Robot Operating System Tutorial
ROS 基础

符国和

Task and Objective

- ROS 核心框架
- 命令行工具
- ROS package
- Catkin Workspace

What is ROS?

- 分布式的进程框架，属次级操作系统

底层：硬件抽象描述、底层驱动程序管理，进程间消息传递和程序包管理
顶层：开发者提供各种软件功能包

- ROS = 通信管道 + 工具 + 功能库+ 生态系统.

ROS History

目的：

- 复用性
- 模块化

发展经历

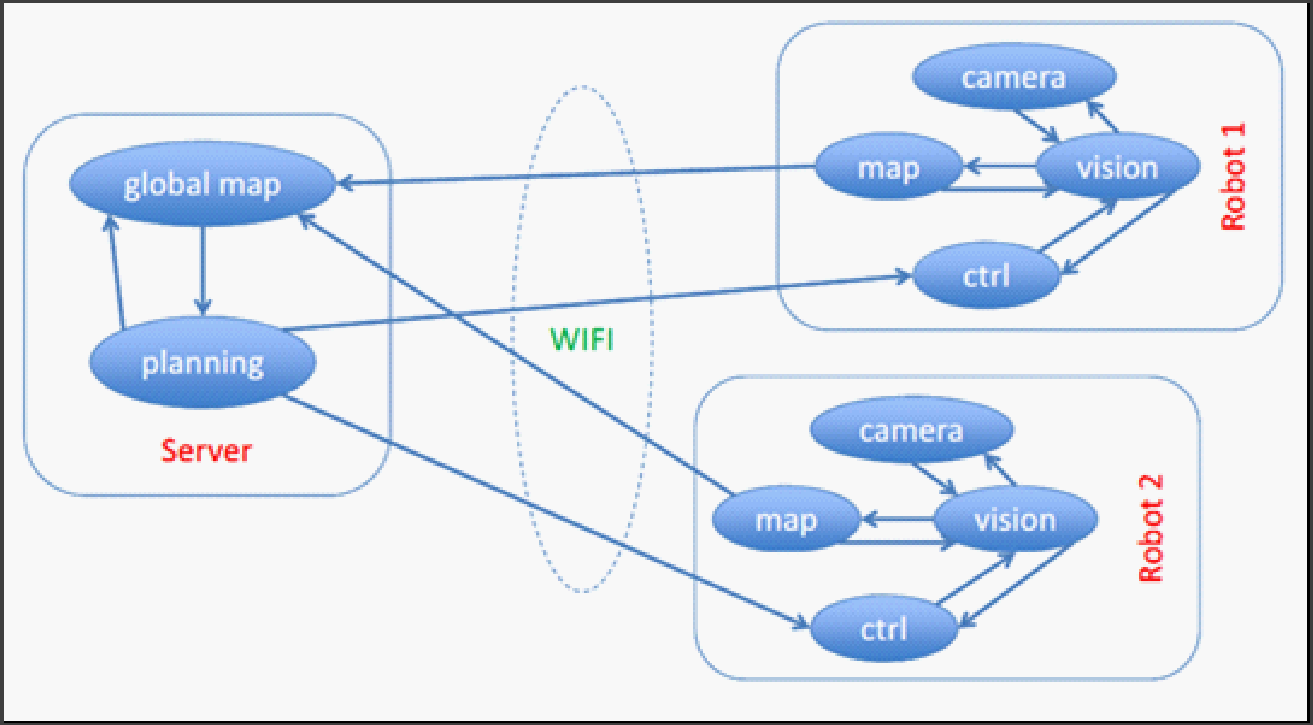
- 最早起源于2007年
- 2009年 ROS0.4
- 2010年Willow Garage正式发布ROS1.0
- 2008-2013 由Willow Garage维护管理
- 2013年-至今 由Open Source Robotics Foundation维护管理
- ROS2.0 ？

What is ROS?

KINETIC KAME
ROS
JADE TURTLE
ROS
INDIGO IGLOO
ROS
HYDRO MEDUSA
ROS
GROOVY GALAPAGOS
ROS FUERTE
ELECTRIC EMYS
ROS
DIAMONDBACK LAUNCH!
ROS
C TURTLE
Box Turtle

ROS 特点

- 点对点的设计



ROS 特点

- 不依赖编程语言

  C++

  Python

  Lisp

  Java

ROS 特点

- 精简与集成

封装：复杂重复使用的驱动和算法

模块化：单独编译

ROS 特点

测试方便

1. 使用模拟器替代底层硬件模块，独立测试顶层部分
2. 按时间戳回放记录的传感器数据和消息数据

ROS 特点

- 丰富的工具包

Gazebo

Rviz

rxplot

rxgraph

qt

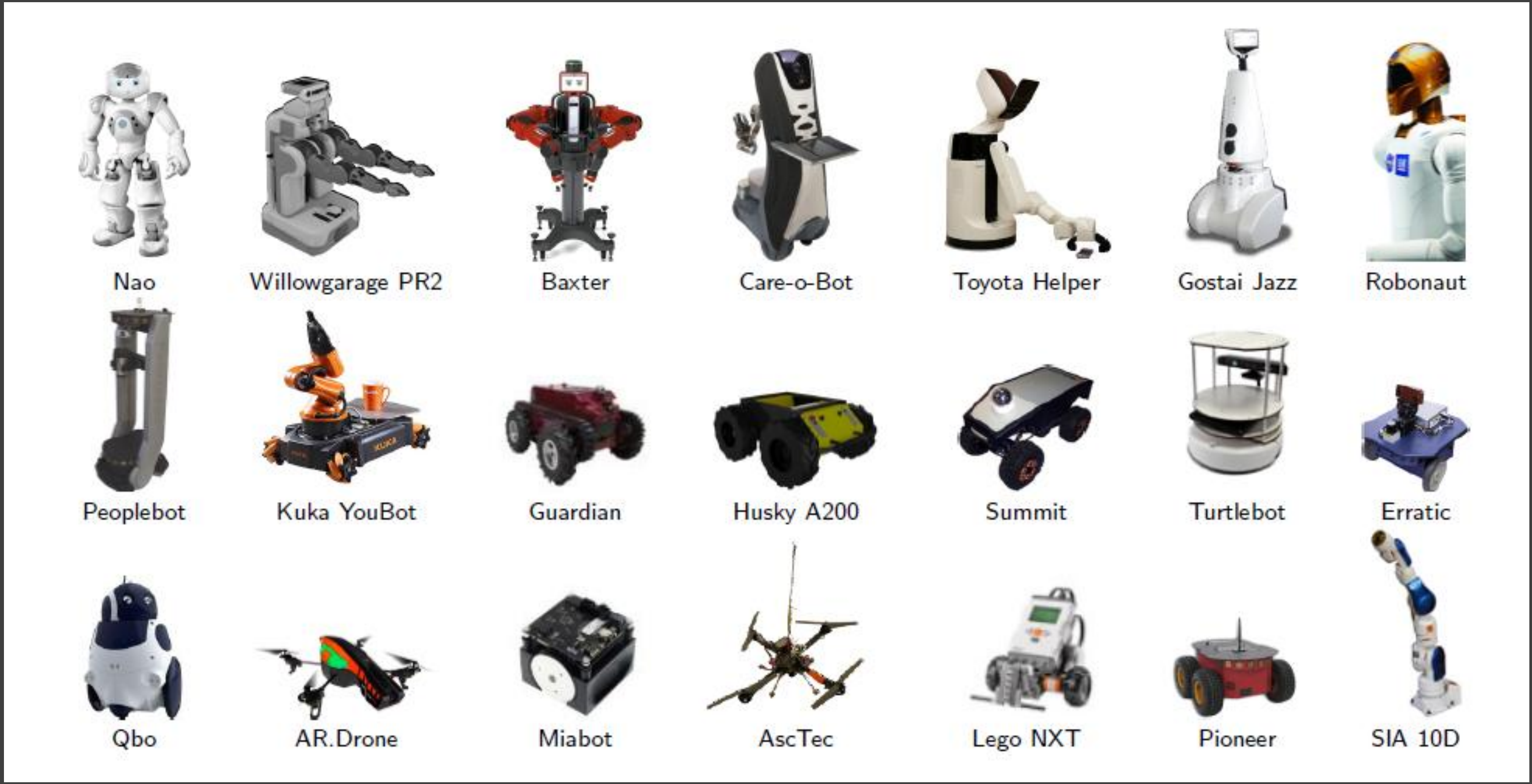
…

Supported operating systems

- 支持最好的操作系统
  - Ubuntu (14.04 LTS + ROS Kinetic)
- 实验性的操作系统

Arch	Mac OS X
Debian	OpenSuse
Fedora	Windows
Gentoo	

Supported robots



A lot more on http://www.ros.org/wiki/Robots

Sensors

- 1D/2D/3D range finders
  - ➢红外测距
  - ➢Hokuyo 、Sick激光雷达
  - ➢Microsoft Kinect
  - ➢Asus Xtion

Sensors

- 1D/2D/3D range finders
- Cameras
  - ➢RGB、RGB-D
  - ➢单目、双目

Sensors

- 1D/2D/3D range finders
- Cameras
- Force/torque/touch sensors
- Pose estimation (IMU/GPS)
- RFID
- Sensor/actuator interfaces
- And many more. . .

Installation - ROS (Indigo) on Ubuntu 14.04

Setup sources.list

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Setup keys

```
$ sudo apt-key adv --keyserver hkp://pool.sks-keyservers.net --recv-key 0xB01FA116
```

Install ROS Desktop-Full, and standalone tools

```
$ sudo apt-get update
$ sudo apt-get install ros-indigo-desktop-full
$ sudo rosdep init
$ rosdep update
```

Setup environment (shell)

```
$ echo \source /opt/ros/indigo/setup.bash" >> /.bashrc
$. /.bashrc
```

Installation - ROS (Indigo) on Ubuntu 14.04 (Trusty)

http://www.aicrobo.com/ubuntu_for_ros.html

Getting started

http://wiki.ros.org/

ROS Concepts

- 几个重要概念：节点（node）、 节点管理器（Master）、主题（topic）、服务（service）、包（package）、堆（stack）、消息（message）…

- 节点（Node）：
  1.每个进程称之为节点（node）
  2.一个机器人有多个节点
  3.所有的节点（node）由节点管理器（Master）管理

ROS Concepts　Node

- 节点（Node）：

- control robot wheel motors
- acquire data from laser scanner
- acquire images from camera
- perform localisation
- perform path planning
- provide graphical visualisation of the system

ROS Master

- 节点管理器（Master）：

  1.是ROS 的核心节点，称为roscore

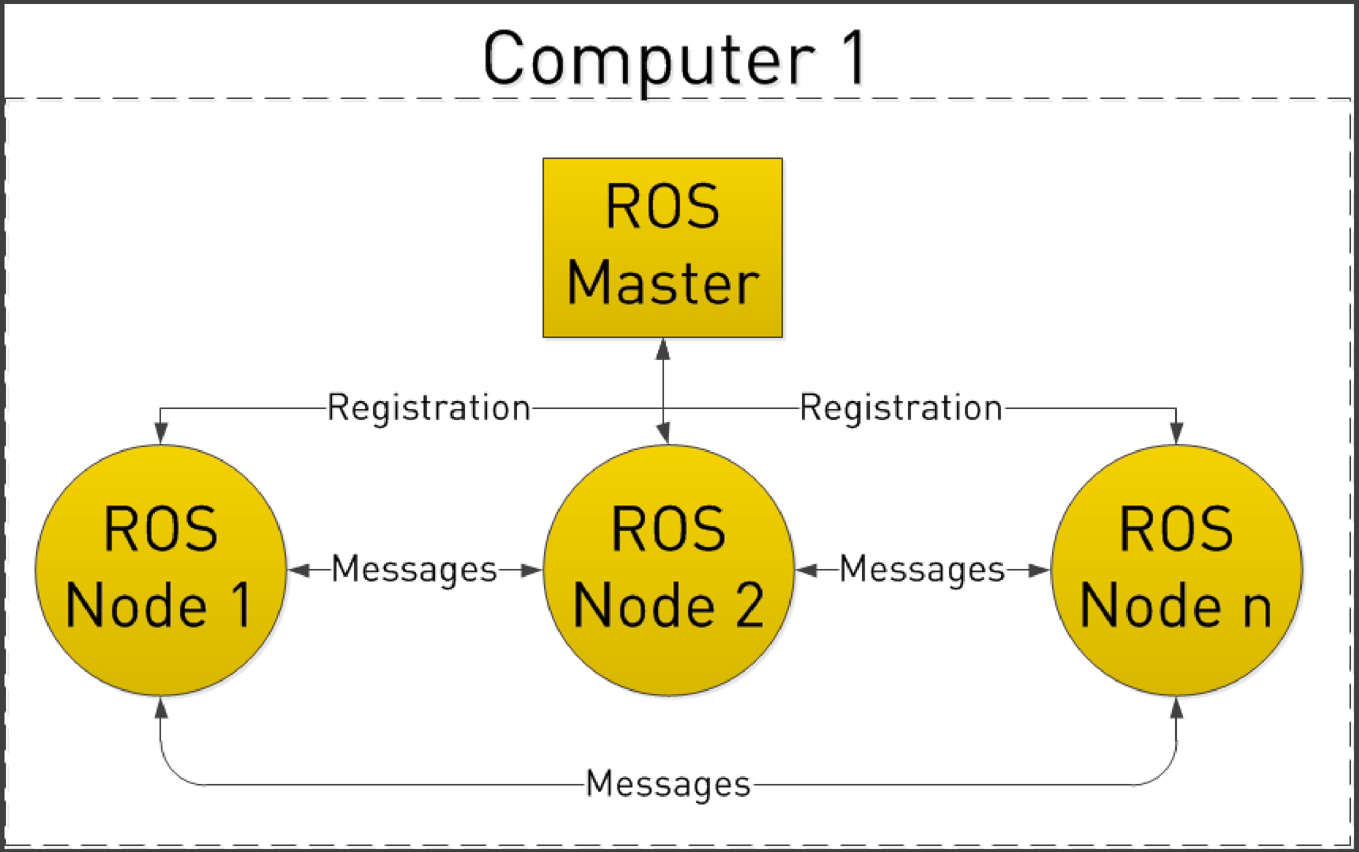
  2. 用于保存节点话题与服务的注册信息和查找表

  3. 运行方法：

```
$ roscore
```

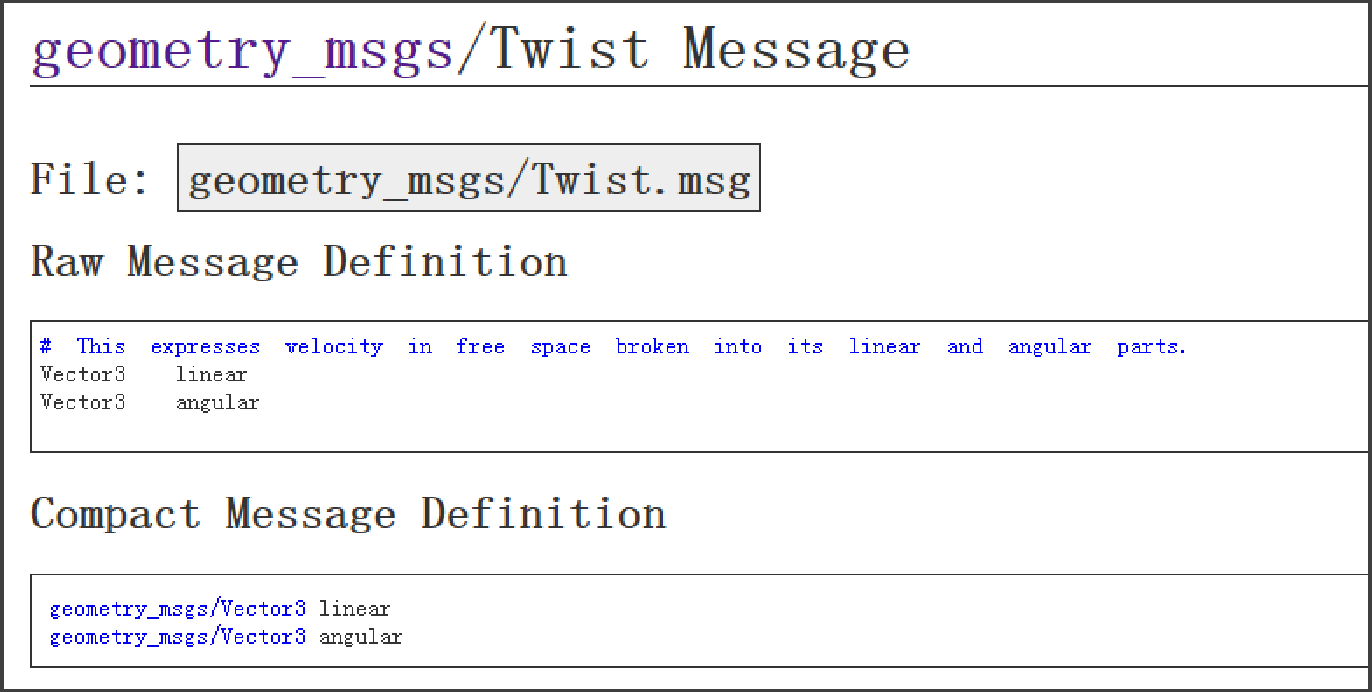
ROS Message

- 消息message：
  1.节点之间是通过传送消息进行通讯的



ROS Message

- 消息message：
  2.每一个消息都是一个数据结构



geometry_msgs/Twist Message

File: geometry_msgs/Twist.msg

Raw Message Definition

```
# This expresses velocity in free space broken into its linear and angular parts.
Vector3  linear
Vector3  angular
```

Compact Message Definition

```
geometry_msgs/Vector3 linear
geometry_msgs/Vector3 angular
```

ROS Message

- 消息message：

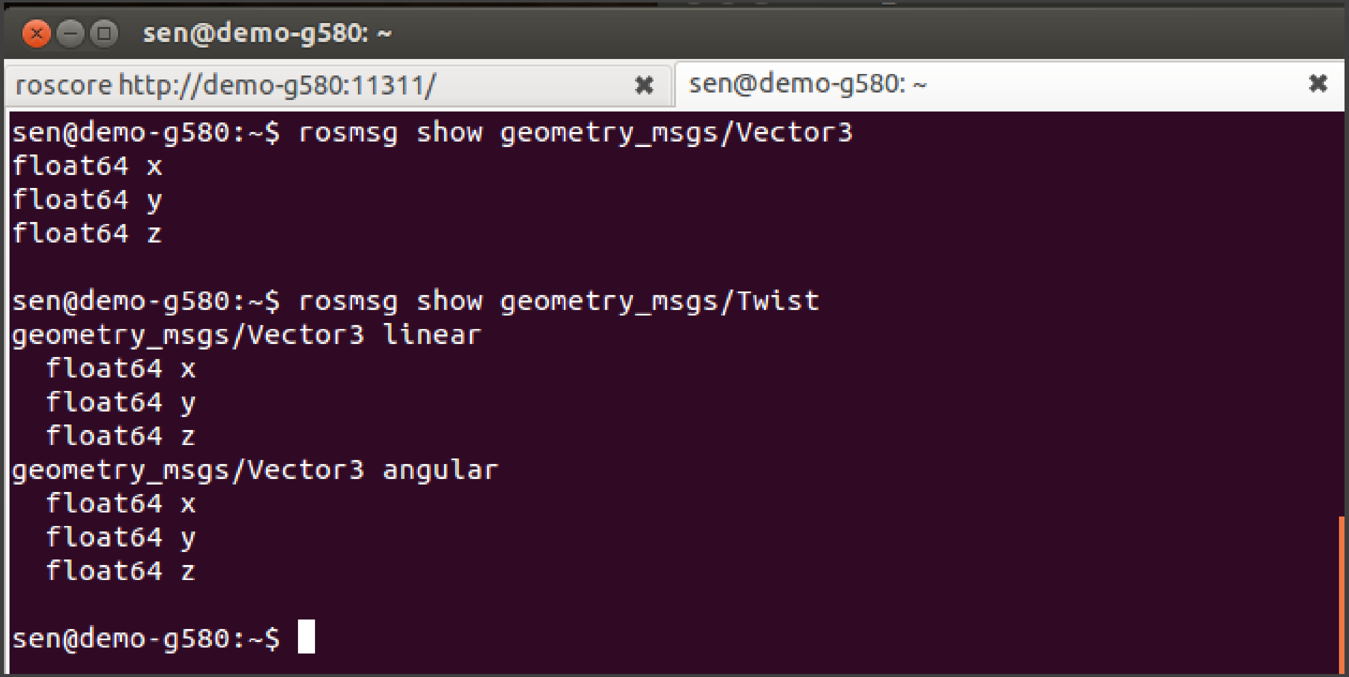
  3.支持标准的数据类型：
  - int8， 16， 32， 64
  - float32，64
  - string
  - time
  - duration
  - array[]
  - 更多信息, go to http://wiki.ros.org/msg

Messages-more ROS command line goodies

- Message over Topics

```
$ rosmsg list
$ rosmsg show geomemtry_msgs/Vector3
$ rosmsg show geomemtry_msgs/Twist
```

- Vector3.msg and Twist.msg from package geometry_msgs



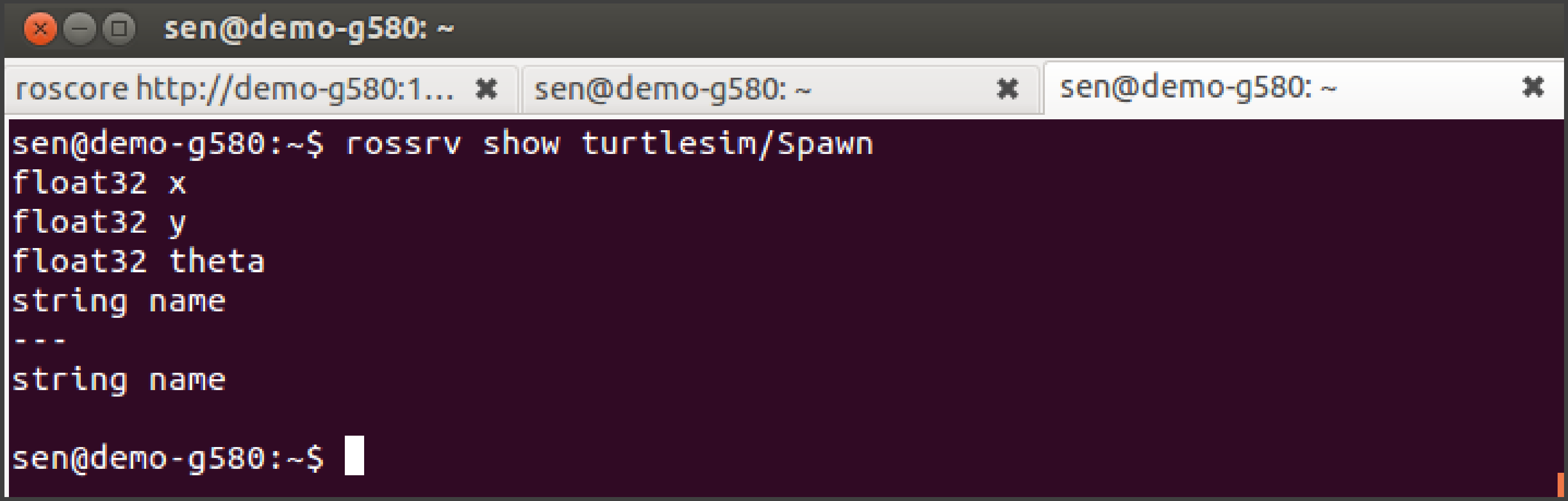
```
sen@demo-g580:~$ rosmsg show geometry_msgs/Vector3
float64 x
float64 y
float64 z

sen@demo-g580:~$ rosmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z

sen@demo-g580:~$ 
```

Messages-more ROS command line goodies

- Message over Services

```
$ rossrv list
$ rossrv   show turtlesim/Spawn
```

- Spawn.srv from package geometry



```
sen@demo-g580:~$ rossrv show turtlesim/Spawn
float32 x
float32 y
float32 theta
string name
---
string name

sen@demo-g580:~$
```

ROS Topic

- 主题（topic）：

  1.消息以一种**publish/subscribe**的方式传递

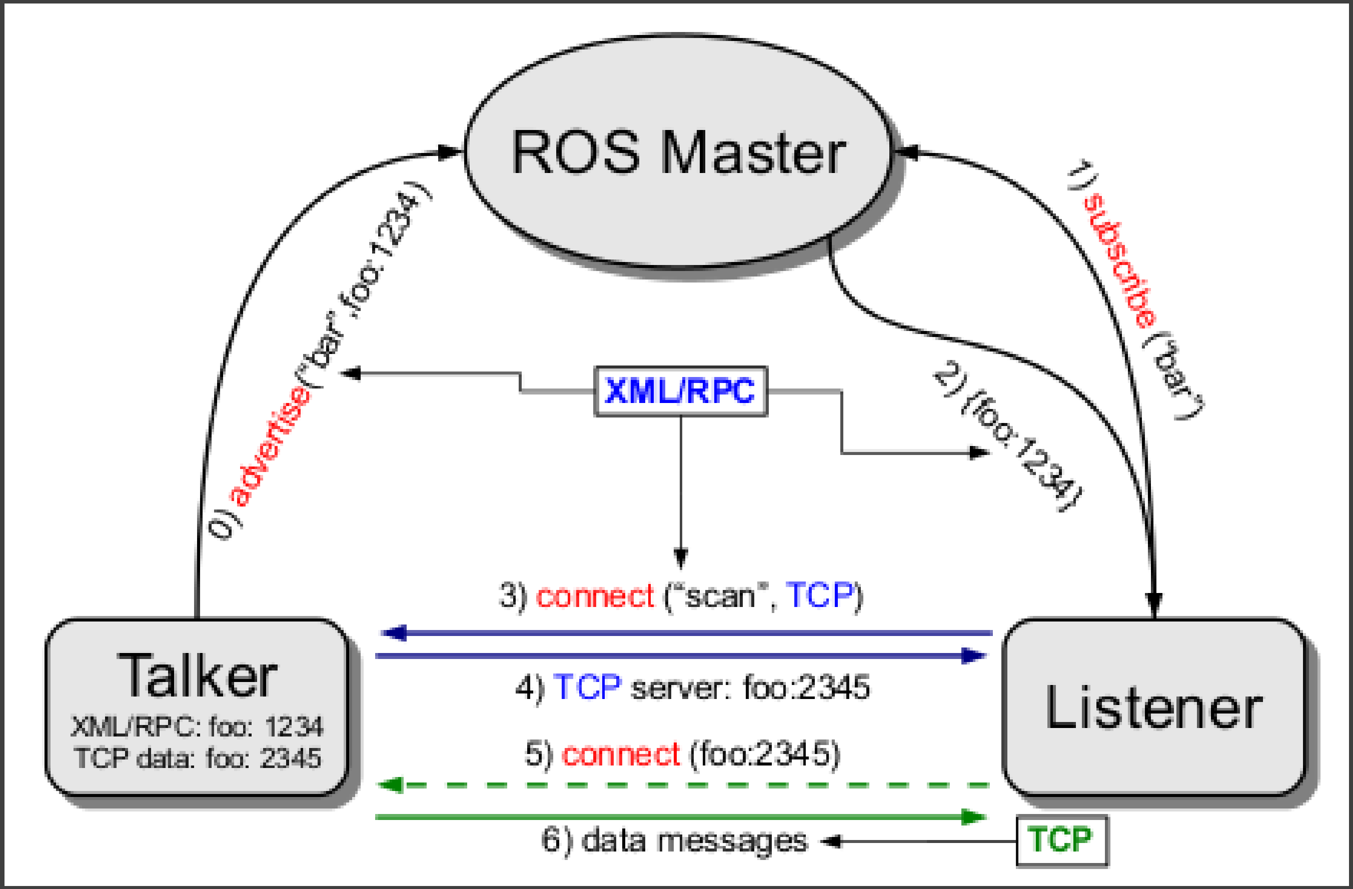
  2.节点可以在给定的主题中发布/订阅消息

  3.一个节点可以订阅/发布多个不同的主题

  4. 允许多个节点订阅/发布同一个主题

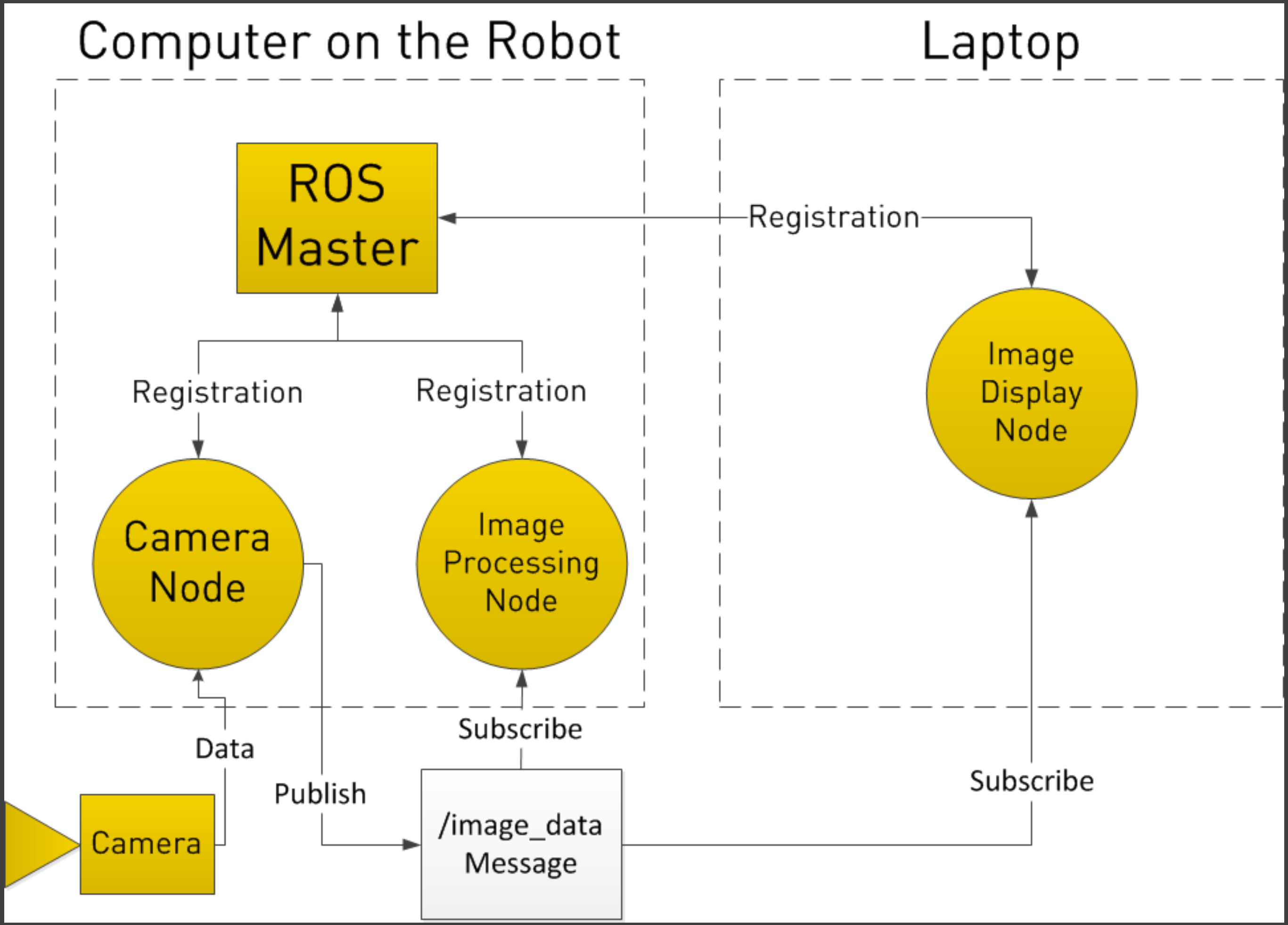
  5.订阅节点和发布节点并不知道相互之间的存在

Topics -diagrammatic representation



Xml/RPC: http://en.wikipedia.org/wiki/XML-RPC

Topics -diagrammatic representation



ROS service

- 服务（service ）：

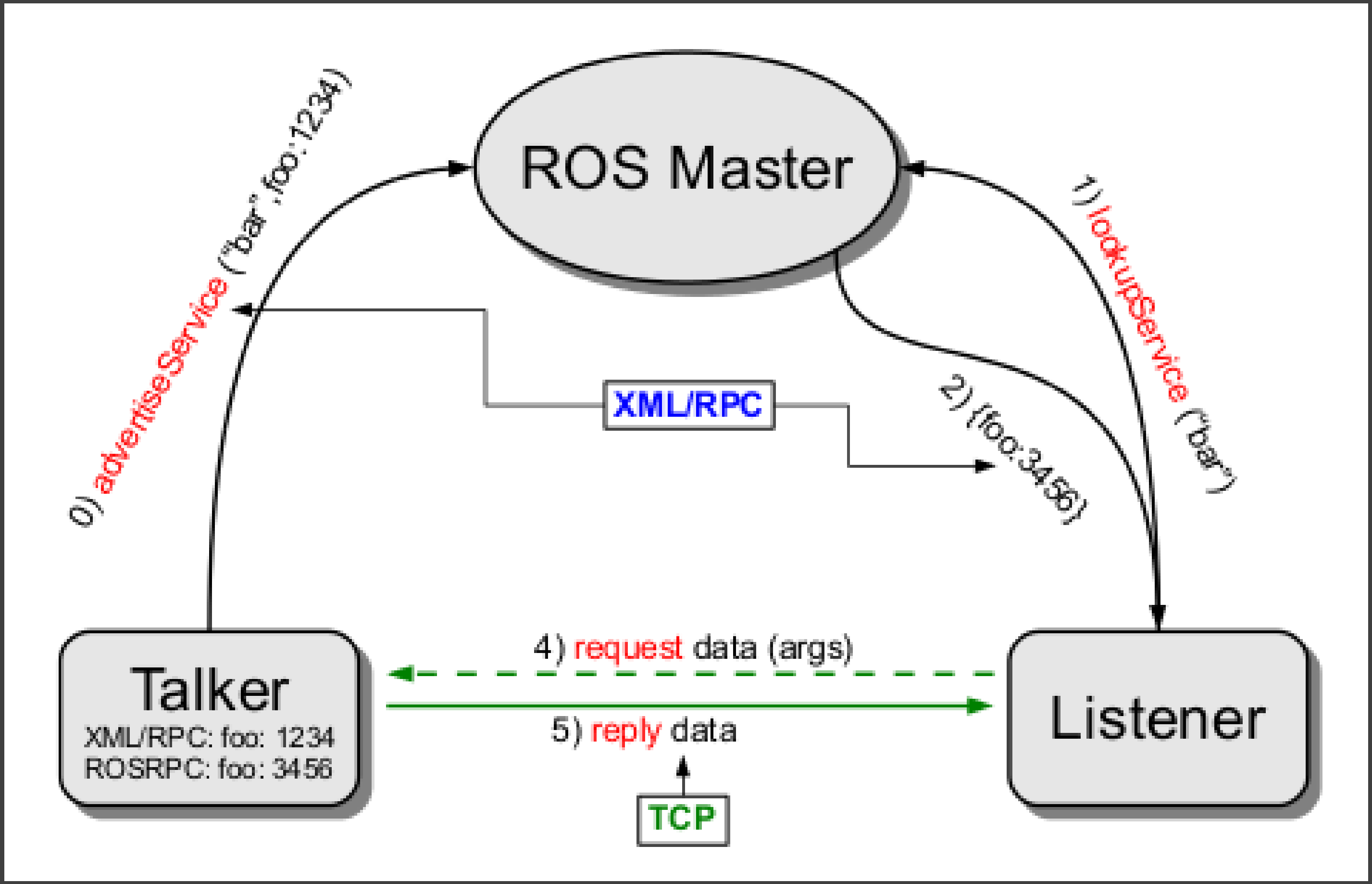
  1.消息以一种request/reply的方式传递

  2. 节点之间发送请求和接受应答

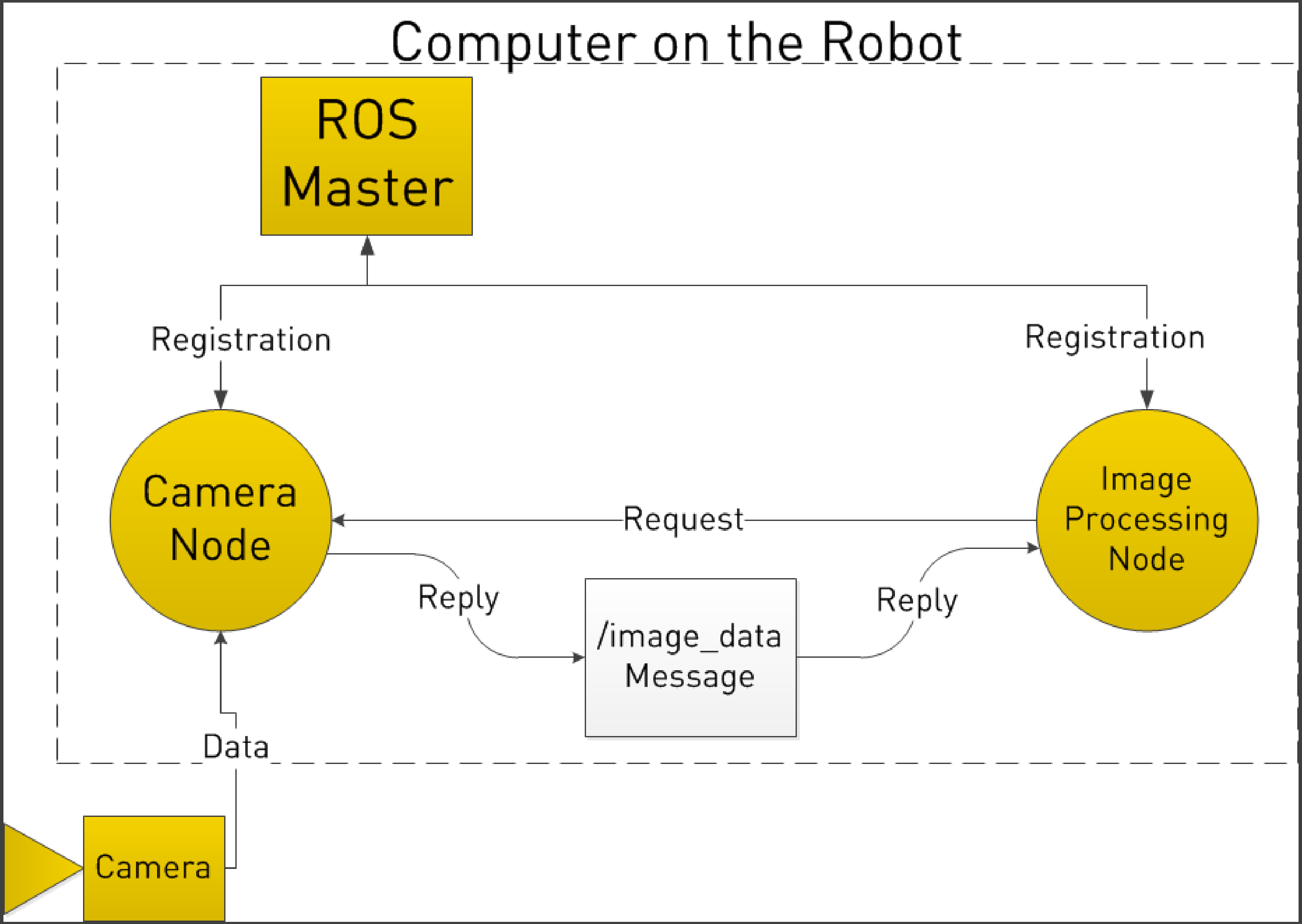
  3. 一对一模式：一个请求，一个响应

  4. 远程过程调用（ remote procedure call, rpc）

Services - diagrammatic representation



Services - diagrammatic representation



ROS Concepts

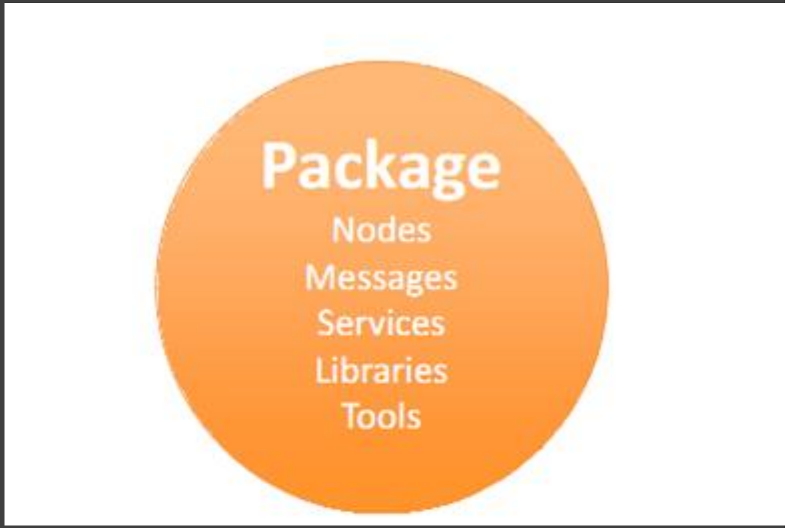
Package

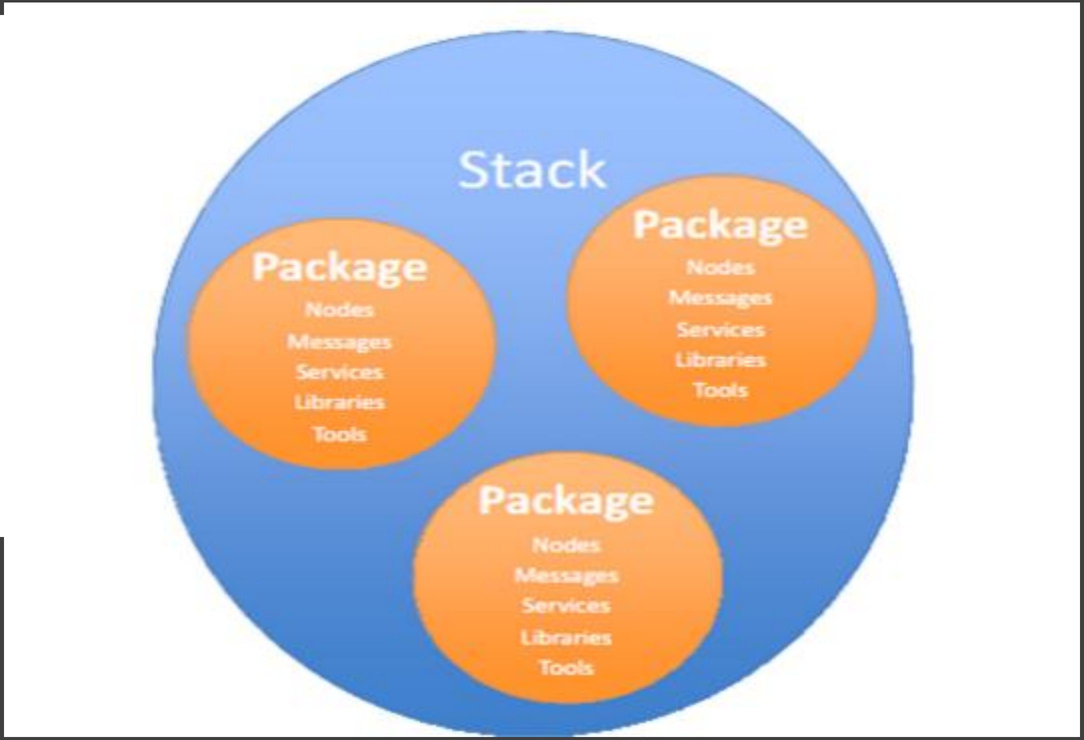
- Packages are the software organization unit of ROS code.
- Each package can contain libraries, configuration file，executables, scripts.
- Manifest: description (metadata) of a package, whose main role is to define dependencies between packages (package.xml)

Meta-packages(stacks)

- Collection of packages forming a higher level library
- Previously called stacks. The concept of stacks was removed with catkin to simplify the growing code base and to support better distribution of packages.

ROS Concepts





ROS Concepts– catkin workspace

```
workspace_folder/                -- WORKSPACE
  build/                 -- BUILD SPACE CMake is invoked to build the catkin packages in the source space
  devel/                 -- DEVEL SPACE where built targets are placed prior to being installed
  src/                             -- SOURCE SPACE
    CMakeLists.txt                 -- 'Toplevel' CMake file, provided by catkin
    package_1/
      CMakeLists.txt               -- CMakeLists.txt file for package_1
      package.xml                  -- Package manifest for package_1
    ...
    package_n/
      CMakeLists.txt               -- CMakeLists.txt file for package_n
      package.xml                  -- Package manifest for package_n
    meta_package/                  --collections of packages
        sub_package_1/
             CMakeLists.txt        -- CMakeLists.txt file for sub_package_1
              package.xml          -- Package manifest for sub_package_1
        …
        sub_package_n/
              CMakeLists.txt       -- CMakeLists.txt file for sub_package_n
              package.xml          -- Package manifest for sub_package_n
        meta_package/
              package.xml          -- Package manifest indicating the meta_package
```

ROS Concepts– create package

catkin_create_pkg <package_name> [depend1] [depend2] [depend3]

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg myPkg std_msgs rospy roscpp
```

ROS Concepts– Package Example

Hypothetical package myPkg/

- **CMakeLists.txt:** CMake build settings for package myPkg
- **package.xml:** metadata and dependencies required by package
- **mainpage.dox:** doc information of package myPkg
- **include/myPkg:** c++ header files
- **src/:** source code directory
- **launch/:** where launch files are stored (if needed)
- **msg/:** message (.msg) types
- **srv/:** service (.srv) types
- **scripts/:** executable scripts

rosbash -ROS command line tools

- rospack: ROS package management tool

```
$ rospack list
$ rospack find myPkg
$ rospack depends myPkg
$ rospack profile
```

- roscd: change directory command for ROS

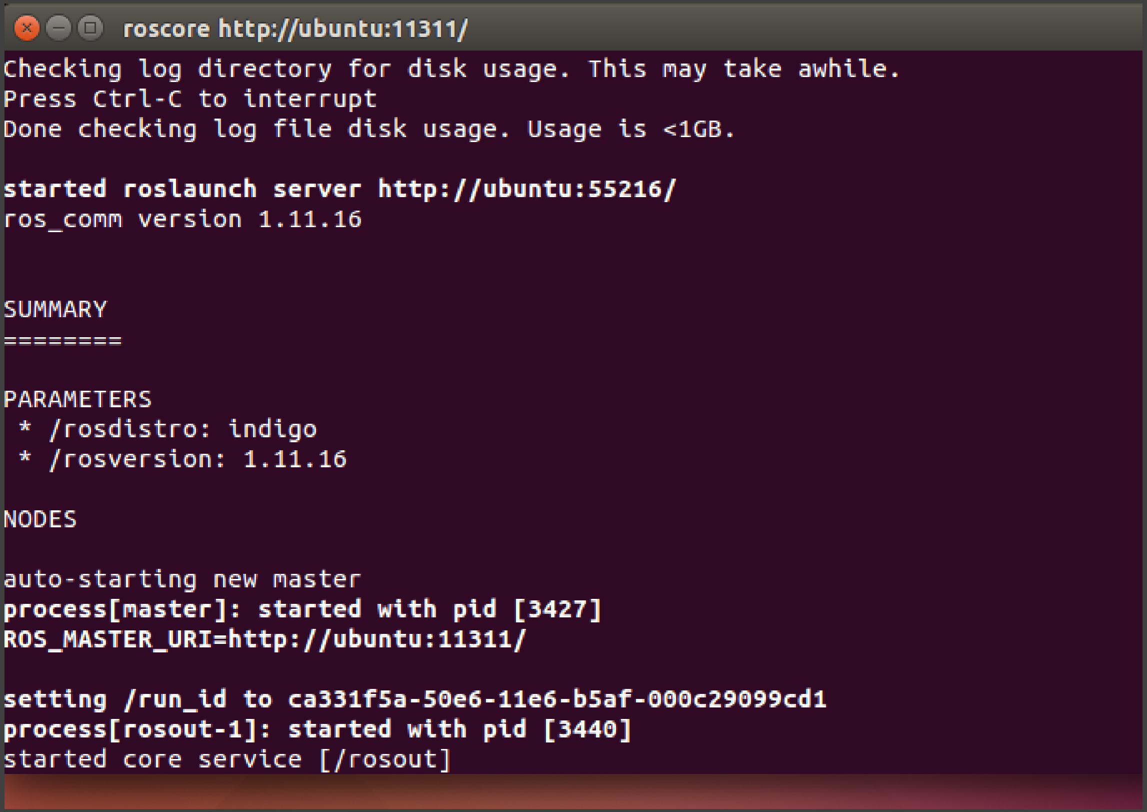
```
$ roscd
$ roscd myPkg
$ ls (standard linux shell command)
```

roscore

roscore

roscore 是ROS 节点程序启动的必要条件

在启动一个节点之前必须启动roscore



roscore http://ubuntu:11311/

```
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu:55216/
ros_comm version 1.11.16


SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.16

NODES

auto-starting new master
process[master]: started with pid [3427]
ROS_MASTER_URI=http://ubuntu:11311/

setting /run_id to ca331f5a-50e6-11e6-b5af-000c29099cd1
process[rosout-1]: started with pid [3440]
started core service [/rosout]
```

rosrun

- **rosrun** 可以运行**package**中的可执行文件，不需要知道可执行文件的位置

rosrun package executable

```
Example: rosrun cmd_vel_publisher cmd_vel_publisher_node
```

- 也可以 带参数服务 **parameters**

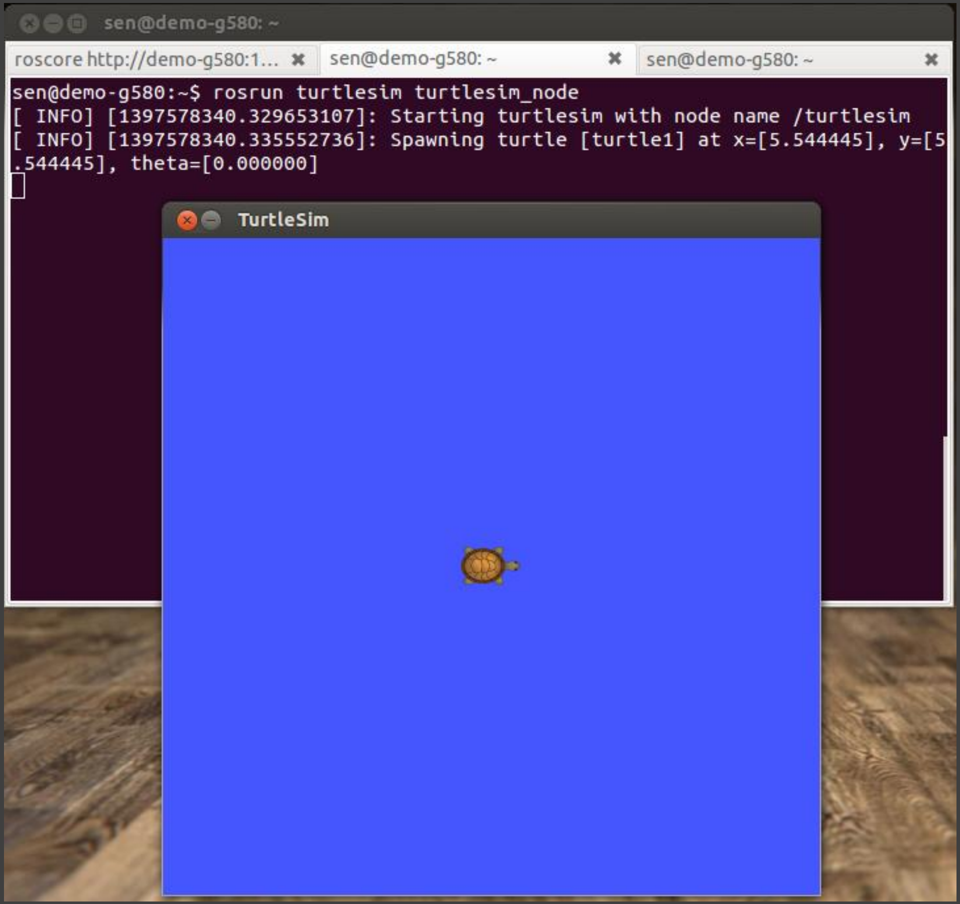
rosrun package node _parameter:=value

```
Example:
rosrun cmd_vel_publisher cmd_vel_publisher_node   _Max_Constant_Vel:=0.5
```

Practice with rosrun

- **rosrun with turtlesim_node**

```
$ rosrun turtlesim turtlesim_node
```



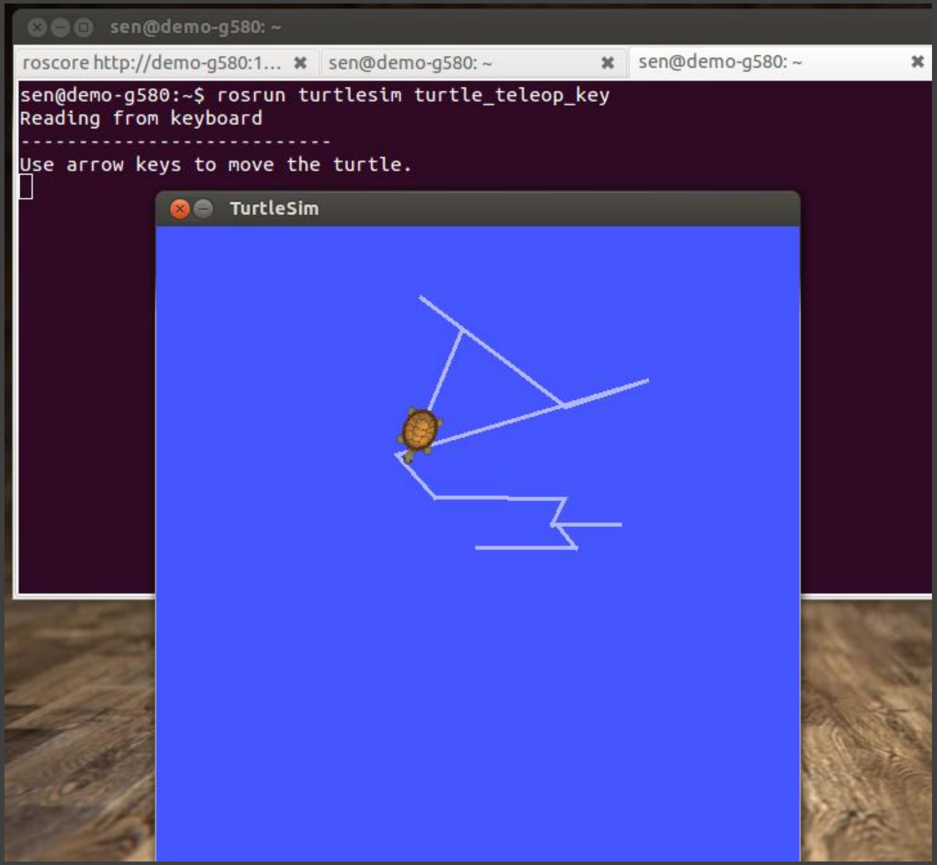
```
sen@demo-g580:~$ rosrun turtlesim turtlesim_node
[ INFO] [1397578340.329653107]: Starting turtlesim with node name /turtlesim
[ INFO] [1397578340.335552736]: Spawning turtle [turtle1] at x=[5.544445], y=[5
.544445], theta=[0.000000]
```

Practice with rosrun

- **rosrun with turtlesim_teleop_key**

  Using the arrow keys to drive the robot

```
$ rosrun turtlesim turtle_teleop_key
```



```
sen@demo-g580:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
```

rosnode

- **The current list of supported commands are**

rosnode kill	kill a running node
rosnode list	list active nodes
rosnode machine	list nodes running on a machines
rosnode ping	test connectivity to node
rosnode info	print information about node

rostopic

The current list of supported commands are

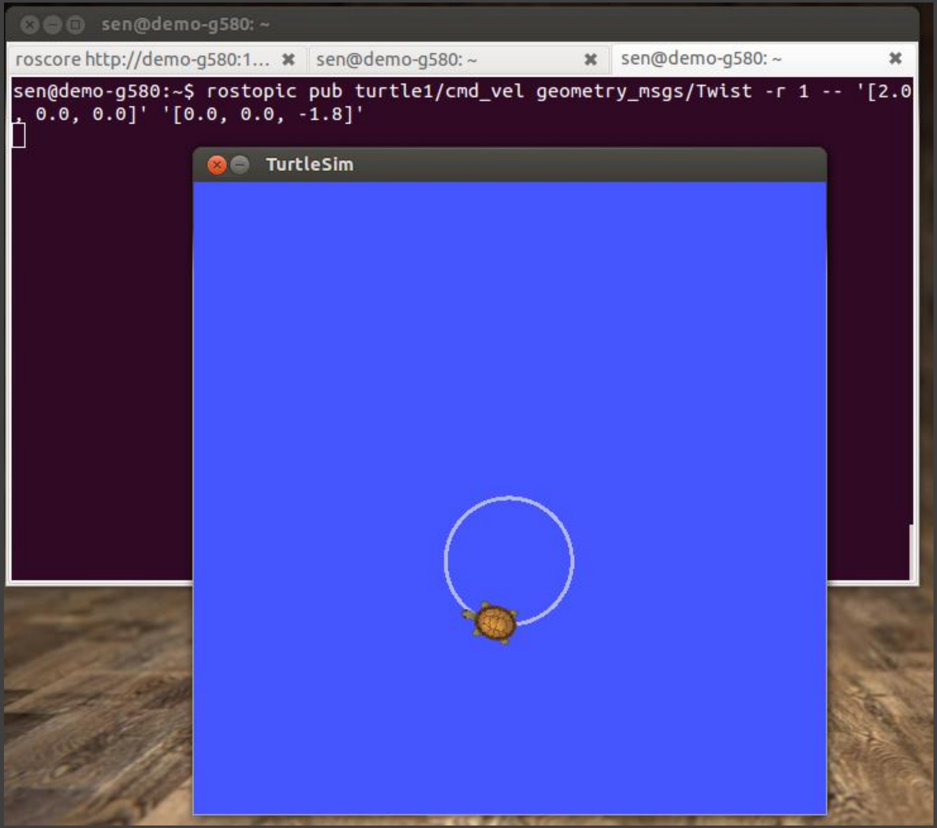
rostopic bw	display bandwidth used by topic
rostopic echo	print messages to screen
rostopic find	find topics by type
rostopic hz	display publishing rate of topic
rostopic info	print information about active topic
rostopic list	print informaion about active topics
rostopic pub	publish data to topic

rostopic

- **rostopic pub**

  $ rostopic pub [topic] [msg_type] [arg]

```
rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 --  '[2.0, 0.0, 0.0]'
'[0.0, 0.0, 1.8]'
```



```
sen@demo-g580:~$ rostopic pub turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0
, 0.0, 0.0]' '[0.0, 0.0, -1.8]'
```

ROS Development Procedures

- 创建一个新的catkin 工作空间
- 创建一个新的ROS package
- 编码
- 修改 make文件
- 编译包

catkin Workspace

- A workspace is a directory in which one or more catkin packages can be built.
- A basic workspace looks like this:

```
workspace_folder/           -- WORKSPACE
  build/                    -- BUILD SPACE CMake is invoked to build the catkin packages in the source space
  devel/                    -- DEVEL SPACE where built targets are placed prior to being installed
  src/                      -- SOURCE SPACE
    CMakeLists.txt          -- 'Toplevel' CMake file, provided by catkin
    package_1/
      CMakeLists.txt        -- CMakeLists.txt file for package_1
      package.xml           -- Package manifest for package_1
    ...
    package_n/
      CMakeLists.txt        -- CMakeLists.txt file for package_n
      package.xml           -- Package manifest for package_n
    meta_package/           --collections of packages
        sub_package_1/
            CMakeLists.txt  -- CMakeLists.txt file for sub_package_1
            package.xml     -- Package manifest for sub_package_1
        ...
        sub_package_n/
            CMakeLists.txt  -- CMakeLists.txt file for sub_package_n
            package.xml     -- Package manifest for sub_package_n
        meta_package/
            package.xml     -- Package manifest indicating the meta_package
```

Creating a catkin Workspace

- http://wiki.ros.org/catkin/Tutorials/create_a_workspace

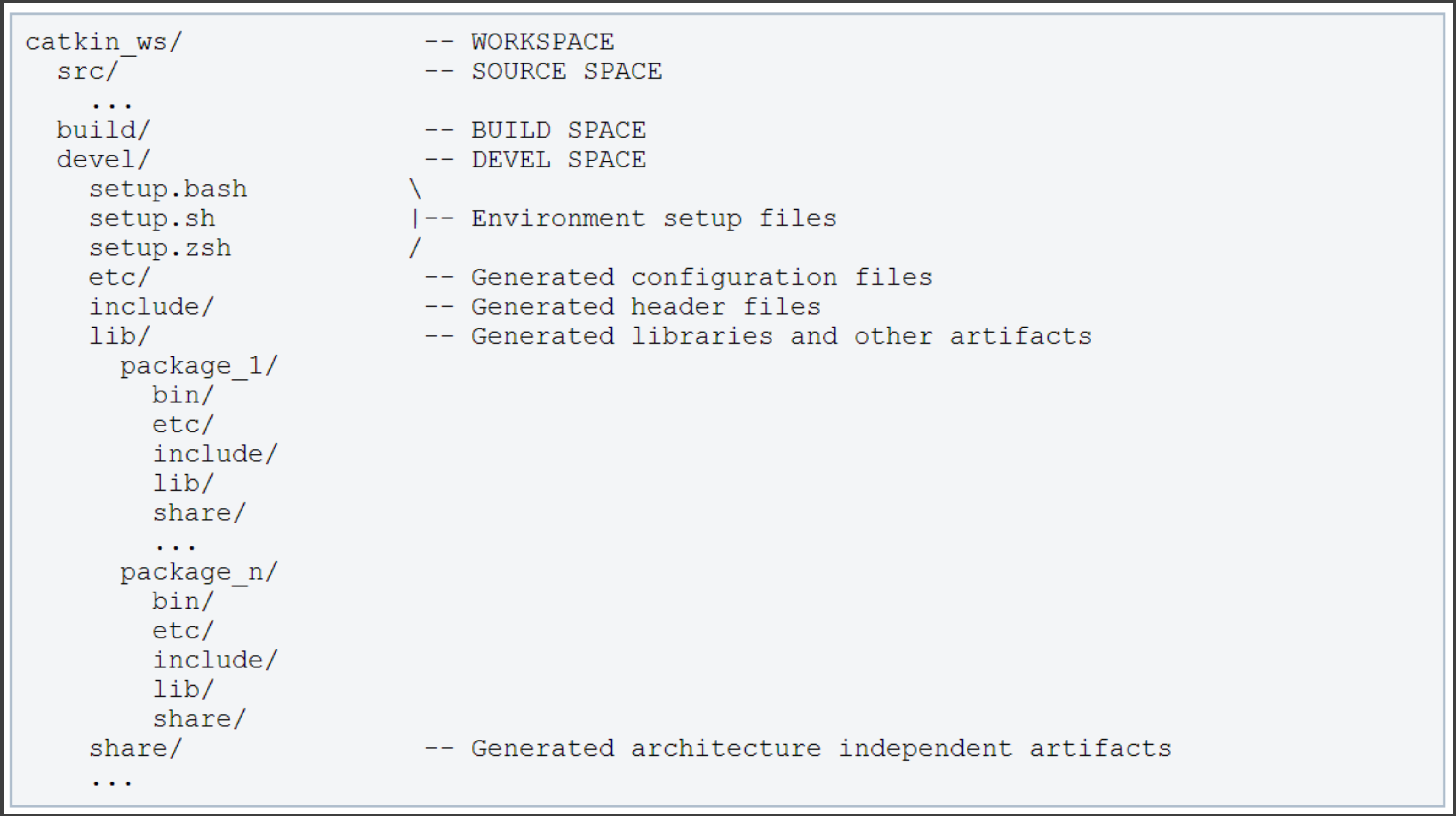
```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
```

- 初始化之后，工作空间 将会生成一个 CMakeLists.txt
- **catkin_make** 编译Workspace中的所有包

```
cd ~/catkin_ws
catkin_make
```

Resulting catkin Workspace

- 所有的构建文件和可执行文件都放在 **devel** 文件夹中



```
catkin_ws/                 -- WORKSPACE
  src/                     -- SOURCE SPACE
    ...
  build/                   -- BUILD SPACE
  devel/                   -- DEVEL SPACE
    setup.bash            \
    setup.sh              |-- Environment setup files
    setup.zsh             /
    etc/                   -- Generated configuration files
    include/               -- Generated header files
    lib/                   -- Generated libraries and other artifacts
      package_1/
        bin/
        etc/
        include/
        lib/
        share/
        ...
      package_n/
        bin/
        etc/
        include/
        lib/
        share/
    share/                 -- Generated architecture independent artifacts
    ...
```

Creating a ROS Package

- http://wiki.ros.org/catkin/Tutorials/CreatingPackage
- Change to the source directory of the workspace

```
$cd ~/catkin_ws/src
```

- **catkin_create_pkg** creates a new package

```
$ catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
```

- Example:

```
$ catkin_create_pkg test_package std_msgs rospy roscpp
```

ROS filesystem – Package Example

Hypothetical package myPkg/

- **CMakeLists.txt**: CMake build settings for package myPkg
- **package.xml:** metadata and dependencies required by package
- **mainpage.dox:** doc information of package myPkg
- **include/myPkg**: c++ header files
- **src**/: source code directory
- **launch**/: where launch files are stored (if needed)
- **msg**/: message (.msg) types
- **srv**/: service (.srv) types
- **scripts**/: executable scripts

The CMakeLists.txt

- cmake_minimum_required #Required CMake Version
- project() #Package Name
- find_package() #Find other CMake/Catkin packages needed for build
- add_message_files(), add_service_files(), add_action_files()

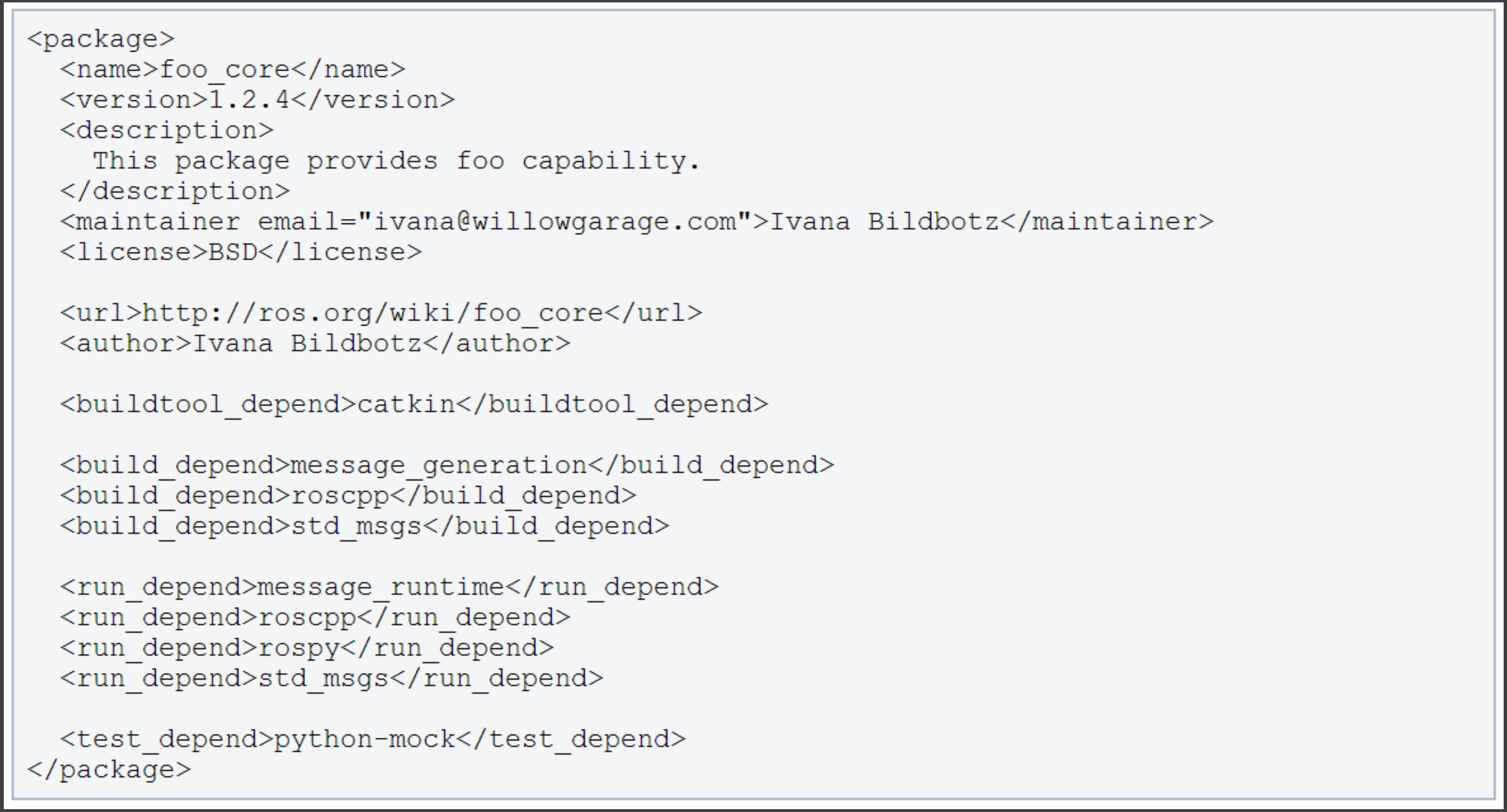
  #Message/Service/Action Generators
- generate_messages() #Invoke message/service/action generation
- catkin_package() #Specify package build info export
- add_library()/add_executable()/target_link_libraries() #Libraries/Executables to build
- catkin_add_gtest() #Tests to build
- install() #Install rules

The **package.xml**

- XML file that defines properties about the package such as:
  - the package name
  - version numbers
  - authors
  - dependencies on other catkin packages

The **package.xml**

- Example for a package manifest:



```
<package>
  <name>foo_core</name>
  <version>1.2.4</version>
  <description>
    This package provides foo capability.
  </description>
  <maintainer email="ivana@willowgarage.com">Ivana Bildbotz</maintainer>
  <license>BSD</license>

  <url>http://ros.org/wiki/foo_core</url>
  <author>Ivana Bildbotz</author>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>message_generation</build_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>

  <run_depend>message_runtime</run_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>

  <test_depend>python-mock</test_depend>
</package>
```

我的博客

http://blog.sina.com.cn/s/articlelist_3285404150_0_1.html

推荐几个初学者博客：

http://blog.sina.com.cn/s/articlelist_1712413141_7_1.html

http://www.guyuehome.com/

THANK YOU!