ROSPLAN TUTORIAL

ROSPlan is a framework for controlling ROS systems with Planning.

Outline of tutorial:

1. Practice with ROS
2. Write PDDL domains for robots
3. Learn how to use ROSPlan
4. Hands-on with the Turtlebot2

"ROSPlan: Planning in the Robot Operating System"
Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS-15). June 2015.

https://github.com/KCL-Planning/ROSPlan/wiki

ROSPLAN TUTORIAL

ROS (the Robot Operating System) is an open-source library for robotics.

Core concepts of ROS:
- Packages
- Nodes
- Messages

Packages contain nodes and messages.
Some example packages:

ros-indigo-navigation
ros-indigo-turtlebot
ros-indigo-mongodb-store
ros-indigo-spacenav-node
rosplan

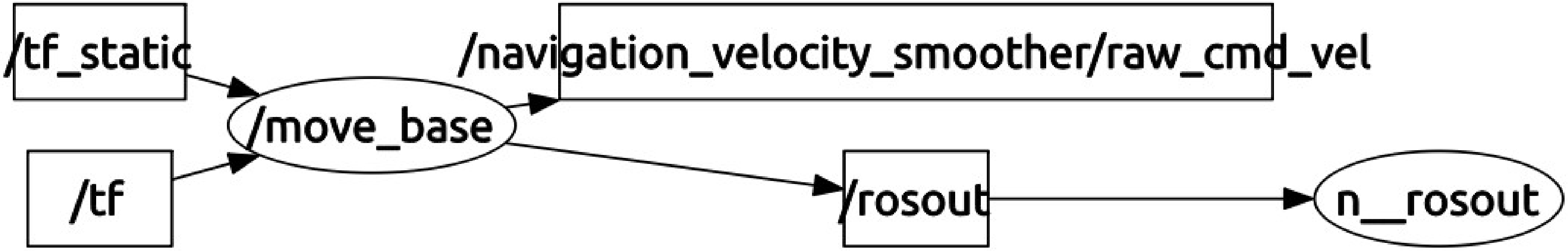
ROSPLAN TUTORIAL

Components in ROS run as nodes. Nodes can be distributed across different computers.

A node might take input and produce output.

Example **node**:

- move_base (navigation)
- - inputs desired location, transforms (positional data)
- - outputs velocity



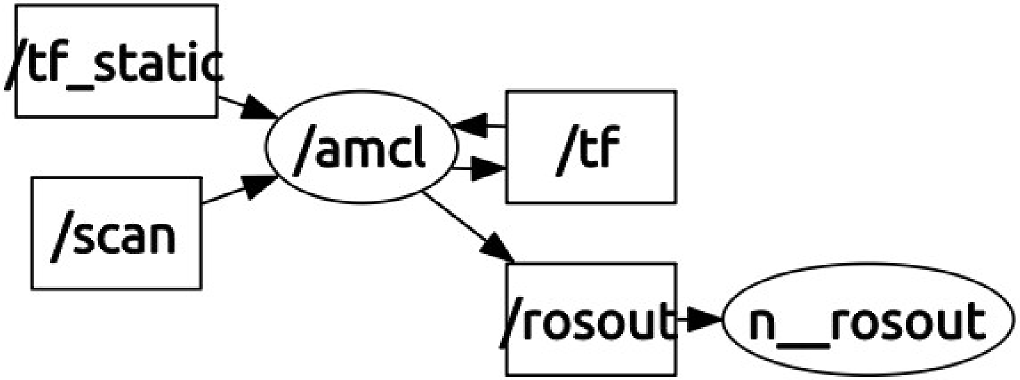
ROSPLAN TUTORIAL

Components in ROS run as nodes. Nodes can be distributed across different computers.

A node might take input and produce output.

Example **node**:

- amcl (odometry)
- - inputs scan data
- - outputs tranform



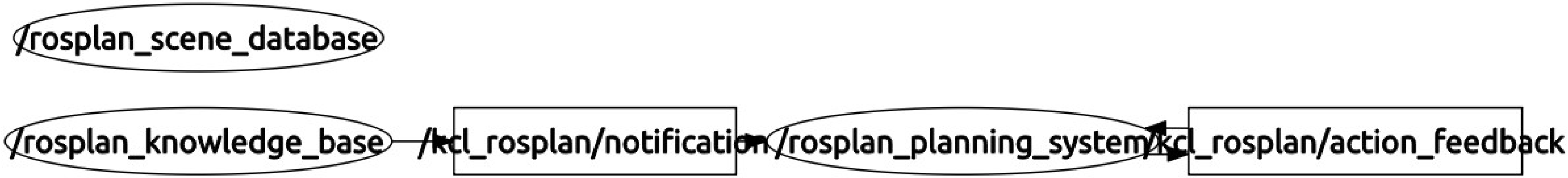
ROSPLAN TUTORIAL

Components in ROS run as nodes. Nodes can be distributed across different computers.

A node might take input and produce output.

Example **node**:

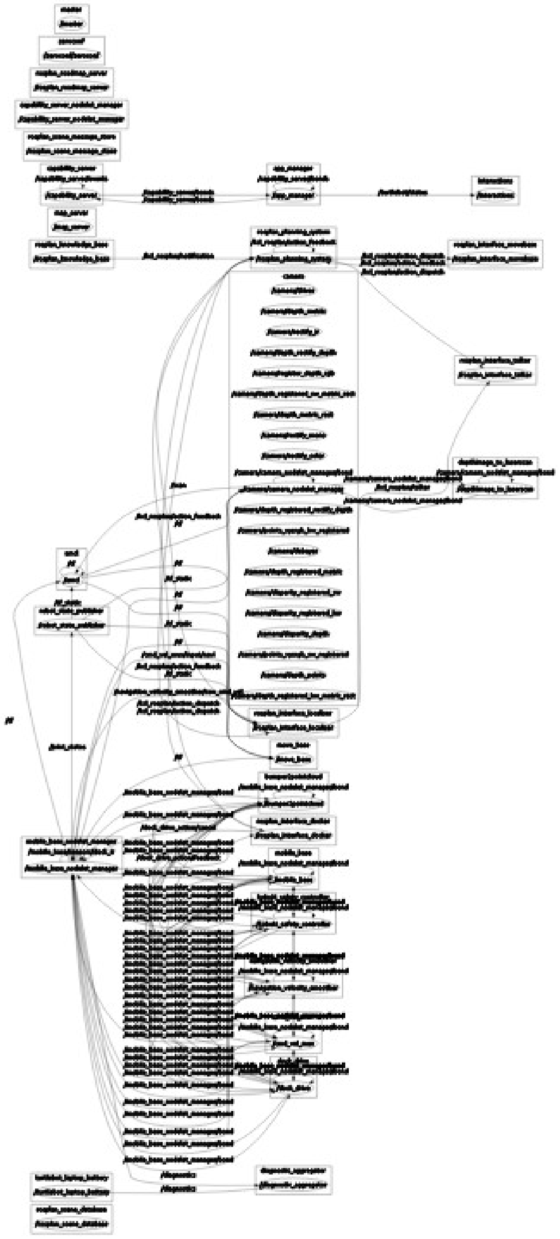
- rosplan (planning and control)
- - inputs problem data
- - outputs actions



ROSPLAN TUTORIAL

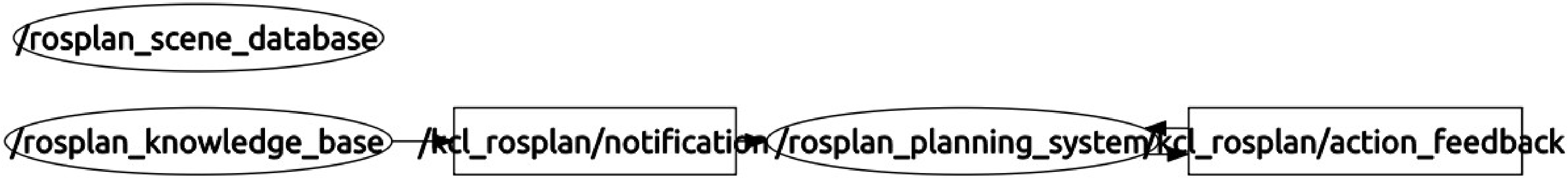
When the whole system is running, there can be a lot of nodes!

Kinect sensor data;
Laser scan;
Odometry;
Transforms;
Velocity commands;
Point cloud data;
Recognised objects;
Octomap data;
PDDL model updates;
PDDL actions;
Action feedback and results;
…

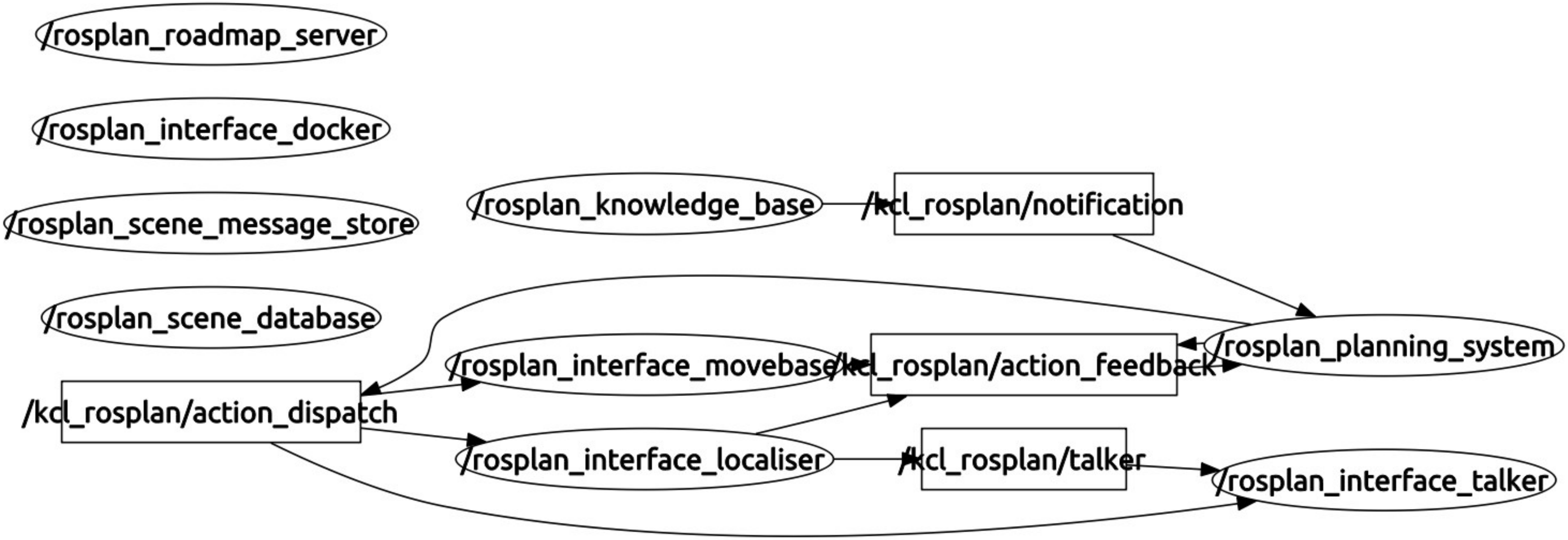


ROSPlan is for controlling a ROS system using a planner.

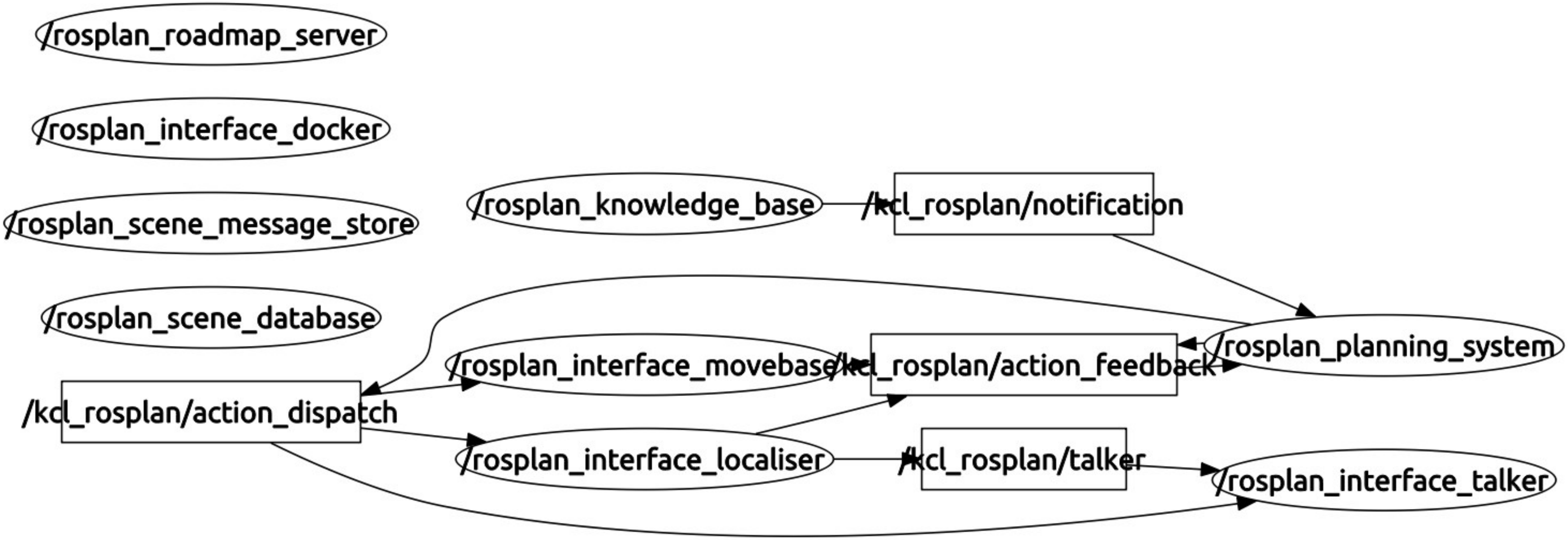
ROSPlan is a set of nodes which control the system.
- ROSPlan
- - input goals, sensor data
- - output actions



ROSPLAN TUTORIAL

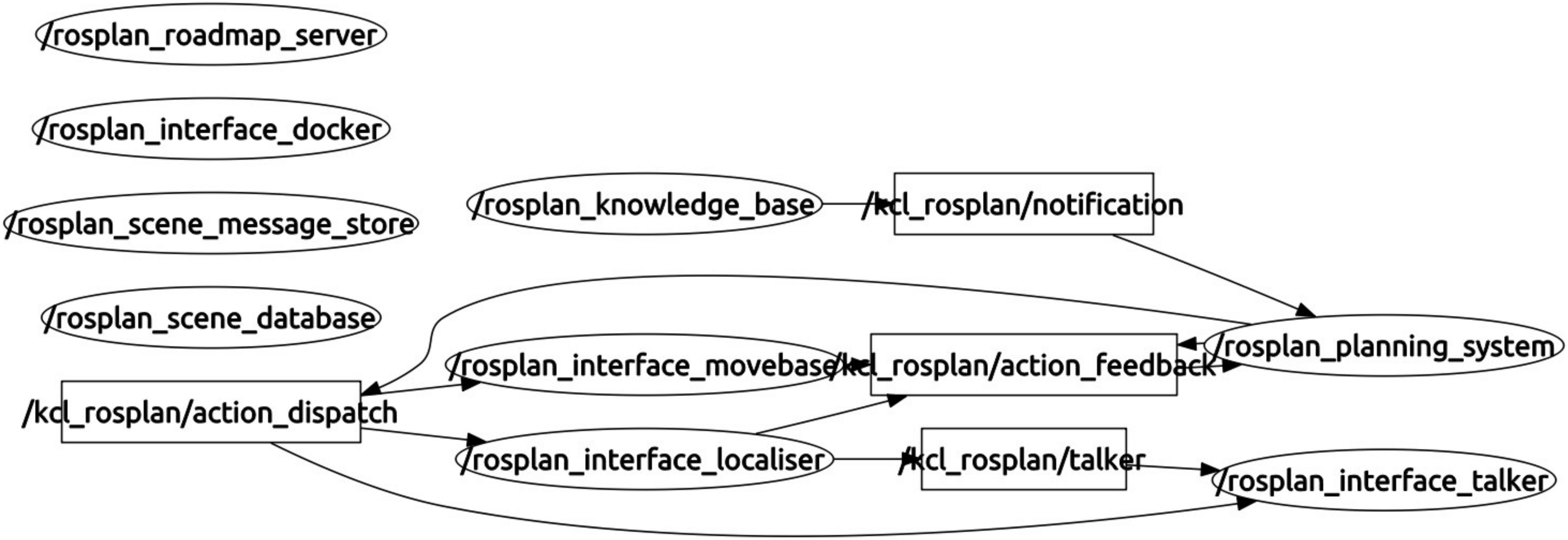

/rosplan_roadmap_server
/rosplan_interface_docker
/rosplan_scene_message_store
/rosplan_scene_database
/kcl_rosplan/action_dispatch
/rosplan_knowledge_base
/kcl_rosplan/notification
/rosplan_interface_movebase
/kcl_rosplan/action_feedback
/rosplan_planning_system
/rosplan_interface_localiser
/kcl_rosplan/talker
/rosplan_interface_talker

ROSPLAN TUTORIAL


/rosplan_roadmap_server
/rosplan_interface_docker
/rosplan_scene_message_store
/rosplan_scene_database
/rosplan_knowledge_base
/kcl_rosplan/notification
/rosplan_planning_system
/rosplan_interface_movebase
/kcl_rosplan/action_feedback
/kcl_rosplan/action_dispatch
/rosplan_interface_localiser
/kcl_rosplan/talker
/rosplan_interface_talker

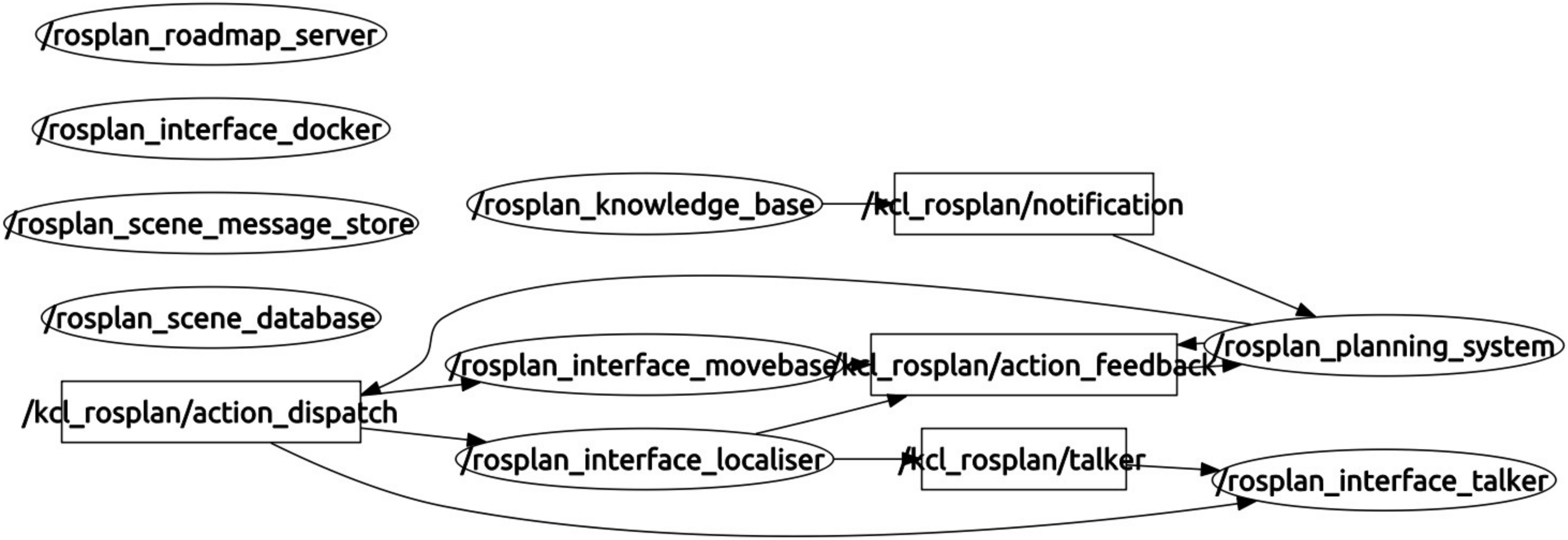
Main Components of ROSPlan:
- PDDL model (rosplan_knowledge_base)
- planning (rosplan_planning_system)
- MongoDB (rosplan_scene_database)
- Plan dispatch (rosplan_planning_system)

ROSPLAN TUTORIAL


/rosplan_roadmap_server
/rosplan_interface_docker
/rosplan_scene_message_store
/rosplan_scene_database
/rosplan_knowledge_base
/kcl_rosplan/notification
/rosplan_planning_system
/rosplan_interface_movebase
/kcl_rosplan/action_feedback
/kcl_rosplan/action_dispatch
/rosplan_interface_localiser
/kcl_rosplan/talker
/rosplan_interface_talker

1.
rosplan_knowledge_base is a node that stores the PDDL domain and problem.
The domain is read from file.
The problem is generated from sensor data.

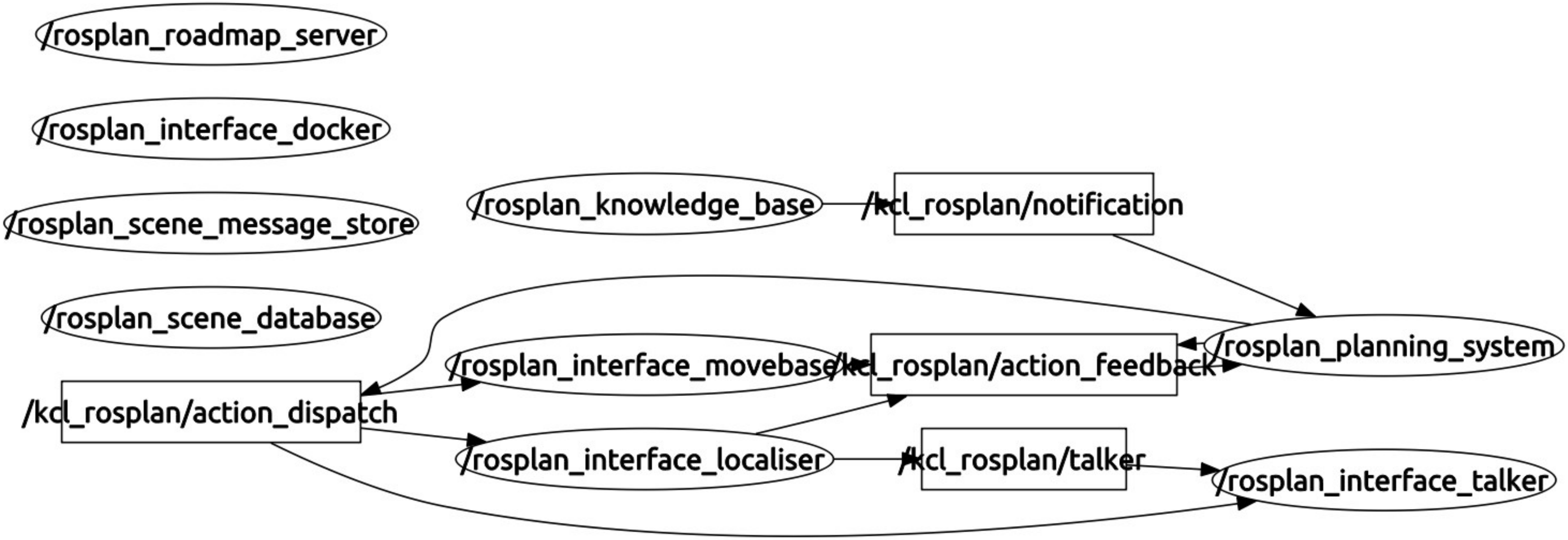
ROSPLAN TUTORIAL


/rosplan_roadmap_server
/rosplan_interface_docker
/rosplan_scene_message_store
/rosplan_scene_database
/rosplan_knowledge_base
/kcl_rosplan/notification
/rosplan_planning_system
/rosplan_interface_movebase
/kcl_rosplan/action_feedback
/kcl_rosplan/action_dispatch
/rosplan_interface_localiser
/kcl_rosplan/talker
/rosplan_interface_talker

2.
rosplan_scene_database is a node that stores real (non-PDDL) data.
For example, coordinates.
waypoint0 (PDDL object) -> [0.1, 2.0, 3.4]

(This mongodb_server **node** is from the ros-indigo-mongodb-store **package.**)

ROSPLAN TUTORIAL



/rosplan_roadmap_server
/rosplan_interface_docker
/rosplan_scene_message_store
/rosplan_scene_database
/rosplan_knowledge_base
/kcl_rosplan/notification
/rosplan_planning_system
/rosplan_interface_movebase
/kcl_rosplan/action_feedback
/kcl_rosplan/action_dispatch
/rosplan_interface_localiser
/kcl_rosplan/talker
/rosplan_interface_talker

3.
rosplan_planning_system is a node that handles top-level control.

- (re)planning
- dispatching plans

ROSPLAN TUTORIAL

3.
rosplan_planning_system is a node that handles top-level control.

- (re)planning
- dispatching plans

The simplest loop is:

```
while( !goal_reached )
{
    request_problem()
    generate_plan()
    goal_reached = dispatch_plan()
}
```

What are the potential problems with this?

ROSPLAN TUTORIAL

Part of the planning system is the **dispatch of plans**.

The dispatcher sends PDDL actions as ROS **messages** to other ROS **nodes** for execution.

A PDDL plan might look like:

```
0.000: goto_waypoint ...
1.345: inspect_location ...
2.192: pickup_object ...
```

How can this be dispatched (robustly)?

ROSPLAN TUTORIAL

ROSPlan stores plans as an Esterel program.

Actions in a plan form nodes in a temporal network.
Actions have:
- causal links
- upper and lower constraints on dispatch time
- start, over all, and end conditions (PDDL conditions)

The dispatch of the plan is an interpretation of the Esterel program.

External conditions are checked using queries to rosplan_knowledge_base.

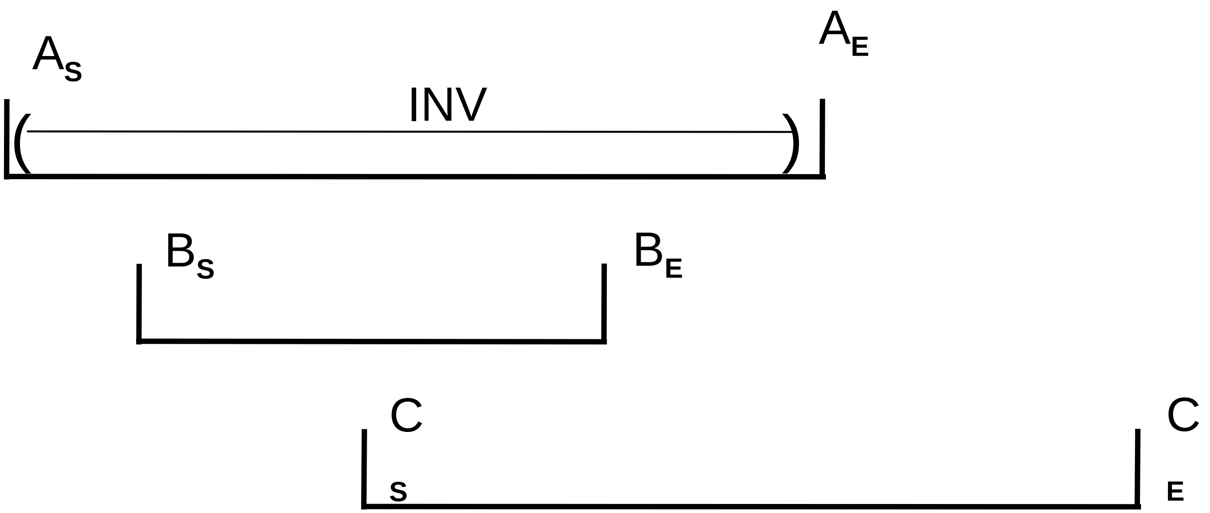
If an action's preconditions were not achieved in time to dispatch it during its time window, then the plan has failed.

What about when an action fails to execute?

ROSPLAN TUTORIAL

External conditions are checked using queries to rosplan_knowledge_base.

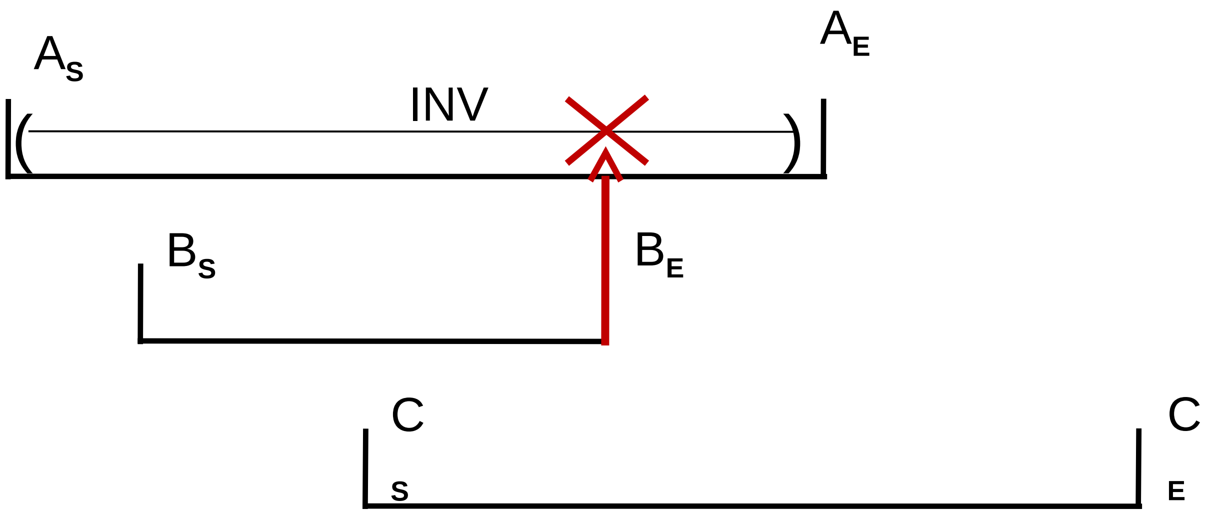
Example:



/rosplan_planning_system

External conditions are checked using queries to rosplan_knowledge_base.

Example:

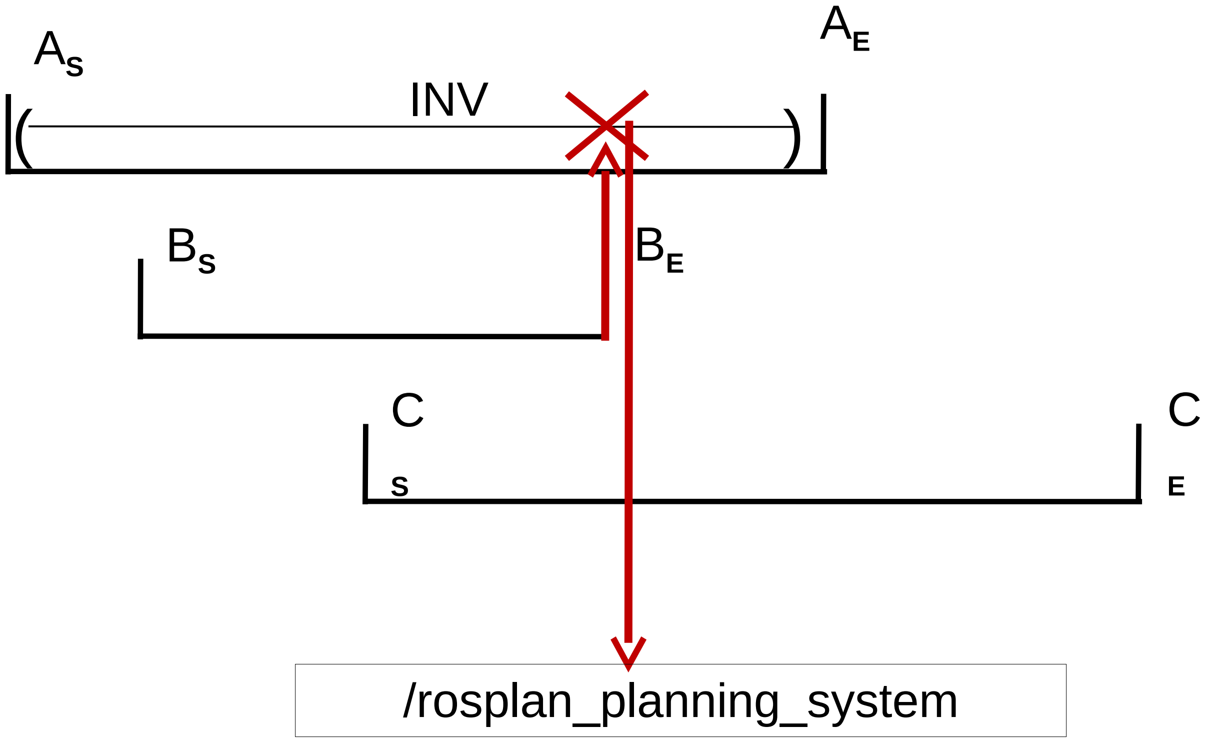


/rosplan_planning_system

ROSPLAN TUTORIAL

External conditions are checked using queries to rosplan_knowledge_base.

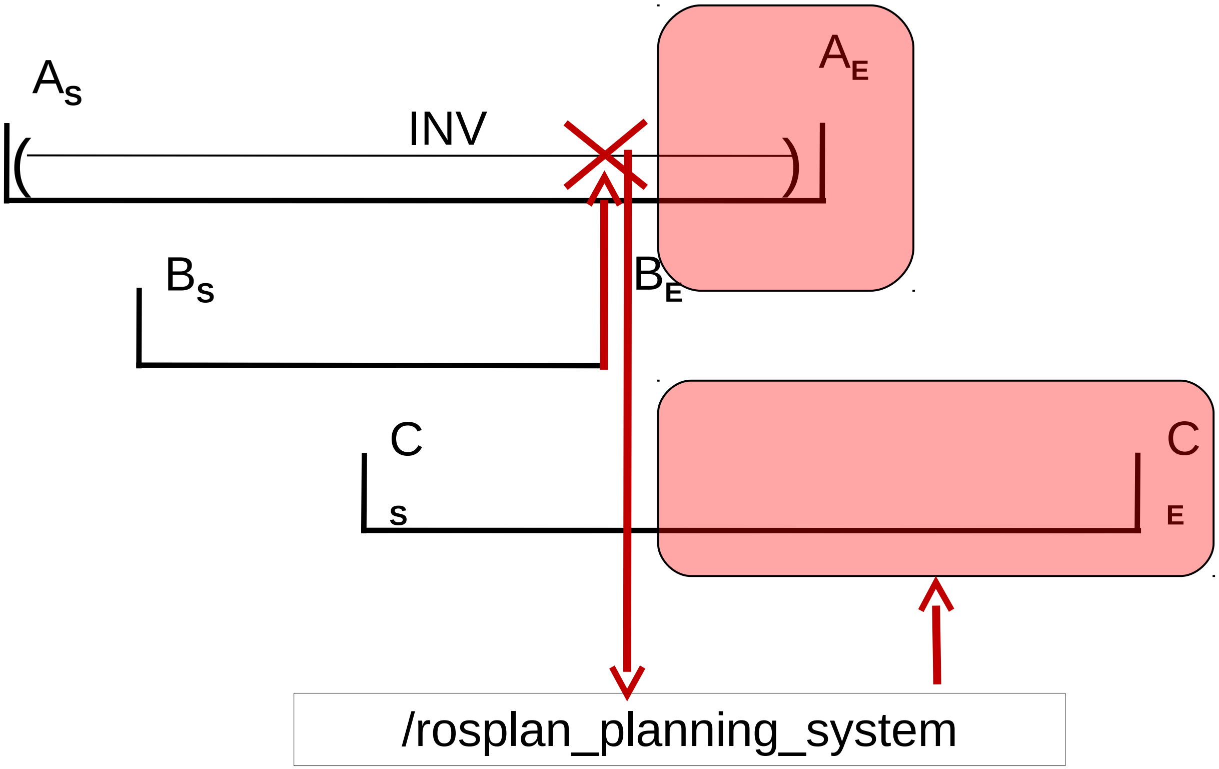
Example:



ROSPLAN TUTORIAL

External conditions are checked using queries to rosplan_knowledge_base.

Example:



ROSPLAN TUTORIAL

Back to that top-level loop:

```
while( !goal_reached )
{
    request_problem()
    generate_plan()
    goal_reached = dispatch_plan()
}
```

This won't always work in the real world.

- errors in the PDDL model
- dead-ends
- new discoveries
- the PDDL domain is just not good enough

ROSPLAN TUTORIAL

Back to that top-level loop:

```
while( !goal_reached )
{
    request_problem()
    generate_plan()
    goal_reached = dispatch_plan()
}
```

Fortunately this is not a fixed behaviour.

The rosplan_planning_system node is activated by a ROS message with:
- conditions
- goals
- timeout constraints

... just like any other action.

ROSPLAN TUTORIAL

Now we will look directly at a ROS system. (Turtlebot 2)

TurtleBot™

ROSPLAN TUTORIAL

Now we will look directly at a ROS system. (Turtlebot 2)

Setting up ROS to run on multiple computers (and robots)

ROSPLAN TUTORIAL

Now we will look directly at a ROS system. (Turtlebot 2)

Setting up ROS to run on multiple computers (and robots)

How to launch it?

- run a single node.
- launch files for groups of nodes and parameters.

ROSPLAN TUTORIAL

Now we will look directly at a ROS system. (Turtlebot 2)

Setting up ROS to run on multiple computers (and robots)

How to launch it?
- run a single node.
- launch files for groups of nodes and parameters.

From the command line:
- view nodes.
- view topics and messages.

ROSPLAN TUTORIAL

Now we will look directly at a ROS system. (Turtlebot 2)

Setting up ROS to run on multiple computers (and robots)

How to launch it?
- run a single node.
- launch files for groups of nodes and parameters.

From the command line:
- view nodes.
- view topics and messages.

Graphical User interfaces
- rviz
- rqt

ROSPLAN TUTORIAL

Now we will look directly at a ROS system. (Turtlebot 2)

Setting up ROS to run on multiple computers (and robots)

How to launch it?
- run a single node.
- launch files for groups of nodes and parameters.

From the command line:
- view nodes.
- view topics and messages.

Graphical User interfaces
- rviz
- rqt

ROSPlan
- the ROSPlan rqt plugin.
- dispatching single PDDL actions to command the robot.
- sending PDDL goals and letting the robot do what it wants.

ROSPLAN TUTORIAL

Now we will look directly at a ROS system. (Turtlebot 2)

"ROSPlan: Planning in the Robot Operating System"
Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS-15). June 2015.

https://github.com/KCL-Planning/ROSPlan/wiki