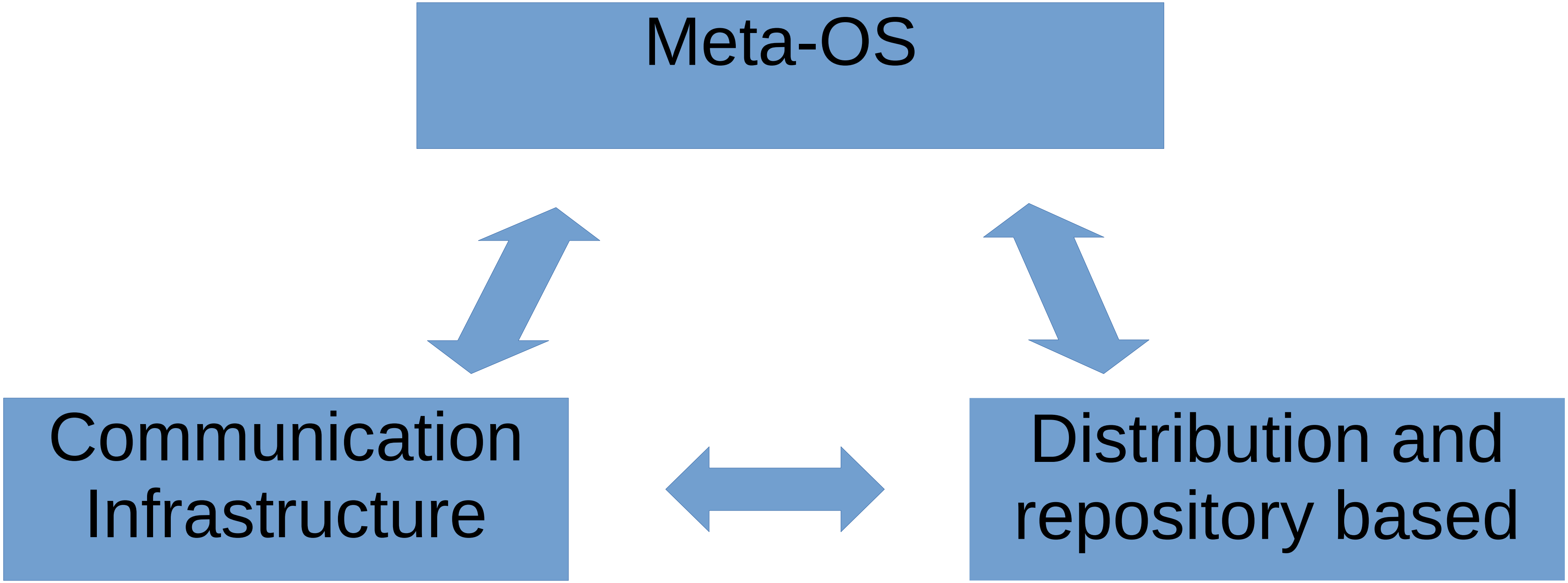
- No Real-Time
- No Synchronization constraints

Imagine a distributed scenario



- Propose an implementation
  - Topics
  - Nodes, Master
  - Services (RPC)
  - Messages

ROS In a Nutshell



Meta-OS

Communication Infrastructure

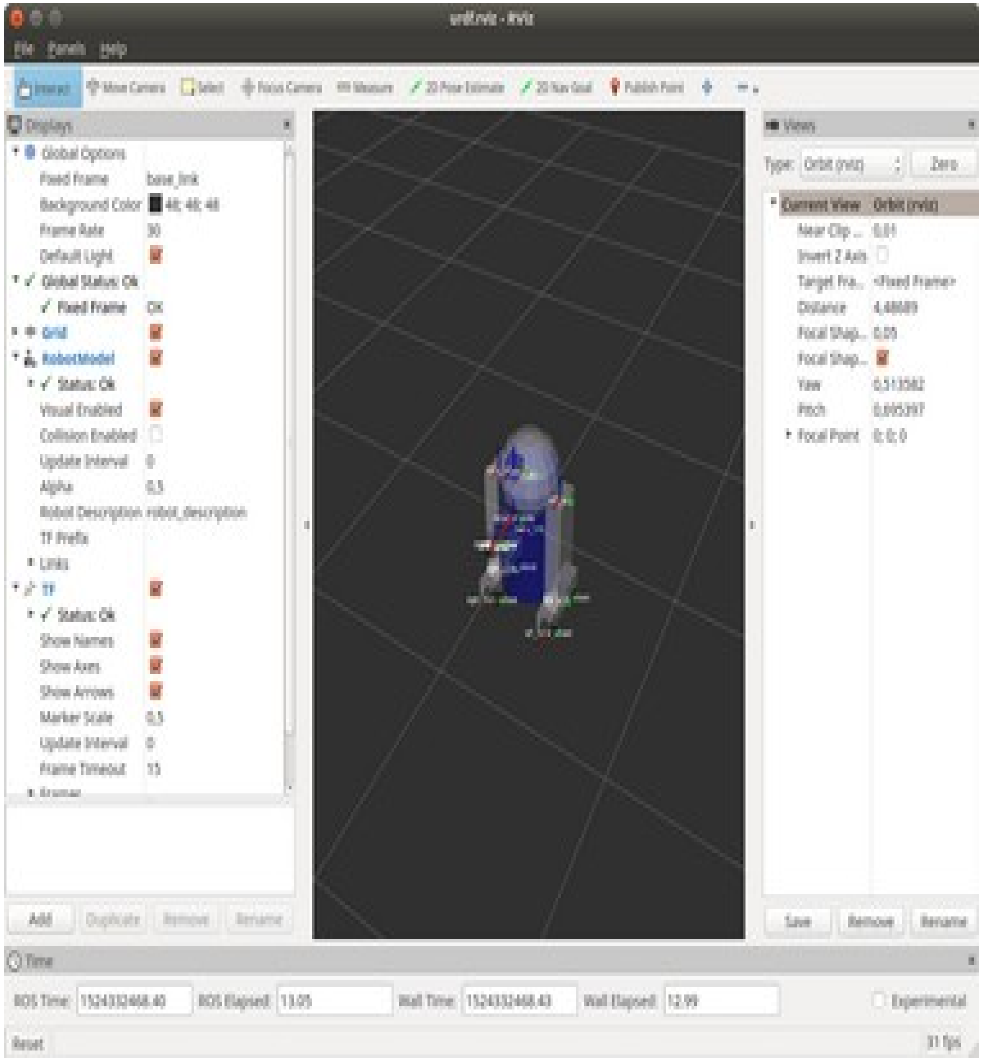
Distribution and repository based

Implementing a robot

Geometry, kinematic and control

Visualization 3D,2D : RViz

- It is a debugging visualization tool for robot
  - Sensor values
  - Visualize SLAM
  - Visualize 3D state of the robot
  - ...



Global Options
Fixed Frame base_link
Background Color 48; 48; 48
Frame Rate 30
Default Light ☑
✓ Global Status: Ok
✓ Fixed Frame OK
Grid ☑
RobotModel ☑
✓ Status: Ok
Visual Enabled ☑
Collision Enabled ☐
Update Interval 0
Alpha 0,5
Robot Description robot_description
TF Prefix
Links
TF ☑
✓ Status: Ok
Show Names ☑
Show Axes ☑
Show Arrows ☑
Marker Scale 0,5
Update Interval 0
Frame Timeout 15
Frames

Type: Orbit (rviz) Zero
Current View Orbit (rviz)
Near Clip ... 0,01
Invert Z Axis ☐
Target Fra... <Fixed Frame>
Distance 4,48689
Focal Shap... 0,05
Focal Shap... ☑
Yaw 0,513582
Pitch 0,695397
Focal Point 0; 0; 0

ROS Time: 1524332468.40 ROS Elapsed: 13.05 Wall Time: 1524332468.43 Wall Elapsed: 12.99 ☐ Experimental
31 fps

tf : Transform

- Position
- Rotation (quaternion)
  - x*cos(a/2) y*cos(a/2) z*cos(a/2) sin(a/2)
  - Roll Pitch Yaw : RPY(r, p, y)

http://wiki.ros.org/tf/Tutorials/Introduction to tf

http://wiki.ros.org/tf/Tutorials/Writing a tf broadcaster

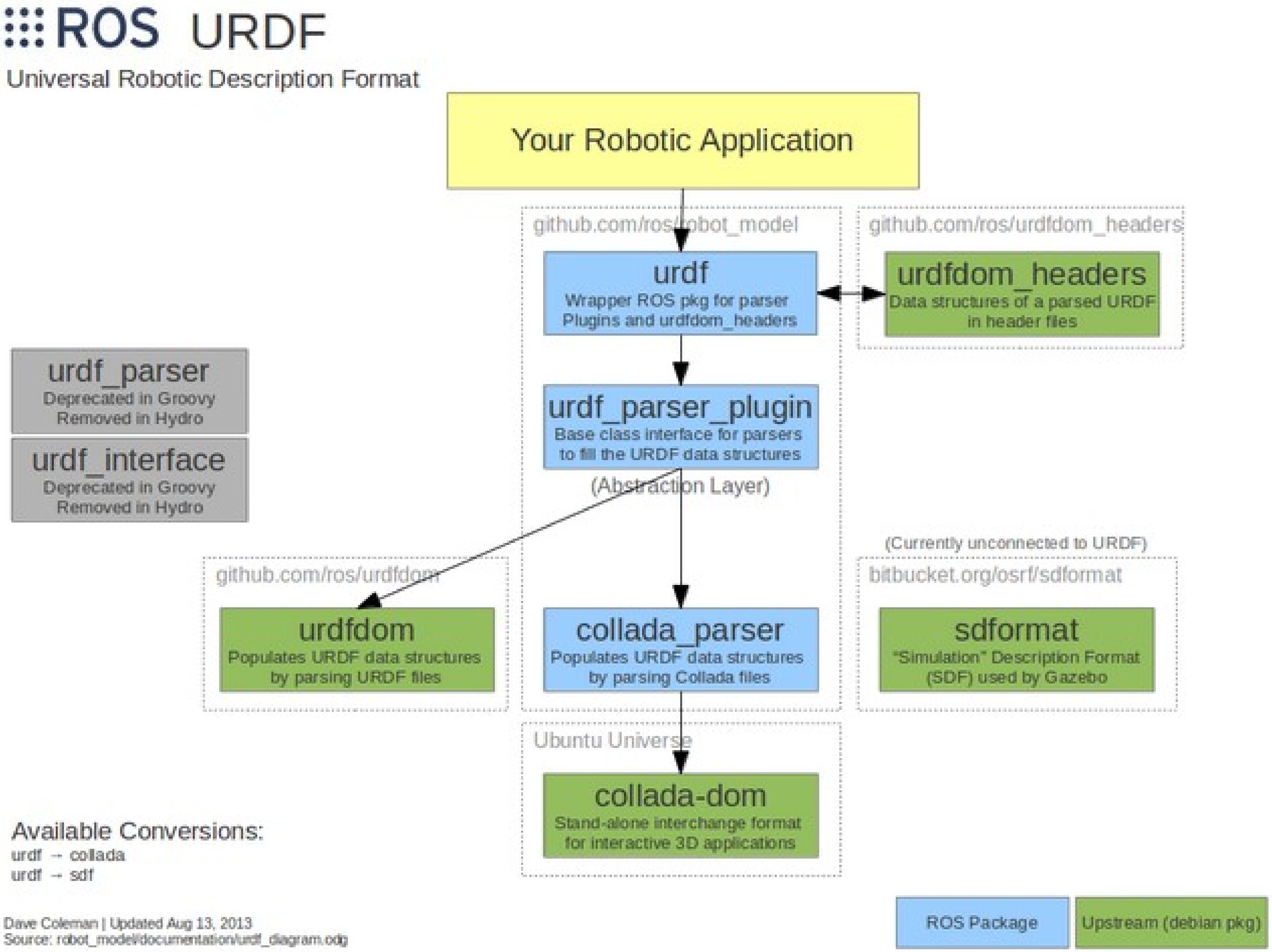
tf

```
#!/usr/bin/env python
import roslib
roslib.load_manifest('learning_tf')
import rospy
import tf
import turtlesim.msg
def handle_turtle_pose(msg, turtlename):
    br = tf.TransformBroadcaster()
    br.sendTransform((msg.x, msg.y, 0),
                     tf.transformations.quaternion_from_euler(0, 0, 
msg.theta),
                     rospy.Time.now(),
                     turtlename,
                     "world")
if __name__ == '__main__':
    rospy.init_node('turtle_tf_broadcaster')
    turtlename = rospy.get_param('~turtle')
    rospy.Subscriber('/%s/pose' % turtlename,
                     turtlesim.msg.Pose,
                     handle_turtle_pose,
                     turtlename)
    rospy.spin()
```

Robot Modeling : URDF

XML Specifications

- <link>
  - Describes the kinematic and dynamic properties of a link.
- <transmission>
  - Transmissions link actuators to joints and represents their mechanical coupling
- <joint>
  - Describes the kinematic and dynamic properties of a joint.
- <gazebo>
  - Describes simulation properties, such as damping, friction, etc
- <sensor>
  - Describes a sensor, such as a camera, ray sensor, etc
- <model_state>
  - Describes the state of a model at a certain time
- <model>
  - Describes the kinematic and dynamic properties of a robot structure.



ROS URDF

Universal Robotic Description Format

Your Robotic Application

github.com/ros/robot_model

urdf
Wrapper ROS pkg for parser Plugins and urdfdom_headers

urdf_parser_plugin
Base class interface for parsers to fill the URDF data structures

(Abstraction Layer)

collada_parser
Populates URDF data structures by parsing Collada files

github.com/ros/urdfdom_headers

urdfdom_headers
Data structures of a parsed URDF in header files

urdf_parser
Deprecated in Groovy
Removed in Hydro

urdf_interface
Deprecated in Groovy
Removed in Hydro

github.com/ros/urdfdom

urdfdom
Populates URDF data structures by parsing URDF files

(Currently unconnected to URDF)

bitbucket.org/osrf/sdformat

sdformat
"Simulation" Description Format (SDF) used by Gazebo

Ubuntu Universe

collada-dom
Stand-alone interchange format for interactive 3D applications

Available Conversions:
urdf → collada
urdf → sdf

Dave Coleman | Updated Aug 13, 2013
Source: robot_model/documentation/urdf_diagram.odg

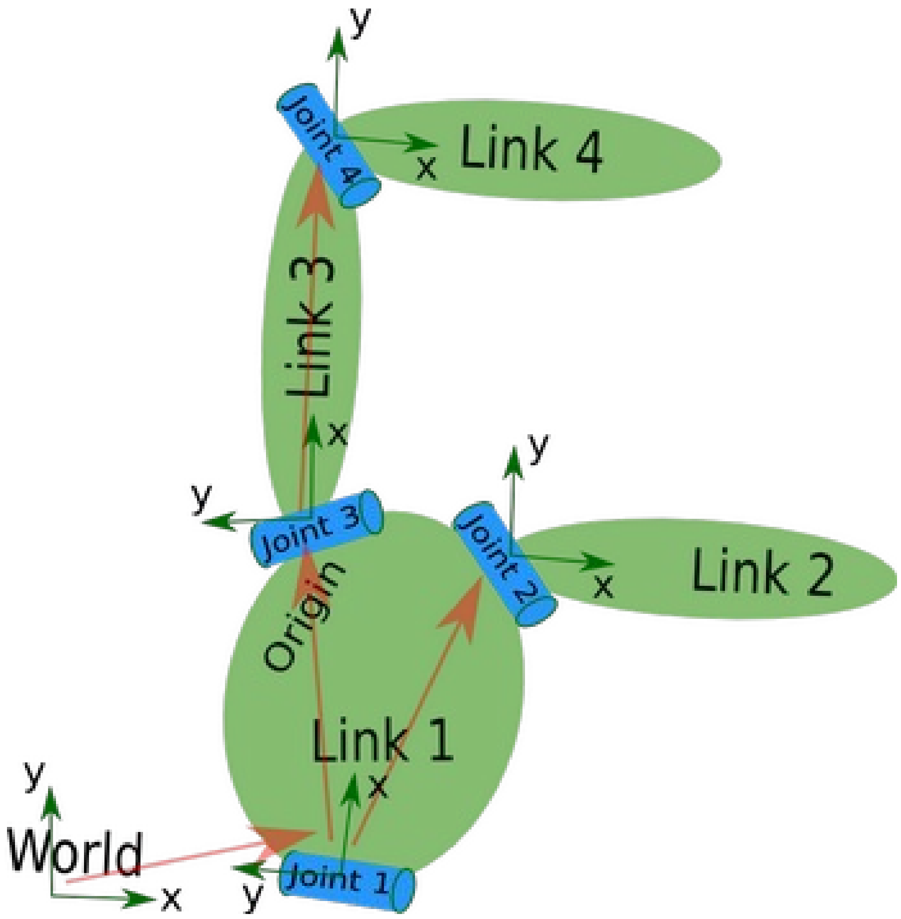
ROS Package

Upstream (debian pkg)

http://wiki.ros.org/urdf

URDF : Links and Joints

```
<robot name="test_robot">
  <link name="link1" />
  <link name="link2" />
  <link name="link3" />
  <link name="link4" />

  <joint name="joint1" type="continuous">
    <parent link="link1"/>
    <child link="link2"/>
  </joint>

  <joint name="joint2" type="continuous">
    <parent link="link1"/>
    <child link="link3"/>
  </joint>

  <joint name="joint3" type="continuous">
    <parent link="link3"/>
    <child link="link4"/>
  </joint>
</robot>
```



http://wiki.ros.org/urdf/Tutorials/Create your own urdf file

http://wiki.ros.org/urdf/Tutorials/Parse a urdf file

http://wiki.ros.org/urdf/Tutorials/Building a Movable Robot Model with URDF

URDF : Shapes

- 1 shape

  roscd urdf_tutorial
  roslaunch urdf_tutorial display.launch model:=urdf/01-myfirst.urdf

```
<?xml version="1.0"?>
<robot name="myfirst">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>
</robot>
```

- Multiple shapes

  02-multipleshapes.urdf

```
<?xml version="1.0"?>
<robot name="multipleshapes">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>
  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
    </visual>
  </link>
  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
  </joint>
</robot>
```

- Origin

  03-origins.urdf

```
...
    <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
...
```

http://wiki.ros.org/urdf/Tutorials/Building a Visual Robot Model with URDF from Scratch

URDF : Physics and Collisions

- Materials

  04-materials.urdf

- Collision
- Inertial

  07-physics.urdf

- Visual

  05-visual.urdf

```
...
 <material name="blue">
    <color rgba="0 0 0.8 1"/>
  </material>
  <material name="white">
    <color rgba="1 1 1 1"/>
  </material>
  <link name="base_link">
    <visual>
      ...
      <material name="blue"/>
      ...
    </visual>
...
```

```
 <link name="link1">
    <collision>
      <origin xyz="0 0 ${height1/2}" rpy="0 0 0"/>
      <geometry>
        <box size="${width} ${width} ${height1}"/>
      </geometry>
    </collision>
    <visual>
      <origin xyz="0 0 ${height1/2}" rpy="0 0 0"/>
      <geometry>
        <box size="${width} ${width} ${height1}"/>
      </geometry>
      <material name="orange"/>
    </visual>
    <inertial>
      <origin xyz="0 0 1" rpy="0 0 0"/>
      <mass value="1"/>
      <inertia
        ixx="1.0" ixy="0.0" ixz="0.0"
        iyy="1.0" iyz="0.0"
        izz="1.0"/>
    </inertial>
  </link>
```

http://wiki.ros.org/urdf/Tutorials/Adding Physical and Collision Properties to a URDF Model

URDF : Joints

- Head

```
<joint name="head_swivel" type="continuous">
  <parent link="base_link"/>
  <child link="head"/>
  <axis xyz="0 0 1"/>
  <origin xyz="0 0 0.3"/>
</joint>
```

- Gripper

```
<joint name="left_gripper_joint" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort="1000.0" lower="0.0" upper="0.548" velocity="0.5"/>
  <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
  <parent link="gripper_pole"/>
  <child link="left_gripper"/>
</joint>
```

```
<joint name="gripper_extension" type="prismatic">
  <parent link="base_link"/>
  <child link="gripper_pole"/>
  <limit effort="1000.0" lower="-0.38" upper="0" velocity="0.5"/>
  <origin rpy="0 0 0" xyz="0.19 0 0.2"/>
</joint>
```

http://wiki.ros.org/urdf/Tutorials/Building a Movable Robot Model with URDF

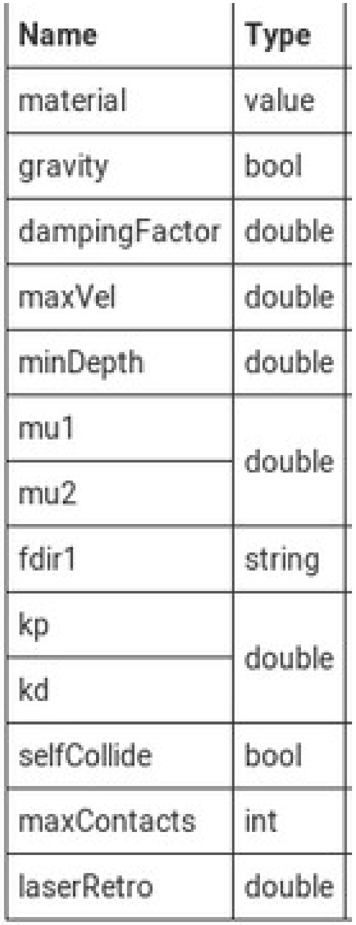
URDF : Gazebo

- links

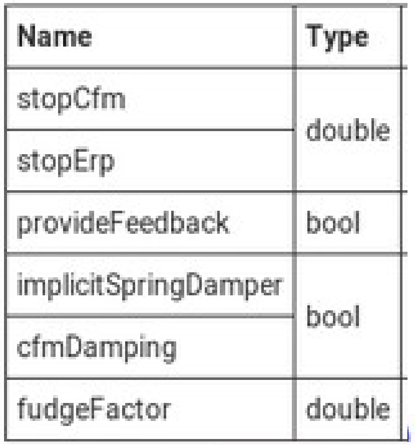
```
<gazebo reference="link2">
   <mu1>0.2</mu1>
   <mu2>0.2</mu2>
   <material>Gazebo/Black</material>
 </gazebo>
```

- joints

```
<joint name="joint2" type="continuous">
  <parent link="link2"/>
  <child link="link3"/>
  <origin xyz="0 ${width} ${height2 - axel_offset*2}" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
  <dynamics damping="0.7"/>
</joint>
```



Name	Type
material	value
gravity	bool
dampingFactor	double
maxVel	double
minDepth	double
mu1	double
mu2	
fdir1	string
kp	double
kd	
selfCollide	bool
maxContacts	int
laserRetro	double



Name	Type
stopCfm	double
stopErp	
provideFeedback	bool
implicitSpringDamper	bool
cfmDamping	
fudgeFactor	double

URDF : Gazebo Ros Control

Controllers → controller_manager spawner

- effort_controllers
  - JointEffortController
  - JointPositionController
  - JointVelocityController
- joint_state_controller
  - JointStateController
- position_controllers
  - JointPositionController
- velocity_controllers
  - JointVelocityController

Config File .yaml → rosparam

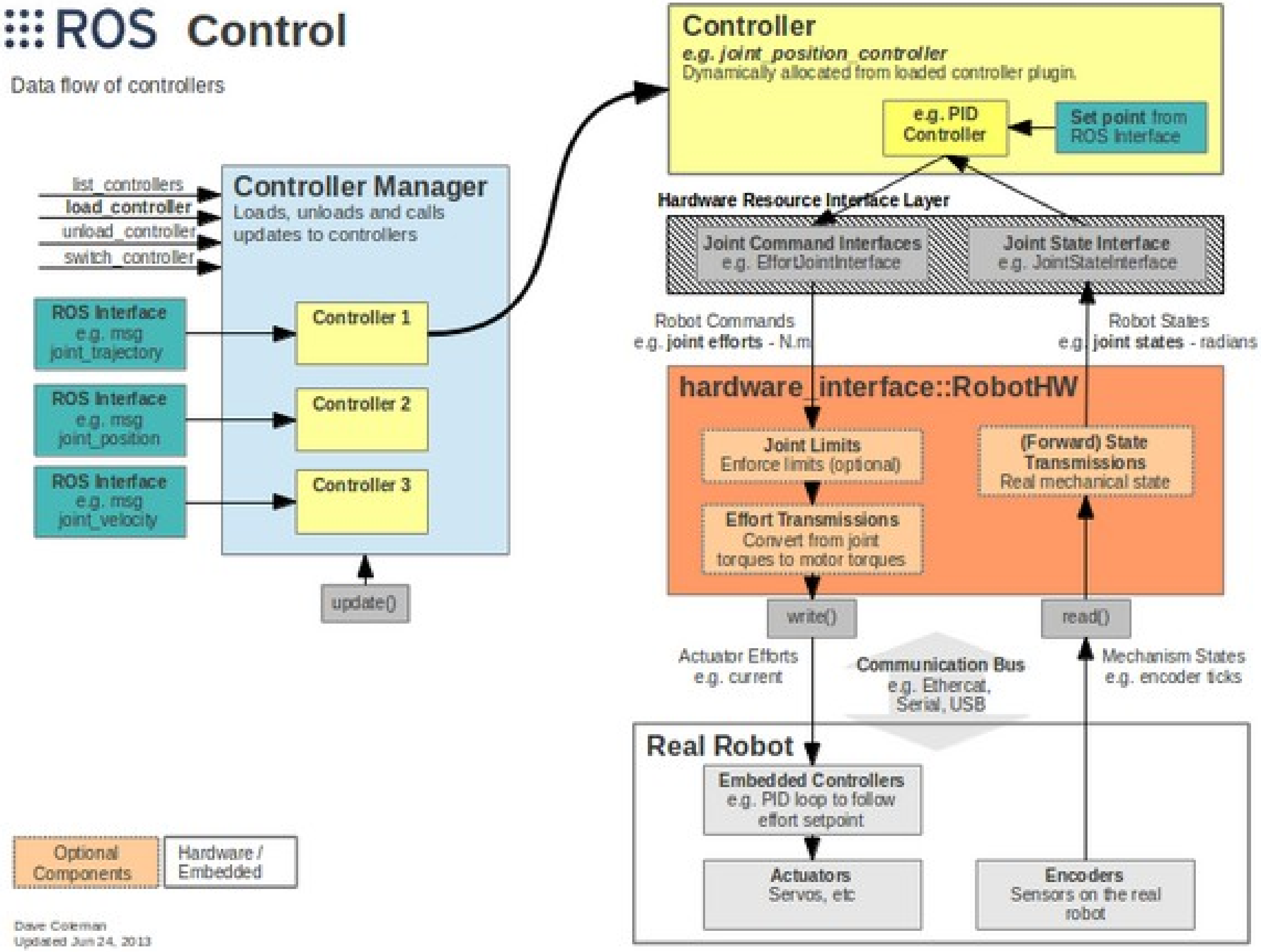
- Loaded in .launch

```
<gazebo>
  <plugin name="gazebo_ros_control"
filename="libgazebo_ros_control.so">
    <robotNamespace>/</robotNamespace>
  </plugin>
</gazebo>
```

```
 # Publish all joint states
-----------------------------------
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50
  # Position Controllers
---------------------------------------
  joint1_position_controller:
    type: effort_controllers/JointPositionController
    joint: joint1
    pid: {p: 100.0, i: 0.01, d: 10.0}
  joint2_position_controller:
    type: effort_controllers/JointPositionController
    joint: joint2
    pid: {p: 100.0, i: 0.01, d: 10.0}
```

```
  <node name="controller_spawner"
pkg="controller_manager" type="spawner" respawn="false"
    output="screen" args="joint1_position_controller
joint2_position_controller joint_state_controller"/>
```

http://wiki.ros.org/ros_control#Hardware_Interfaces
http://gazebosim.org/tutorials/?tut=ros_control


ROS Control
Data flow of controllers
Controller
e.g. joint_position_controller
Dynamically allocated from loaded controller plugin.
e.g. PID Controller
Set point from ROS Interface
Controller Manager
Loads, unloads and calls updates to controllers
list_controllers
load_controller
unload_controller
switch_controller
ROS Interface e.g. msg joint_trajectory
ROS Interface e.g. msg joint_position
ROS Interface e.g. msg joint_velocity
Controller 1
Controller 2
Controller 3
update()
Hardware Resource Interface Layer
Joint Command Interfaces e.g. EffortJointInterface
Joint State Interface e.g. JointStateInterface
Robot Commands e.g. joint efforts - N.m
Robot States e.g. joint states - radians
hardware_interface::RobotHW
Joint Limits Enforce limits (optional)
Effort Transmissions Convert from joint torques to motor torques
(Forward) State Transmissions Real mechanical state
write()
read()
Actuator Efforts e.g. current
Communication Bus e.g. Ethercat, Serial, USB
Mechanism States e.g. encoder ticks
Real Robot
Embedded Controllers e.g. PID loop to follow effort setpoint
Actuators Servos, etc
Encoders Sensors on the real robot
Optional Components
Hardware / Embedded
Dave Coleman
Updated Jun 24, 2013

URDF – Transmission & Control

Hardware Interfaces

- Joint Command Interfaces
  - EffortJointInterface
  - VelocityJointInterface
  - PositionJointInterface
- Joint State Interfaces
- Actuator Command Interfaces
  - EffortActuatorInterface
  - VelocityActuatorInterface
  - PositionActuatorInterface

…

```
<joint name="head_swivel" type="continuous">
  <parent link="base_link"/>
  <child link="head"/>
  <axis xyz="0 0 1"/>
  <origin xyz="0 0 ${bodylen/2}"/>
  <limit effort="30" velocity="1.0"/>
</joint>
...
 <transmission name="head_swivel_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="$head_swivel_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="head_swivel">
<hardwareInterface>PositionJointInterface</hardwareInterface>
  </joint>
</transmission>
```

```
type: "position_controllers/JointPositionController"
joint: head_swivel
```

http://wiki.ros.org/urdf/Tutorials/Using a URDF in Gazebo

URDF : Sensor examples

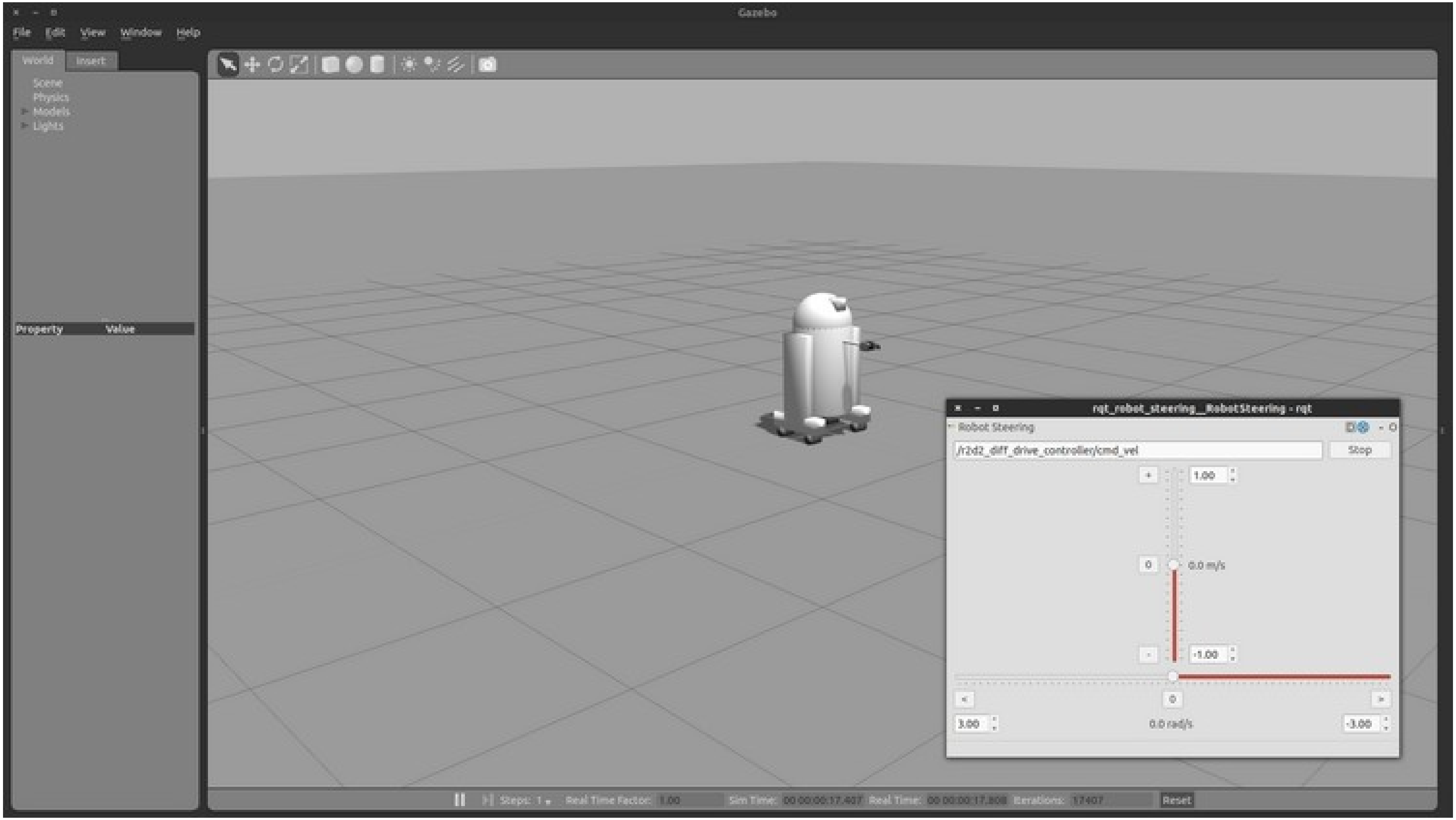
- Supported descriptions :
  - Camera
  - Ray
- Proposal Descriptions
  - IMU
  - Magnetometer
  - Gps
  - Contact
  - Force Torque
  - Sonar
  - Rfidtag
  - Rfid

```
<sensor name="my_ray_sensor" update_rate="20">
  <parent link="optical_frame_link_name"/>
   <origin xyz="0 0 0" rpy="0 0 0"/>
   <ray>
     <horizontal samples="100" resolution="1"
min_angle="-1.5708" max_angle="1.5708"/>
     <vertical samples="1" resolution="1" min_angle="0"
max_angle="0"/>
   </ray>
</sensor>
```

```
<sensor name="my_camera_sensor"
update_rate="20">
   <parent link="optical_frame_link_name"/>
   <origin xyz="0 0 0" rpy="0 0 0"/>
   <camera>
     <image width="640" height="480"
hfov="1.5708" format="RGB8" near="0.01"
far="50.0"/>
   </camera>
 </sensor>
```

http://wiki.ros.org/urdf/XML/sensor
http://wiki.ros.org/urdf/XML/sensor/proposals

R2D2 – From URDF to Gazebo



/r2d2_diff_drive_controller/cmd_vel

1.00

0.0 m/s

-1.00

0

3.00

0.0 rad/s

-3.00

Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:00:00:17.407 Real Time: 00 00:00:00:17.808 Iterations: 17407

http://wiki.ros.org/urdf/Tutorials/Using a URDF in Gazebo

References

- wiki.ros.org
- cmake.org
- design.ros2.org

Robotics Control Softwares

Ludovic Saint-Bauzel (saintbauzel@isir.upmc.fr)
2021