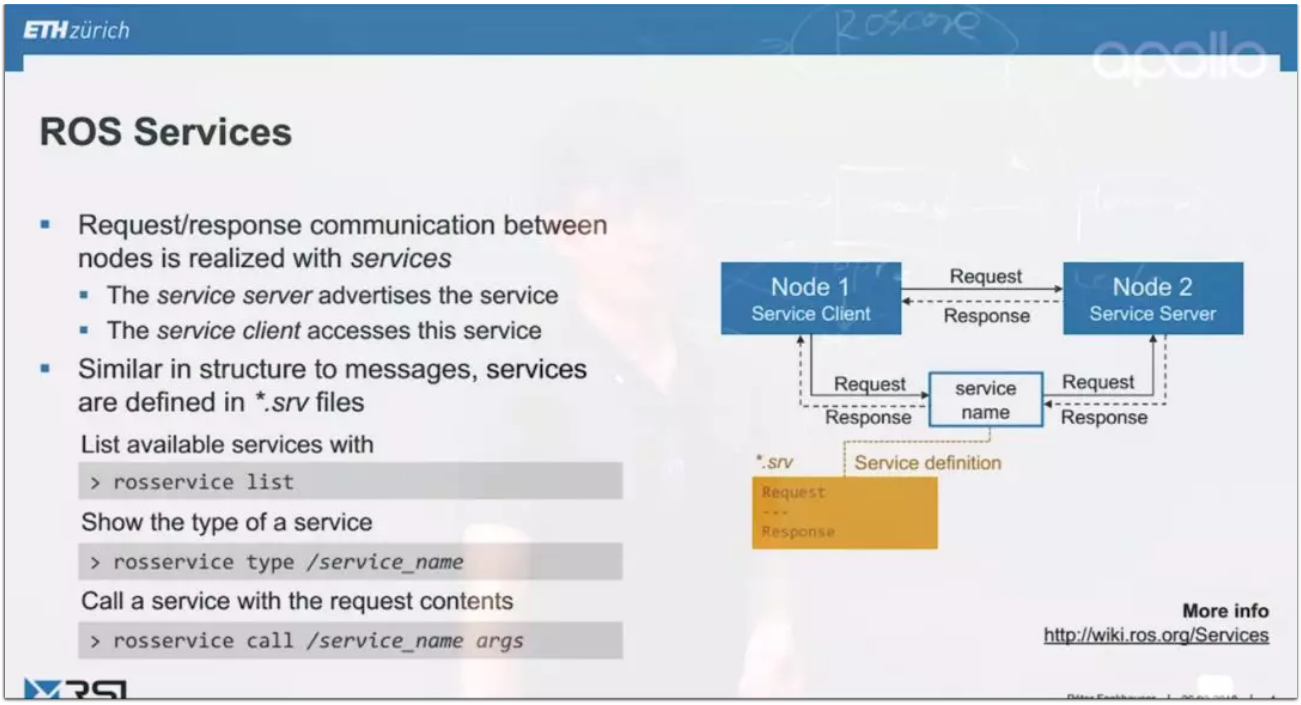
进阶课程㉟丨Apollo ROS原理—3

ROS是一个强大而灵活的机器人编程框架，从软件构架的角度说，它是一种基于**消息传递通信的分布式多进程框架**。ROS本身是基于消息机制的，可以根据功能把软件拆分成为各个模块，每个模块只是负责读取和分发消息，模块间通过消息关联。

目前ROS仅适用于Apollo 3.0之前的版本，最新代码及功能还请参照Apollo 3.5及5.0版本。

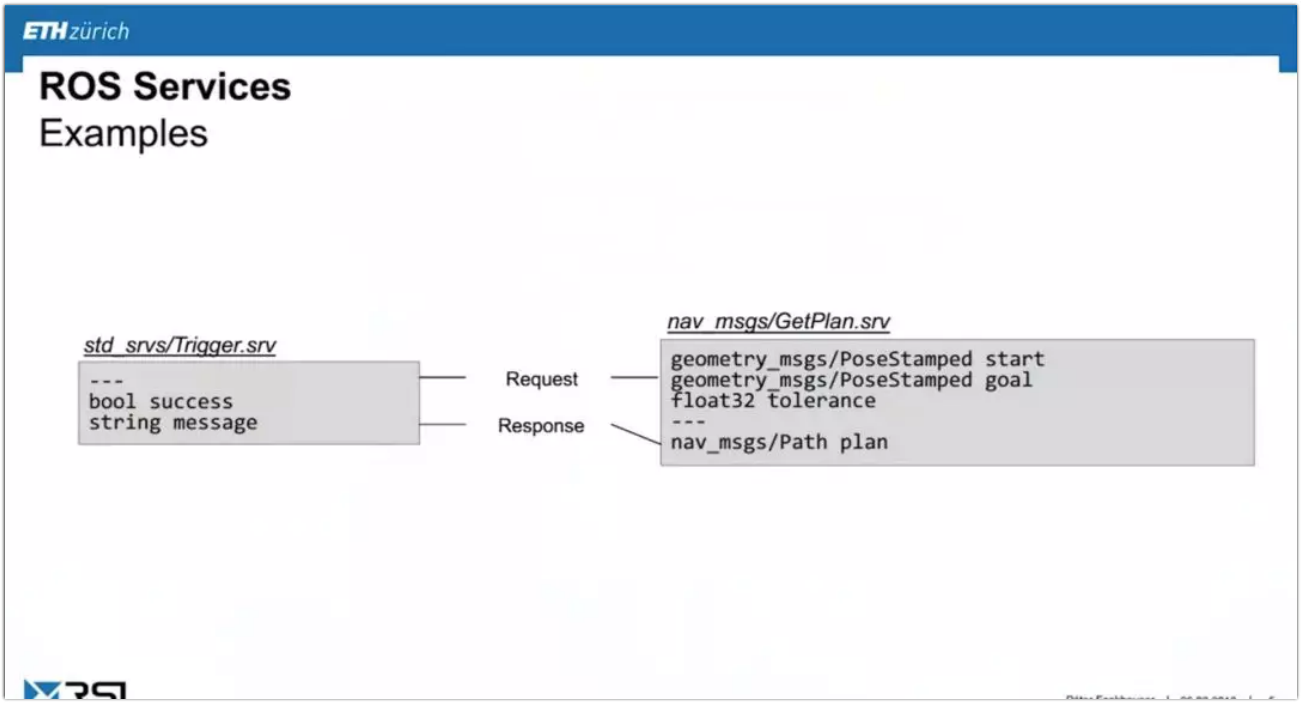
以下，ENJOY

本节主要介绍几个在实际开发调试过程中使用比较广泛的一些概念。

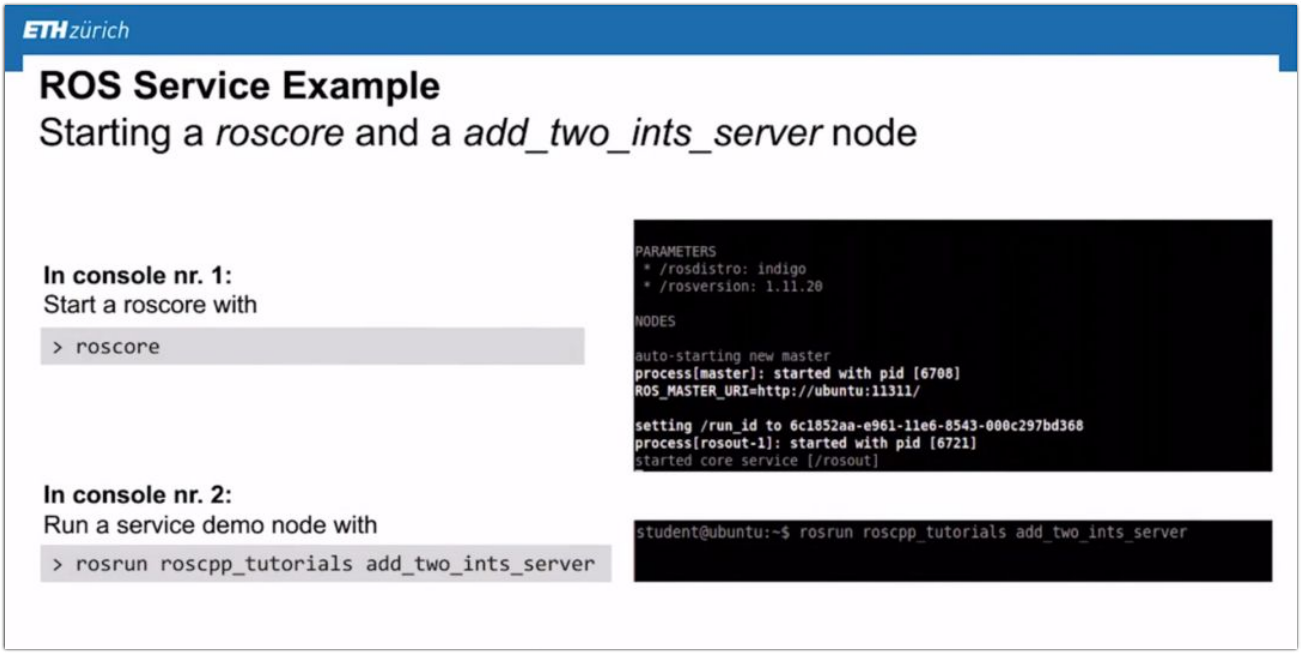


ROS提供了三种节点之间通信的方式：第一种是大家最常用的基于消息的订阅发布模型，第二种就是ROS Service，第三种Param，它借鉴了Service的思想。Service在自动驾驶系统里面使用的比较广泛，与基于消息发布订阅模型类似， Service有一个service name，同时Service底层是一个SRV描述文件，它和MSG描述文件比较类似，不同是SRV描述文件定义了两种消息：请求信息的消息格式和响应格式。请求是Client向Server发出请求的消息定义格式，与Response逻辑类似。

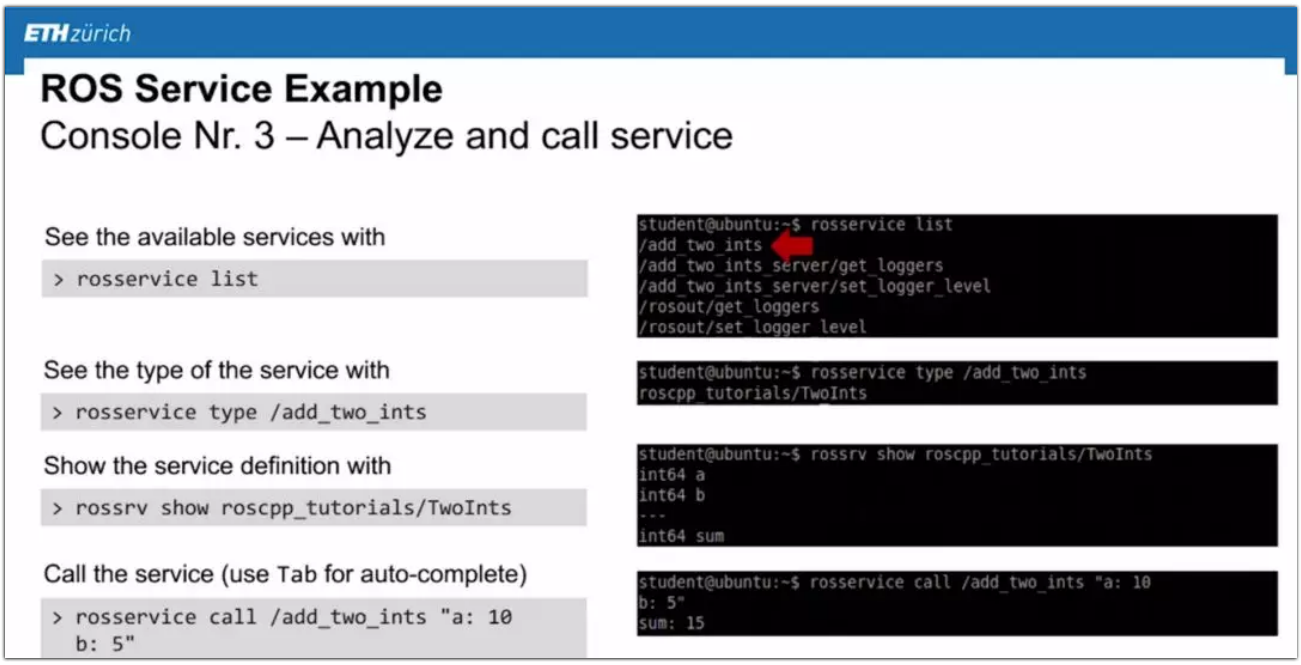
对应Service，Rosservice提供了一系列命令行工具，例如常用的像List、Call等一些基本的功能响应。



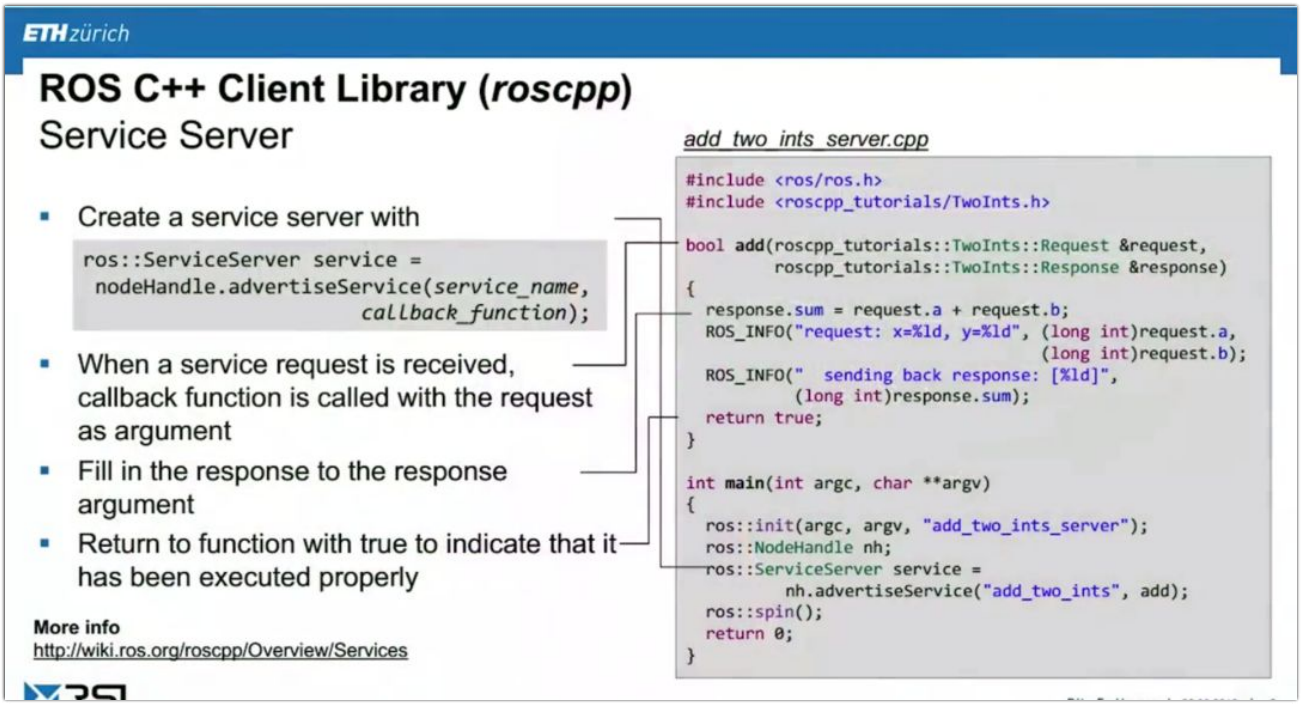
这是一个SRV文件，可以看到所有的Service对应的SRV文件描述都有一个Request和Response方式。当然这两个都可以置为空，置空就没有意义了。此外，也可以写一些具体的类型，比如我向你发送一个什么样的消息请求，你在接受我的对应消息请求之后会返回一个什么样的响应的数据格式指令。



上面结合一个实际的例子，给出Service具体的使用方法。Service启动的时候，需要提前启动Roscore，即**节点管理器**。第二步启动Service的一个例子程序，之后通过List和Type命令可以看到在启动某一个节点后，这个节点里面注册了某个Service的一个实际展示。

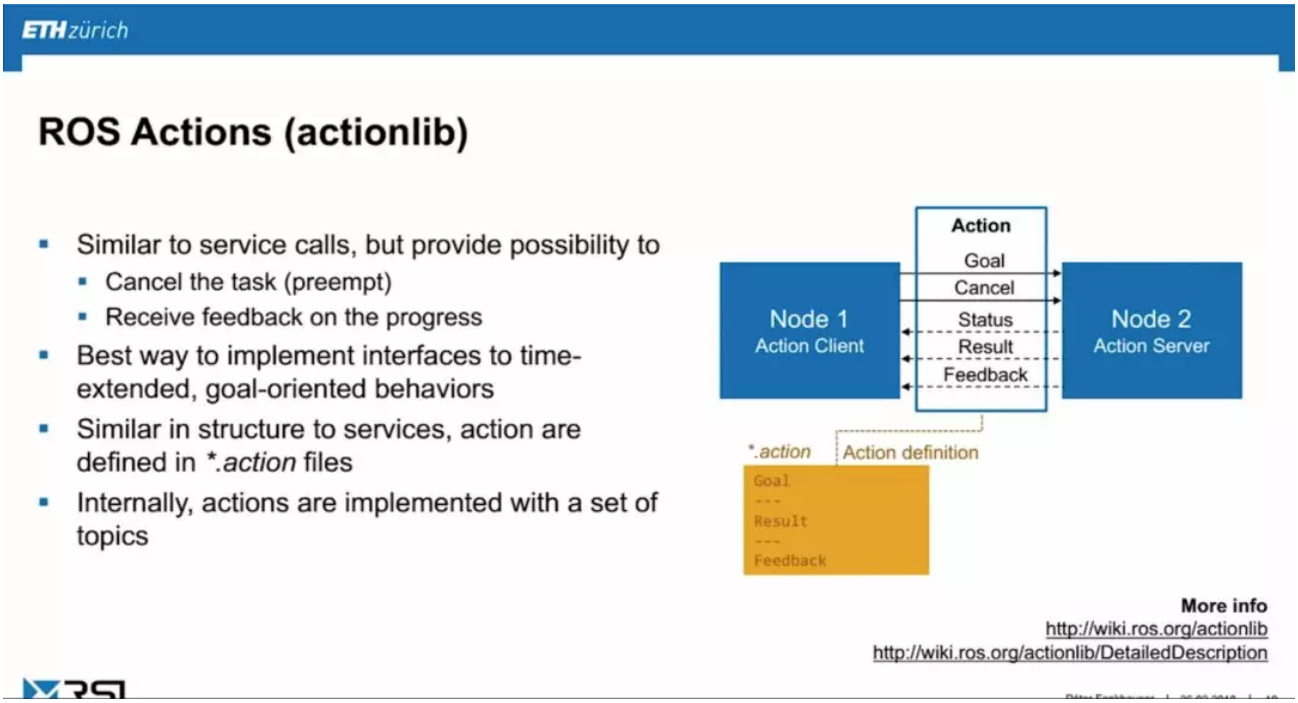


与前面提到的Rostopic对应，ROS也提供了命令行方式调用一个Service，当然命令行方式调用Service也是把它当成了一个节点的方式进行Service、Client链路的建立和响应。

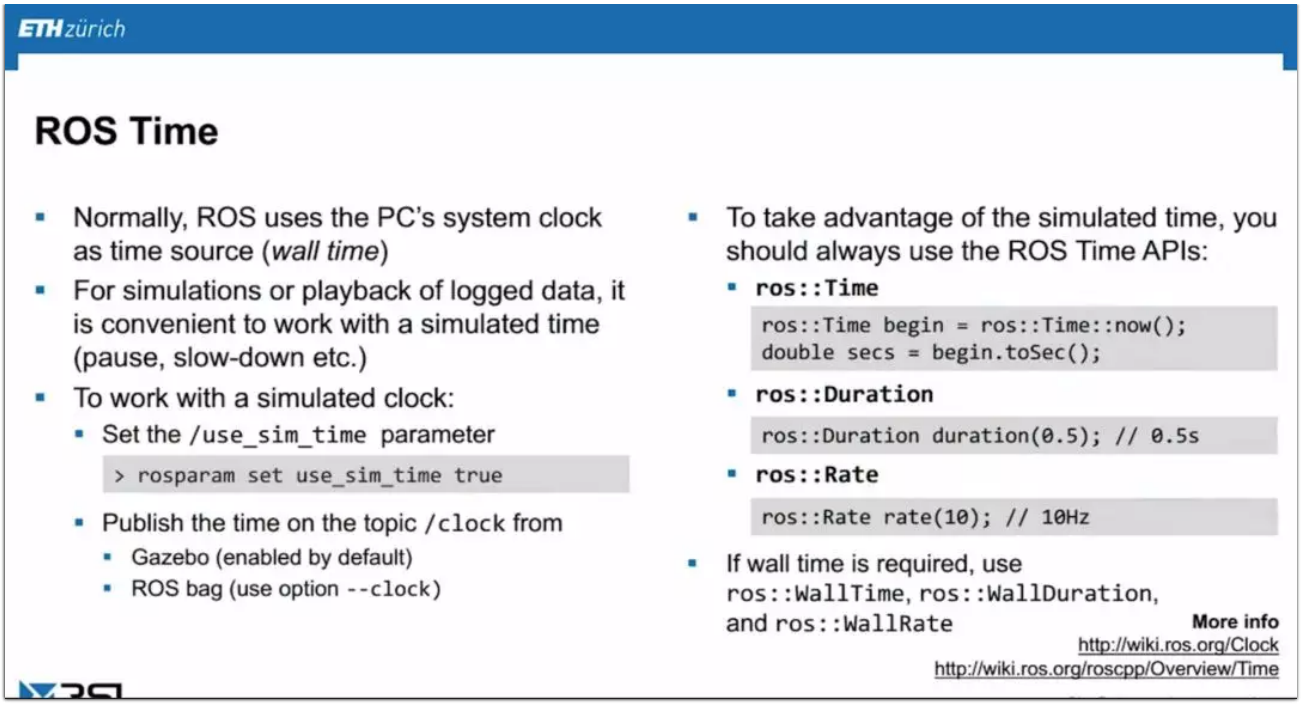


这个是结合实际的C++例子，描述Service的具体使用方法，可以看到Service和普通的Publisher、Subscriber比较类似，尤其是和Subscriber比较类似。但不同的是它有一个Service方法，定义了节点提供的服务类型。以及当Client向它发出请求时，节点会做出什么样的行为同时把这个结果再发送给Client。第二个不同点是在注册Service时，把Service的Name，和Service所提供的函数注册即可。与注册Server的节点对应，Client的节点也比较简单，只需要进行两步就可以完成一个Service的使用，第一步是声明Client对象，第二步是直接去调用Service,传入对应的Request就可以拿到对应的Response结果。

ROS Actions



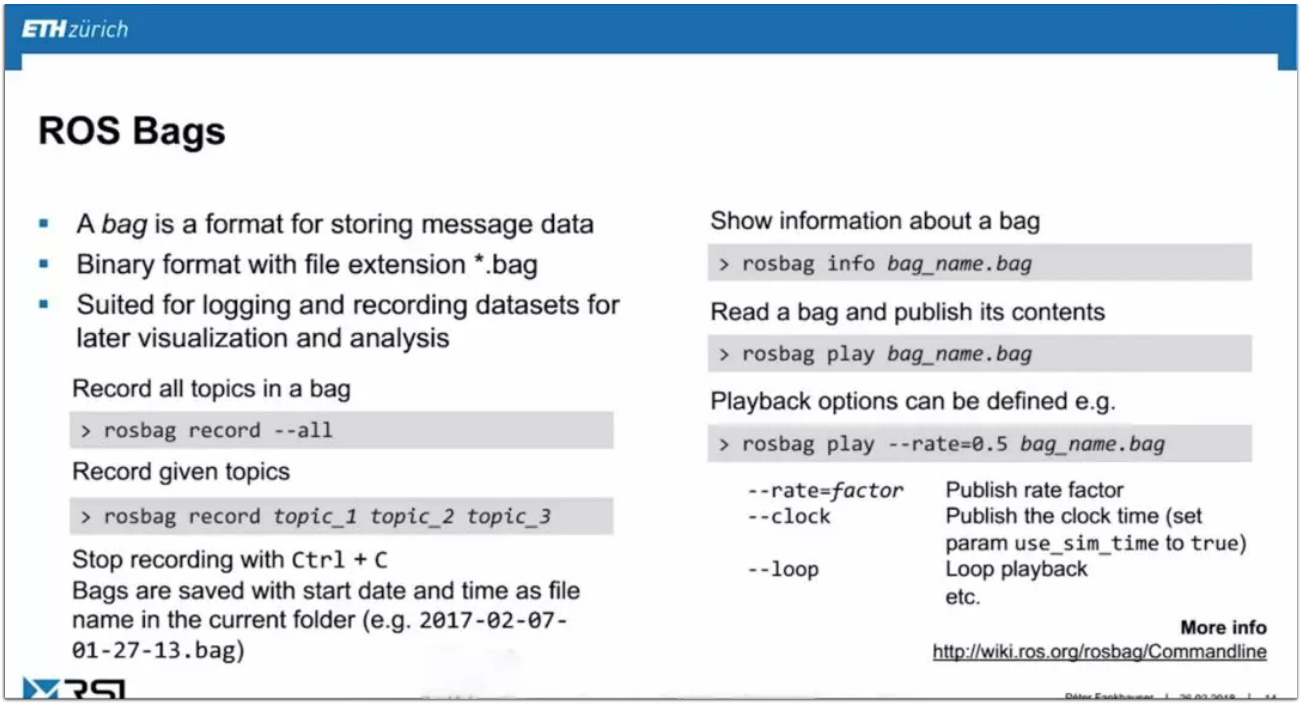
ROS还提供了另外一种通讯方式，这种不常见通讯方式就是Actions，相比Service，它多了一个取消的功能和带有反馈机制。对于Service，发起一个Service请求需要等到返回一个正确的Response结果才会退出。Actions在发送一个Service请求之后，它可以发送取消的命令，取消这个Service请求，可用于一些较长时间的Service场景。当然目前这种场景在Apollo自动驾驶系统里面比较少。ROS Actions在.action文件里定义了action，跟srv其实类似，只不过是在.action文件里面，定义了更多的类型。



ROS系统供了一套time机制，这个time的时间源来自于PC机的系统时间。Rostime基于此提供了一个重要的功能点：**仿真时间**。用ROS系统进行自动驾驶开发的时候，仿真模拟是一个不可缺少的环节。如刚才提到的Rviz、Gazebo都是为开发者进行离线仿真和模拟的强大调试工具，在使用这些调试工具的时候，实验数据可能是很早之前录制的，也有可能是在其他地方录制的。这种之前的数据在仿真环境下进行模拟时，如何回放当时的场景，或者是如何把当时的时间转化到现在的时间？Rostime就供了虚拟时钟功能，保证在回放一些历史实验数据，或者其他地方实验数据时，让整个仿真系统认为现在的场景就是所需要的那个系统时间和系统场景。

04

ROS Bags



ROS Bags有两个比较重要的功能，第一个是把实际车上调试的数据或者是把自动驾驶进行道路测试的原始传感器数据按一定格式录制到某个bag文件里。实验室或者开发环境可以根据bag文件不断的回放，去复现当时的网络场景。例如，有一个场景，车在某一个特定的地方做了一个错误的决策，我们想改一版算法去验证这个场景有没有被覆盖，这时可以拿那个Rosbag回放，验证新版算法的输出是不是符合预期。另外Rosbag数据对算法进行模型训练和调优也是非常有必要的。

05

调试工具

最后了解一下调试工具，如下图所示，ROS提供了一些简单的功能，比如说ROS WTF这种功能，可以让用户很简单地查看当前系统是环境变量设置的问题，还是其他的一些核心库链接的位置问题，还是其他的问题导致的一些运行失败，通过WTF都可以很快的定位，同时，ROS也提供了一些其他的Debug诊断功能供开发者在实际开发过程当中去使用。

Debugging Strategies

Debug with the tools you have learned

- Compile and run code often to catch bugs early
- Understand compilation and runtime error messages
- Use analysis tools to check data flow (`rosnode info`, `rostopic echo`, `roswtf`, `rqt_graph` etc.)
- Visualize and plot data (RViz, RQT Multiplot etc.)
- Divide program into smaller steps and check intermediate results (`ROS_INFO`, `ROS_DEBUG` etc.)
- Make your code robust with argument and return value checks and catch exceptions
- If things don't make sense, clean your workspace

```
> catkin clean --all
```

Learn new tools

- Build in *debug* mode and use GDB or Valgrind

```
> catkin config --cmake-args
                -DCMAKE_BUILD_TYPE=Debug
```

- Use Eclipse breakpoints
- Maintain code with unit tests and integration tests

More info

http://wiki.ros.org/UnitTesting
http://wiki.ros.org/gtest
http://wiki.ros.org/rostest
http://wiki.ros.org/roslaunch/Tutorials/Roslaunch%20Nodes%20in%20Valgrind%20or%20GDB