Presentation 12/29/2021

Path Planning

Earlier we've seen how to create a Map and Locoalization for SLAM purposes

Today we'll focus on the path planning part

The move_base pkg

move_base node: moves robot from its current position to goal position

It takes a goal pose w/ msg type: **geometry_msgs/PoseStamped**

When node receives goal pose, it links to components (global, planner, local planner, costmaps and recovery behaviors) and outputs a velocity command

It sends **geometry_msgs/Twist** to **/cmd_vel** Topic

Navigation Stack Setup

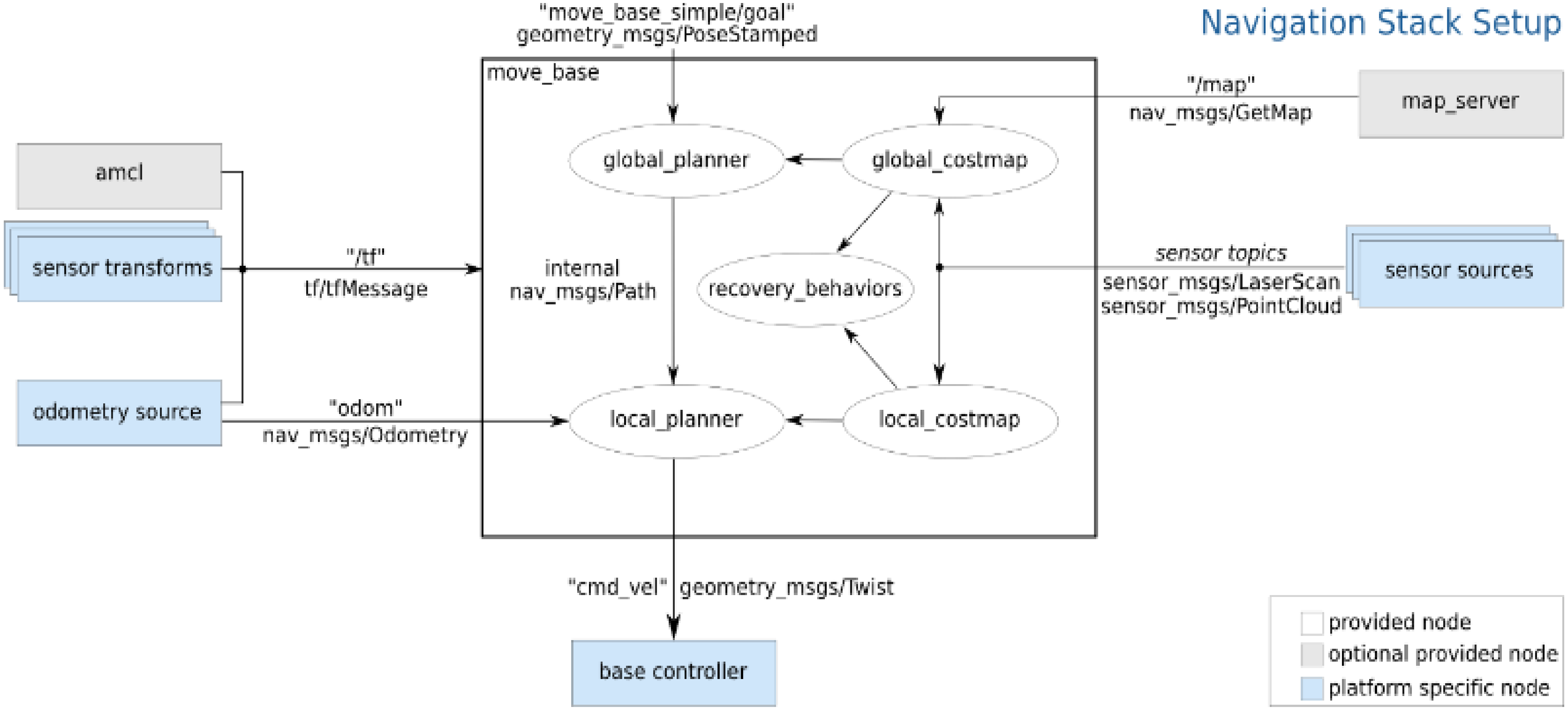

Fig.1.1 - Navigation Stack Setup (figure from ROS Wiki)

Costmaps

DEF: Map representing cost of traversing different areas

Represents either a free space or places where robot would be in collision {0,255}

2 types: **global costmap** (static map) & **local costmap** (sensor's readings)

Costmap params defined in 3 files:

1) global_costmap_params.yaml
2) local_costmap_params.yaml
3) common_costmap_params.yaml

Planners

2 types: **global planner** & **local planner**

Gp: NavfnROS, Carrot Planner, GlobalPlanner
Lp: DWA (Dynamic Window Approach)

gp in charge of calculating safe path to arrive at goal pose (static map)

Each planner has its own params, which modify the way planner behaves

Note: **gp** won't calculate dynamic obstacles (bc not in static map)

Send a goal pose

1) RViz: 2D Nav Goal Tool

2) rostopic pub /move_base/goal move_base_msgs/MoveBaseActionGoal {}

3) Create Action Server :

Use **actionlib** pkg to create servers executing long-running goals
Need to define goal, result, feedback msgs (in **action** dir --> **.action** file)
See **send_goal_client.py** file

Here, action server: move_base

Next: Path Planning part 2: **Obstacle Avoidance**