ROS Introduction

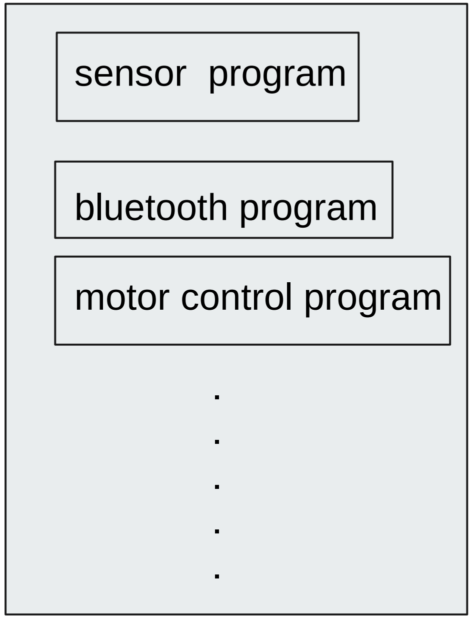
What is ROS(Robot Operating System)?

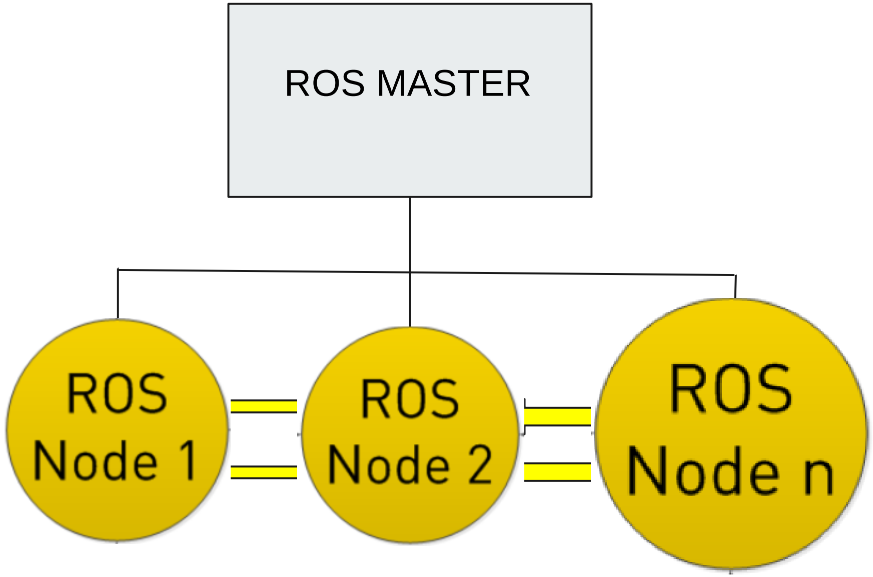
a flexible framework for writing robot software(tools 、 libraries)

Open source(code reuse)

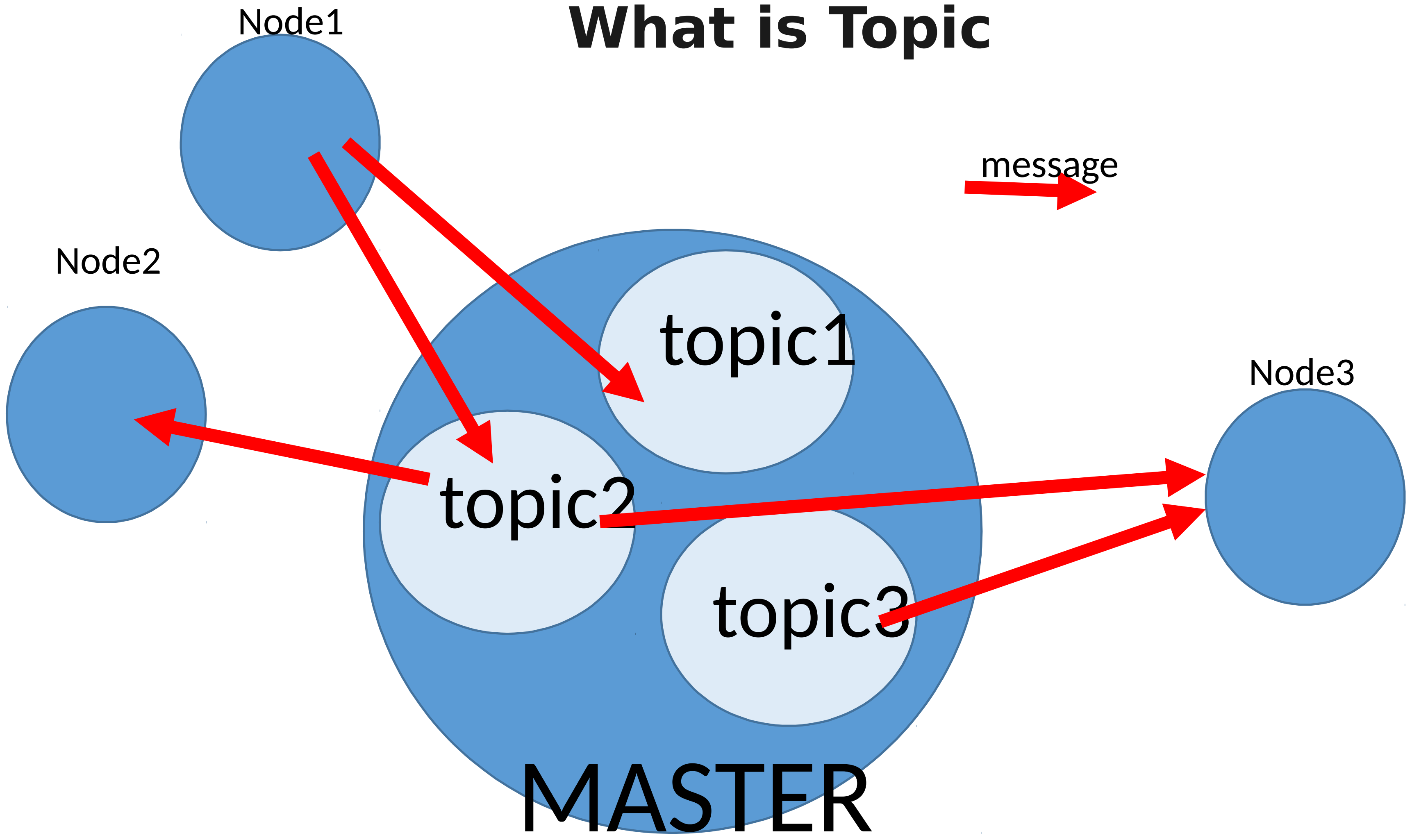
Executables can be **individually** designed and easily **connected** at runtime

General Concepts

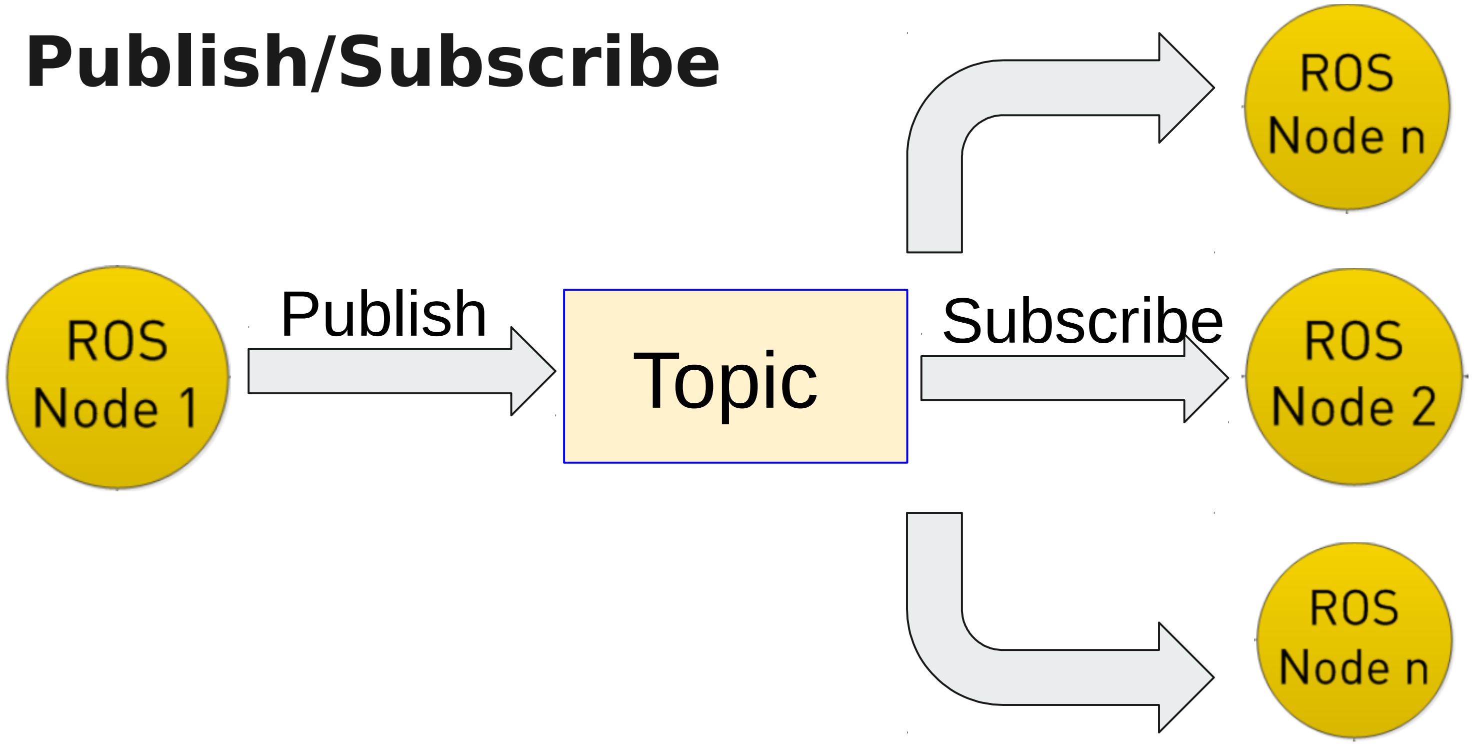




What is Topic



Publish/Subscribe



Tutorial Time

Create a ROS Workspace

```
$ mkdir -p ~/catkin_ws/src      # 創建資料夾
$ cd ~/catkin_ws/                        # 移到 catkin_ws 資料夾（已有）
$ catkin_make                            # 編譯

$ source devel/setup.bash           # 覆蓋此 workspace 至環境
```

Understanding ROS Nodes

```
$ sudo apt-get install ros-kinetic-ros-tutorials        # 下載 tutorial 所需之 package

$ roscore

$ rosnode list                  # 可得知目前所執行之 node

$ rosrun turtlesim turtlesim_node      # 執行 turtlesim 此 package 底下的 node turtlesim_node

$ rosnode list
```

Understanding ROS Topics

```
$ rosrun turtlesim turtlesim_node       # 開啟 node
$ rosrun turtlesim turtle_teleop_key    # 開啟鍵盤控制 node
```

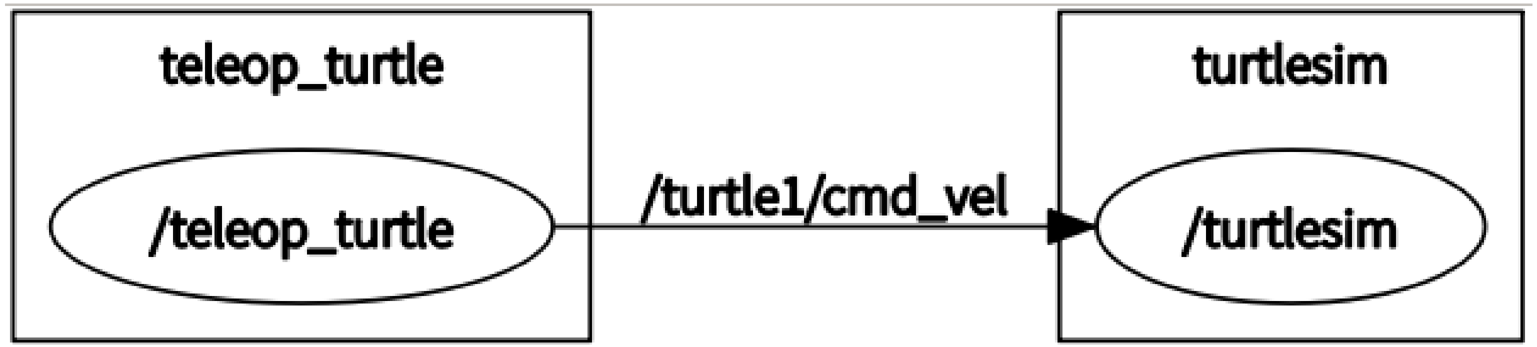
Understanding ROS Topics

```
$ sudo apt-get install ros-kinetic-rqt            # 下載 rqt package

$ sudo apt-get install ros-kinetic-rqt-common-plugins

$ rosrun rqt_graph rqt_graph            #rqt_graph
```

Understanding ROS Topics



Rostopic

$ rostopic type /turtle1/cmd_vel # 可得知此 topic 的 type

geometry_msgs/Twist # 此 topic 的 type

$ rosmsg show geometry_msgs/Twist # 可得知此 message 的內容， message 即為我們傳的資料

```
geometry_msgs/Vector3 linear
float64 x
float64 y
float64 z
geometry_msgs/Vector3 angular
 float64 x
float64 y
float64 z
```

Rostopic

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

# publish velocity and angular velocity to topic by once

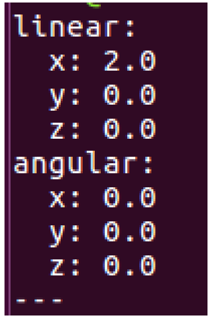
$ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, -1.8]'

# publish velocity and angular velocity to topic by 1Hz

then see rqt again(remember to reset)

Rostopic

$ rostopic echo /turtle1/cmd_vel # 可得知有什麼資料被 publish 到此 topic



```
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

$ rostopic list # 可得知目前有哪些 topic

Creating a ROS Package

```
$ cd ~/catkin_ws/src                    # 移至 src 資料夾

$ catkin_create_pkg first_pkg std_msgs rospy roscpp

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

$ cd ~/catkin_ws

$ catkin_make

$ . ~/catkin_ws/devel/setup.bash
```

Writing a Simple Publisher and Subscriber (C++)

到 first_pkg/src 資料夾底下

創建一空白文件 test.cpp

talker.cpp (a publisher node) 、 listener.cpp (a subscriber node)

test.cpp

```cpp
#include <ros/ros.h>


int main(int argc, char **argv)
{
  ros::init(argc, argv, "test");
  ros::NodeHandle nh;


  ROS_INFO("Hello world!");
}
```

talker.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <sstream>
int main(int argc, char **argv)
{
  ros::init(argc, argv, "talker");
  ros::NodeHandle n;
  ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);
  ros::Rate loop_rate(10);
  int count = 0;
  while (ros::ok())
  {
    std_msgs::String msg;
    std::stringstream ss;
    ss << "I love NCRL " << count;
    msg.data = ss.str();
                        ROS_INFO("%s", msg.data.c_str());
                        chatter_pub.publish(msg);
                        ros::spinOnce();
                        loop_rate.sleep();
    ++count;
  }
            return 0;
}
```

listener.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"
void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
  ROS_INFO("I heard: [%s]", msg->data.c_str());
}
int main(int argc, char **argv)
{
  ros::init(argc, argv, "listener");
  ros::NodeHandle n;
  ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);
  ros::spin();
  return 0;
}
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(first_pkg)

## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp rospy std_msgs genmsg)


## Declare a catkin package
catkin_package()

## Build
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(atest src/test.cpp)
target_link_libraries(talker ${catkin_LIBRARIES})

add_executable(talker src/talker.cpp)
target_link_libraries(talker ${catkin_LIBRARIES})


add_executable(listener src/listener.cpp)
target_link_libraries(listener ${catkin_LIBRARIES})
```

Examining the Simple Publisher and Subscriber

```
$ cd ~/catkin_ws

$ catkin_make

$rosrun  first_pkg atest

$rosrun first_pkg talker
```