深蓝学院 shenlanxueyuan.com

古月居 GYH.AI

ROS理论与实践

—— 第2讲：认识ROS

主讲人 胡春旭

机器人博客“古月居”博主

《ROS机器人开发实践》作者

武汉精锋微控科技有限公司 联合创始人

华中科技大学 自动化学院 硕士

目录

1. ROS的定义与组成

1. ROS的定义与组成

ROS，Robot Operating System，机器人操作系统

1. What is ROS?

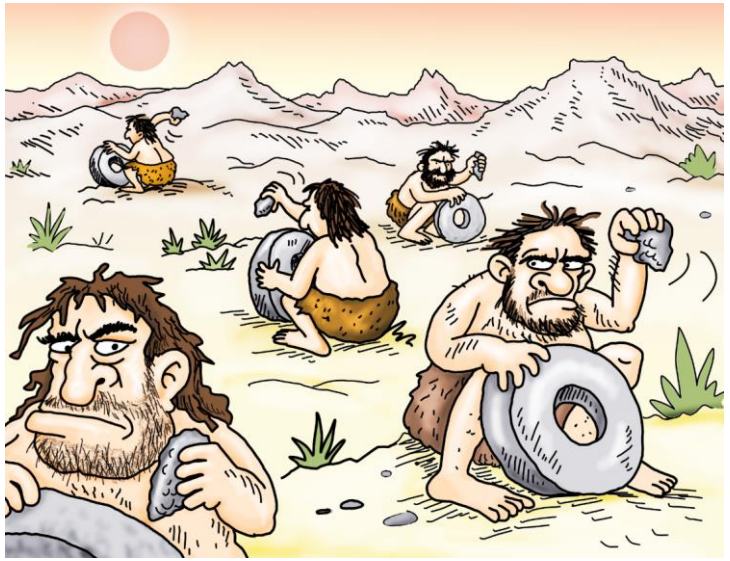
ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is similar in some respects to 'robot frameworks,' such as Player, YARP, Orocos, CARMEN, Orca, MOOS, and Microsoft Robotics Studio.

The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server. These are explained in greater detail in our Conceptual Overview.

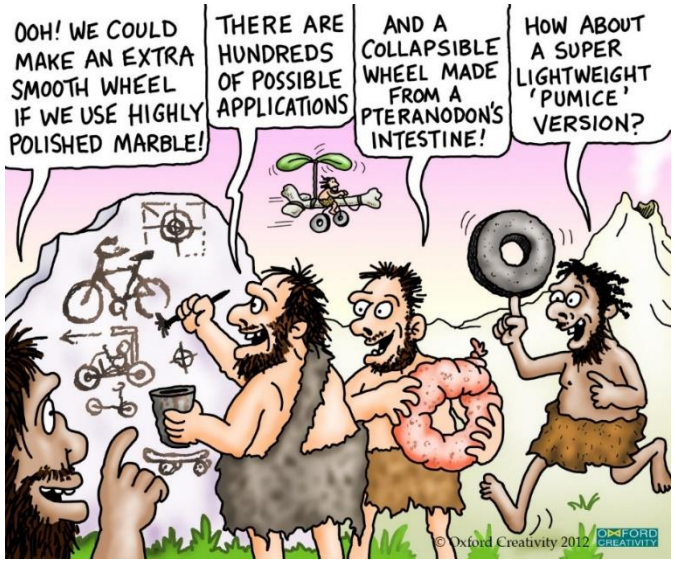
ROS is not a realtime framework, though it is possible to integrate ROS with realtime code. The Willow Garage PR2 robot uses a system called pr2_etherCAT, which transports ROS messages in and out of a realtime process. ROS also has seamless integration with the Orocos Real-time Toolkit.

（来源：http://wiki.ros.org/ROS/Introduction ）

1. ROS的定义与组成



传统模式



现代模式

提高机器人研发中的软件复用率

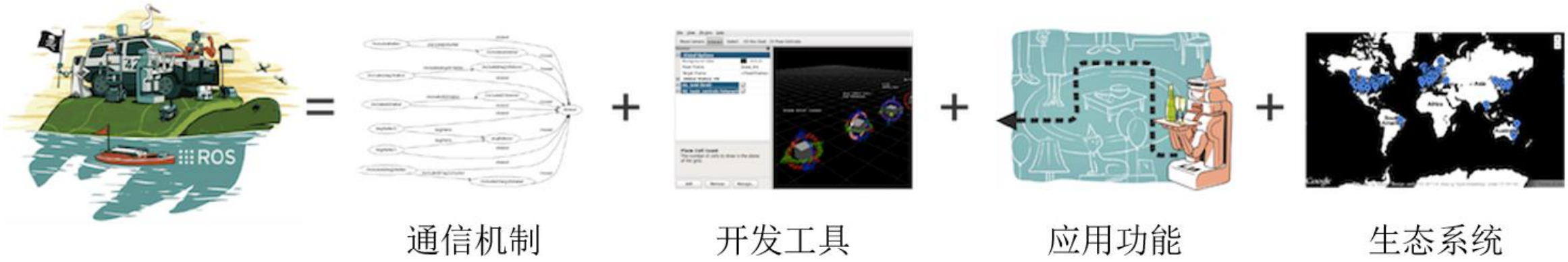
（1）点对点的设计

（2）多语言支持

（3）架构精简、集成度高

（4）组件化工具包、功能包丰富

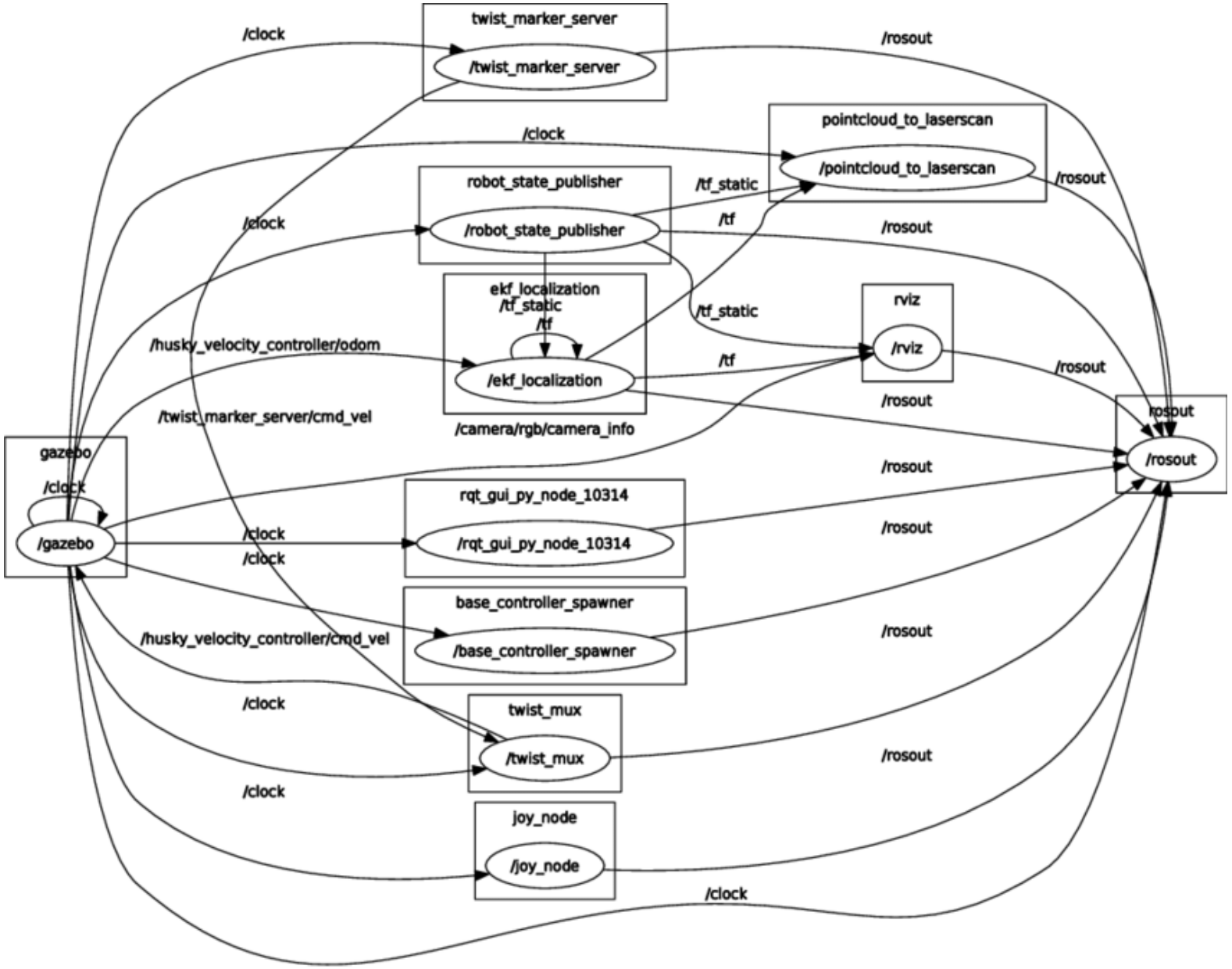
（5）免费并且开源



通信机制　开发工具　应用功能　生态系统

1. ROS的定义与组成

松耦合分布式通信



1. ROS的定义与组成

WORKSPACES

Create Workspace

```
mkdir catkin_ws && cd catkin_ws
wstool init src
catkin_make
source devel/setup.bash
```

Add Repo to Workspace

```
roscd; cd ../src
wstool set repo_name \
--git http://github.com/org/repo_name.git \
--version=kinetic-devel
wstool up
```

Resolve Dependencies in Workspace

```
sudo rosdep init  # only once
rosdep update
rosdep install --from-paths src --ignore-src \
--rosdistro=${ROS_DISTRO} -y
```

PACKAGES

Create a Package

```
catkin_create_pkg package_name [dependencies ...]
```

Package Folders

include/package_name	C++ header files
src	Source files. Python libraries in subdirectories
scripts	Python nodes and scripts
msg, srv, action	Message, Service, and Action definitions

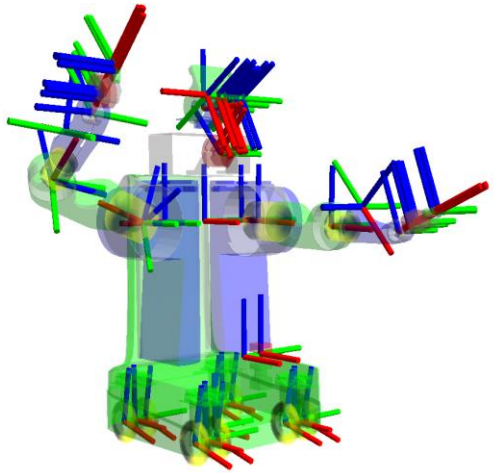
Release Repo Packages

```
catkin_generate_changelog
# review & commit changelogs
catkin_prepare_release
bloom-release --track kinetic --ros-distro kinetic repo_name
```

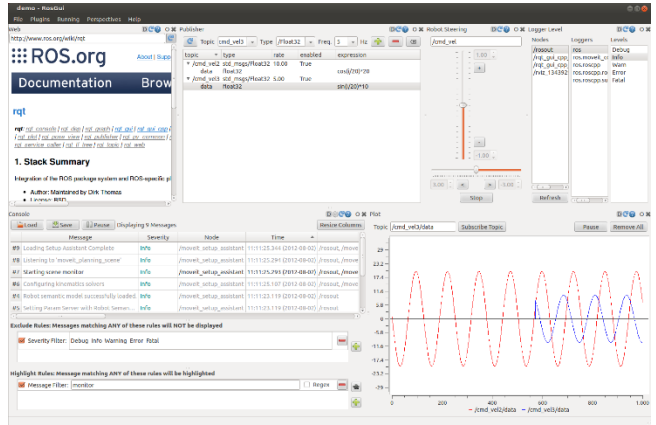
Reminders

- Testable logic
- Publish diagnostics
- Desktop dependencies in a separate package

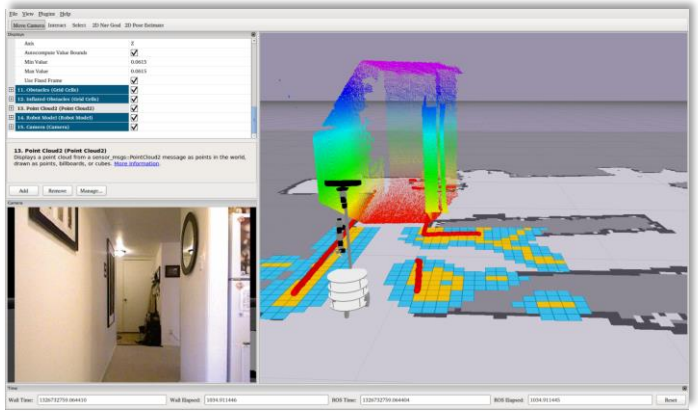
命令行&编译器



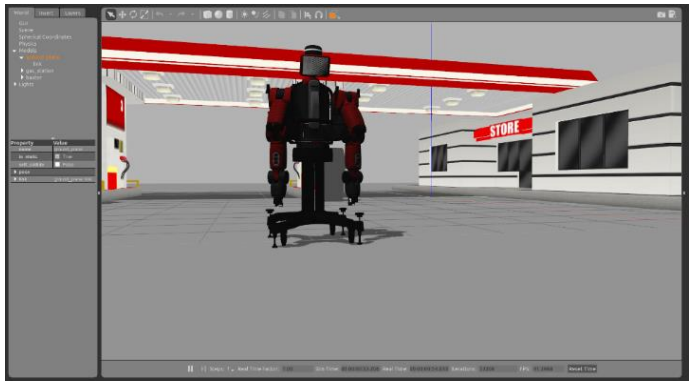
TF坐标变换



QT工具箱

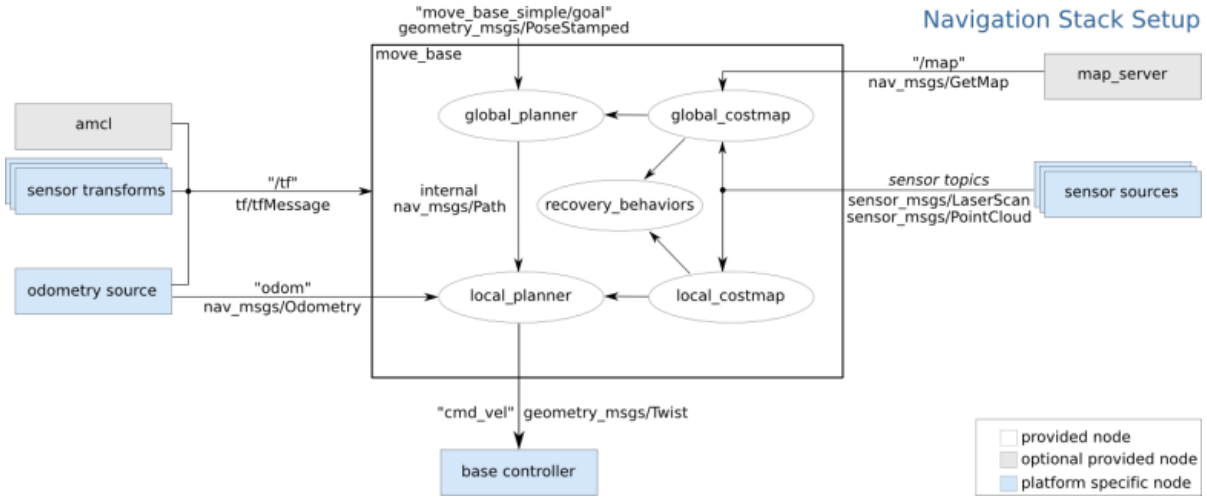


Rviz

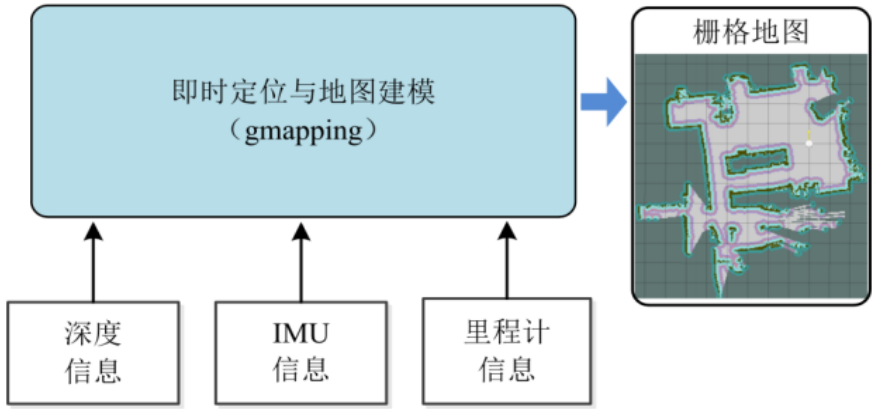


Gazebo

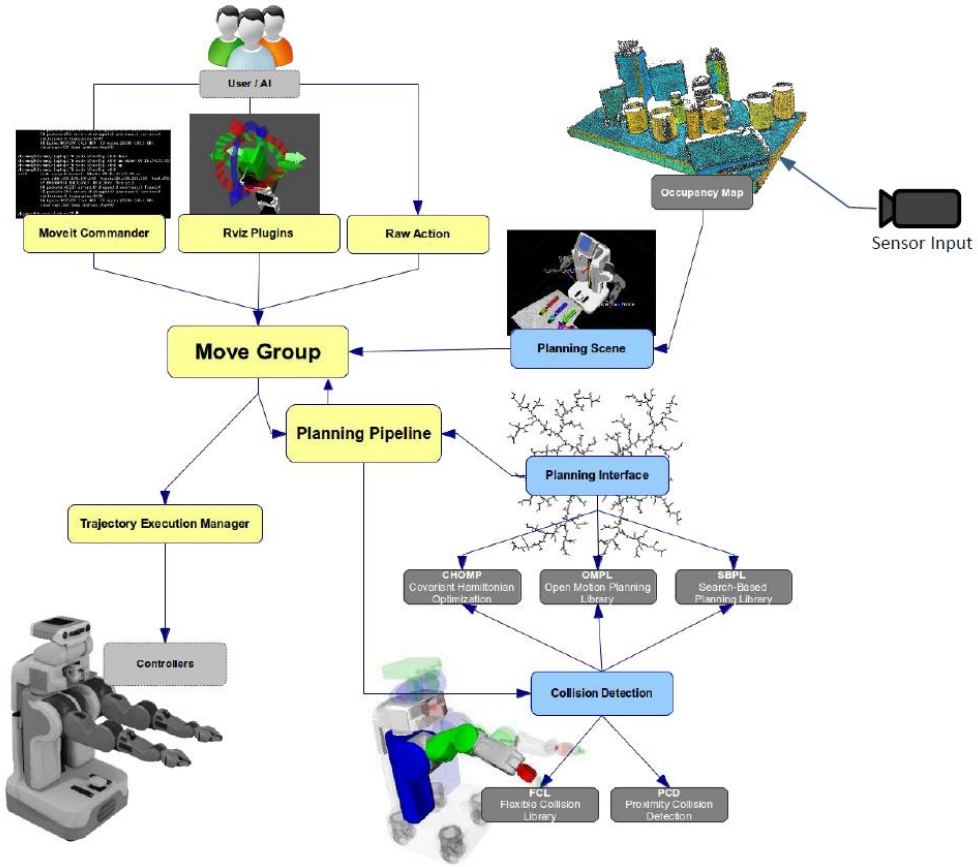
1. ROS的定义与组成



Navigation



SLAM



MoveIt!

1. ROS的定义与组成

1. 发行版（Distribution）：ROS发行版包括一系列带有版本号、可以直接安装的功能包。

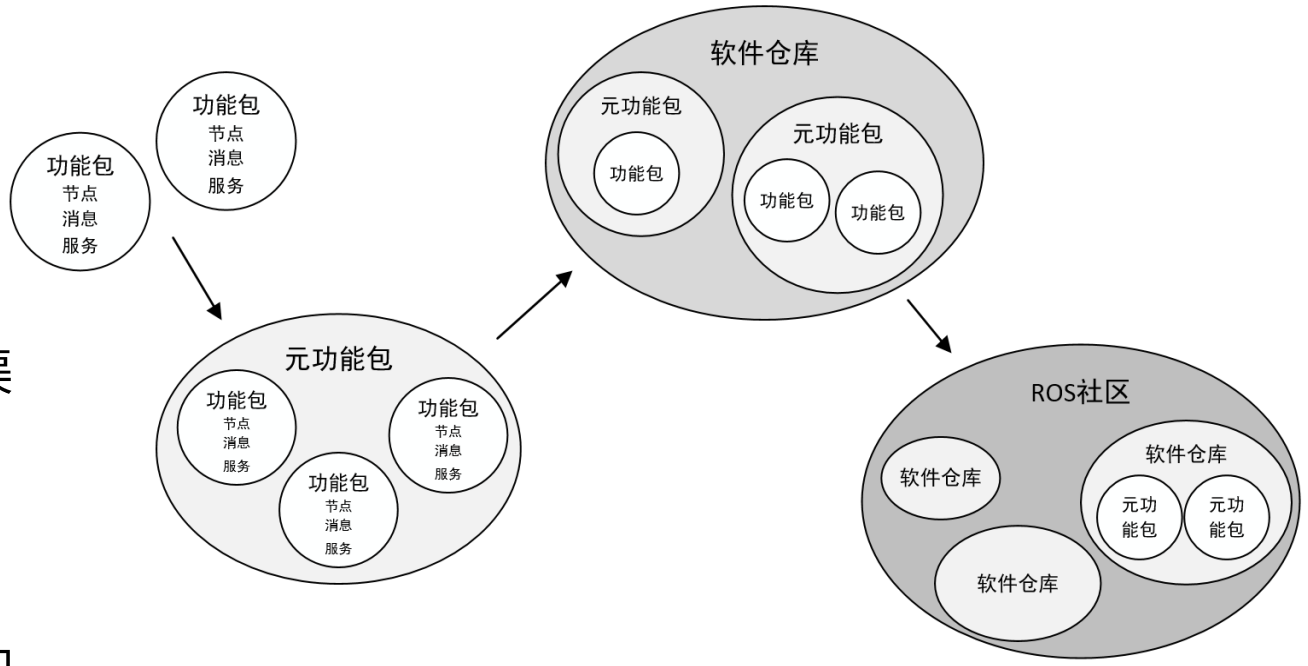
2. 软件源（Repository）：ROS依赖于共享网络上的开源代码，不同的组织机构可以开发或者共享自己的机器人软件。

3. ROS wiki：记录ROS信息文档的主要论坛。

4.邮件列表（Mailing list）：交流ROS更新的主要渠道，同时也可以交流ROS开发的各种疑问。

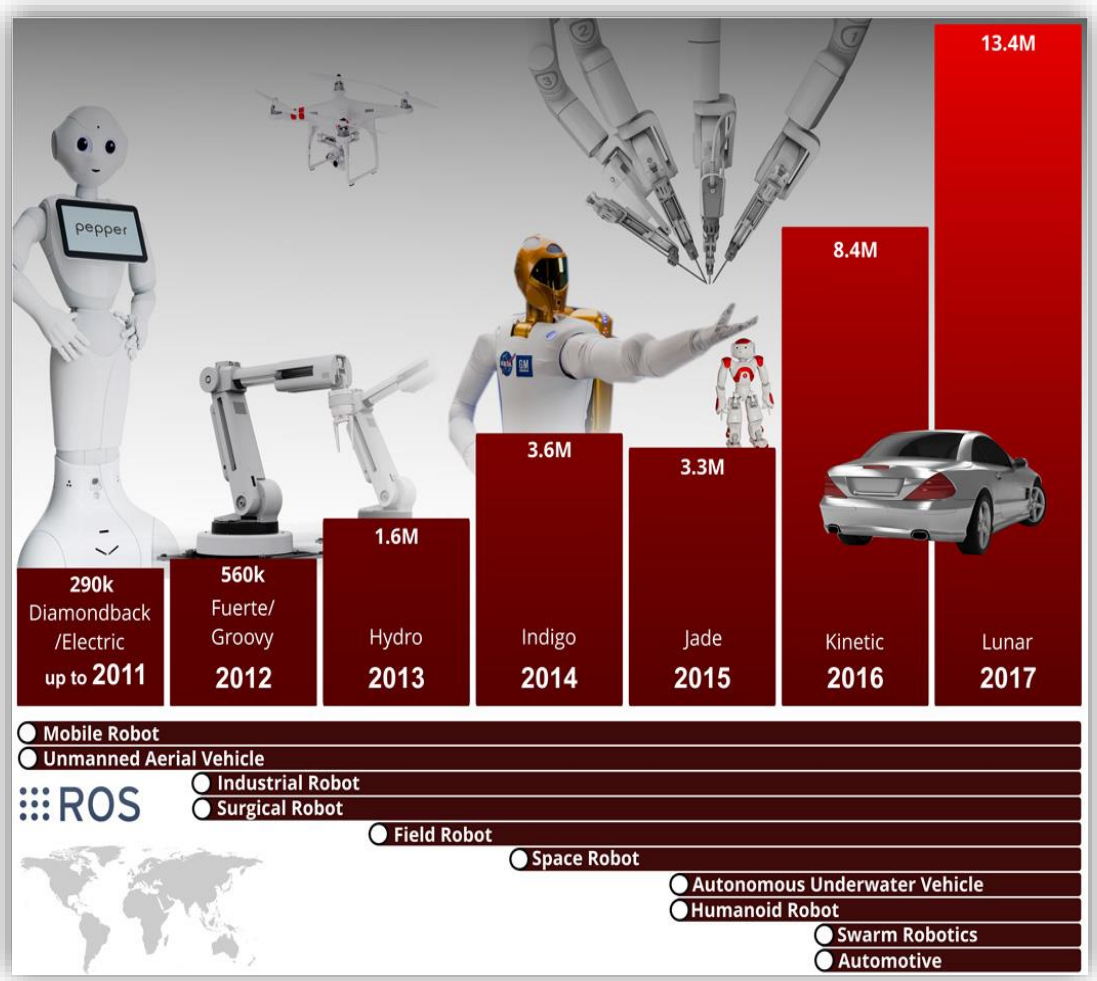
5. ROS Answers：咨询ROS相关问题的网站。

6. 博客（Blog）：发布ROS社区中的新闻、图片、视频（http://www.ros.org/news）



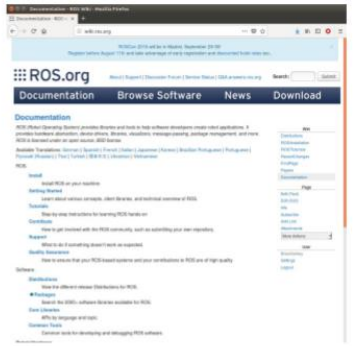
ROS社区资源的组织形式

1. ROS的定义与组成

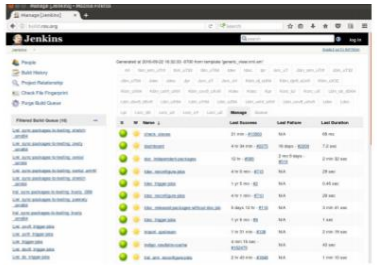


1.	United States	34,710 (19.08%)
2.	China	31,946 (17.56%)
3.	Japan	15,518 (8.53%)
4.	Germany	12,711 (6.99%)
5.	India	8,400 (4.62%)
6.	Philippines	7,235 (3.98%)
7.	South Korea	6,790 (3.73%)
8.	United Kingdom	4,325 (2.38%)
9.	Taiwan	4,233 (2.33%)
10.	France	3,725 (2.05%)

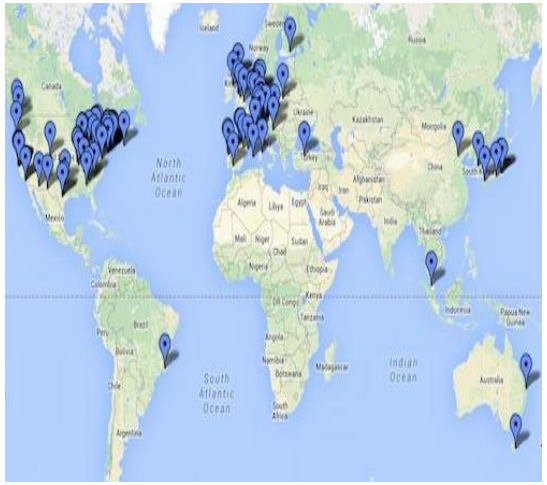
- Global impact: USA constitutes 19% of users***
- Partial translations in 14 languages



- 178K monthly users**
  - 35% annual increase*
- 1.49M annual users***
- 24.48M annual page views***



- 327K monthly downloaders**
  - 41% annual increase*
- 16.2M monthly binary packages downloaded**
  - 21% annual increase*
- 5.9TB monthly download traffic**
  - 25% annual increase*

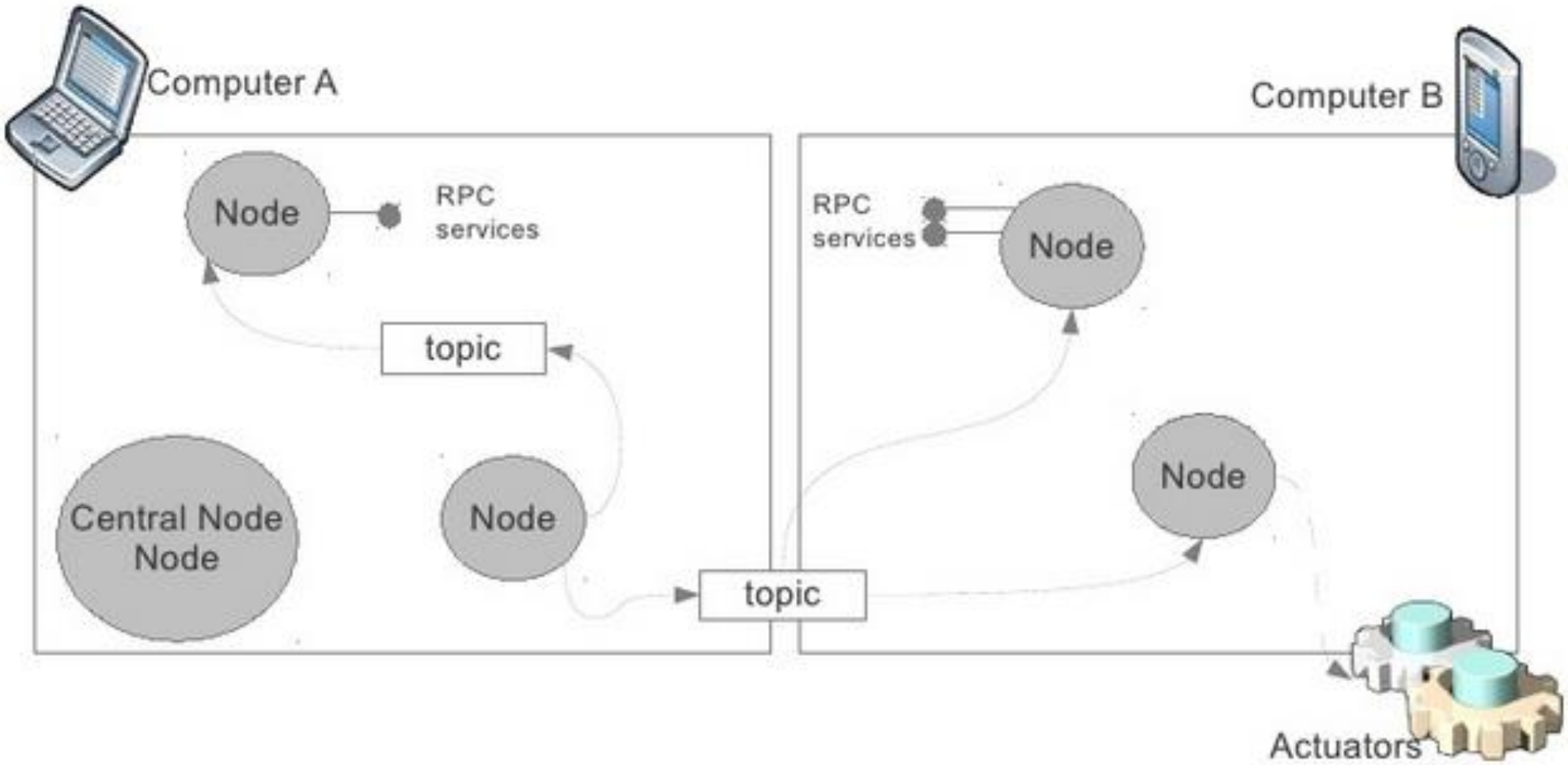


ROS社区内的功能包数量、下载量、wiki访问量、相关文章均呈大幅度上涨

（来源：ROSCon2017、2018）

2. ROS核心概念与通信机制

2. ROS核心概念与通信机制



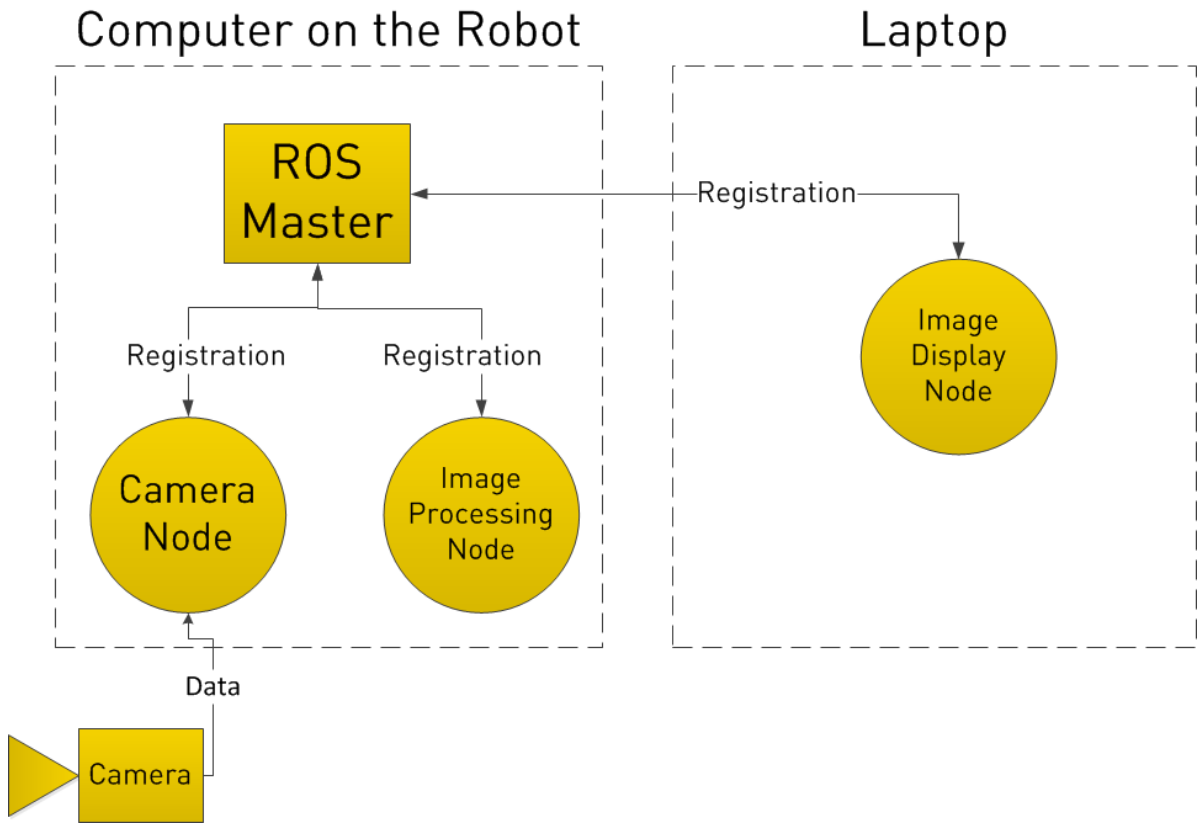
2. ROS核心概念与通信机制

节点（Node）—— 执行单元

- 执行具体任务的进程、独立运行的可执行文件；
- 不同节点可使用不同的编程语言，可分布式运行在不同的主机；
- 节点在系统中的名称必须是唯一的。

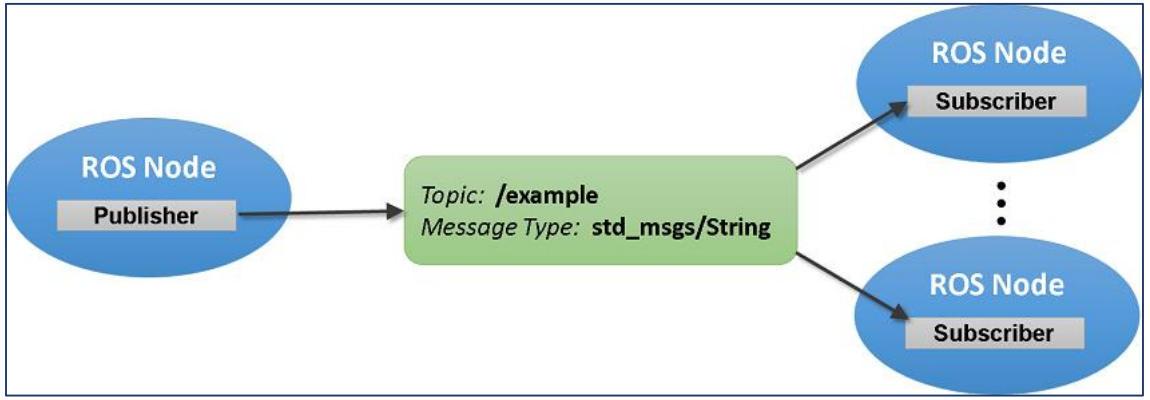
节点管理器 （ROS Master）—— 控制中心

- 为节点提供命名和注册服务；
- 跟踪和记录话题/服务通信，辅助节点相互查找、建立连接；
- 提供参数服务器，节点使用此服务器存储和检索运行时的参数。

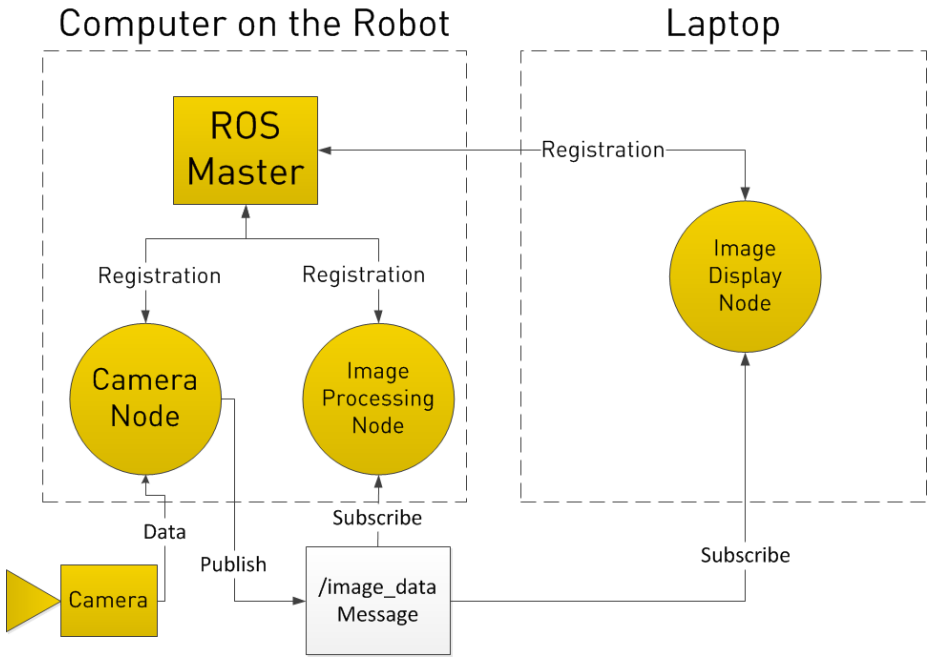


2. ROS核心概念与通信机制

- **话题（Topic）—— 异步通信机制**
  - 节点间用来传输数据的重要总线；
  - 使用发布/订阅模型，数据由发布者传输到订阅者，同一个话题的订阅者或发布者可以不唯一。
- **消息（Message）—— 话题数据**
  - 具有一定的类型和数据结构，包括ROS提供的标准类型和用户自定义类型；
  - 使用编程语言无关的.msg文件定义，编译过程中生成对应的代码文件。



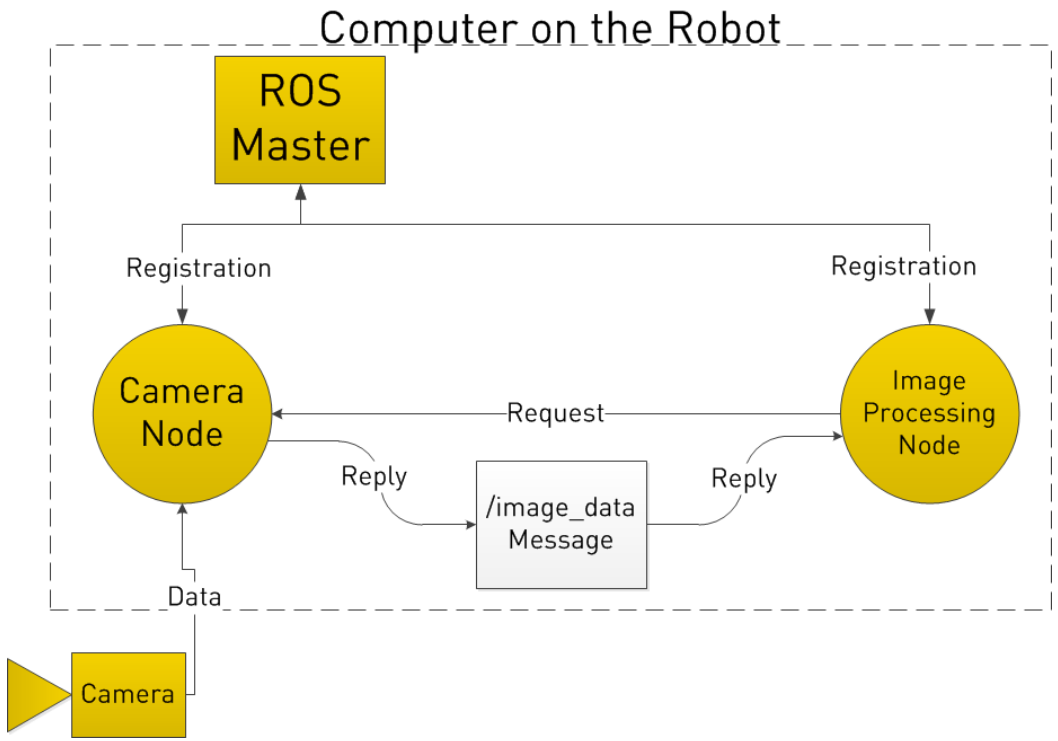
话题模型（发布/订阅）

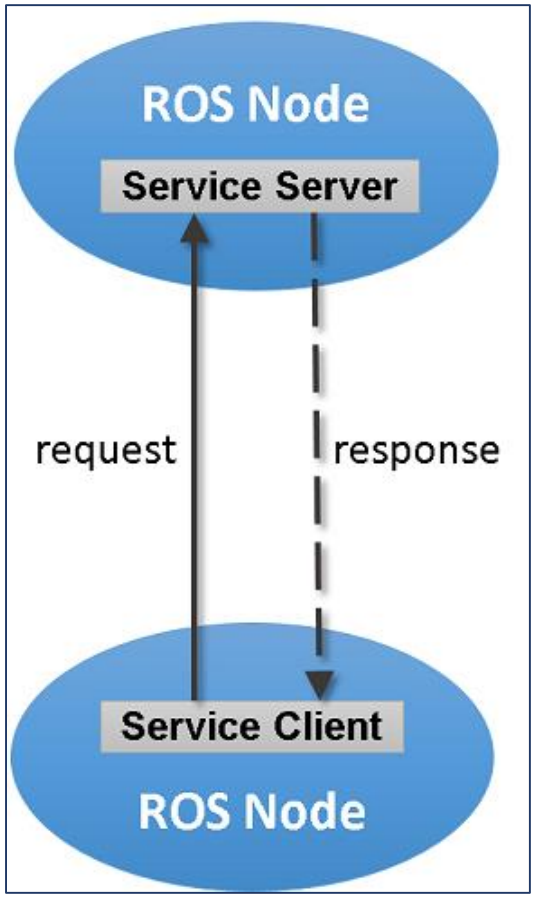


2. ROS核心概念与通信机制

服务（Service）—— 同步通信机制

- 使用客户端/服务器（C/S）模型，客户端发送请求数据，服务器完成处理后返回应答数据；
- 使用编程语言无关的.srv文件定义请求和应答数据结构，编译过程中生成对应的代码文件。





服务模型（请求/应答）

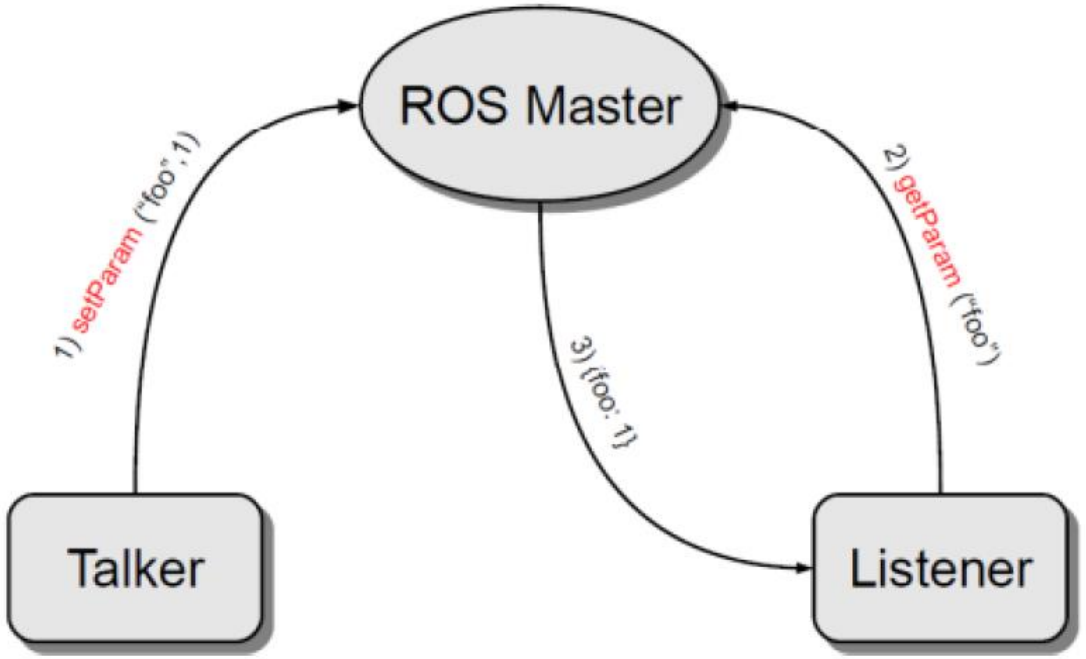
话题与服务的区别

	话题	服务
同步性	异步	同步
通信模型	发布/订阅	服务器/客户端
底层协议	ROSTCP/ROSUDP	ROSTCP/ROSUDP
反馈机制	无	有
缓冲区	有	无
实时性	弱	强
节点关系	多对多	一对多（一个server）
适用场景	数据传输	逻辑处理

2. ROS核心概念与通信机制

■ 参数（Parameter）—— 全局共享字典

- 可通过网络访问的共享、多变量字典；
- 节点使用此服务器来存储和检索运行时的参数；
- 适合存储静态、非二进制的配置参数，不适合存储动态配置的数据。



参数模型（全局字典）

2. ROS核心概念与通信机制

■ 功能包（Package）

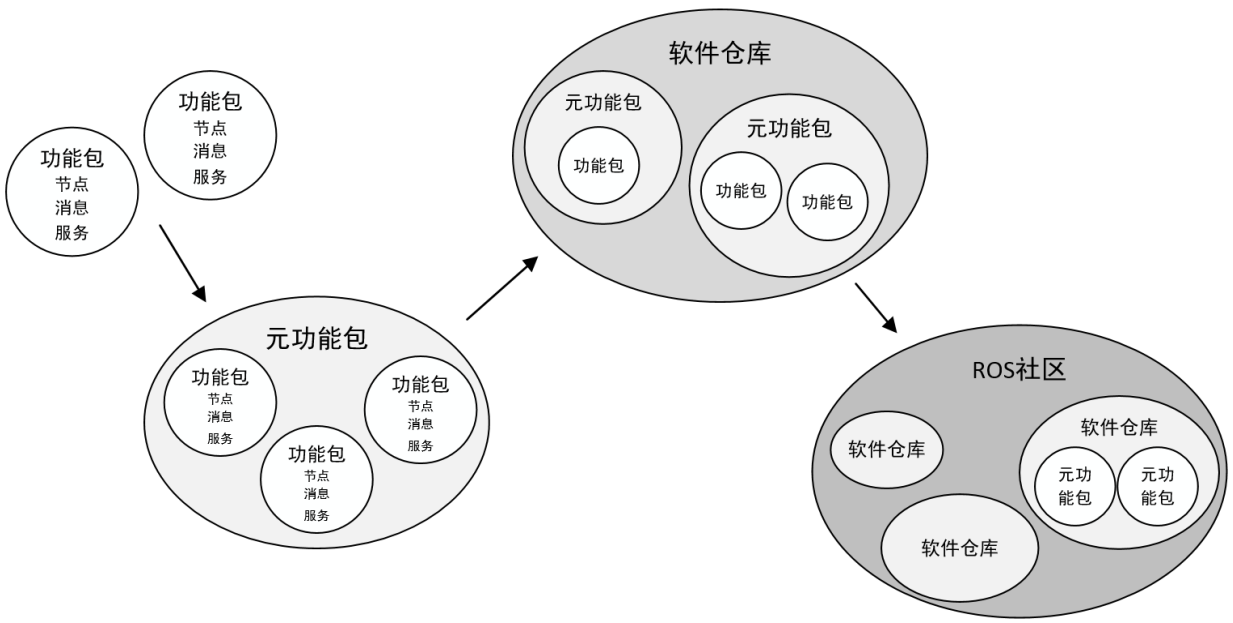
- ROS软件中的基本单元，包含节点源码、配置文件、数据定义等

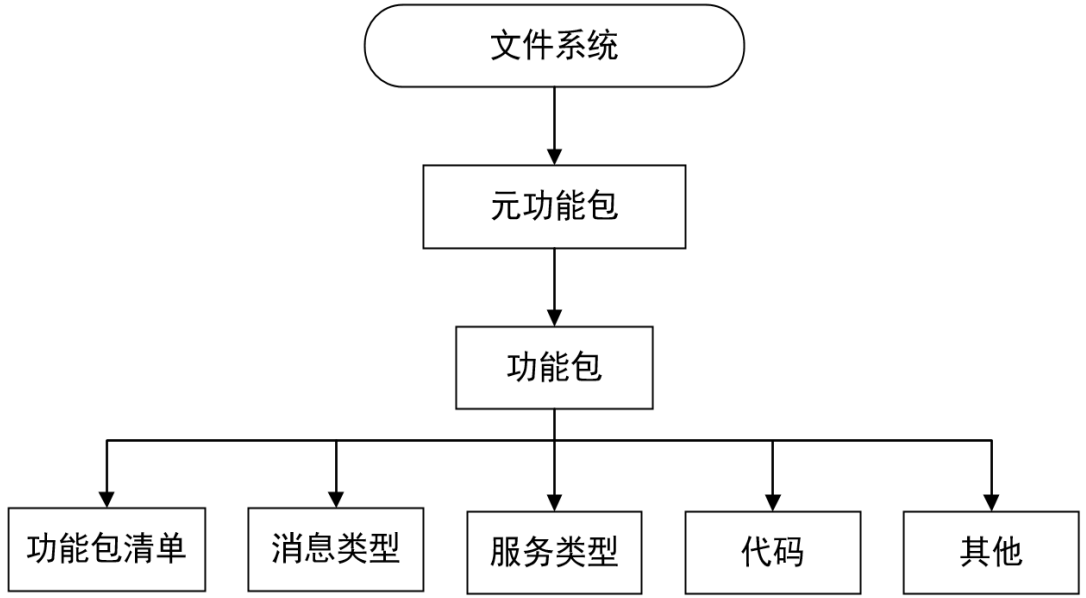
■ 功能包清单（Package manifest）

- 记录功能包的基本信息，包含作者信息、许可信息、依赖选项、编译标志等

■ 元功能包（Meta Packages）

- 组织多个用于同一目的功能包





3.第一个ROS例程——小海龟仿真分析

3. 第一个ROS例程——小海龟仿真分析

古月居 GYH.AI

常用命令

- rostopic
- rosservice
- rosnode
- rosparam
- rosmsg
- rossrv

WORKSPACES

Create Workspace

```
mkdir catkin_ws && cd catkin_ws
wstool init src
catkin_make
source devel/setup.bash
```

Add Repo to Workspace

```
roscd; cd ../src
wstool set repo_name \
--git http://github.com/org/repo_name.git \
--version=kinetic-devel
wstool up
```

Resolve Dependencies in Workspace

```
sudo rosdep init  # only once
rosdep update
rosdep install --from-paths src --ignore-src \
--rosdistro=${ROS_DISTRO} -y
```

PACKAGES

Create a Package

```
catkin_create_pkg package_name [dependencies ...]
```

Package Folders

include/package_name	C++ header files
src	Source files. Python libraries in subdirectories
scripts	Python nodes and scripts
msg, srv, action	Message, Service, and Action definitions

Release Repo Packages

```
catkin_generate_changelog
# review & commit changelogs
catkin_prepare_release
bloom-release --track kinetic --ros-distro kinetic repo_name
```

Reminders

- Testable logic
- Publish diagnostics
- Desktop dependencies in a separate package

CMakeLists.txt

Skeleton

```
cmake_minimum_required(VERSION 2.8.3)
project(package_name)
find_package(catkin REQUIRED)
catkin_package()
```

Package Dependencies

To use headers or libraries in a package, or to use a package's exported CMake macros, express a build-time dependency:

```
find_package(catkin REQUIRED COMPONENTS roscpp)
```

Tell dependent packages what headers or libraries to pull in when your package is declared as a catkin component:

```
catkin_package(
  INCLUDE_DIRS include
  LIBRARIES ${PROJECT_NAME}
  CATKIN_DEPENDS roscpp)
```

Note that any packages listed as CATKIN_DEPENDS dependencies must also be declared as a <run_depend> in package.xml.

Messages, Services

These go after find_package(), but before catkin_package().
Example:

```
find_package(catkin REQUIRED COMPONENTS message_generation
std_msgs)
add_message_files(FILES MyMessage.msg)
add_service_files(FILES MyService.msg)
generate_messages(DEPENDENCIES std_msgs)
catkin_package(CATKIN_DEPENDS message_runtime std_msgs)ww
```

Build Libraries, Executables

Goes after the catkin_package() call.

```
add_library(${PROJECT_NAME} src/main)
add_executable(${PROJECT_NAME}_node src/main)
target_link_libraries(
  ${PROJECT_NAME}_node ${catkin_LIBRARIES})
```

Installation

```
install(TARGETS ${PROJECT_NAME}
  DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION})
install(TARGETS ${PROJECT_NAME}_node
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
install(PROGRAMS scripts/myscript
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
install(DIRECTORY launch
  DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION})
```

RUNNING SYSTEM

Run ROS using plain:

```
roscore
```

Alternatively, roslaunch will run its own roscore automatically if it can't find one:

```
roslaunch my_package package_launchfile.launch
```

Suppress this behaviour with the --wait flag.

Nodes, Topics, Messages

```
rosnode list
rostopic list
rostopic echo cmd_vel
rostopic hz cmd_vel
rostopic info cmd_vel
rosmsg show geometry_msgs/Twist
```

Remote Connection

Master's ROS environment:

- ROS_IP or ROS_HOSTNAME set to this machine's network address.
- ROS_MASTER_URI set to URI containing that IP or hostname.

Your environment:

- ROS_IP or ROS_HOSTNAME set to your machine's network address.
- ROS_MASTER_URI set to the URI from the master.

To debug, check ping from each side to the other, run roswtf on each side.

ROS Console

Adjust using rqt_logger_level and monitor via rqt_console. To enable debug output across sessions, edit the $HOME/.ros/config/rosconsole.config and add a line for your package:

```
log4j.logger.ros.package_name=DEBUG
```

And then add the following to your session:

```
export ROSCONSOLE_CONFIG_FILE=$HOME/.ros/config/rosconsole.config
```

Use the roslaunch --screen flag to force all node output to the screen, as if each declared <node> had the output="screen" attribute.

CLEARPATH ROBOTICS™

www.clearpathrobotics.com/ros-cheat-sheet

3. 第一个ROS例程——小海龟仿真分析

启动ROS Master

```
$ roscore
```

⬇

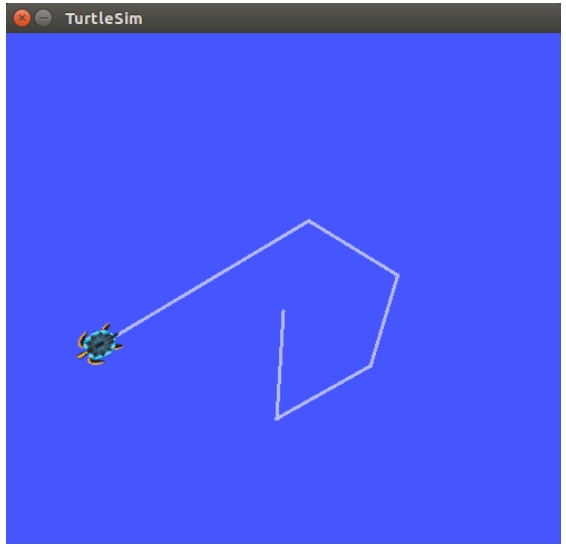
启动小海龟仿真器

```
$ rosrun turtlesim turtlesim_node
```

⬇

启动海龟控制节点

```
$ rosrun turtlesim turtle_teleop_key
```



小海龟仿真器界面

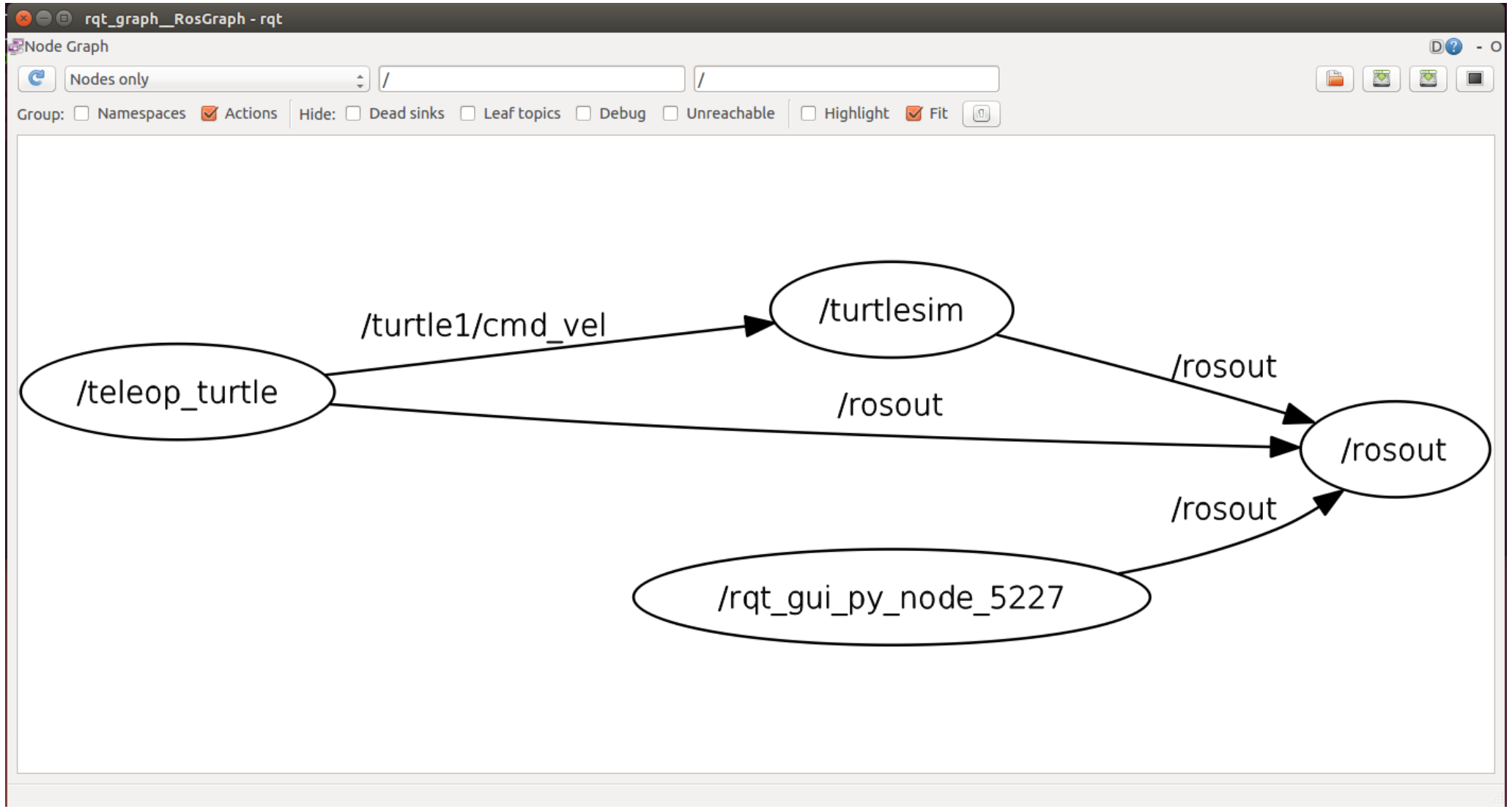
```
hcx@hcx-vpc:~$ rosrun turtlesim turtlesim_node
[ INFO] [1561200736.947992315]: Starting turtlesim with node name /turtlesim
[ INFO] [1561200736.954437402]: Spawning turtle [turtle1] at x=[5.544445], y=[5.
544445], theta=[0.000000]
```

启动海龟仿真器节点

```
hcx@hcx-vpc:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
```

启动海龟控制节点

3. 第一个ROS例程——小海龟仿真分析



使用rqt_graph可视化工具查看系统中运行的计算图

3. 第一个ROS例程——小海龟仿真分析

古月居 GYH.AI

查看话题列表

```
$ rosnode list
```

发布话题消息

```
$ rostopic pub -r 10 /turtle1/cmd_vel geometry_msgs/Twist "linear:
    x: 1.0
    y: 0.0
    z: 0.0
  angular:
    x: 0.0
    y: 0.0
    z: 0.0"
```

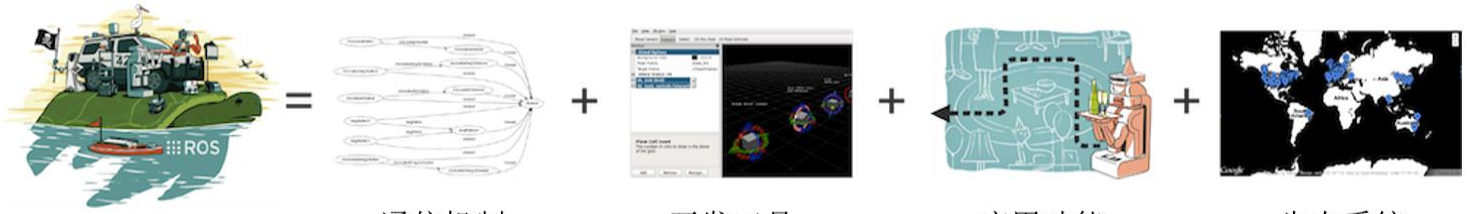
发布服务请求

```
$ rosservice call /spawn "x: 5.0
    y: 5.0
    theta: 0.0
    name: 'turtle2'"
```

本讲小结

古月居 GYH.AI

ROS定义与组成



ROS核心概念与通信机制

节点、节点管理器、话题、消息、服务、参数服务器、
功能包、功能包清单、元功能包

第一个ROS例程

ROS常用命令行的使用方法
（rqt_graph、rostopic、rosservice、rosbag …）

本讲作业

1. 运行海龟仿真器，并使用命令行工具新产生2只海龟；

2. 查看3只海龟的计算图结构；

3. 使用命令行工具控制一只海龟做圆周运动，并显示位置曲线；

扩展阅读

古月居 GYH.AI

- 古月·ROS入门21讲
https://www.bilibili.com/video/av59458869

- A Gentle Introduction to ROS
http://wiki.ros.org/Books/AGentleIntroductiontoROS_Chinese

- ROS Introduction
http://wiki.ros.org/ROS/Introduction

- 知乎 | 如何学习ROS?
https://mp.weixin.qq.com/s/Yuku2YGlDFKnFzLki3f7Wg

- "Powering the world’s robots" 的ROS是什么?
https://mp.weixin.qq.com/s/f9QZLfMWD3TbxRH85xAqXA

Thank You

怕什么真理无穷，进一寸有一寸的欢喜

更多精彩，欢迎关注

古月居

古月春旭