ROS

Robot Operating System

Course of

Robot Programming with ROS 2

Day 3

3. Nav2

ikerlan

Universidad
Rey Juan Carlos

Intelligent
Robotics
Lab

Description

NAV2

ROS2 navigation system designed to be:

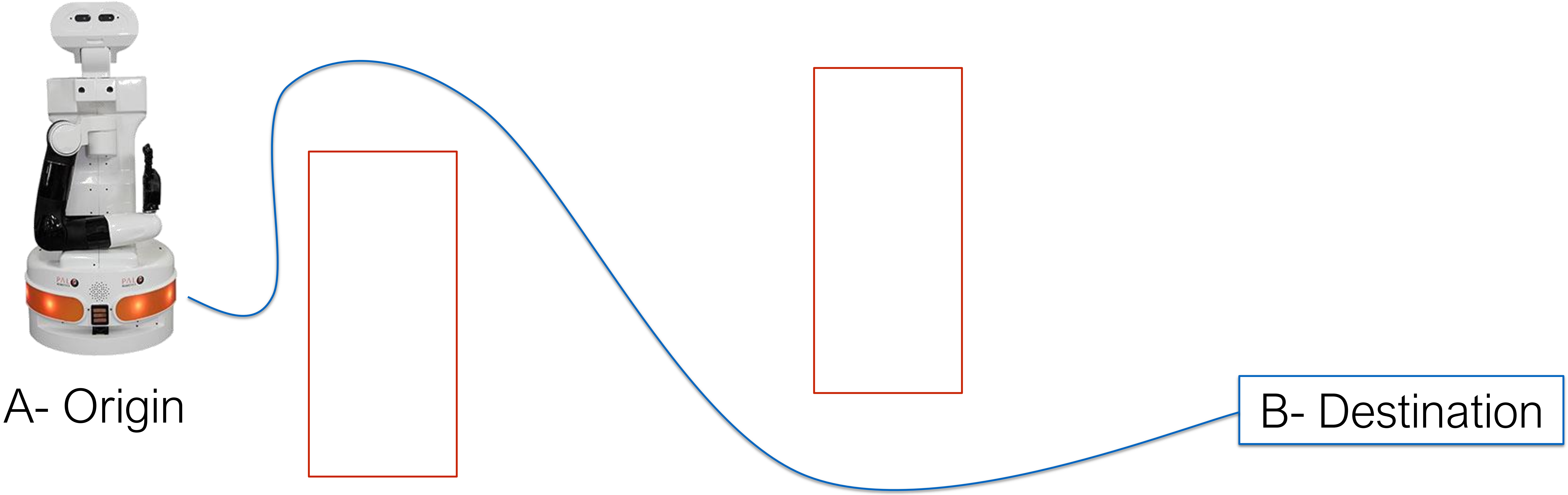
- Modular
- Configurable
- Scalable

Description

It aspires to be the modst widely used navigation sofware, it supports major robot types:

- Holonomic, differential-drive, legged and Ackermann

Main Goal



- Move the robot from point A to point B in a safe way

Introduction

NAV2

- The project seeks to find a safe way to have a mobile robot move to complete complex tasks through many types of environments and classes of robot kinematics.
- Not only can it move from Point A to Point B, but it can have intermediary poses, and represent other types of tasks like object following and more.
- Nav2 is a production-grade and high-quality navigation framework trusted by 50+ companies worldwide.

Functionalities

- It provides perception, planning, control, localization, visualization, and much more to build highly reliable autonomous systems. This will complete environmental modeling from sensor data, dynamic path planning, compute velocities for motors, avoid obstacles, represent semantic regions and objects, and structure higher-level robot behaviors.

Behavior Trees

- Nav2 uses behavior trees to create customized and intelligent navigation behavior via orchestrating many independent modular servers.
- A task server can be used to compute a path, control effort, recovery, or any other navigation related task.
- These separate servers communicate with the behavior tree (BT) over a ROS interface such as an action server or service.
- A robot may utilize potentially many different behavior trees to allow a robot to perform many types of unique tasks.

Tools

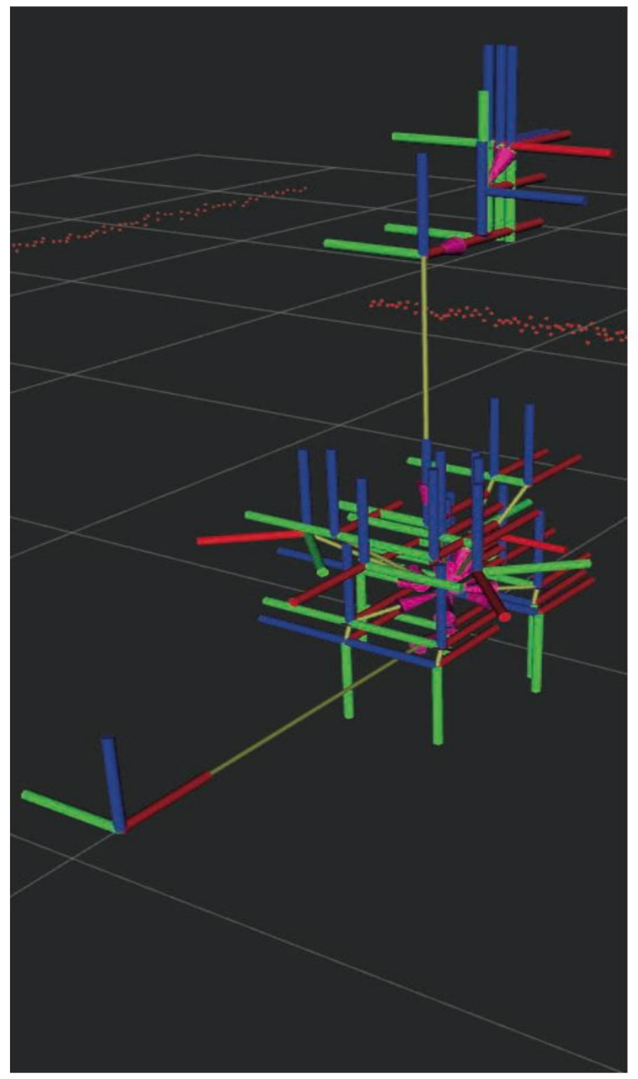
- Load, serve, and store maps (Map Server)
- Localize the robot on the map (AMCL)
- Plan a path from A to B around obstacles (Nav2 Planner)
- Control the robot as it follows the path (Nav2 Controller)
- Smooth path plans to be more continuous and feasible (Nav2 Smoother)
- Convert sensor data into a costmap representation of the world (Nav2 Costmap 2D)
- Build complicated robot behaviors using behavior trees (Nav2 Behavior Trees and BT Navigator)

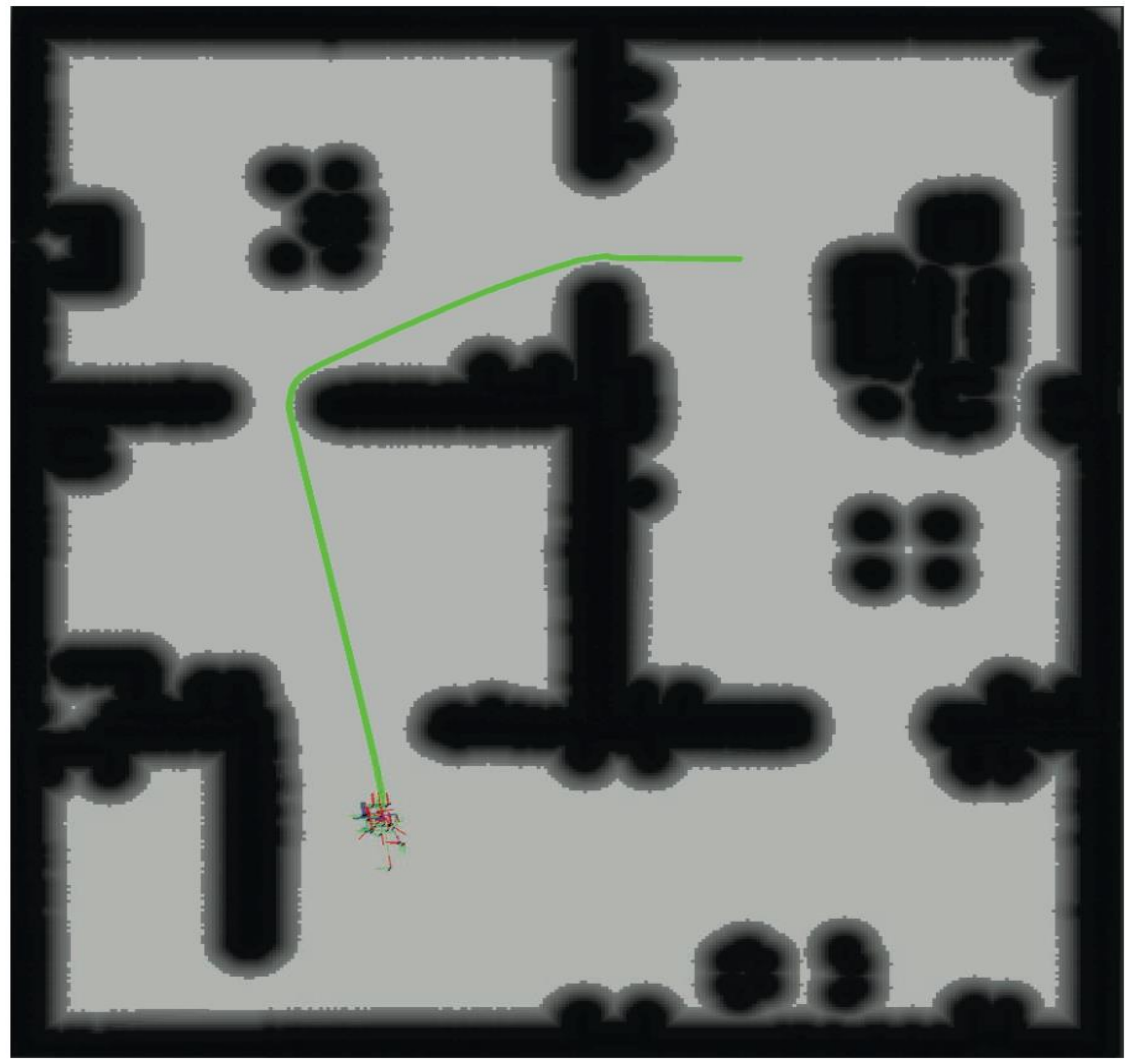
Tools

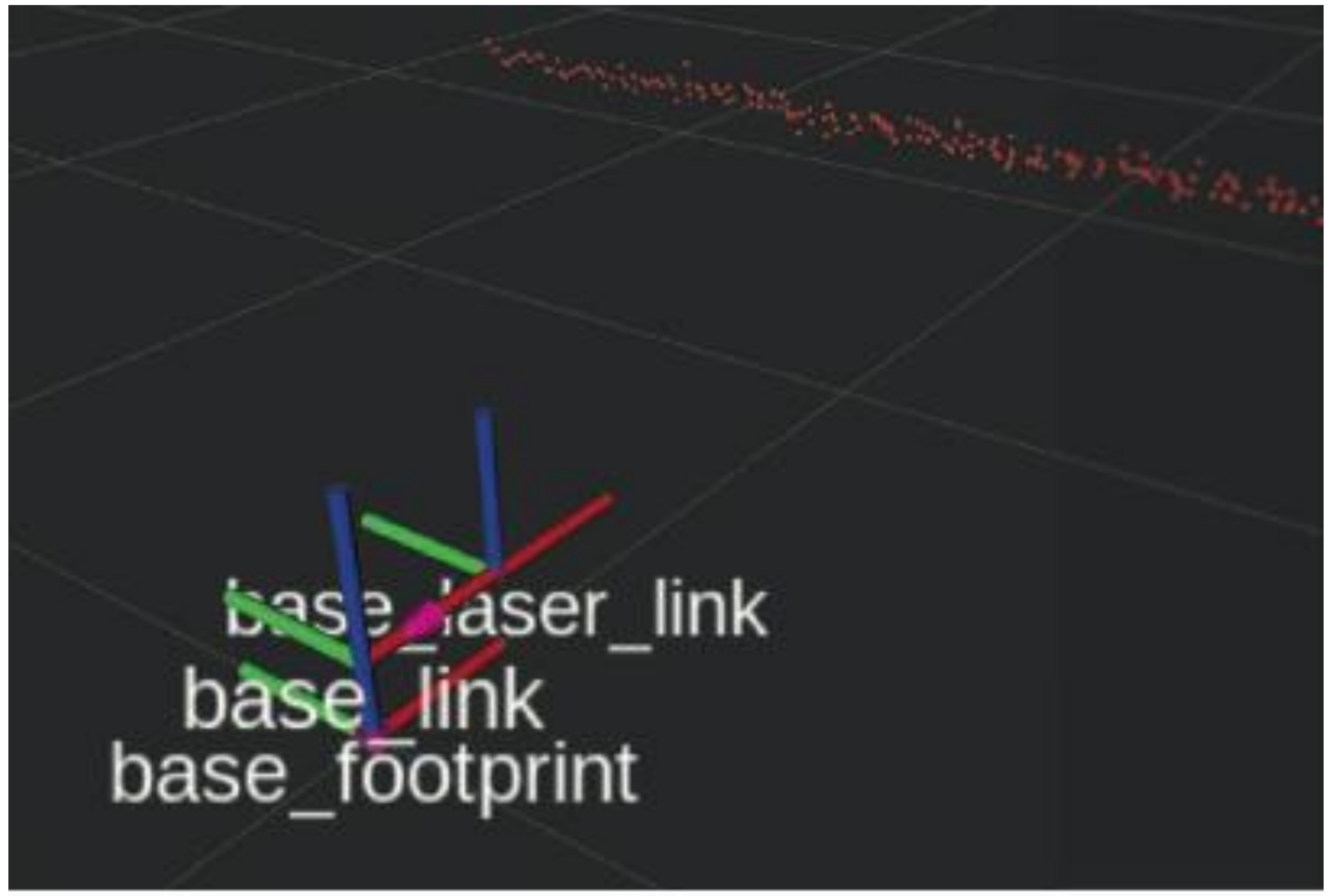
- Compute recovery behaviors in case of failure (Nav2 Recoveries)
- Follow sequential waypoints (Nav2 Waypoint Follower)
- Manage the lifecycle and watchdog for the servers (Nav2 Lifecycle Manager)
- Plugins to enable your own custom algorithms and behaviors (Nav2 Core)
- Monitor raw sensor data for imminent collision or dangerous situation (Collision Monitor)
- Python3 API to interact with Nav2 in a pythonic manner (Simple Commander)
- A smoother on output velocities to guarantee dynamic feasibility of commands (Velocity Smoother)

Inputs

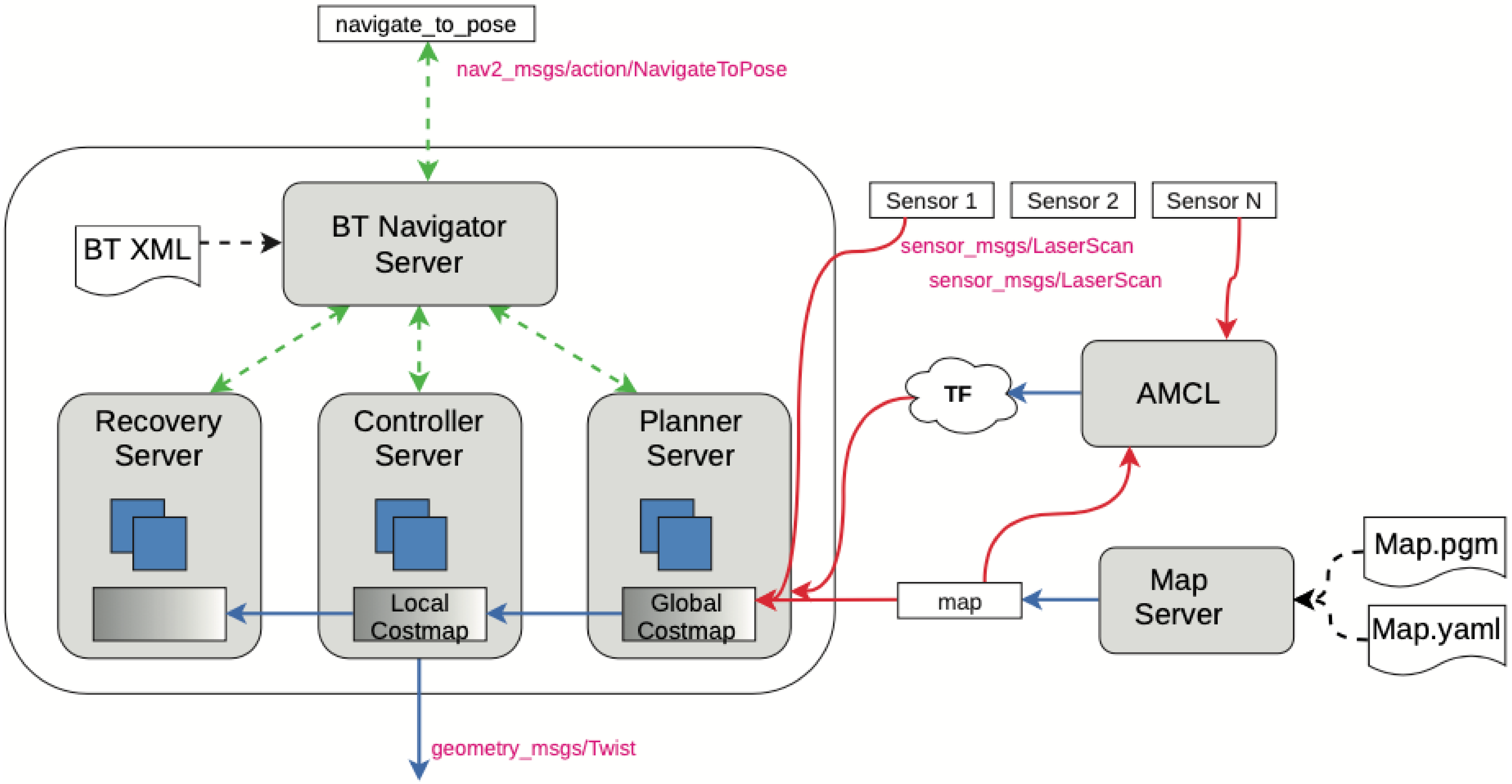
- TF transformations
- Map
- Relevant sensor data sources
- Navigation logic codede as a Behavior Tree XML file coded





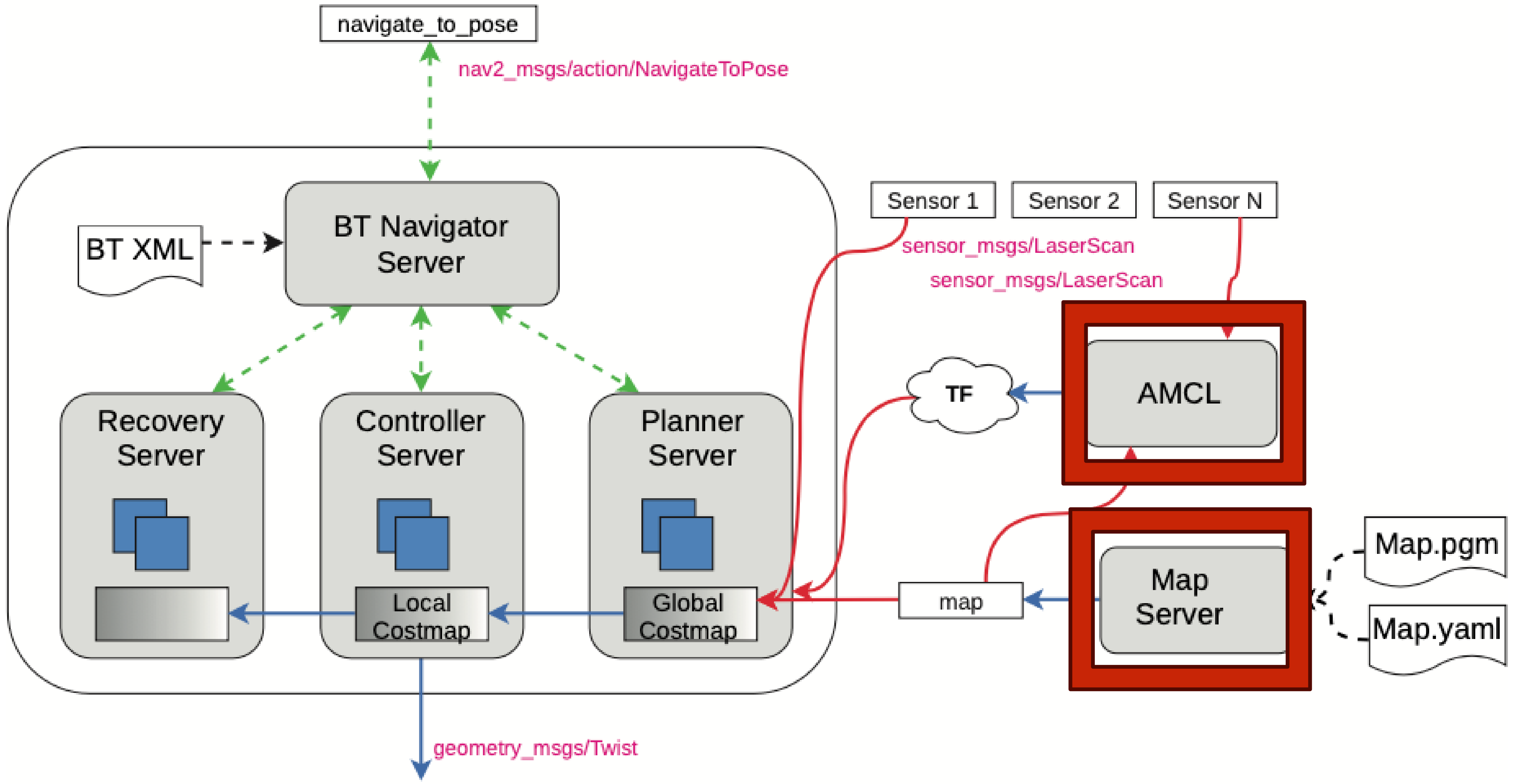


Architecture



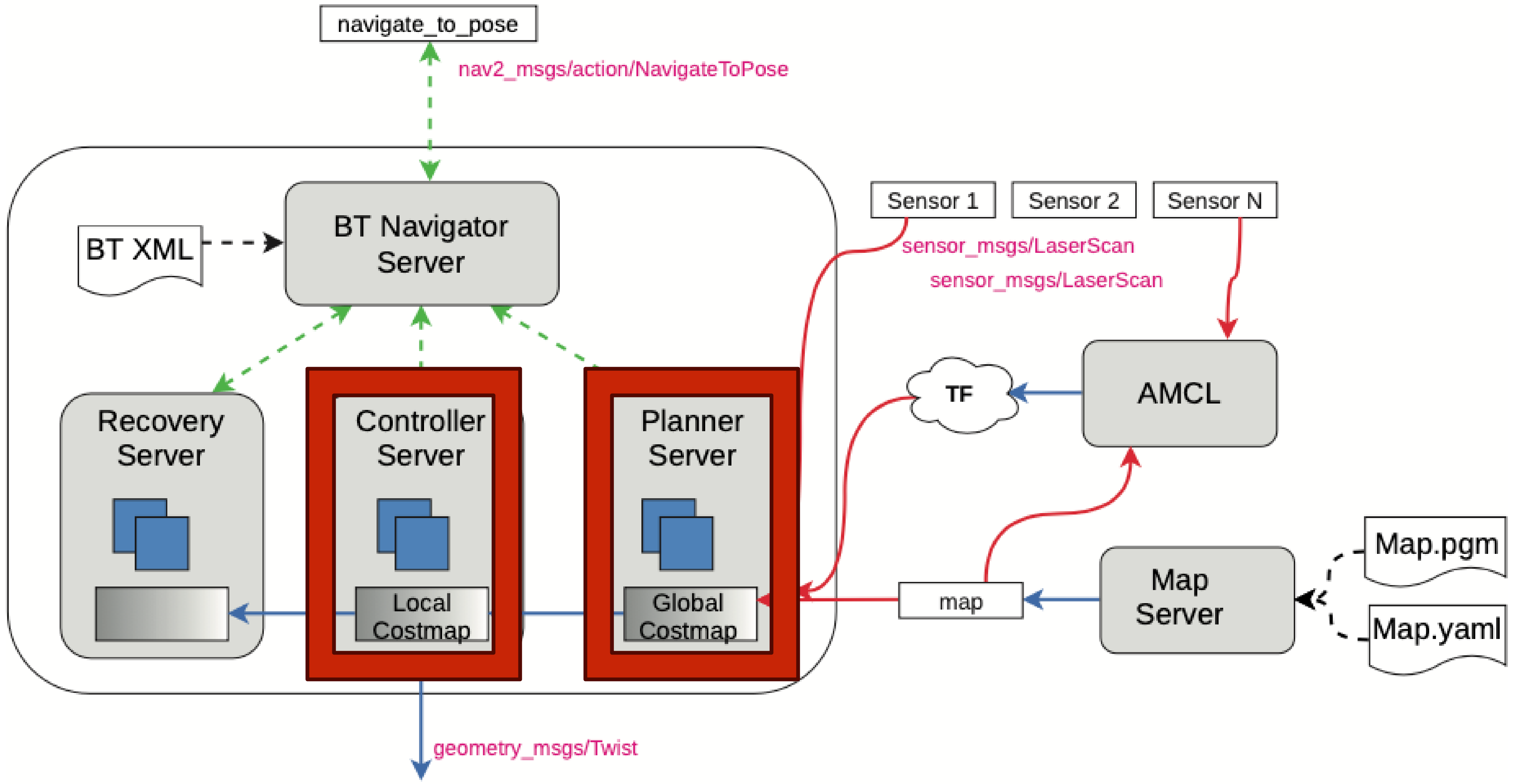
Architecture

- Map Server: This component reads a map from two files and publishes it as a nav_msgs/msg/OccupancyGrid,
- AMCL: This component implements a localization algorithm based on Adaptive Monte-Carlo (AMCL). It uses sensory information, primarily distance readings from a laser and the map, to calculate the robot's position.



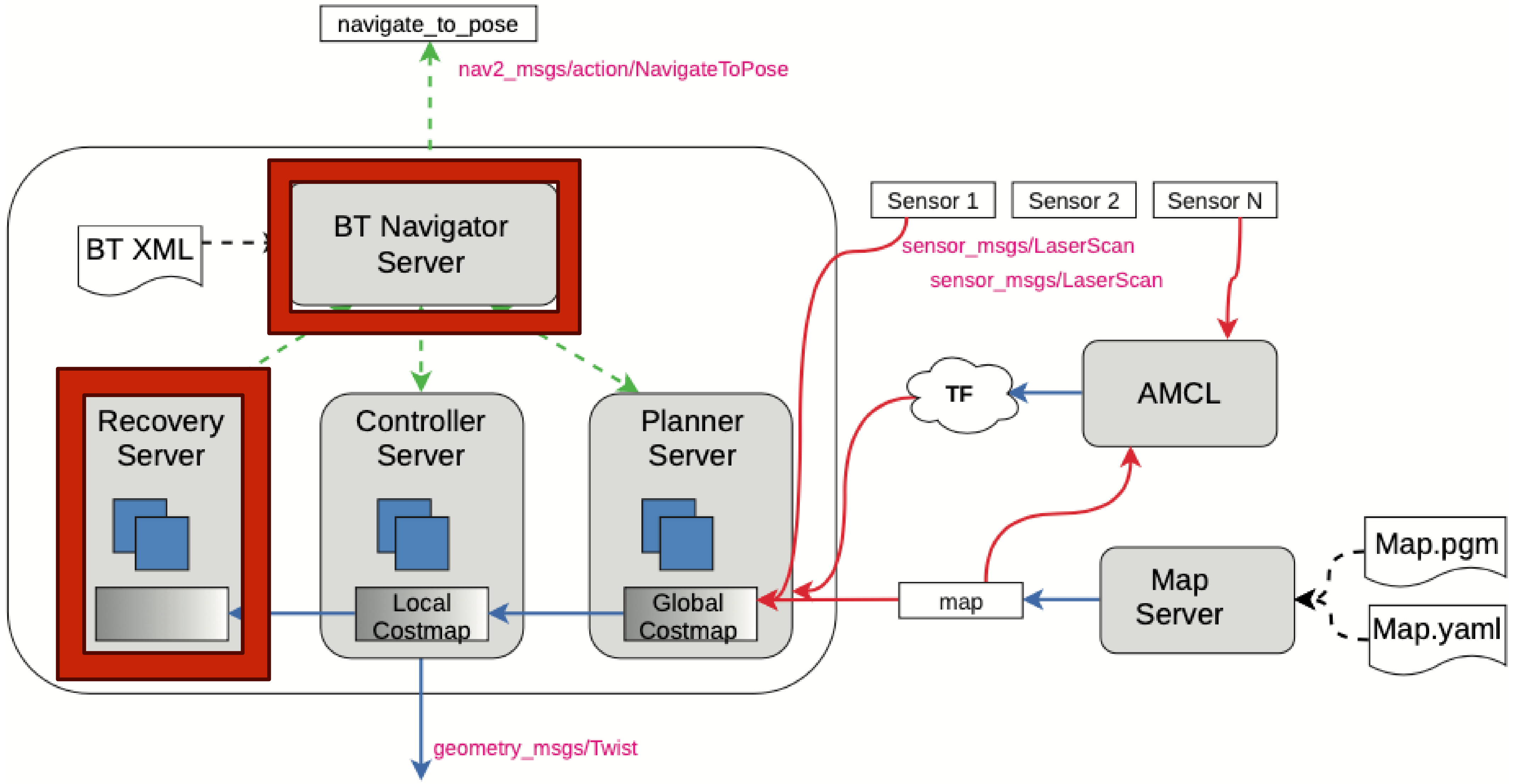
Architecture

- Planner Server: The function of this component is to calculate a route from the origin to the destination
- Controller Server:This component receives the route calculated by the Planner Server and publishes the speeds sent to the robot base



Architecture

- Recovery Server: This component has several helpful recovery strategies.
- BT Navigator Server: This is the component that orchestrates the rest of the navigation components.



Nav2 in practice: Marathon2

- The robots successfully navigated over 60 kilometres in under 23 hours in a Dynamic campus environment
- Neither the robot suffered a collision o dangerous situation requiring an emergency stop.