routing模块分析

**1. 功能：**

基于请求，输出两点之间导航信息。

**2. 输入：**

（1） 用户的起始和终止位置。/apollo/routing\_request [pb\_msgs/RoutingRequest]。

（2） HD map

（3） /apollo/monitor [pb\_msgs/MonitorMessage]

**3. 输出：**

（1） 请求的响应：/apollo/routing\_response [pb\_msgs/RoutingResponse]

（2） /apollo/monitor [pb\_msgs/MonitorMessage]

话题/apollo/routing\_response包含了最终的路由信息。

**4. 节点I/O：**



**5. routing 源码：**

（1） routing.cc和routing.h

1） Name()函数：返回节点名字。

2） Routing()：构造函数，初始化monitor\_。

3） Init()：节点初始化函数。

a） 获取路由地图文件routing\_map\_file。

b） 创建导航器navigator\_ptr\_。

c） 读取导航配置文件：modules/routing/conf/routing.pb.txt到 routing\_conf\_。

d） 根据Adapter配置文件：modules/routing/conf/adapter.conf，创建 nodehandle和响应topics。

e） 设置路由请求话题的回调函数OnRouting\_Request。这里为啥不直接使 用rosservice？可能不是直接控制，而是将提供数据出去，共后续模块， 也就是planning使用，所以使用topic。

4） OnRouting\_Request()：处理请求。

a） 输入路由请求。

b） 执行SearchRoute()函数，根据routing\_request，寻找路由，并设置 routing\_response。

c） 发布routing\_response。

（2） navigation.cc和navigation.h。

1） SearchRoute()函数：

a） 显示请求命令的信息。

b） 检查导航器是否ready。构造函数时打开。

c） 通过初始化函数，将请求中的点转化为graph中的格式，并存储在 way\_nodes和way\_s.

d） 通过a star 算法由 way\_nodes和 way\_s计算出路由点result\_nodes。

e） 将路由点result\_nodes合并到response结构。

2） SearchRouteByStrategy()函数：

主要是利用A star算法寻找一个全局路径。

3） GetWayNodes()函数：

将request转化为拓扑图graph\_中的点。