TECHNISCHE UNIVERSITÄT DRESDEN

CeTI Centre for Tactile Internet with Human-in-the-Loop

Software Technology Group
U2 / TP5

CeTI Summer School

Developing robotic applications with ROS

// 02.09.2020

Robot Operation System (ROS)

Motivation

“Hey Robot, please pick up a piece of garbage and put it into the box.”

(1)

Robot Operation System (ROS)

Motivation

“Hey Robot, please pick up a piece of garbage and put it into the box.”

(1)

- *How to control the robot?*
- *How to understand the environment?*
- *How to create applications?*

The Robot Operation System (ROS)

Robot Operation System (ROS)

What is ROS?

"A middleware for robotic software development, designed for heterogenous computing environments."

ROS (5)

- Provides **services** accordingly designed for:
  - hardware abstraction
  - low-level device control
  - message-passing
  - ...

(6)

- **Distributed:** programs running on multiple devices & communication via network → Peer-2-Peer communication
- **Multi-lingual:** programs can be written in any language for which a client library exists (C++, Python, Java ...)
- **Free and Open Source**

Basic Concepts of ROS

Robot Operation System (ROS)

ROS Nodes & Master

Starting ROS

ROS Master:

- Central registry for all ROS based processes (nodes)
  - Every node registers automatically on startup
- Communication management between nodes
- Provided by every ROS installation

Important Command Line Commands:

```
> roscore
```

Robot Operation System (ROS)

ROS Nodes & Master

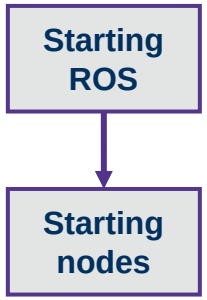
Starting ROS

ROS Master:

```
sebastian@jarvis:~/ros-workspaces/gripper_sim_ws_3/panda_gazebo_workspace$ roscore
... logging to /home/sebastian/.ros/log/de5d5ace-e79e-11ea-8d8b-f875a49f9fe4/roslaunch-jarvis-5872.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://jarvis:35659/
ros_comm version 1.14.6


SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.6

NODES

auto-starting new master
process[master]: started with pid [5910]
ROS_MASTER_URI=http://jarvis:11311/

setting /run_id to de5d5ace-e79e-11ea-8d8b-f875a49f9fe4
process[rosout-1]: started with pid [5922]
started core service [/rosout]
```

Robot Operation System (ROS)

ROS Nodes & Master

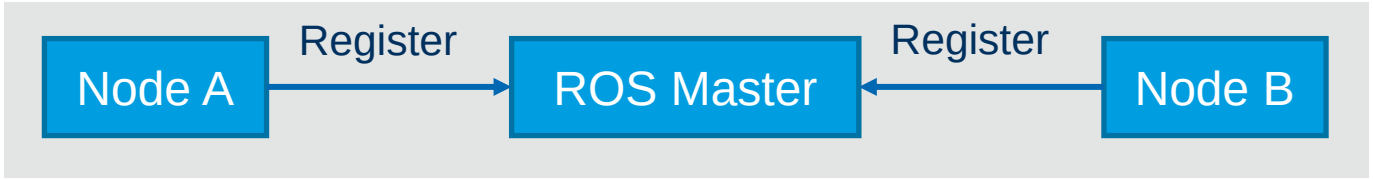


ROS Node:

- Single-process and single-purpose executable program
- Separately compiled, executed, and managed
- Bundled as packages

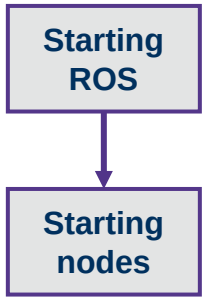
Important Command Line Commands:

```
> rosrun package_name node_name
> rosnode list
```



Robot Operation System (ROS)

ROS Nodes & Master



ROS Node:

- Single-process and single-purpose executable program
- Separately compiled, executed, and managed
- Bundled as packages

Important Command Line Commands:

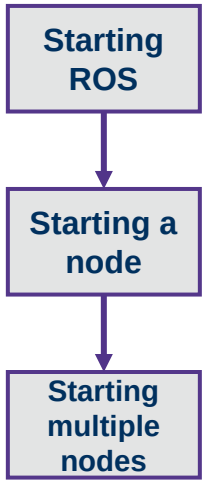
```
> rosrun package_name node_name
> rosnode list
```

```
sebastian@jarvis:~/ros-workspaces/gripper_sim_ws_3/panda_gazebo_workspace$ rosrun rviz rviz
[ INFO] [1598448353.859517776]: rviz version 1.13.13
[ INFO] [1598448353.859559738]: compiled against Qt version 5.9.5
[ INFO] [1598448353.859572402]: compiled against OGRE version 1.9.0 (Ghadamon)
[ INFO] [1598448353.862363538]: Forcing OpenGl version 0.
[ INFO] [1598448354.367592639]: Stereo is NOT SUPPORTED
[ INFO] [1598448354.367714549]: OpenGl version: 3 (GLSL 1.3).
```

```
sebastian@jarvis:~/ros-workspaces/gripper_sim_ws_3/panda_gazebo_workspace$ rosnode list
/controller_spawner
/controller_spawner_hand
/gazebo
/gazebo_gui
/joint_position_launcher
/joint_state_desired_publisher
/move_group
/robot_state_initializer_node
/robot_state_publisher
/rosout
/rqt_console
/rviz_jarvis_8819_3715647439979158483
```

Robot Operation System (ROS)

ROS Launch



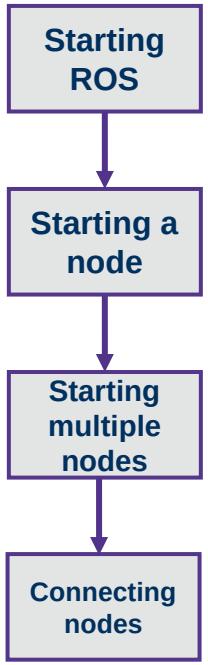
Problem: Launching all nodes of an application by console can get messy.

- **Solution: ROS Launch**
  - Allows launching multiple nodes and parameter configuration
  - Nodes to launch and parameters are in included in XML-formatted files
    - *.launch -files
  - Launches ROS master (roscore) automatically

```
1  <launch>
2      <include file="$(find panda_simulation)/launch/simulation.launch"/>
3      <include file="$(find panda_moveit_config)/launch/moveit_rviz.launch" />
4      <node pkg="sample_applications" type="BasicJointSpacePlanner"
5              name="BasicJointSpacePlannerInstance" respawn="false" output="screen"/>
6  </launch>
```

Robot Operation System (ROS)

ROS Topics, Publishers & Subscribers

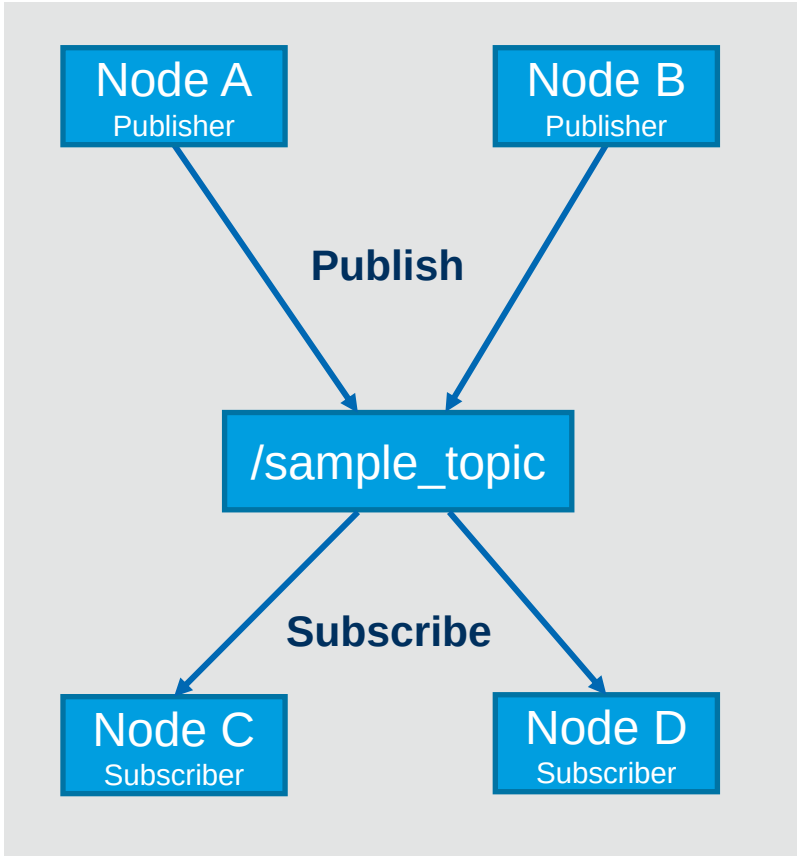


ROS Topics:

- Named buses over for messages exchange
  - Nodes publish or subscribe to topics
  - Nodes are not aware of who they are communicating with
  - Communication protocols: UDP or TCP/IP
  - Usually: 1 publisher & n subscribers

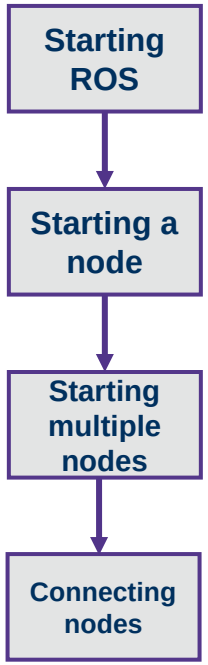
Important Command Line Commands:

```
> rostopic list
> rostopic info /topic_name
> rostopic echo /topic_name
```



Robot Operation System (ROS)

ROS Topics, Publishers & Subscribers



ROS Topics:

- Named buses over for messages exchange
  - Nodes publish or subscribe to topics
  - Nodes are not aware of who they are communicating with
  - Communication protocols: UDP or TCP/IP
  - Usually: 1 publisher & n subscribers

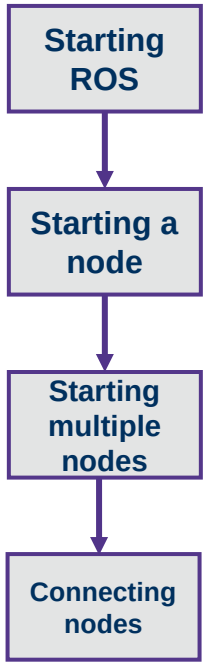
Important Command Line Commands:

```
> rostopic list
> rostopic info /topic_name
> rostopic echo /topic_name
```

```
sebastian@jarvis:~/ros-workspaces/gripper_sim_ws_3/panda_gazebo_workspace$ rostopic info /joint_states
Type: sensor_msgs/JointState

Publishers:
 * /gazebo (http://jarvis:36675/)

Subscribers:
 * /joint_state_desired_publisher (http://jarvis:41239/)
 * /robot_state_publisher (http://jarvis:43237/)
```

Robot Operation System (ROS)

ROS Messages



ROS Messages:

- ROS defines description language for message types
  - Integers, floats, booleans, strings
  - Arrays and objects (nested messages)

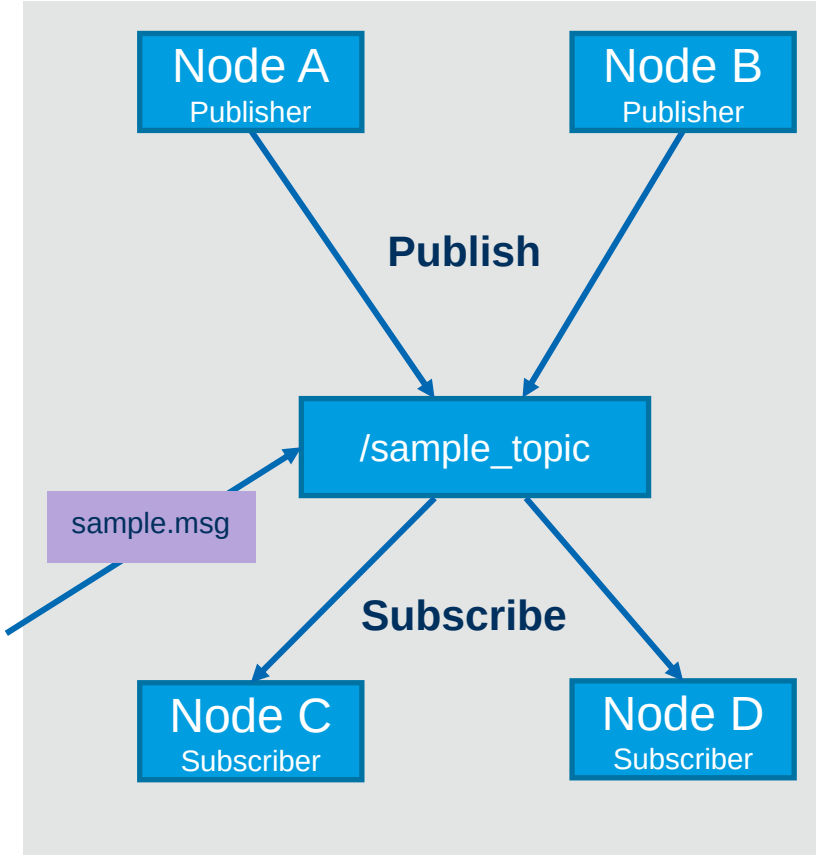
Important Command Line Commands:

```
> rostopic type /topic_name
> rostopic pub /topic_name data_type data
```

```
int64     counter
uint32    height
string    name
double[]  data
```

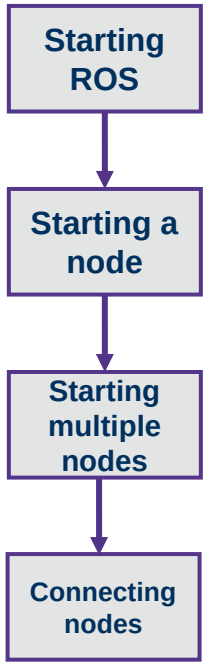
sample.msg

Message Definition



Robot Operation System (ROS)

ROS Services



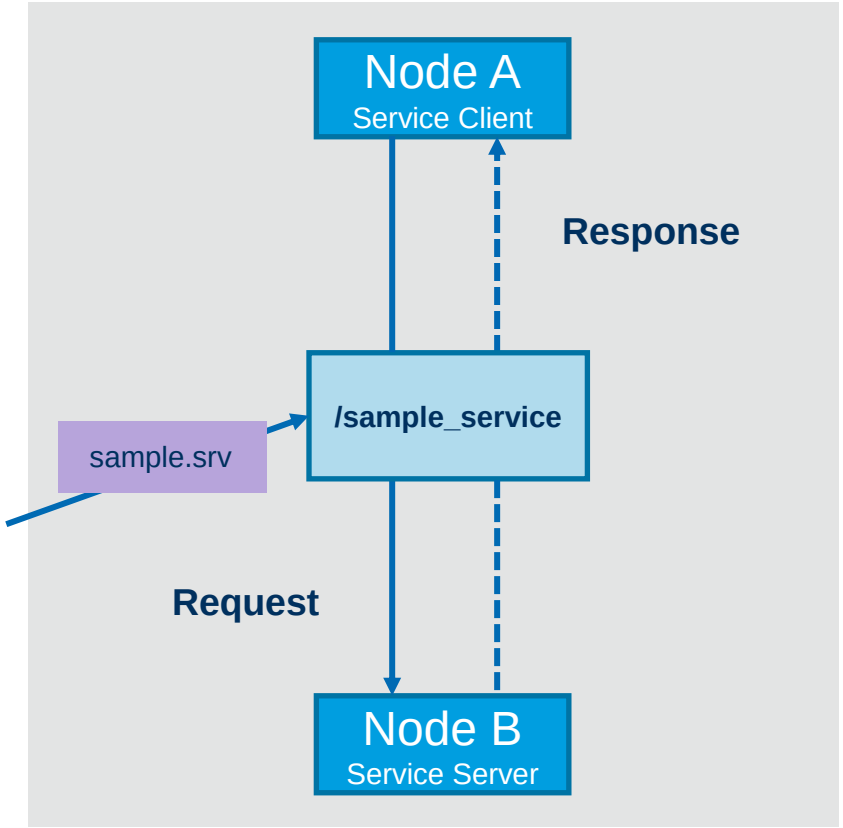
- Request / response –based communication between nodes
  - **Service Server:** advertises / provides services to nodes
  - **Service Client:** uses advertised services
- Structure of service messages similar to topic messages:

```
int64    param1
uint32   param2
---
string   result
double[] data
```

sample.srv

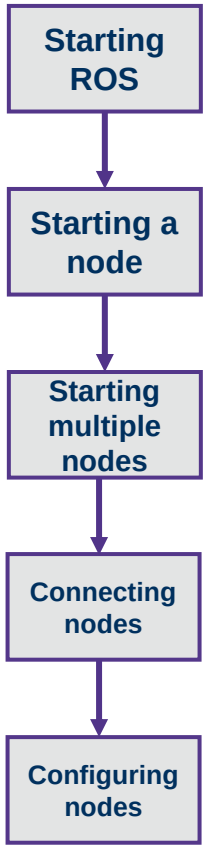
Important Command Line Commands:

```
> rosservice list
> rosservice call /service_name /args
```



Robot Operation System (ROS)

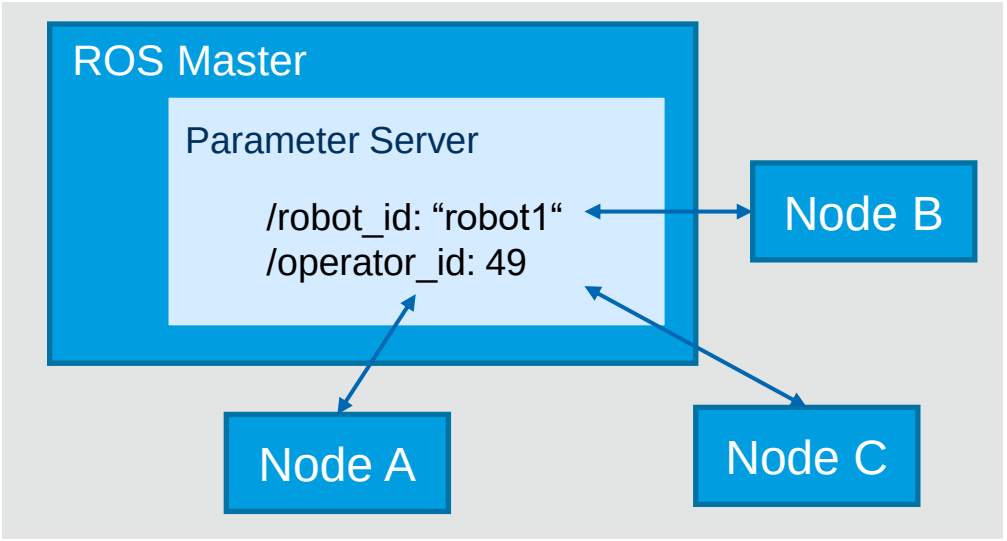
ROS Parameter Server



- Shared key-value-storage
  - Performance optimized
  - Encapsulation into name spaces
  - Optimal for configuration parameters
- Nodes use the parameter server to store / retrieve parameters at runtime
- Multiple possibilities to define parameters:
  - In code of ROS nodes
  - In YAML configuration files
  - Console

Important Command Line Commands:

```
> rosparam list
> rosparam get param_name
> rosparam set param_name new_value
```



ROS Packages and how to build them

Robot Operation System (ROS)

Catkin Build System

Creating Workspaces

- **Catkin:**
  - The build system infrastructure of ROS
    - Creates executables and libraries
  - Included by default when ROS is installed
  - Based on CMake and Python (wraps CMake)

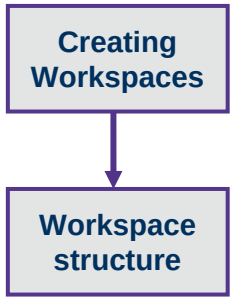
- **Catkin Workspace:**
  - Folder where you modify, build, and install catkin packages
  - Defines the context for the catkin build system

Creating a workspace:

```
> source /opt/ros/<ros-distro>/setup.bash
> mkdir –p ~/catkin_ws/src
> cd ~/catkin_ws
> catkin build
```

Robot Operation System (ROS)

Catkin Build System

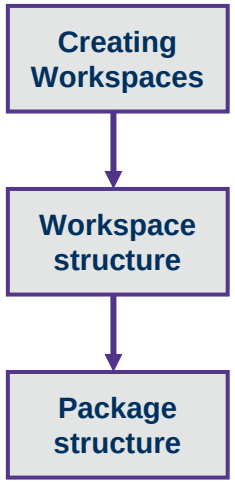


- **Main directories of catkin:**
  - src:
    - Contains the source code as packages
  - build:
    - Build space, used by CMake to build packages
    - Contains cached information and temporary files
  - devel:
    - Contains built targets

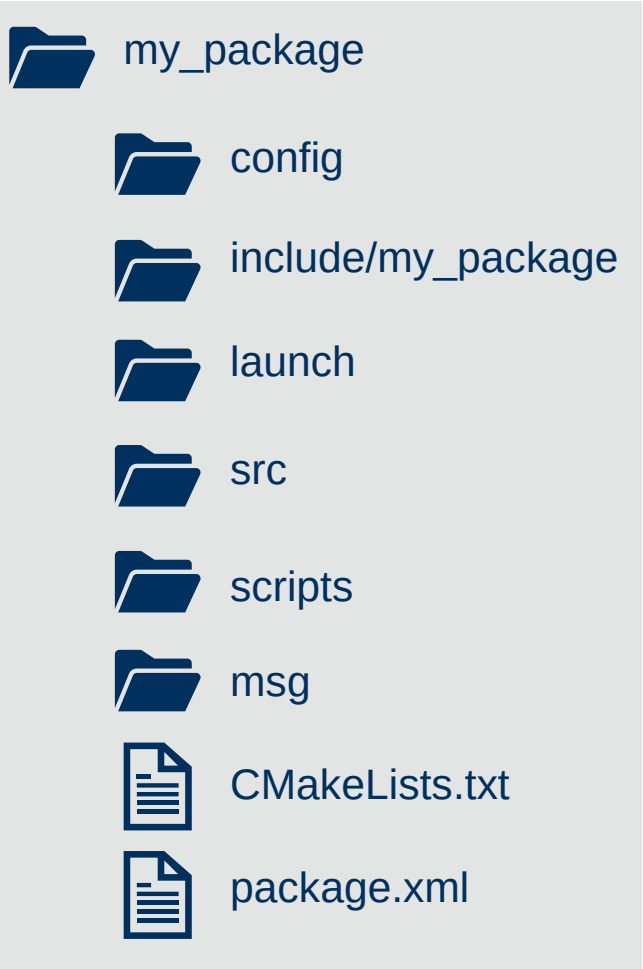
"Developing packages only requires modification of the src-directory!"

Robot Operation System (ROS)

ROS Packages



- Nodes, libraries and other ROS software is organized in packages
- Possible parts of a package:
  - Source code (C++, Python, Java, …)
  - Launch files
  - Configuration files → e.g. YAML, URDF, …
  - Message / Service / Action – definitions
  - Documentation
  - Data (images, text, …)
  - Tests
- Packages can depend on / require other packages
  - ➢ Dependencies are checked during build process via Catkin



Robot Operation System (ROS)

ROS Packages

Creating Workspaces

Workspace structure

Package structure

- **package.xml**
  - Definition of meta-information like package name, version, contact-information
  - Licensing information
  - Listing of dependencies to other packages
- **CMakeLists.txt**
  - Build-recipe for the CMake build system
  - Meta-information: project-name, C++ -Version, CMake-Version
  - Listing of Message-, Service- and Action-Classes to generate
  - Definition of executables and libraries to build and link
  - Definition of Tests to build and link

Robot Operation System (ROS)

Catkin Build System

- Creating Workspaces
- Workspace structure
- Package structure
- Creating Packages

Creating Packages from Command Line:

> catkin create pkg package_name {dependencies}

Building Packages from Command Line:

> catkin build package_name

Resolving caching problems:

> catkin clean package_name

Exposing the Package to the Command Line:

> source devel/setup.bash

```
sebastian@jarvis:~/ros-workspaces/public_ros_2/panda_gazebo_workspace$ catkin build
--------------------------------------------------------------------------------
Profile:                     default
Extending:          [cached] /opt/ros/melodic
Workspace:                   /home/sebastian/ros-workspaces/public_ros_2/panda_gazebo_workspace
--------------------------------------------------------------------------------
Build Space:        [exists] /home/sebastian/ros-workspaces/public_ros_2/panda_gazebo_workspace/build
Devel Space:        [exists] /home/sebastian/ros-workspaces/public_ros_2/panda_gazebo_workspace/devel
Install Space:      [unused] /home/sebastian/ros-workspaces/public_ros_2/panda_gazebo_workspace/install
Log Space:          [exists] /home/sebastian/ros-workspaces/public_ros_2/panda_gazebo_workspace/logs
Source Space:       [exists] /home/sebastian/ros-workspaces/public_ros_2/panda_gazebo_workspace/src
DESTDIR:            [unused] None
--------------------------------------------------------------------------------
Devel Space Layout:          linked
Install Space Layout:        None
--------------------------------------------------------------------------------
Additional CMake Args:       None
Additional Make Args:        None
Additional catkin Make Args: None
Internal Make Job Server:    True
Cache Job Environments:      False
--------------------------------------------------------------------------------
Whitelisted Packages:        None
Blacklisted Packages:        None
--------------------------------------------------------------------------------
Workspace configuration appears valid.
--------------------------------------------------------------------------------
[build] Found '4' packages in 0.0 seconds.
[build] Package table is up to date.
Starting  >>> franka_description
Starting  >>> panda_simulation
Starting  >>> sample_applications
Finished  <<< panda_simulation                [ 0.2 seconds ]
Finished  <<< sample_applications             [ 0.2 seconds ]
Finished  <<< franka_description              [ 0.1 seconds ]
Starting  >>> panda_moveit_config
Finished  <<< panda_moveit_config             [ 0.1 seconds ]
[build] Summary: All 4 packages succeeded!
[build]   Ignored:   None.
[build]   Warnings:  None.
[build]   Abandoned: None.
[build]   Failed:    None.
[build] Runtime: 0.4 seconds total.
```

ROS Tooling

Robot Operation System (ROS)

ROS Tooling: rqt

- **rqt: Univeral User Interfaces for ROS**
  - Qt-based user interface & framework for GUI development for ROS
  - Allows development of GUI-Plugins integrated with ROS
  - Many Plugins on the market

http://wiki.ros.org/rqt_image_view

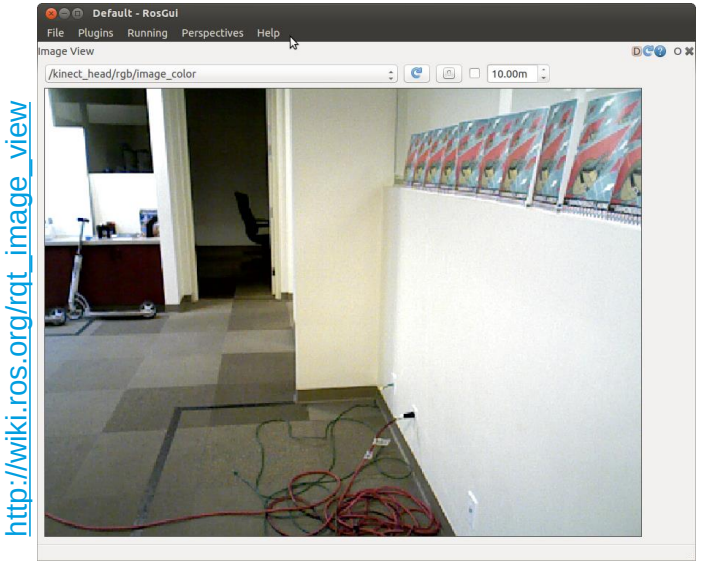
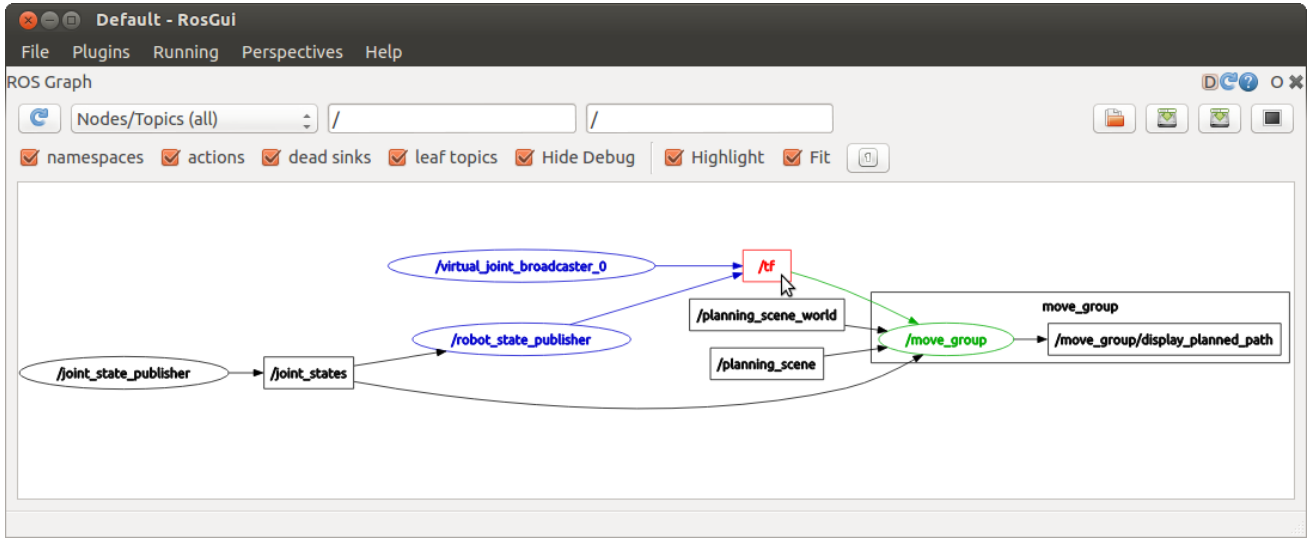


Image visualization with **rqt_image_view**

http://wiki.ros.org/rqt_graph



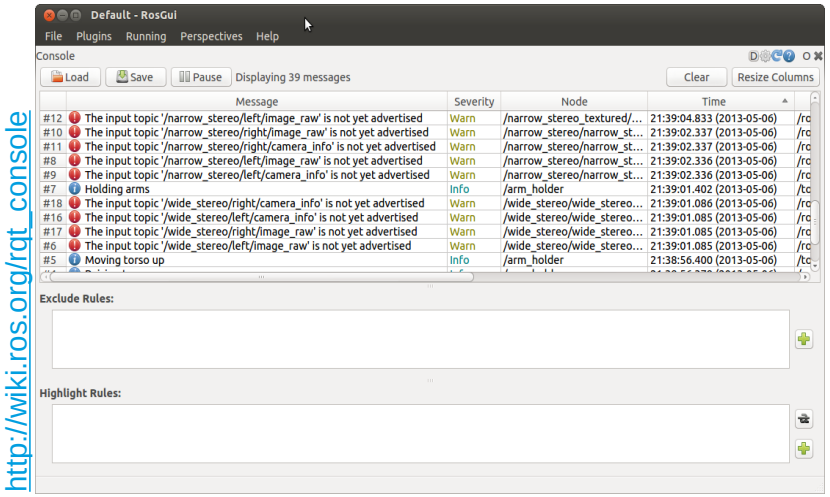
ROS computation graph visualization with **rqt_graph**

Robot Operation System (ROS)

ROS Tooling: rqt

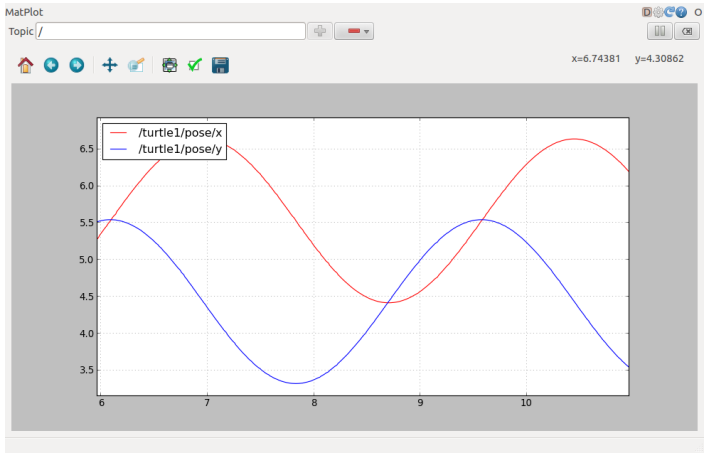
- **rqt: Univeral User Interfaces for ROS**
  - Qt-based user interface & framework for GUI development for ROS
  - Allows development of GUI-Plugins integrated with ROS
  - Many Plugins on the market

http://wiki.ros.org/rqt_console

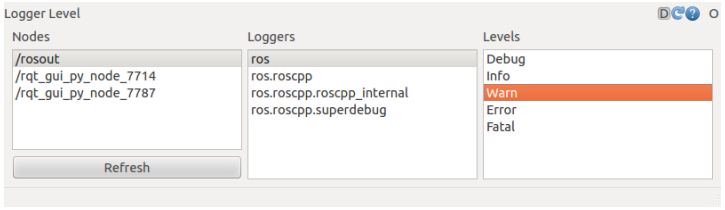


Message visualization with **rqt_console**

http://wiki.ros.org/rqt_plot



Numeric value visualization with **rqt_plot**



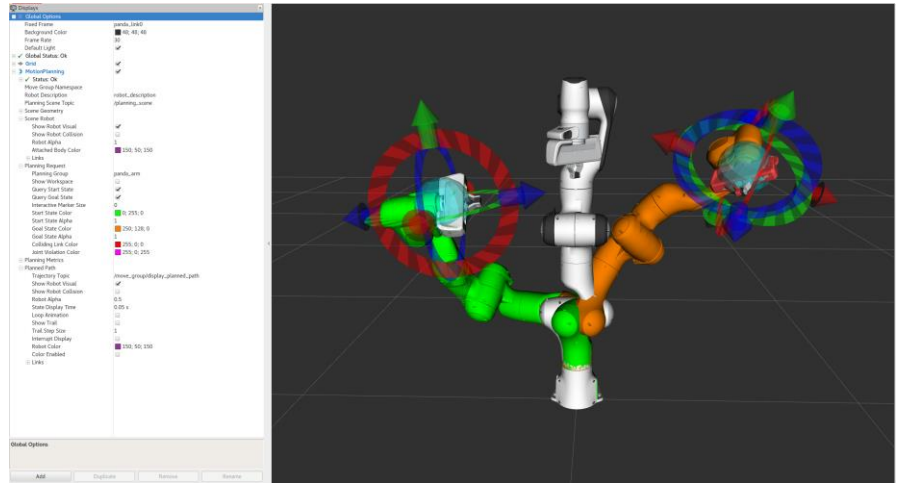
http://wiki.ros.org/rqt_logger_level (8)

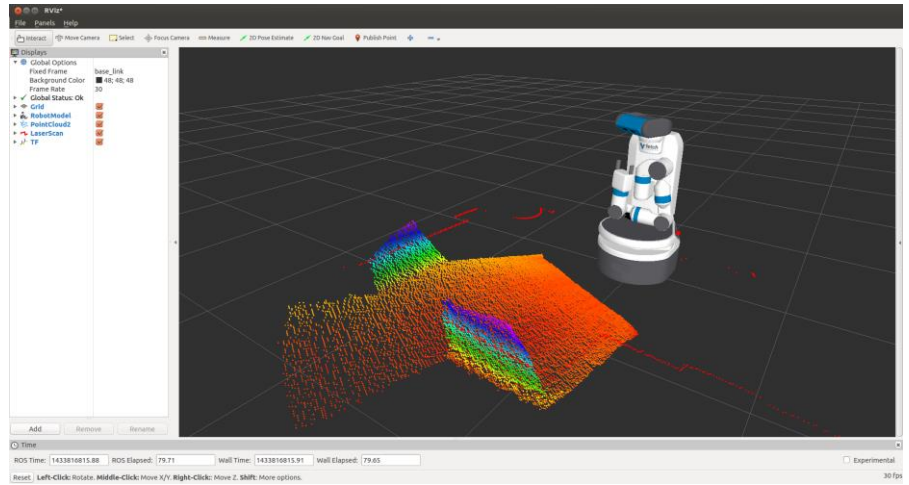
Logger level adaption with **rqt_logger_level**

Robot Operation System (ROS)

ROS Tooling: RViz

- **Rviz: A 3D visualization tool for ROS**
  - Ability to visualize camera-images, stereo-data, lidar-sensor data
  - Visualization of data by subscription to topics
  - Tools to publish additional information
    - Trajectory paths
    - Text
    - …
  - Multiple camera viewpoints (orthographic, top-down, …)
  - Setup/Configuration can be saved in a file
  - Extensible with plugins





Important Command Line Command:

```
> rosrun rviz rviz
```

(7)

Robot Operation System (ROS)

ROS Tooling: Bags

- Allows **recording of messages** over time
  - Only messages → not services
- **Storage format** for storing message data
  - *.bag –files
- **Use cases:** logging, dataset recording for later visualization / analysis and debugging

```
> rosbag record --all
> rosbag record topic_name_1 topic_name_2
```

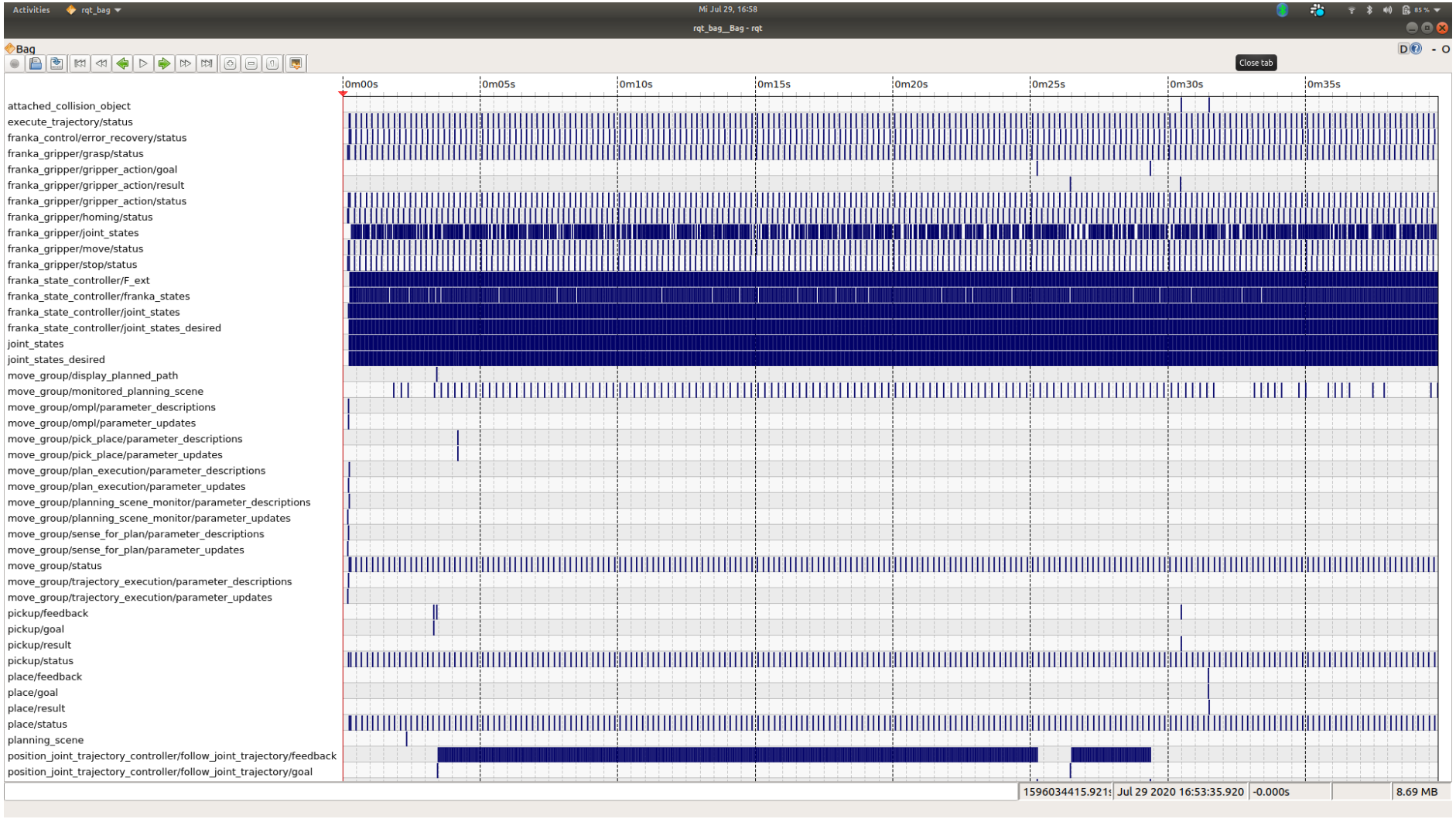
- Possibility to **replay saved messages**

```
> rosbag play my_bag.bag
```

Robot Operation System (ROS)

ROS Tooling: Bags

Bags visualization with **rqt_bag**



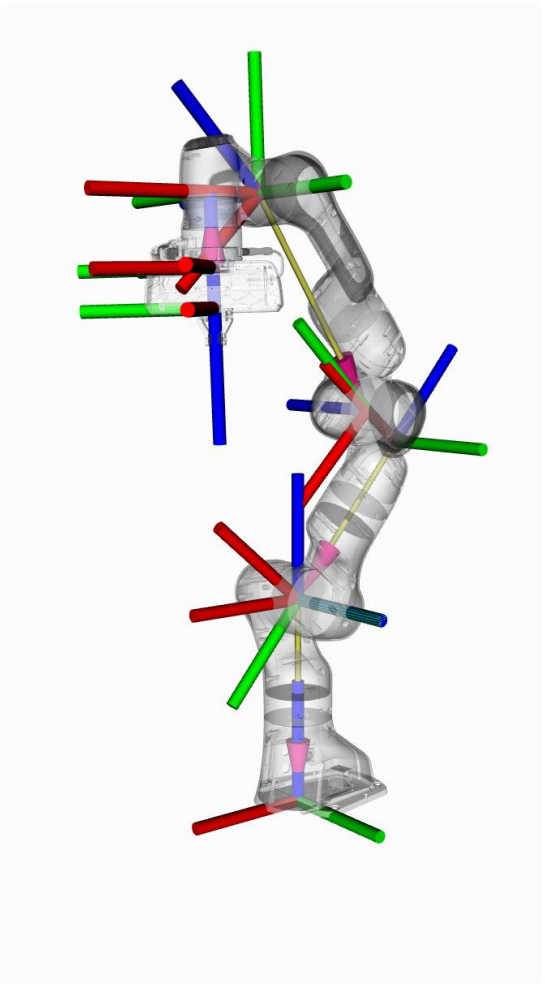
Robot Operation System (ROS)

ROS Tooling: TF Transformation System

- Every joint and every link maintains its own coordinate frame
  - Rational: Transformations between them are easy

- TF maintains relationships between coordinate frames
  - Structure: Tree
  - Ability to transform vectors and points between frames

- Implementation based on ROS Publishers and Subscribers
  - /tf –topic: transformation at a time
  - /tf_static –topic: transformations that will stay unchanged
  - New transforms are added by publishing to the tf topics



(9)

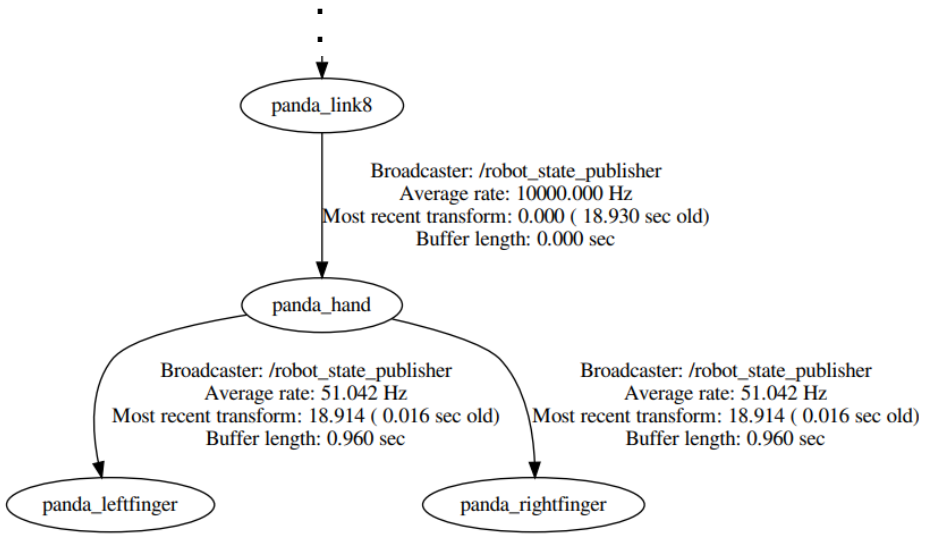
Robot Operation System (ROS)

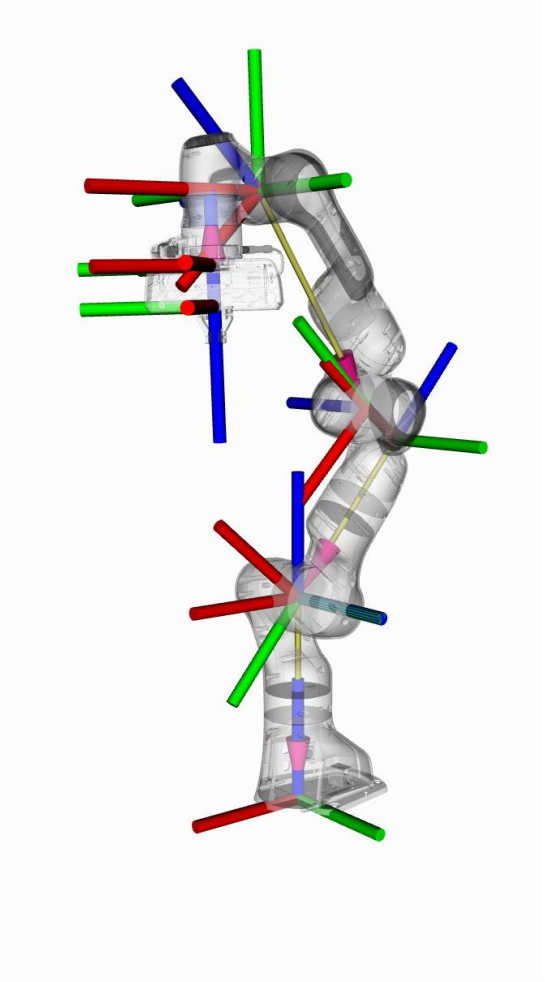
ROS Tooling: TF Transformation System

- TF2 Tooling:
  - C++ / Python API
  - Command Line

    > rosrun tf tf_monitor

    > rosrun tf tf_echo source target
  - View Frames rqt tool





(9)

The Motion Planning Framework MoveIt

MoveIt
Motivation

“Hey Robot, please pick up a piece of garbage and put it into the box.”

(1)

- *How to pick up an object?*
- *How to move between positions?*
- *How to understand the surroundings?*
- *How to avoid obstacles?*

MoveIt
Motivation

"Hey Robot, please pick up a piece of garbage and put it into the box."

Required building blocks:

- Generation of motion trajectories
- Obstacle detection and avoidance
- Ability to grasp objects
- Parametrization (velocity / forces)
- Constraining of motions

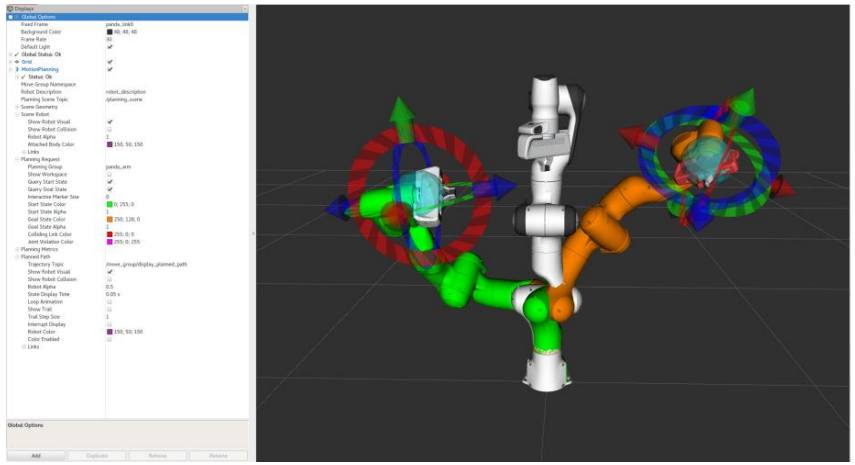
MoveIt

What is MoveIt?

Motion-Planning-Framework for ROS

- **Motion Planning** (**Navigation** and **Manipulation**)
  - Construction of robotic trajectories
  - Environment modelling (Primitives, Meshes, Octomaps)
  - Obstacle Avoidance
- Computation of **inverse and forward kinematics**
  - Different algorithms usable, Time-parameterizable
- **Many provided robots** (for example Franka Emika Panda)
  - Integration with user-defined robotic controllers
- Integrated plan **visualization** (RViz) and connection to **simulators**

MoveIt (2)



MoveIt

Basic robotic concepts

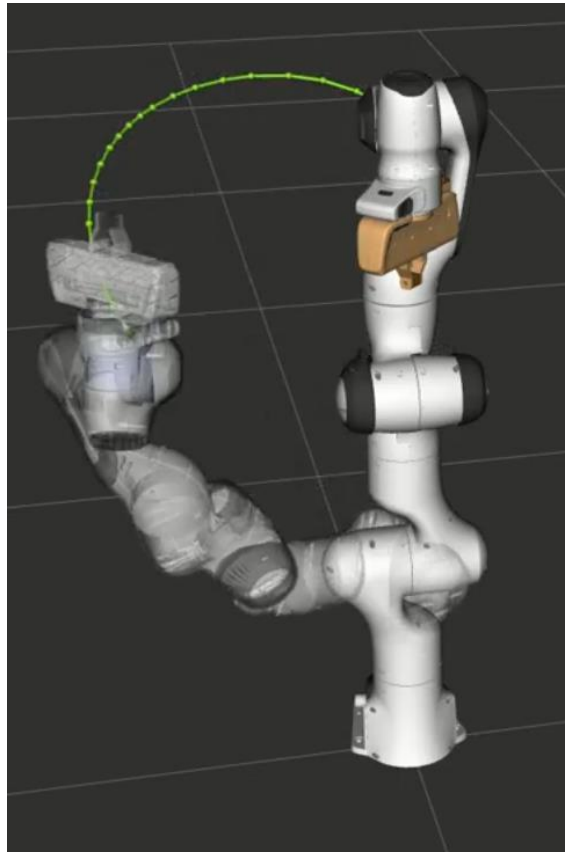
- **Pose:** Position & Orientation of an object (e.g. joint, end-effector, cube,...)
- **Path:** pure geometric description of motions (could e.g. contain list of poses)
  - Globally planned, accounts obstacle avoidance
- **Trajectory:** path + velocities & accelerations in each of its points
- **Forward kinematic:** computation of the position of the end-effector from specified values for the joint parameters (e.g. joint-angles)
- **Inverse kinematics:** calculation of joint parameters needed to place the end-effector in a given position and orientation (relative to the start of the kinematic chain)

MoveIt

Basic robotic concepts

- **Pose in Joint Space:**
  - Description of a robot's pose using the rotation angles
  - Description for each individual joint of a robot

- **Pose in Cartesian Space:**
  - Description of a robot's pose using position and orientation of the end effector
  - Representation is not complete for defining a pose
    - ➢ An inverse kinematic solver must compute a joint space trajectory



MoveIt

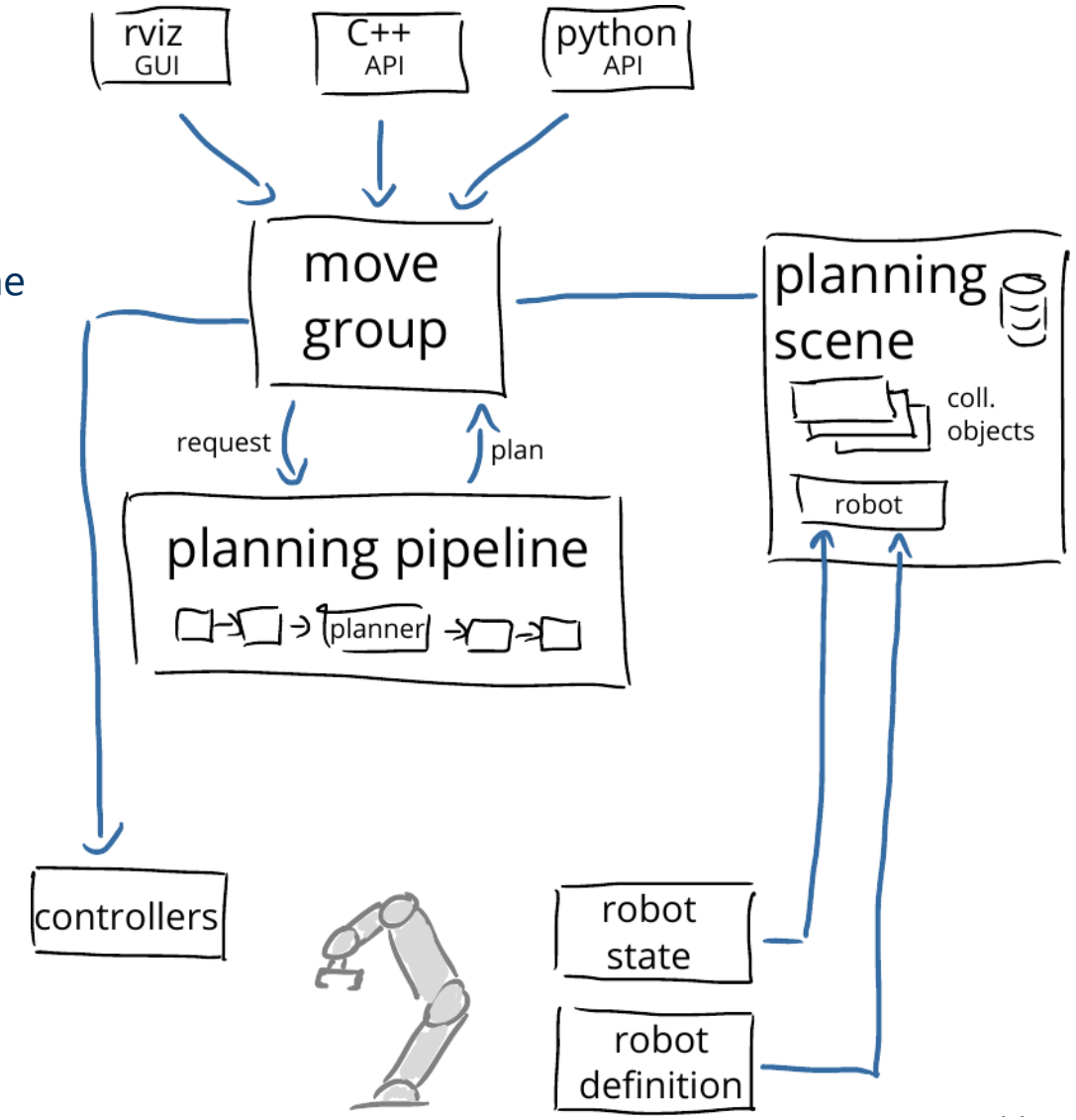
Architecture

Move Group:

- Central integrator pulling and coordinating functionalities of the other components
- Interface for the user based on ROS services and actions

Planning Scene:

- Representation of robotic state & environmental information
- Retrieves robotic state information via joint states topics
- Retrieves sensor information:
  - ➢ Build in support for point clouds & depth images
- Retrieves world geometry from the user:
  - ➢ Collision objects (meshes, shapes, octomaps)



MoveIt

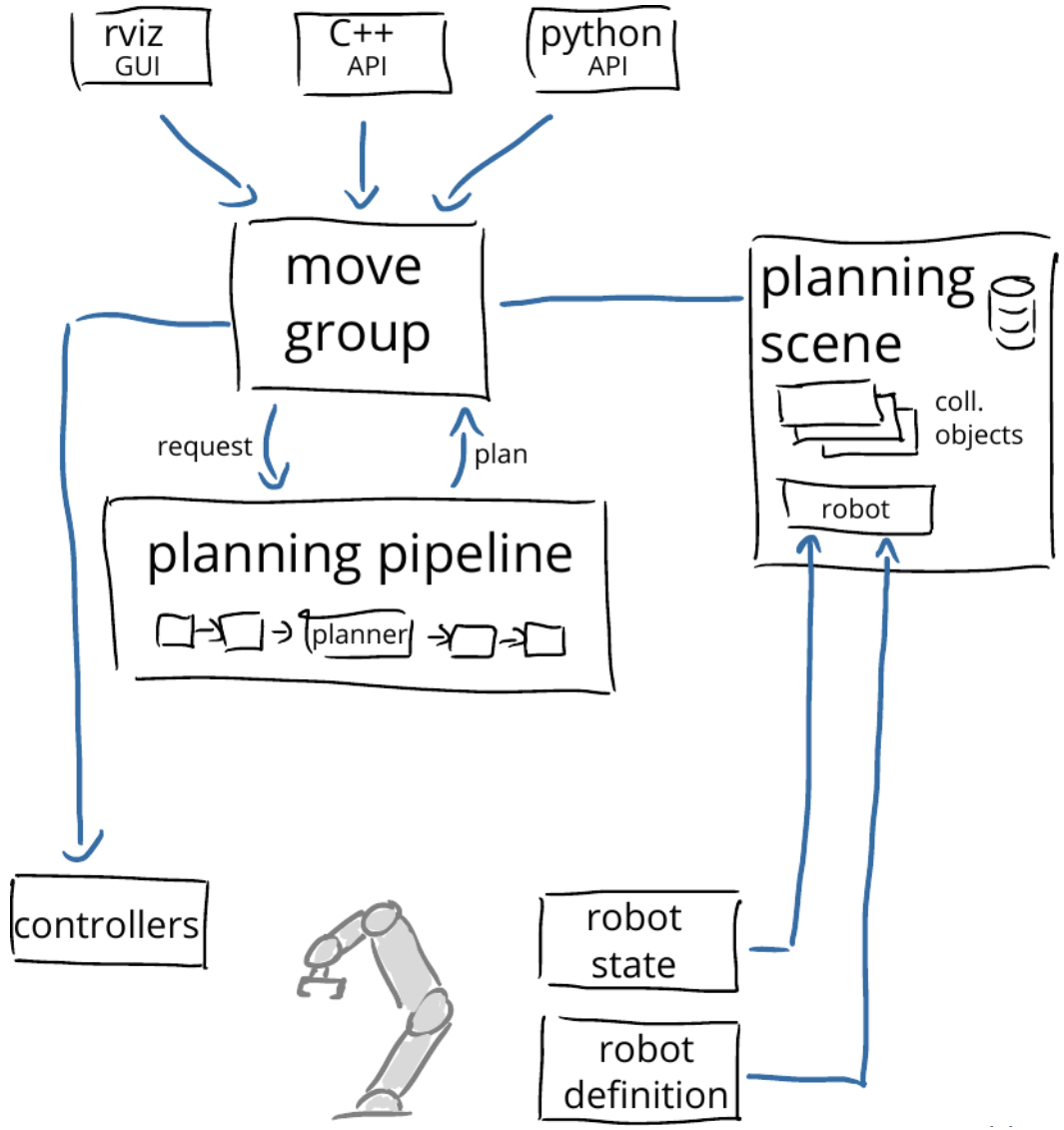
Architecture

Planning Pipeline:

- Responsible for trajectory computation
- Chains together motion planner with request adapters (pre-/post procession of motion plans)

Controllers:

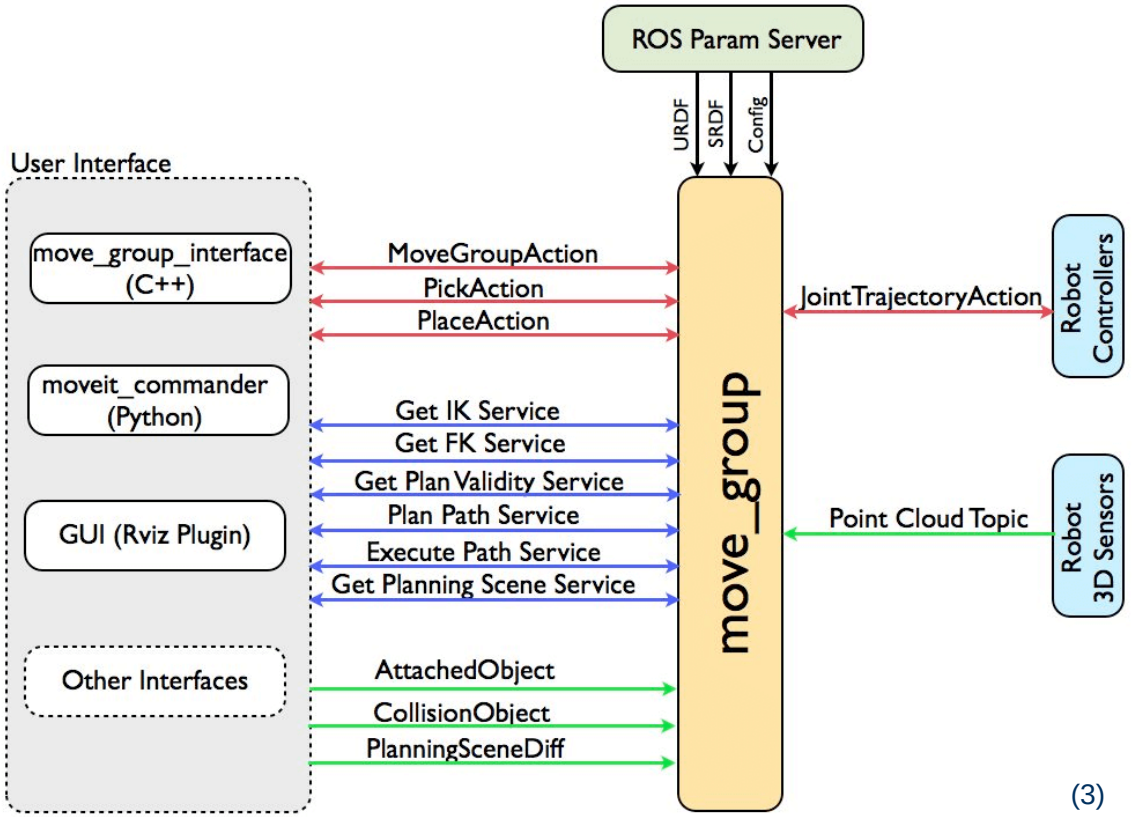
- Execution of the trajectories computed within Move Group
  - Receive `moveit_msgs::RobotTrajectory` messages
- Developed for specific robots
  - Typically provided by robot manufacturer



MoveIt

Details: Move Group

- Access to actions / services provided in three ways:
  - Python (moveit_commander)
  - C++ (move_group_interface)
  - GUI (Rviz Plugin)
- Communicates with robot through ROS topics and actions:
  - Retrieve current state (joint positions, orientation)
  - Get sensor data, Point Clouds, …
  - Receive commands
  - To instruct a robot's controller
  - …
- Configuration (details later):
  - Via ROS parameter server
  - Via MoveIt Config Package

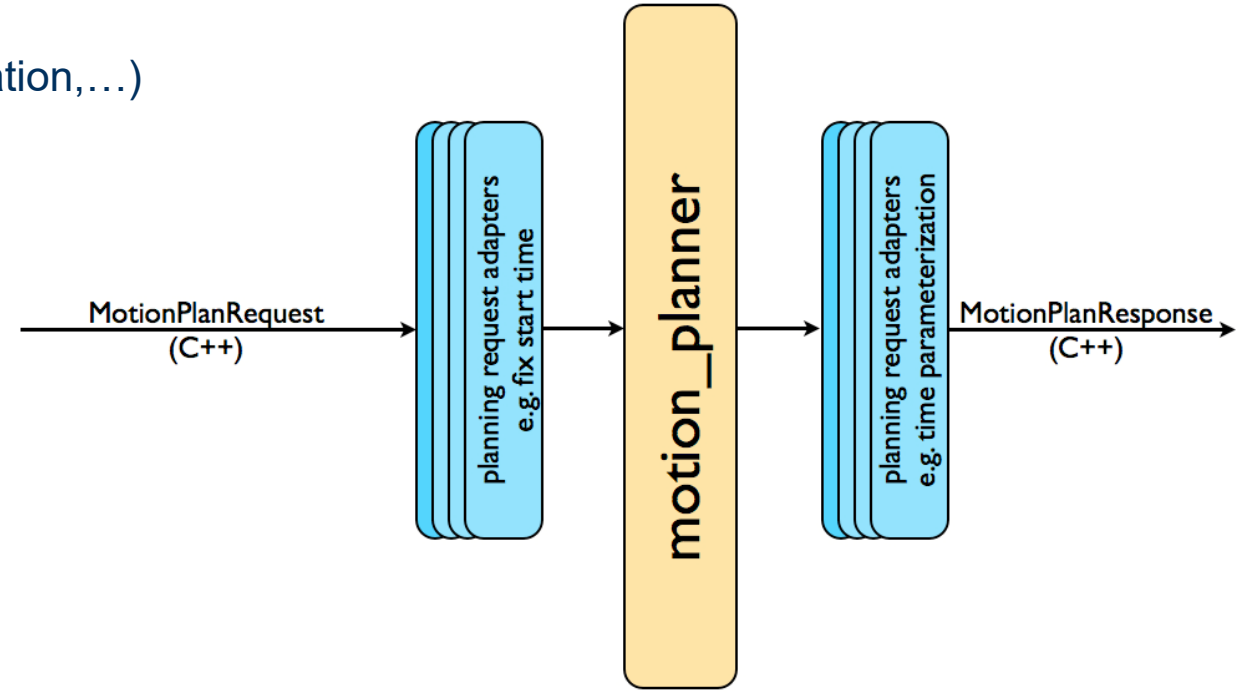


(3)

MoveIt

Details: Motion Planning Pipeline

- **Input:** Motion Plan Request
  - Definition of where to move robot to (position/orientation)
  - Possibility to attach objects
  - Possibility to add constraints (on position, orientation,…)
- **Planning request adapters**
  - Allows preprocessing of requests
    - Useful for example to modify start state
  - Allows postprocessing of plans
    - Useful to add time-parametrization
- **Motion Planner**
  - Various planners available (configured by user)
  - Solves inverse kinematic equations
  - Collision checking (configured by user)
- **Output:** Motion Plan for controller
  - Contains messages of type **moveit_msgs::RobotTrajectory**



(3)

MoveIt

Configuration

General Configuration: Robot description ROS package

- ➢ provided by manufacturer

Use case specific Configuration: MoveIt configuration ROS package

- Important concept: **Planning Group**
  - Set of joints and links, considered together for trajectory generation
  - Alternative: “joint model group”
- Configuration file types:
  - YAML
  - URDF (Unified Robot Description Format)
  - SRDF (Semantic Robot Description Format)

MoveIt

Configuration

- **Robot description ROS package**
  - Configuration of robot geometry via URDF
  - Configuration of surface meshes (used e.g. for visualization in Gazebo)
  - …

```
255     <link name="${arm_id}_link7">
256       <visual>
257         <geometry>
258           <mesh filename="package://${description_pkg}/meshes/visual/link7.dae"/>
259         </geometry>
260       </visual>
261       <collision>
262         <origin xyz="0 0 0.01" rpy="0 0 0"/>
263         <geometry>
264           <cylinder radius="${0.04+safety_distance}" length="0.14" />
265         </geometry>
266       </collision>
267       <collision>
268         <origin xyz="0 0 0.08" rpy="0 0 0"/>
269         <geometry>
270           <sphere radius="${0.04+safety_distance}"  />
271         </geometry>
272       </collision>
273       <collision>
274         <origin xyz="0 0 -0.06" rpy="0 0 0"/>
275         <geometry>
276           <sphere radius="${0.04+safety_distance}"  />
277         </geometry>
278       </collision>
279     </link>
```

MoveIt

Configuration

- **MoveIt configuration ROS package**
  - Configuration of Planning Groups
  - Configuration of used robot controllers
  - Configuration of joint limits
  - Configuration of used kinematic solver & collision checker
  - Semantic description of robot (via SRDF), linking to robot description package

```yaml
joint_limits:
  panda_joint1:
    has_velocity_limits: true
    max_velocity: 2.1750
    has_acceleration_limits: true
    max_acceleration: 3.75
  panda_joint2:
    has_velocity_limits: true
    max_velocity: 2.1750
    has_acceleration_limits: true
    max_acceleration: 1.875
  panda_joint3:
    has_velocity_limits: true
    max_velocity: 2.1750
    has_acceleration_limits: true
    max_acceleration: 2.5
  panda_joint4:
```

Extracts from: [1] Joint limit config

```yaml
controller_list:
 - name: position_joint_trajectory_controller
   action_ns: follow_joint_trajectory
   type: FollowJointTrajectory
   default: true
   joints:
     - panda_joint1
     - panda_joint2
     - panda_joint3
     - panda_joint4
     - panda_joint5
     - panda_joint6
     - panda_joint7
```

[2] Controller config

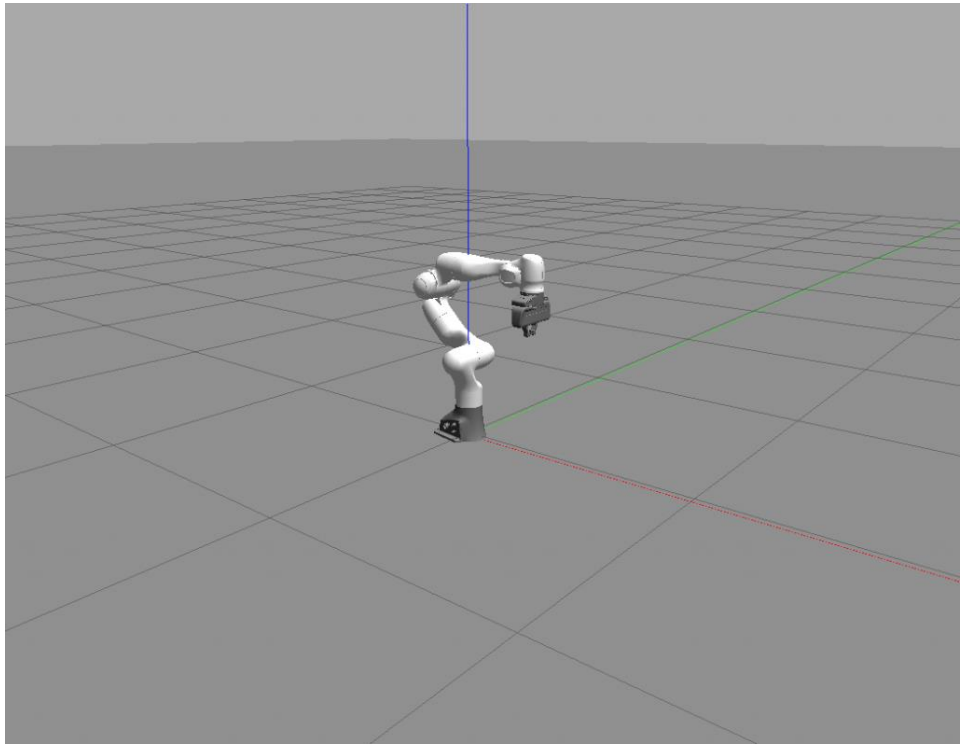
```xml
<group name="panda_arm_hand">
  <group name="panda_arm" />
  <group name="hand" />
</group>
<group_state name="open" group="hand">
  <joint name="panda_finger_joint1" value="0.035" />
  <joint name="panda_finger_joint2" value="0.035" />
</group_state>
<group_state name="close" group="hand">
  <joint name="panda_finger_joint1" value="0" />
  <joint name="panda_finger_joint2" value="0" />
</group_state>
```

[3] Semantic description

Simulation

Simulation
Gazebo

- **3D physics simulator**
  - Simulation of 3D rigid bodies
  - Simulation of motion dynamics
  - 3D visualization
  - User-Interface to introspect / manipulate variables
- **Gazebo Worlds**
  - Database containing predefined environments & robots
- **Connectivity**
  - ROS Integration
  - Can be extended with plugins
- Connection between a MoveIt and Gazebo possible
  - ➢ Based on Gazebo's ROS interface & Gazebo Plugins



Sources

Sources

(1) http://wiki.ros.org
(2) http://wiki.ros.org/ROS/Tutorials
(3) http://gazebosim.org
(4) https://www.theconstructsim.com/history-ros/

Image Sources

(1) https://i.ytimg.com/vi/MqSKb7cuvnc/maxresdefault.jpg
(2) https://camo.githubusercontent.com/c825d6376efd0510944399c3ae2687dcaefb9686/68747470733a2f2f6d6f766569742e726f732e6f72672f6173736574732f6c6f676f2f6d6f766569745f6c6f676f2d626c61636b2e706e67
(3) https://moveit.ros.org/documentation/concepts/
(4) https://www.cobofact.ch/wp-content/uploads/2018/09/Panda_Cobofact_web.png
(5) https://miro.medium.com/max/800/1*lsbkF4ybE4jDj2eOsqgeSg.png
(6) https://www.ros.org/news/assets_c/2016/10/1--XgoPd36umkXi6lXTGkCng-thumb-480x311-1681.png
(7) https://docs.fetchrobotics.com/_images/rviz.png
(8) http://wiki.ros.org/ROS/Tutorials/UsingRqtconsoleRoslaunch
(9) http://docs.ros.org/kinetic/api/moveit_tutorials/html/_images/panda_tf.png

Backup Slides

Robot Operation System (ROS)

A brief history of ROS

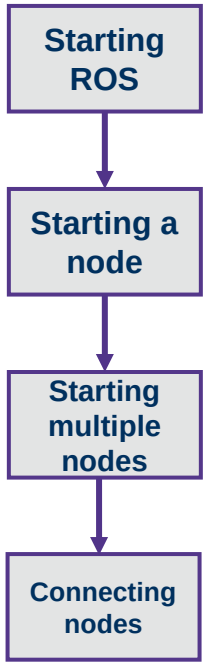
- **First development** in 2007 at Stanford Artificial Intelligence Laboratory
- **2007 – 2014:** Lead development by Willow Garage together with external contributors
  - **2009** first distribution release: ROS Mango Tango (ROS 0.4)
- **Since 2013** managed by OSRF (Open Robotics), new release every year
- **Since 2015:** Lack of support for real-time and security has been addressed in the creation of **ROS 2.0**
- **Current releases:**
  - ROS 1: Noetic & Melodic
  - ROS 2: Foxy Fitzroy

ROS (5)

(6)

Robot Operation System (ROS)

ROS Actions (Actionlib)

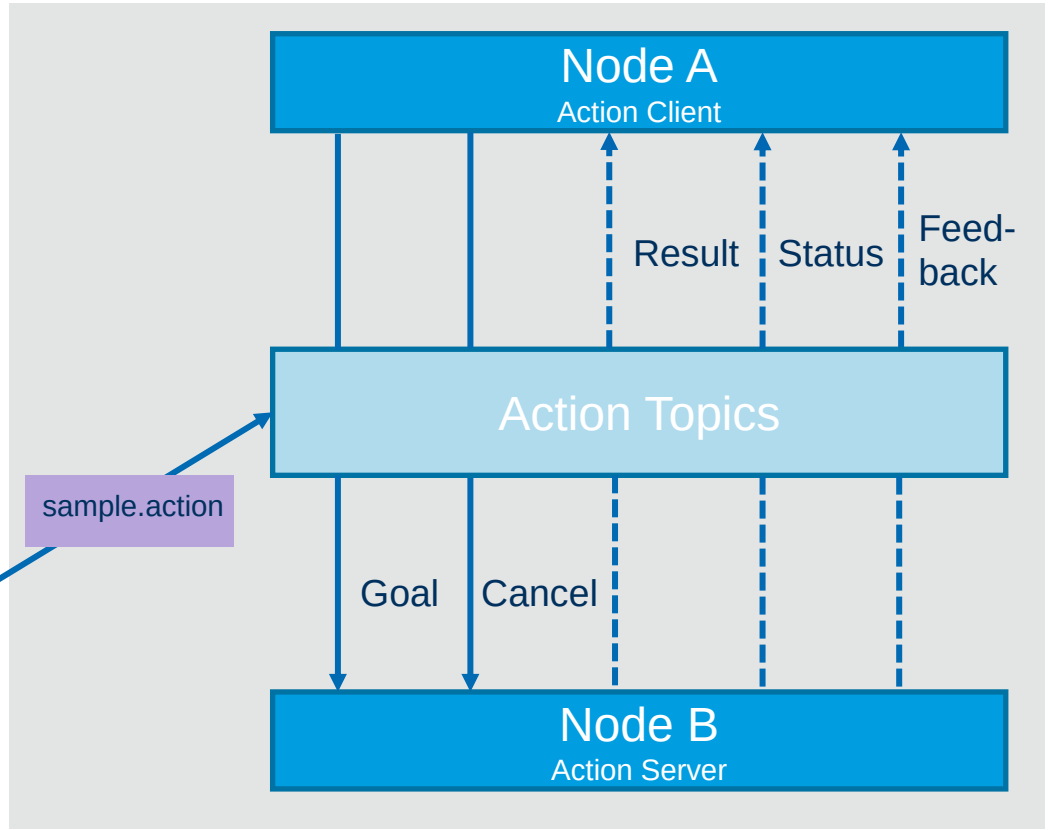


- Similar to the service pattern of ROS
  - Request / response –based communication
  - **Action Server:** provides actions to nodes
  - **Action Client:** uses provided actions
- Additional features:
  - Cancel a task
  - Receive feedback on task progress → stateful
  - Usage: time-extended & goal-oriented programs
  - ➢ Realized by 5 topics per action
- Structure of action messages:

```
int64[] input_data
---
int64   result
---
int64[] feeback
```

sample.action

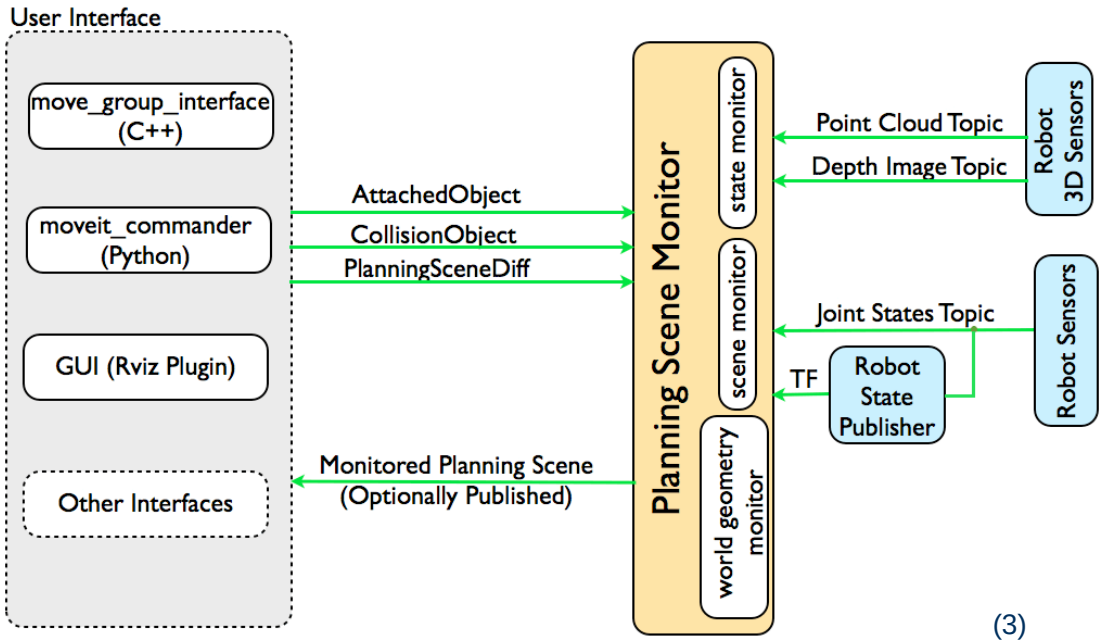
Action Definition



MoveIt

Details: Planning Scene

- Part of the Move Group (as planning scene monitor)
- Representation of robotic state & environmental information
- Retreives robotic state information:
  - ➢ Listens to joint states topic
- Retreives sensor information:
  - ➢ Build in support for point clouds & depth images
- Retreives world geometry from the user:
  - ➢ Collision objects (meshes, shapes, octomaps)



(3)