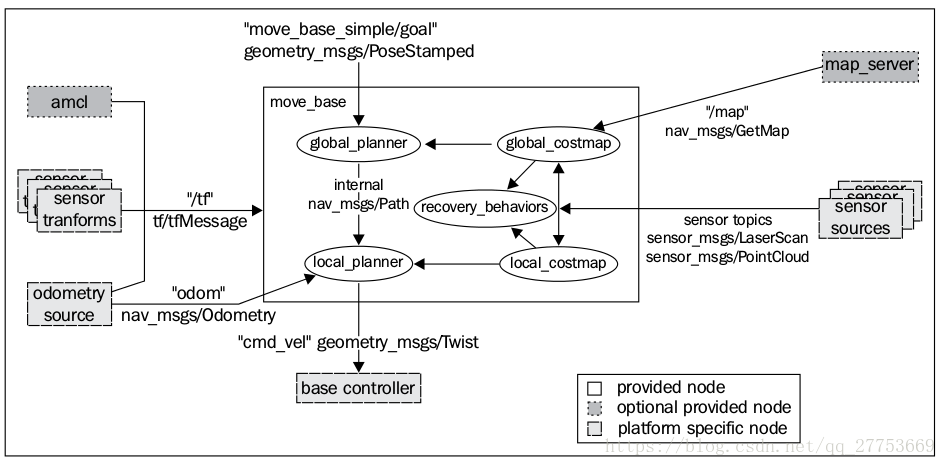
**Navigation stack总体结构**



这是2D的导航功能包集，通过接收里程计数据、tf坐标变换树以及传感器数据，为移动机器人输出目标位置以及安全速度。概念层面上讲，导航功能包集是相当简单的。 它从里程计和传感器数据流获取信息，并将速度命令发送给移动基站（比如你的机器人）。但是，想要在任意机器人上使用导航功能包集可能有点复杂。使用导航功能包集的先决条件是，机器人必须运行ROS，有一个tf变换树，使用正确的ROS Message types发布传感器数据。而且，我们需要在高层为一个具有一定形状和动力学特点的机器人配置导航功能包集。

1.acml：是一个针对在二维移动的机器人的基于概率定位系统。它实现了自适应蒙特卡罗滤波的定位方法，并使用粒子滤波器去跟踪在已知地图中机器人的位置。

2.base\_local\_planner：这个包提供了Trajectory Rollout以及Dynamic Window两种在2D平面局部导航的方法。通过提供一个跟随的规划路径和一个代价地图，控制器生成速度指令并发送至机器人。它支持全向和非全向机器人，以及任何可以用多边形或圆描述的机器人轮廓，并且可以在launch文件中设置这些参数。这个包已经被封装好，可以通过nav\_core包的BaseLocolPlanner接口来调用。

3.carrot\_planner：这个规划器是一个简单的全局规划器，可以通过nav\_core::BaseGlobalPlanner 来进行调用，并且被move\_base节点用作一个全局规划的插件。这个规划器从用户处采集到一个目标点，之后检查用户指定的目标点是否是障碍物，如果是的话沿着robot与目标点构成的向量向后退，直到找到一点没有障碍物位置。之后它会将此目标点作为目标发送给局部规划器和控制器。这个规划器允许机器人尽可能到达离用户指定的目标点最近的位置。

4.clear\_costmap\_recovery：为导航包提供了一种自救行为，试图通过将代价地图还原成已知区域外的静态地图从而清除出空间。

5.costmap\_2d：通过激光或点云的数据，投影到2D平面上，创建代价地图，并可以设置膨胀半径。

6.dwa\_local\_planner：局部规划器，提供动态窗口方法（ Dynamic Window Approach）在平面上局部导航。与base\_local\_planner类似。

7.fake\_localization：提供了一个简单节点fake\_localization node，可以代替一个定位系统，并提供了acml包的ROS API的子集。由于较低的计算量，这个节点非常频繁的用于在仿真环境中提供完美的定位。这个节点将里程计数据转换为位置、粒子云，并以acml发布的数据格式发布。

8.global\_planner：全局路径规划节点。

9.map\_server：将代价地图作为ROS Service发布，提供了map\_saver节点，可以通过命令行存储地图。

10.move\_base：提供了action动作的实现（actionlib包），即给定一个世界系下的目标位置，机器人会试图移动到该位置。另外，move\_base节点中包含了两个代价地图（全局、局部），以及一个全局规划器和一个局部规划器，以便实现导航任务。

11.move\_base\_msgs： 通过MoveBase.action文件定义产生的消息文件，用于actionlib与move\_base的通信。

12.move\_slow\_and\_clear：为robot提供一种自救行为，即清除代价地图的信息并限制机器人速度，但这不绝对安全，robot可能会撞到某些障碍。但这是唯一一种可以与允许最大速度动态设置的局部规划器兼容的自救行为。

13.nav\_core：为robot实现导航任务提供了通用的接口，包括BaseGlobalPlanner，BaseLocalPlanner,， RecoveryBehavior interfaces等，这有利于方便的更新规划器或者自救行为的版本。

14.navfn：全局规划器，提供了一个快速插值的函数，可以在起始点到目标点之间快速插值，并找到代价最小的一条路径。

15.robot\_pose\_ekf：这个包用于估计robot的三维位置，利用扩展卡尔曼滤波的方法，建立了一个六维模型，联合了轮子里程计、IMU、视觉里程计的数据。

16.rotate\_recovery：提供了一种自救行为，通过旋转360度来清除空间。

17.voxel\_grid：提供一个有效的三维体素网格的实现。

使用Navigation导航包的一个整体框架，其核心是move\_base节点，包含了global\_planner、local\_planner、global\_costmap、local\_costmap、recovery\_behaviors五个模块，move\_base节点订阅tf（坐标系转换）、odom（里程计数据）、map（地图）、sensor datas（激光数据或点云）以及goal等话题，之后发布cmd\_vel话题。

tf：利用ROS的tf工具发布坐标系之间的转换关系，包括：/map->/odom，/odom->/base\_link，/base\_link->/sensor等

odom：导航包需要用到里程计的数据，故需将其用tf工具和nav\_msgs/Odometry消息发布出来

map：在导航前，最好可以提供一张全局的地图，因此需要我们提前创建（但这不是必须的）

sensor data：用于避障，建图等等，可以是激光雷达的平面数据或者是点云数据（sensor\_msgs/LaserScan or sensor\_msgs/PointCloud ）

goal：目标在全局地图中的坐标，用geometry\_msgs/PoseStamped消息格式发布