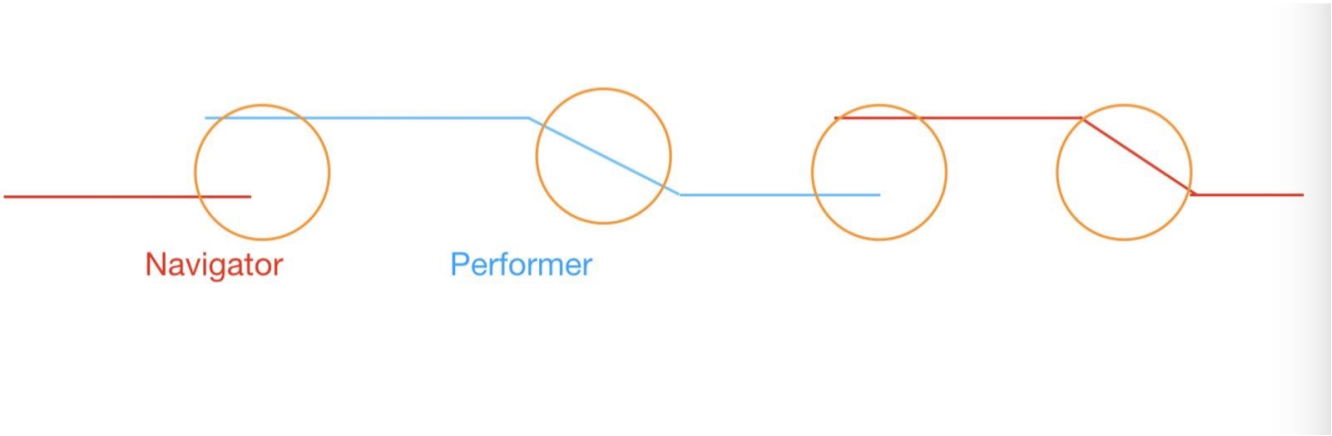
4. navigator和performer的平滑过渡方案

0. 版本

版本	作者	审阅人	更新日期
v 1.0	焦健	代津，王隆潭，刘会良，王好，江瑜	2019-11-04

4.1 切换整体逻辑方案



1）navigator结束任务，切换到performer。

navigator内部逻辑：首先NTT快要结束轨迹任务（某个边界条件），并且后面有伺服任务的时候（该任务的type不为0），降低速度优先级，向上层层汇报到NSA，这中间LPP和NSA都可以将自身状态设置为完成轨迹状态，NSA分配动作任务给Performer。navigator同时订阅速度指令stamp velocity，发现Performer发送stamp velocity为当前时段的速度以后，停止发送速度。该过程navigator一直发送低优先级的速度。

performer逻辑：Performer接受到动作任务以后，同时订阅着NTT的速度输出（根据owner判断），然后把当前的速度输出改为和NTT当前输出一样的值，发出高优先级的速度进行伺服，同时监听navigator的速度输出，如果没有收到，就逐步降低优先级，直到优先级达到最低，衔接过程完成。

2）performer到navigator

performer逻辑：快要完成动作时（某个边界条件），向NSA汇报完成动作任务（finish pallet servo），如果NSA本身有cached Task，那么NSA返回为true，否则返回为false。如果为false，那么performer默认切换失败，继续伺服完余下动作，不做和navigator的切换；如果为true，则继续执行直到收到来自navigator的速度输出，停止发送速度。该过程performer一直发送低优先级的速度。

navigator逻辑：如果切换的finish pallet servo返回为true，那么performer切换成功，NSA向下发送LPP一段轨迹任务，LPP接收到以后，规划当前的速度为performer目前的速度，生成轨迹向下发送给NTT做轨迹跟随。NTT同时监听performer的速度输出，如果没有收到，就逐步降低优先级，直到优先级达到最低，衔接过程完成。

以上两种方式chassis都是同样的处理：

3) chassis处理逻辑

chassis逻辑：chassis一直收听速度指令，然后根据下面的伪代码来操作：

```
if (cmd.owner == current _owner) {execute(cmd.twist); current_priority = cmd.priority}
if (cmd.priority > current_priority) {execute(cmd.twist); current_owner = cmd.owner,}
```