移动机器人开发技术（激光SLAM版）配套教学PPT

序　章

第1课 移动机器人的过去、现在及未来

机器人操作系统

第2课 初识ROS

第3课 ROS编程初步

第4课 机器人的坐标变换

第5课 机器人仿真环境

机器人硬件平台

第6课 TurtleBot3仿真环境实战

第7课 自主搭建机器人小车

机器人核心技术

感知	建图与定位	路径规划与导航
第08课 环境感知基础	第10课 机器人的移动控制	第13课 导航基础
第09课 感知数据融合	第11课 SLAM基础	第14课 ROS中的导航包
	第12课 SLAM实战	第15课 ROS导航实战

机器人应用实战

送餐	物流（专题讲座）	图书盘点（专题讲座）
1 送餐机器人结构设计	1 物流机器人结构设计	1 图书盘点机器人结构
2 送餐机器人环境搭建	2 物流机器人环境模拟	2 图书盘点机器人环境
3 送餐机器人建图	3 物流机器人关键技术	3 图书盘点机器人工作模式
4 送餐机器人导航	4 大规模多机器人调度	4 图书盘点中的视觉分析

移动机器人开发技术（激光SLAM版）配套教学PPT

第三课 ROS编程初步

北京郵電大學
Beijing University of Posts and Telecommunications

移动机器人与智能技术实验室编

宋桂岭 明安龙 2021.9
expsong@qq.com

第3课 ROS编程初步

北邮移动机器人与智能技术实验室 编

1 ROS开发过程回顾

2 ROS的框架结构

3 消息的订阅与发布

4 服务的请求与响应

5 动作的执行与应答

6 ROS的参数服务

第3课 ROS编程初步

北邮移动机器人与智能技术实验室 编

1 ROS开发过程回顾

1.1 ROS项目开发流程

①创建 → **②编码** → **③配置** → **④编译** → **⑤运行**

ROS1：
$catkin_create_pkg my_package [依赖项]

ROS2：
ros2 pkg create --build-type ament_cmake --node-name my_node my_package

注意文件夹结构，代码都位于src文件夹内

```
workspace_folder/
    src/
      package_1/
          CMakeLists.txt
          package.xml

      package_2/
          CMakeLists.txt
          package.xml
      ...
      package_n/
          CMakeLists.txt
          package.xml
```

CMakeLists.txt中配置运行和依赖
package.xml配置依赖

ROS1：
catkin_make

ROS2：
colcon build

ROS1：
先roscore
后rosrun

ROS2:
ros2 run

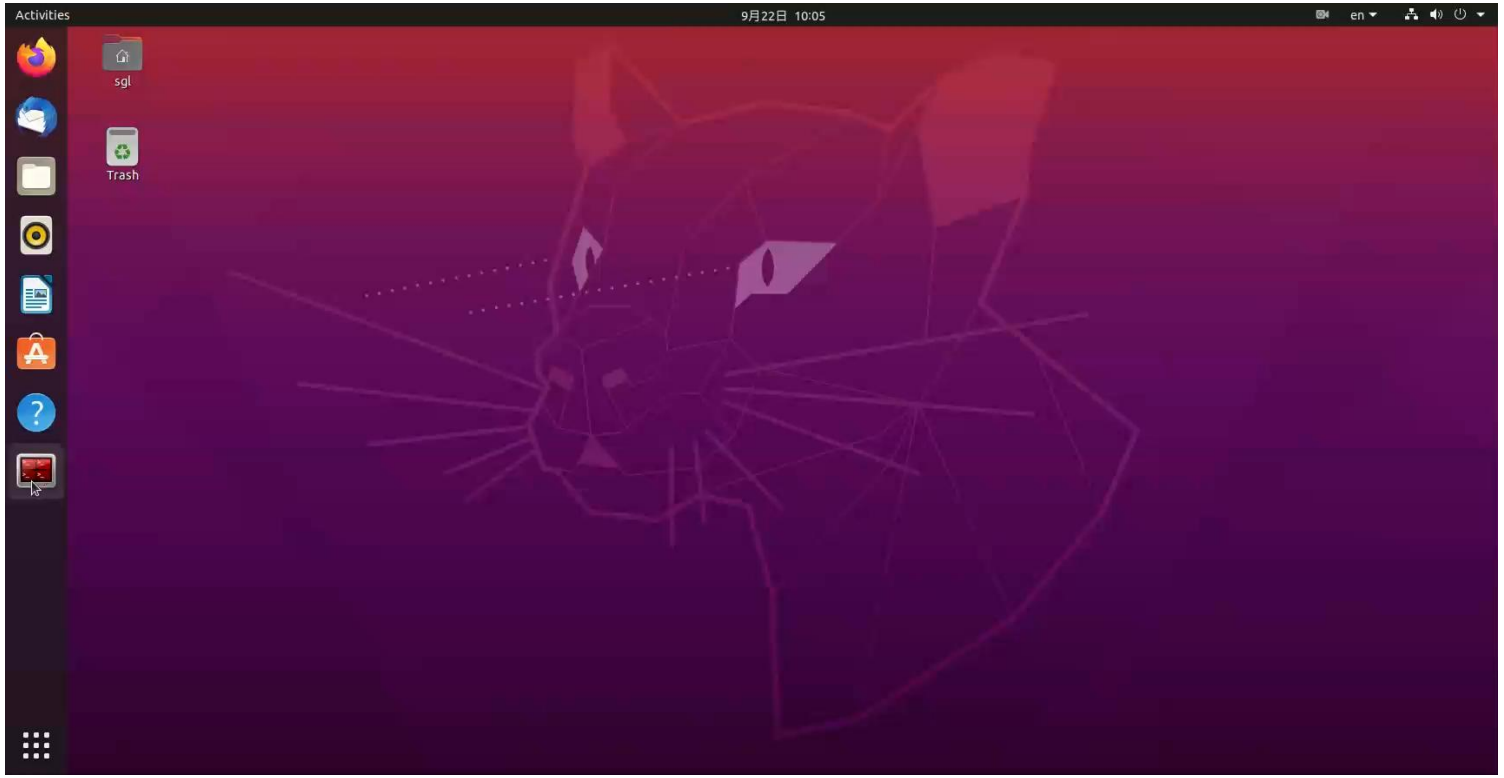
1.2 ROS常用命令行及工具

ROS1

rospack find [package_name]:
返回包所在位置路径
rospack depends[1] [package_name]:
返回包依赖的其他包
roscd <package-or-stack>[/subdir]:
切换当前目录
rosls <package-or-stack>[/subdir]:
当前目录文件列表

ROS2

ros2:在终端中输入ros2可以显示全部的ros2指令，可以发现roscd rosls等在ros2中取消
注意：ros环境切换：通过gedit ~/.bashrc,
修改引用：source /opt/ros/foxy/setup.bash



第3课 ROS编程初步

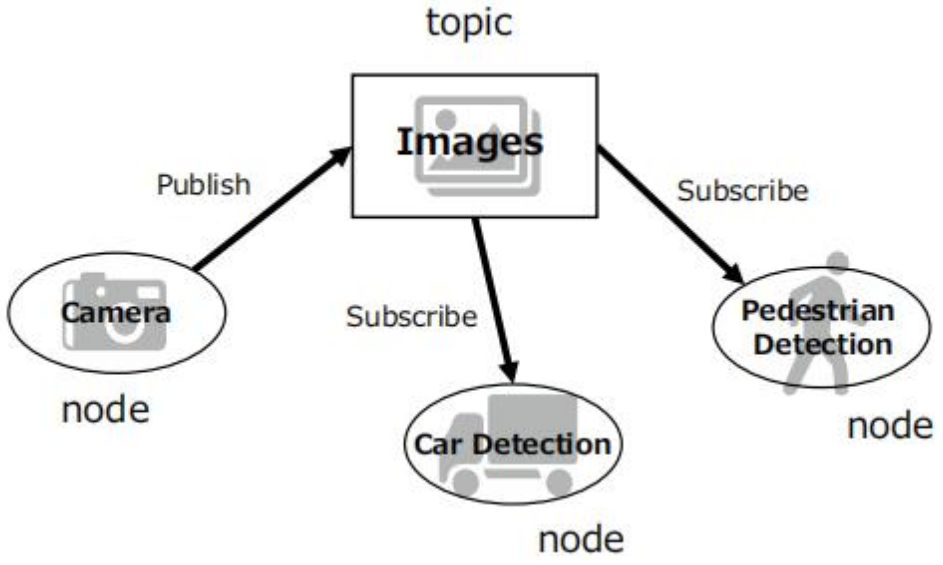
北邮移动机器人与智能技术实验室 编

2 ROS的框架结构

2.1 ROS计算图架构

- **节点Node**：计算单元，独立进程程序，用户开发
- **消息Message**：定义了节点间数据传递的格式规范
- **主题Topic:** 消息具有主题，从而一种消息结构可以被多个主题使用，节点间通过发布和订阅主题实现通信，用于周期性任务
- **服务Service:**用于请求/应答机制，用于单次任务
- **动作Action:**用于请求/应答机制，用于长时间任务的过程反馈，类比：长文件下载过程中的进度响应
- **录像Bags**：可以通过录像将ROS节点间消息进行存储和回访，从而降低传感器数据采集难度，加快程序算法开发和调试速度。

ROS的设计目标：
轻量化、接口友好、语言无关、易测试、易扩展



一种典型的计算图架构（发布者/订阅者模式）

相机传感器**节点**发布了图片消息**主题**,车辆检测和行人检测**节点订阅**该**主题**，各自完成独立任务。每个节点都是独立进程，完成单个或多个任务。

2.1 ROS1框架

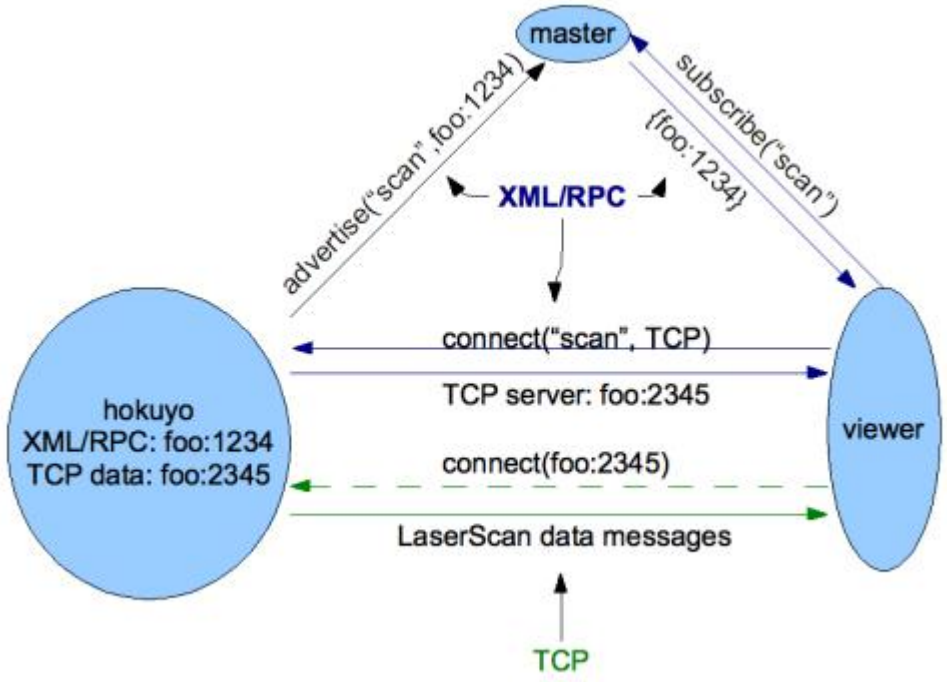
文档阅读：ROS/Introduction - ROS Wiki

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

作业：分享比较不同机器人开发框架，占平时成绩

ROS1对计算图架构的实现：

- **节点Node：**计算单元，独立进程程序，用户开发
- **主节点Master：**系统维护，提供节点注册及查找服务，通过roscore指令启动
- **参数服务器Parameter Server：**提供全局参数存取服务，主节点程序的一部分。
- **消息Messages：**定义了节点间数据传递的格式规范
- **主题Topics:**消息具有主题，从而一种消息结构可以被多个主题使用，节点间通过发布和订阅主题实现通信，用于周期性任务
- **服务Services:**用于请求/应答机制，用于单次任务
- **录像Bags：**可以通过录像将ROS节点间消息进行存储和回访，从而降低传感器数据采集难度，加快程序算法开发和调试速度。



ROS1的 计算图框架实现

http://wiki.ros.org/ROS/Technical%20Overview

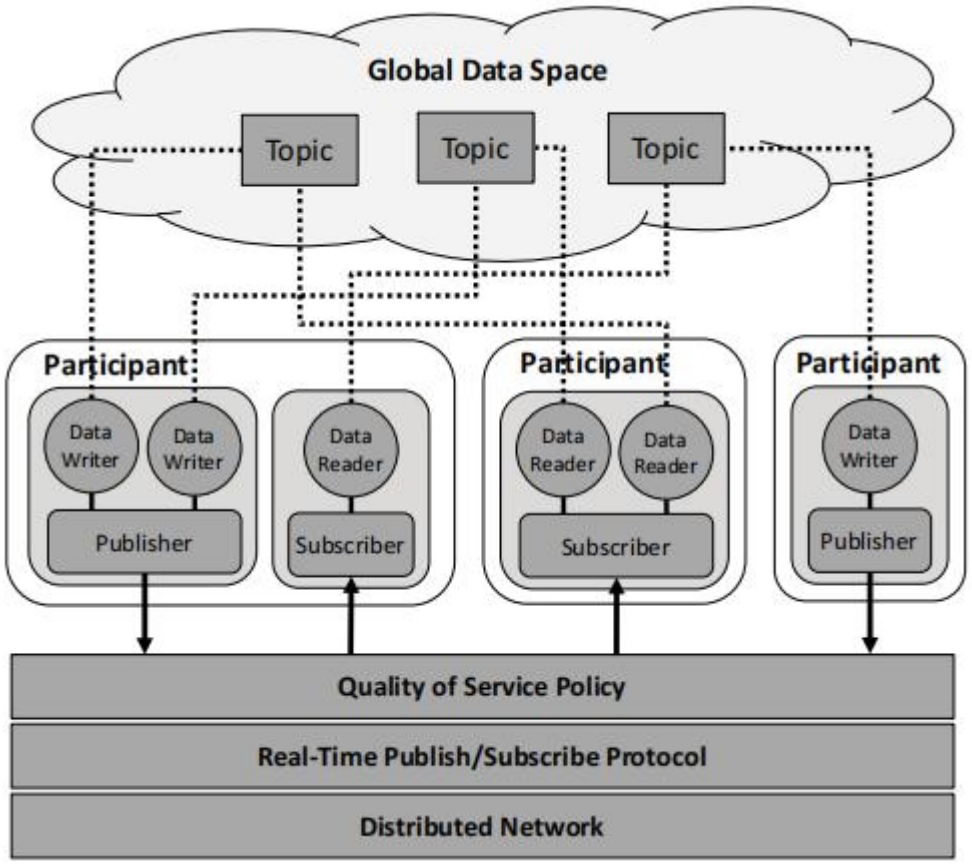
2.2 ROS2框架

文档阅读： Concepts — ROS 2 Documentation

文档阅读： Design (ros2.org)

ROS2对计算图架构的实现：

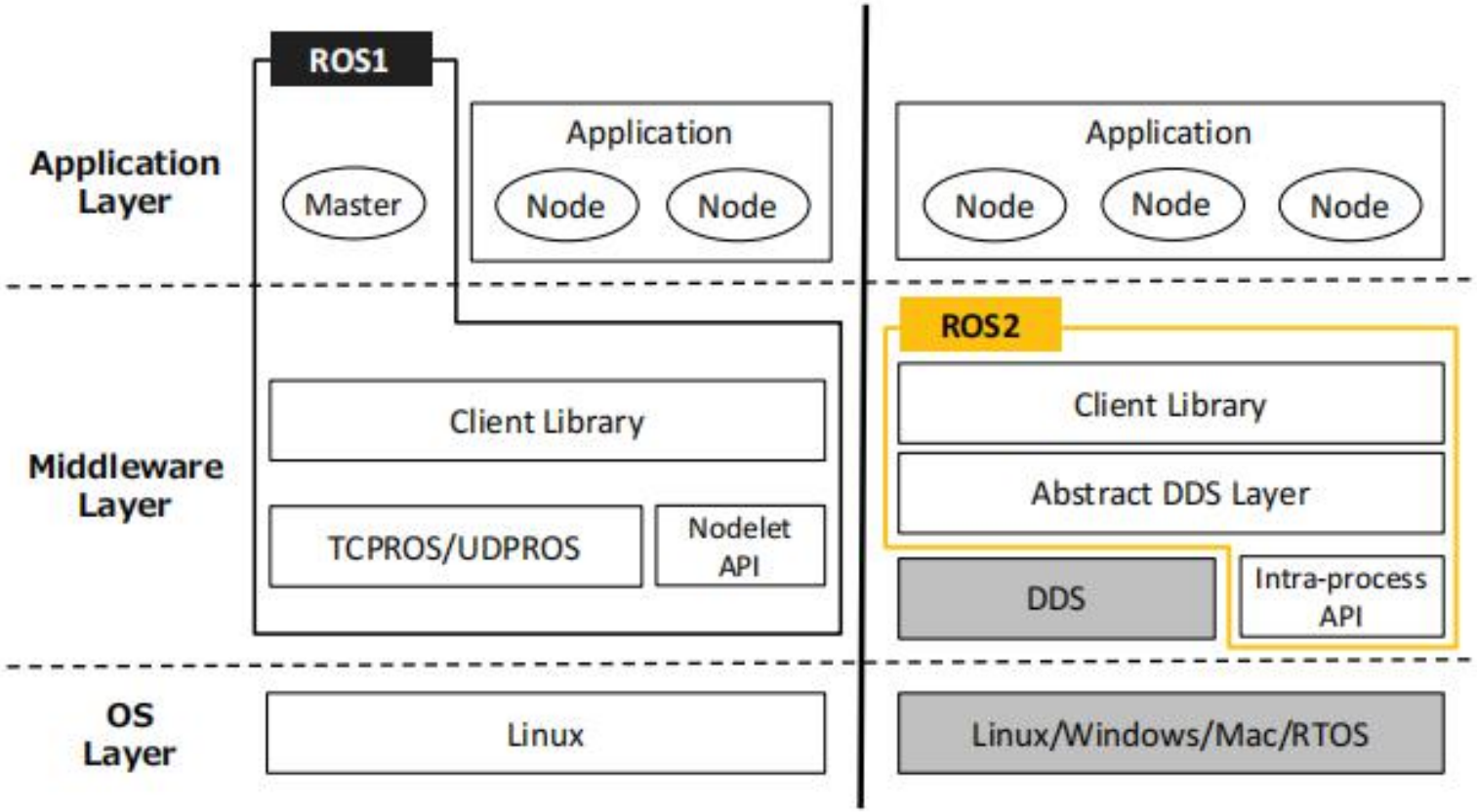
- **节点Node：**计算单元，独立进程程序，用户开发
- **节点参数：**每个节点都可以做参数服务器
- **消息Messages：**定义了节点间数据传递的格式规范
- **主题Topics:**消息具有主题，从而一种消息结构可以被多个主题使用，节点间通过发布和订阅主题实现通信，用于周期性任务
- **服务Services:**用于请求/应答机制，用于单次任务
- **录像Bags：**可以通过录像将ROS节点间消息进行存储和回访，从而降低传感器数据采集难度，加快程序算法开发和调试速度。
- **数据分发服务DDS：**遵循了一种面向实时嵌入式系统的数据服务标准，采用了中间件模式，用户可以采用任意厂家的底层实现。



ROS2 遵循的数据分发服务标准

2.3 ROS1 VS ROS2

论文阅读：Exploring the Performance of ROS2
中文翻译：ROS2测评——探索ROS2的性能



ROS1：耦合的中心节点，单操作系统支持

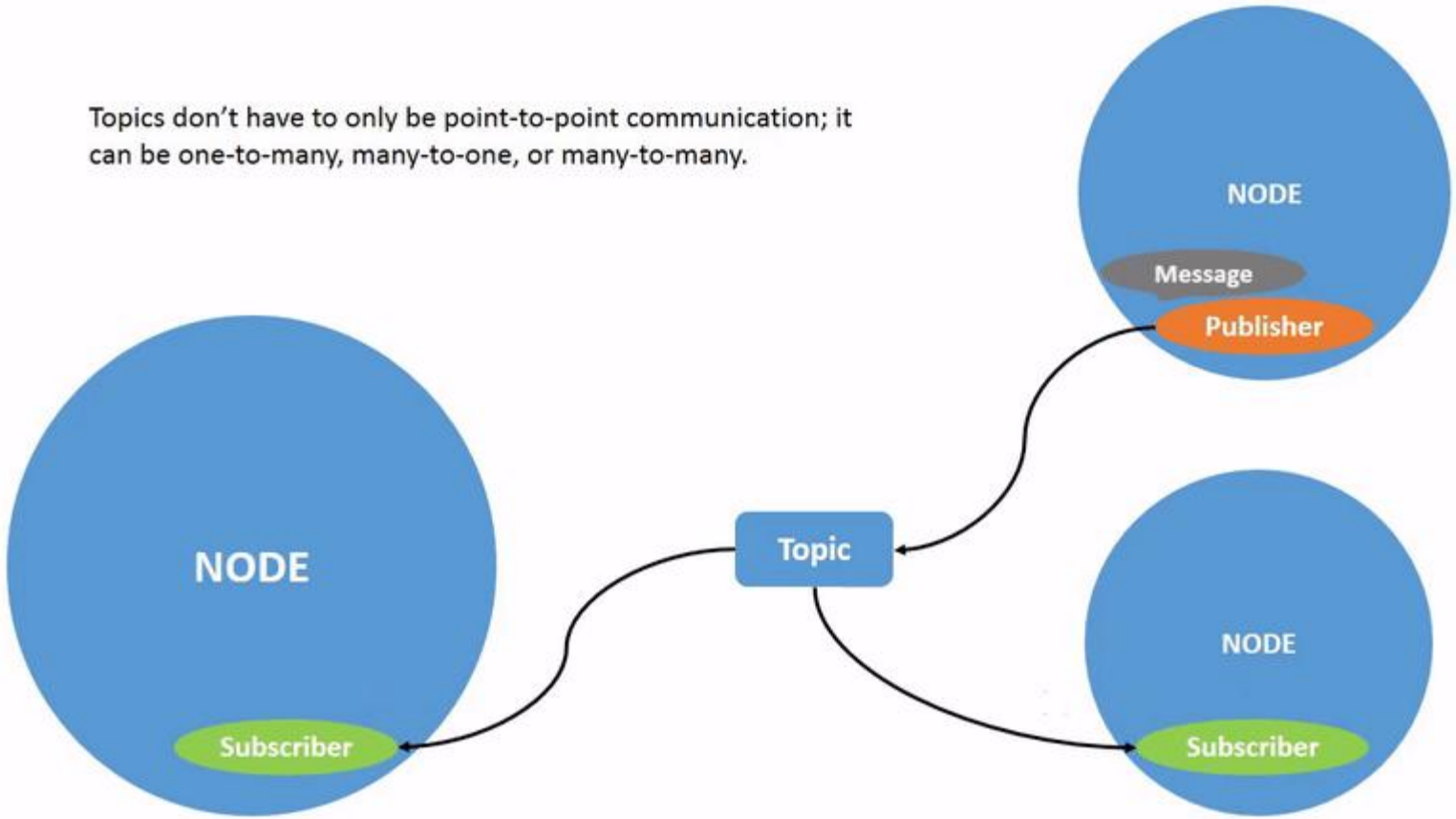
ROS2：去中心节点，多操作系统支持

第3课 ROS编程初步

北邮移动机器人与智能技术实验室 编

3 消息的发布与订阅

3.1 ROS的发布/订阅模型



ROS通过发布和订阅消息的方式，实现节点间通信
通信模式：1对1、1对多、多对多（ROS2新特性）

3.2 ROS1编程实现

程序下载

https://gitee.com/mrobotit/mrobot_book/tree/master/ch2/mrobot_ws/src/learning_topic

发布节点核心代码

```
ros::NodeHandle nh;
ros::Publisher turtle_vel_pub= nh.advertise<geometry_msgs:: Twist> ("/turtle1/cmd_vel",10);
```

订阅节点核心代码：

```
ros::NodeHandle n;//创建节点句柄
ros::Subscriber pose_sub=n.subscribe("/turtle1/pose", 10, poseCallback);
//回调函数
void poseCallback(const turtlesim::Pose::ConstPtr& msg){
        ROS_INFO("Turtle pose: x:%0.6f,y:%0.6f",msg->x,msg->y);
}
```

课后作业：

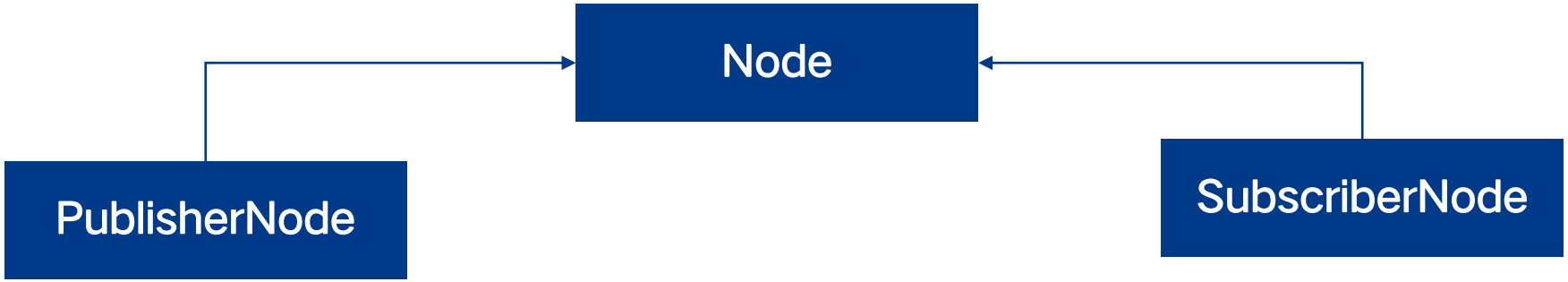
使用rqt_graph工具查看节点间通信图

参照ROS2官方教程

Writing a simple publisher and subscriber (C++) — ROS 2 Documentation: Foxy documentation

将教材中ROS1版本的程序改写为ROS2版本

3.3 ROS2编程实现



①发布节点初始化

```
rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
rclcpp::TimerBase::SharedPtr timer_;
```

②构造函数内定义发布模型：

```
publisher_ = this->create_publisher<std_msgs::msg::String>("topic", 10);
timer_ = this->create_wall_timer(
    500ms, std::bind(&MinimalPublisher::timer_callback, this));
```

③通过定时任务实现周期发布：

```
void timer_callback(){
    //程序处理...
    publisher_->publish(message);
 }
```

①订阅节点初始化

```
rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscription_;
```

②构造函数内定义订阅模型：

```
subscription_ = this->create_subscription<std_msgs::msg::String>(
      “topic”, 10, std::bind(&MinimalSubscriber::topic_callback, this,
_1));
```

③回调函数实现消息处理：

```
void topic_callback(const std_msgs::msg::String::SharedPtr msg) const
 {
      RCLCPP_INFO(this->get_logger(), "I heard: '%s'", msg->data.c_str());
}
```

3.3 自定义主题消息

话题：如何通过ROS Topic名称来了解消息结构？

①rostopic info [消息名字]; 可以得到Type:信息.，例如：

```
$ rostopic info /odom
Type: nav_msgs/Odometry


Publishers:
 * /mobile_base_nodelet_manager ()

Subscribers: None
```

2. 根据上述操作得到 topic 消息类型为
nav_msgs/Odometry;
使用工具 $rosmsg show nav_msgs/Odometry **后如右侧**

3, 如果想要查看msg的说明文件可以使用
$roscd nav_msgs
在该msg文件夹下有很多*.msg文件. 用于描述不同的msg的说明，实际开发时可以从中选用某个消息来扩展

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
  float64[36] covariance
geometry_msgs/TwistWithCovariance twist
  geometry_msgs/Twist twist
    geometry_msgs/Vector3 linear
      float64 x
      float64 y
      float64 z
    geometry_msgs/Vector3 angular
      float64 x
      float64 y
      float64 z
  float64[36] covariance

不同的缩进代表的是级别
```

3.3 自定义主题消息

自定义消息位于：

src/msg文件夹内，文件名后缀.msg

支持的数据类型std_msgs - ROS Wiki：

- int8, int16, int32, int64 (plus uint*)
- float32, float64
- string
- time, duration
- other msg files
- variable-length array[] and fixed-length array[C]

动手实战：

1）ROS/Tutorials/CreatingMsgAndSrv - ROS Wiki
Writing a Simple Publisher and Subscriber (C++)

2）Creating custom ROS 2 msg and srv files — ROS 2 Documentation: Foxy documentation

3）完成教材2.5.7的示例程序

4）将教材2.5.7的代码改写为ROS2版本

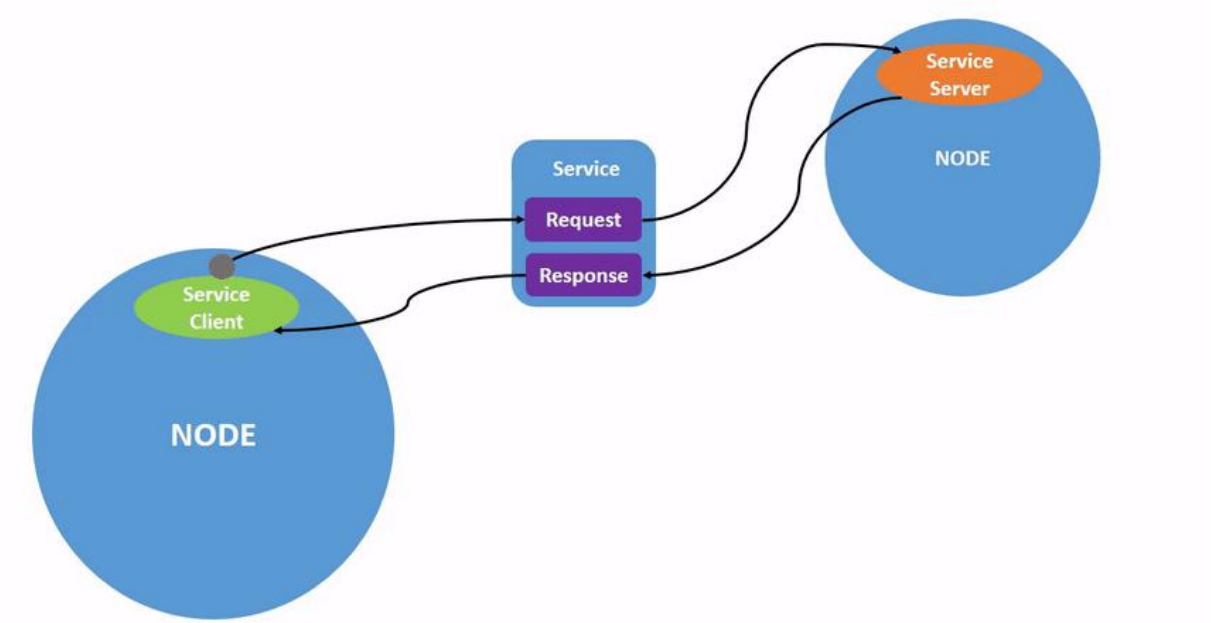
第3课 ROS编程初步

北邮移动机器人与智能技术实验室 编

4 服务的请求与响应

4.1 ROS的请求/响应模型

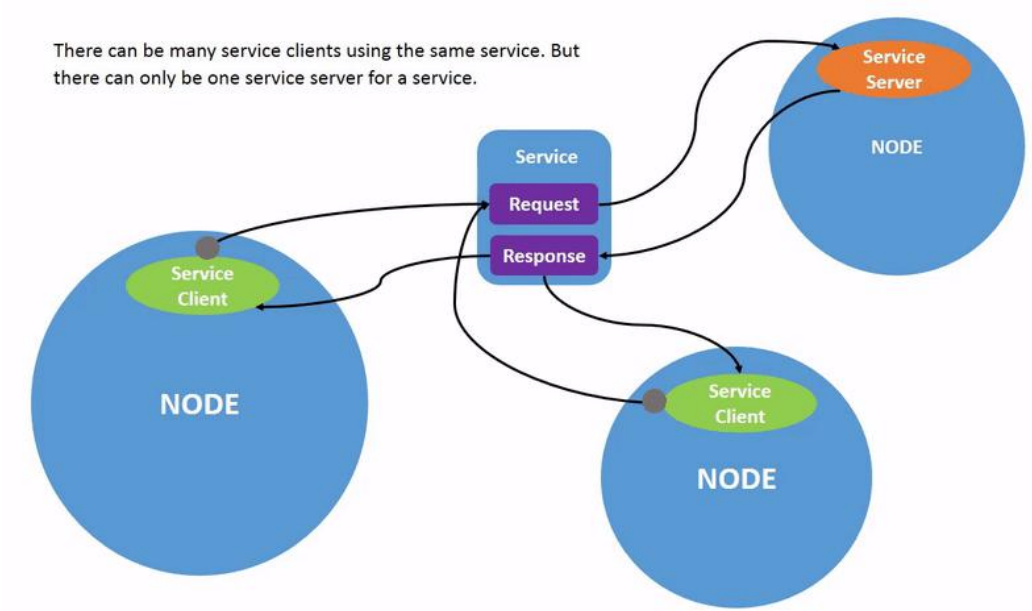
请求应答模型是经典的数据通信模型，ROS也同样支持这种通信方式



1对1

1个服务端

1个客户端



1对多

1个服务端

N个客户端

4.2 ROS1编程实现

程序下载

https://gitee.com/mrobotit/mrobot_book/tree/master/ch2/mrobot_ws/src/learning_service

Client客户端核心代码：

```
ros::NodeHandle nh;
ros::service::waitForService("/spawn");
ros::ServiceClient add_turtle = nh.serviceClient<turtlesim:: Spawn>("/spawn");
add_turtle.call(srv);
ROS_INFO("Spwan turtle successfully [name:%s]", srv.response. name.c_str());
```

Server服务端核心代码

```
ros::NodeHandle nh;
ros::ServiceServer command_service = nh.advertiseService ("/turtle_command", commandCall back);
bool commandCallback(std_srvs::Trigger::Request  &req, std_srvs:: Trigger::Response &res)
{
        ...
        return true;
}
```

备注：教材上的例子没有实现两端程序代码之间的通信，是独立的两个例子

4.3 自定义服务消息实现客户端服务器通信

ROS/Tutorials/CreatingMsgAndSrv - ROS Wiki

ROS/Tutorials/WritingServiceClient(c++) - ROS Wiki

客户端

```
ros::NodeHandle n;
ros::ServiceClient client =
n.serviceClient<beginner_tutorials::AddTwoInts>("add_two_i
nts");
beginner_tutorials::AddTwoInts srv;
srv.request.a = atoll(argv[1]);
srv.request.b = atoll(argv[2]);
if (client.call(srv)){
  ROS_INFO("Sum: %ld", (long int)srv.response.sum);
}
```

服务端

```
ros::NodeHandle n;
ros::ServiceServer service =
n.advertiseService("add_two_ints", add);

add(beginner_tutorials::AddTwoInts::Request  &req,
         beginner_tutorials::AddTwoInts::Response &res)
{
    res.sum = req.a + req.b;
    return true;
}
```

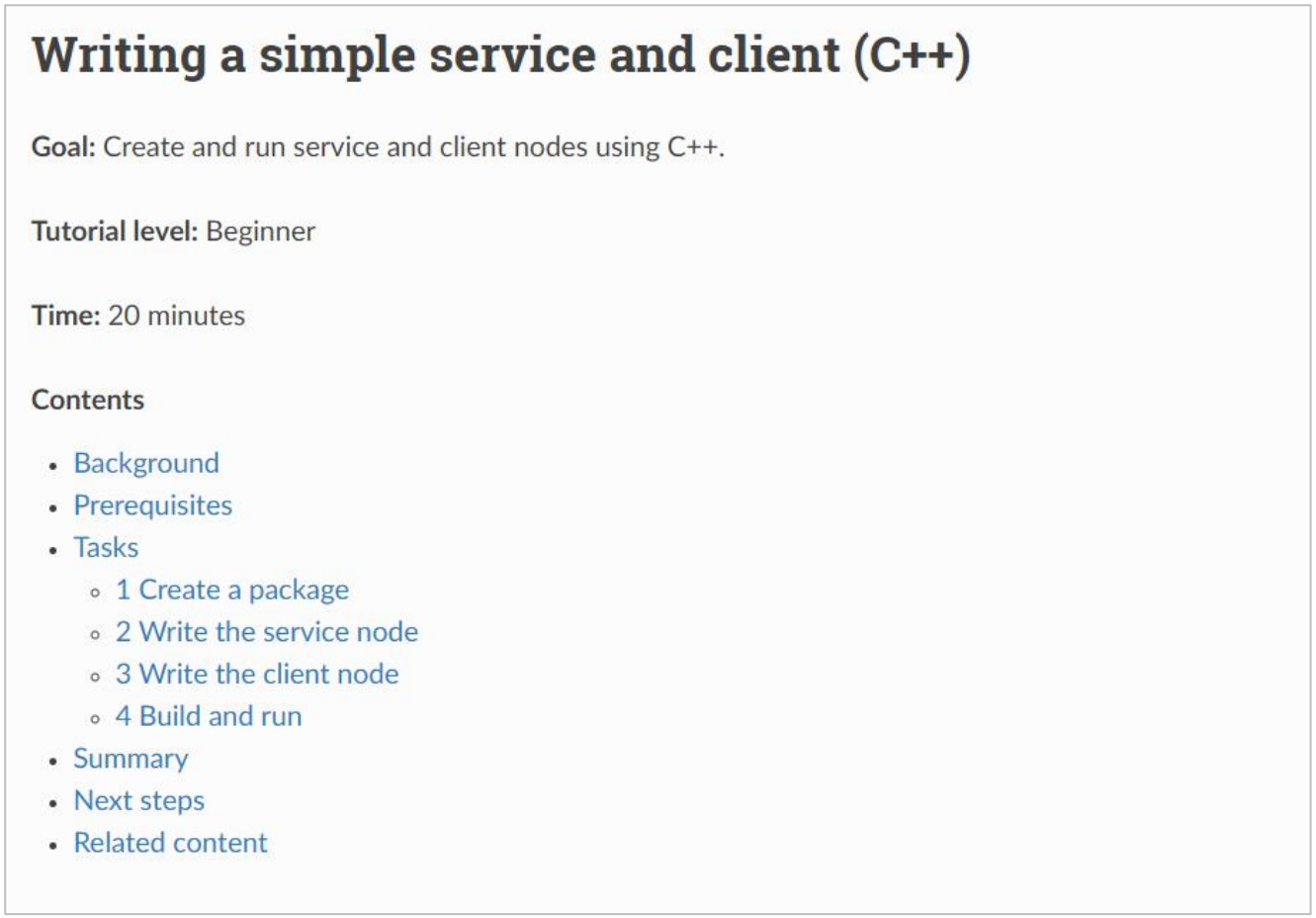
遵循的服务消息格式

```
int64  a
int64  b
---
int64  sum
```

4.4 ROS2编程实现

[Writing a simple service and client (C++) — ROS 2 Documentation: Foxy documentation]

和ROS1实现类似，可以自行参照官网例子



Writing a simple service and client (C++)

Goal: Create and run service and client nodes using C++.

Tutorial level: Beginner

Time: 20 minutes

Contents

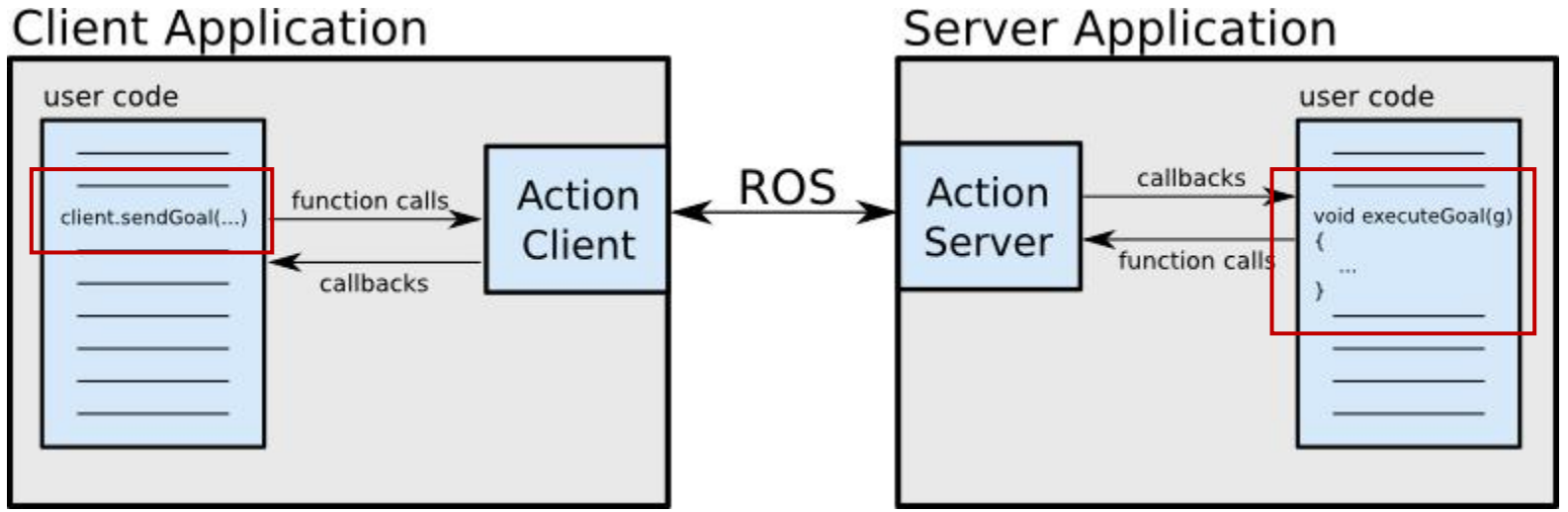
第3课 ROS编程初步

北邮移动机器人与智能技术实验室 编

5 动作的执行与应答

5.1 Action的意义

cn/actionlib - ROS Wiki　服务请求模式下，对于长时间作业的补充应答机制



行为规范: Goal, Feedback, & Result

客户端：Goal
任务目标，例如移动机器人到某个地点
服务端：Feedback
任务反馈，例如移动过程中机器人的位姿
服务端：Result
执行结果，例如返回移动到目标点的环境信息

.action文件格式

```
# 定义目标goal
uint32 dishwasher_id  # 用哪台洗碗机
---
# 定义任务结果
uint32 total_dishes_cleaned id  # 洗了多少
碗
---
# 定义反馈信息
float32 percent_complete
```

5.2 ROS1中Action的具体实现

actionlib_tutorials/Tutorials - ROS Wiki

教材中的代码
https://gitee.com/mrobotit/mrobot_book/tree/master/ch2/mrobot_ws/src/learning_action

2. Beginner Tutorials

2.1 In C++

1. Writing a Simple Action Server using the Execute Callback
   This tutorial covers using the simple_action_server library to create a Fibonacci action server. This example action server generates a Fibonacci sequence, the goal is the order of the sequence, the feedback is the sequence as it is computed, and the result is the final sequence.
2. Writing a Simple Action Client
   This tutorial covers using the simple_action_client library to create a Fibonacci action client. This example program creates an action client and sends a goal to the action server.
3. Running an Action Client and Server
   This tutorial covers running the Fibonacci server and client then visualizing the channel output and node graph.
4. Writing a Simple Action Client (Python)
   This tutorial covers using the action_client library to create a Fibonacci simple action client in Python.
5. Send a Goal to Action Server without Action Client
   This tutorial shows ways of send a Goal to an Action Server without the need of create an Action Client.

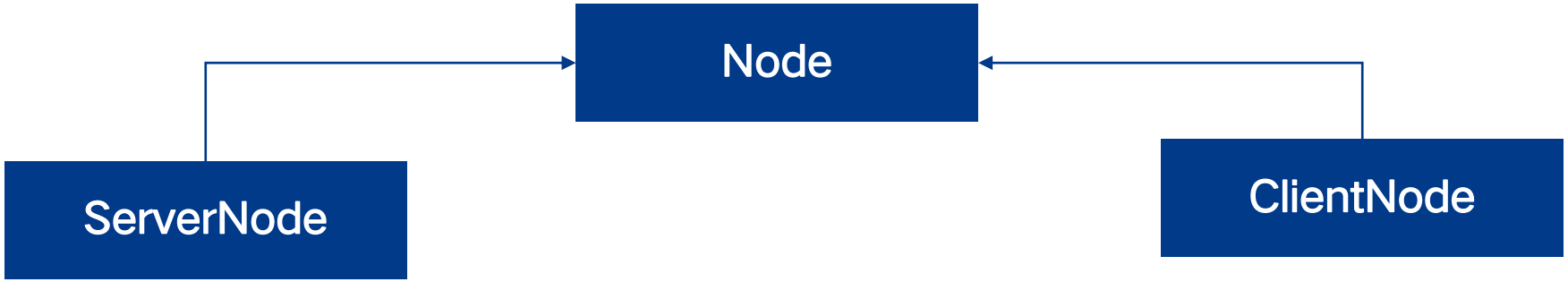
注意事项，其中代码：

```
boost::bind(&FibonacciAction::executeCB, this, _1);
```

其中的_1为占位符

参考理解：C++的占位符 std::placeholder_Erick Lv的笔记-CSDN博客

5.3 ROS2编程实现 Writing an action server and client (C++) — ROS 2 Documentation: Rolling documentation



①服务端节点初始化

```
rclcpp_action::Server<Fibonacci>::SharedPtr action_server_;
```

②三个事件处理函数：

```
rclcpp_action::GoalResponse handle_goal
rclcpp_action::CancelResponse handle_cancel
void handle_accepted→调用其他处理函数
```

③服务处理函数：

```
void execute(const std::shared_ptr<GoalHandleFibonacci> goal_handle)
{  //程序处理...
   //更新过程数据  //给出最终结果
}
```

①客户端节点初始化

```
rclcpp_action::Client<Fibonacci>::SharedPtr client_ptr_;
```

②三个事件回调函数：

```
void goal_response_callback(..)
void feedback_callback(..)
void result_callback(…)
```

③向服务器发送任务目标：

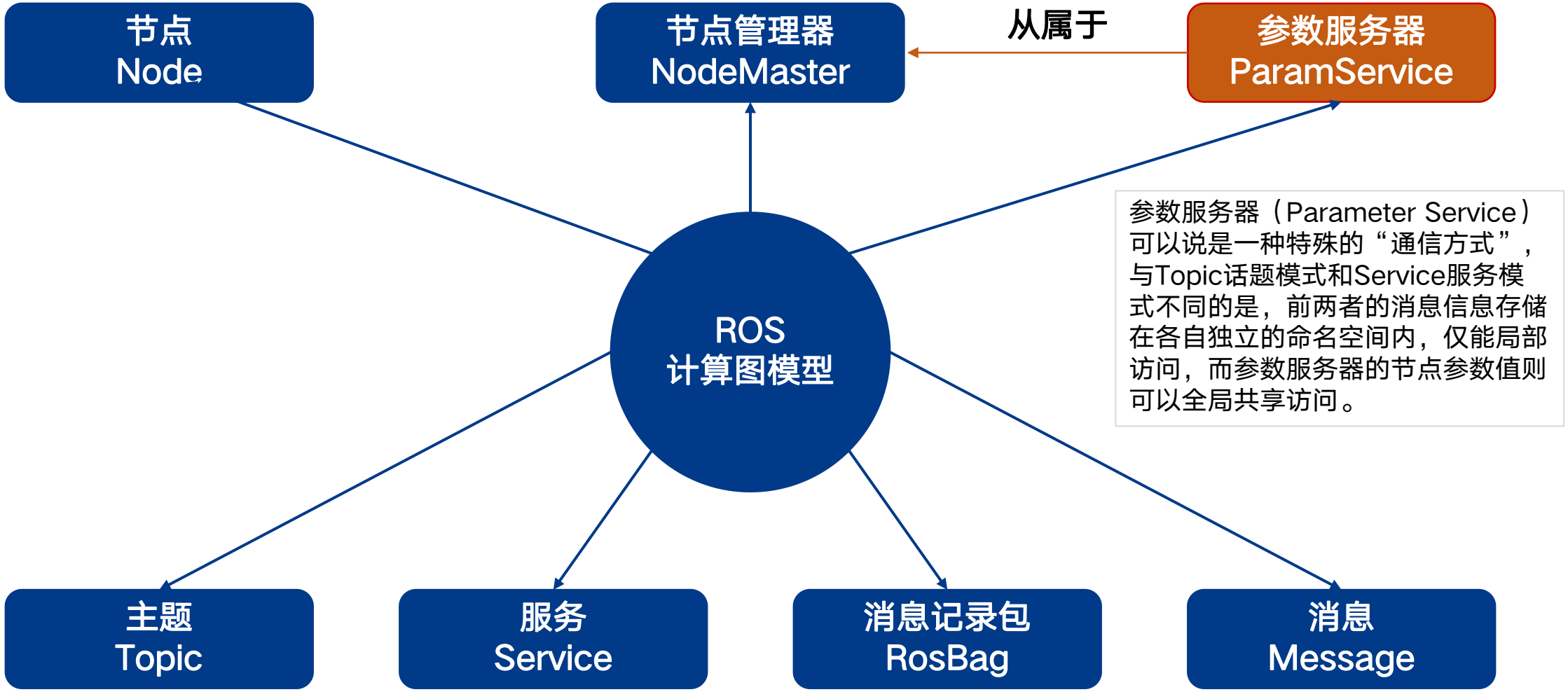
```
//程序处理
// 定义回调引用等
this->client_ptr_->async_send_goal(goal_msg, send_goal_options);
```

第3课 ROS编程初步

北邮移动机器人与智能技术实验室 编

6 ROS的参数

6.1 ROS1的参数服务器



参数服务器（Parameter Service）可以说是一种特殊的“通信方式”，与Topic话题模式和Service服务模式不同的是，前两者的消息信息存储在各自独立的命名空间内，仅能局部访问，而参数服务器的节点参数值则可以全局共享访问。

6.2 ROS1的参数存取

参数服务器通过维护数据字典来存储各种参数和配置，数据字典是键值对列表

命令行

```
rosparam list                                  //列出当前所有参数
rosparam get param_key                          //显示某个参数值
rosparam set param_key param_value             //设置某个参数值
rosparam dump file_name                        //保存参数到文件
rosparam load file_name                         //从文件读取参数
rosparam delete param_key                      //删除参数
```

读取参数代码

```
int red, green, blue;
ros::param::get("/turtlesim/background_r", red);
ros::param::get("/turtlesim/background_g", green);
ros::param::get("/turtlesim/background_b",blue);
```

设置参数代码

```
ros::param::set("/turtlesim/background_r", 255);
ros::param::set("/turtlesim/background_g", 255);
ros::param::set("/turtlesim/background_b",255);
```

6.2 ROS2的参数存取

官方文档

Understanding ROS 2 parameters — ROS 2 Documentation: Foxy documentation

Using parameters in a class (C++) — ROS 2 Documentation: Foxy documentation

参数是节点的配置参数值，是节点配置的一部分。
参数类型：整数，浮点数，布尔值，字符串和列表。

在ROS2中，每个节点都有自己的参数。所有参数都是可动态重新配置的，并且是基于ROS2服务构建的。

命令行

```
ros2 param list          //列出当前所有参数
ros2 param get           //获取某个参数值
ros2 param set           //设置某个参数值
ros2 param dump          //保存当前节点参数到文件
ros2 param load          //从文件中读取节点参数
```

代码

自定义的节点均继承自系统节点，ROS已经写好get_parameter和set_parameter等代码，直接调用即可

```
node->get_parameter("my_parameter", parameter_string_);
```

移动机器人开发技术（激光SLAM版）配套教学PPT

谢 谢 观 看

北京郵電大學
Beijing University of Posts and Telecommunications

移动机器人与智能技术实验室编

宋桂岭 明安龙 2021.9
expsong@qq.com