Introduction to ROS2

Outline

- Introduce background concepts
  - ROS2 concepts
- Run examples
- Create publishers and subscribers
- Create custom messages
- Create launch files

UNIVERSITY OF ARKANSAS

Prerequisites

- Have a running version of Ubuntu 20.04 installed
- Have ROS2 installed on Ubuntu
- Be able to connect to the Internet

ARKANSAS RAZORBOTZ

UNIVERSITY OF ARKANSAS
College of Engineering
Computer Science & Computer Engineering

Why Use Linux?

Windows ROS2 Install

- Use Windows 10
- Install Chocolatey
- Install Python
- Install Visual C++ Redistributables
- Install OpenSSL
- Install Visual Studio 2019
- Install OpenCV
- Install Chocolatey and Python dependencies
- Install RQt dependencies
  - Install PyQt5
  - Install QT
- Install Graphviz

Linux ROS2 Install

- Enter commands in terminal

Why Use Linux?

- Most embedded processors run some form of Linux
  - RoboRIO, Jetson Nano, Raspberry Pi, etc.
- When robots are in arenas, you might not have access to the device
- Need to have the skills necessary to fix the issues

What is ROS?

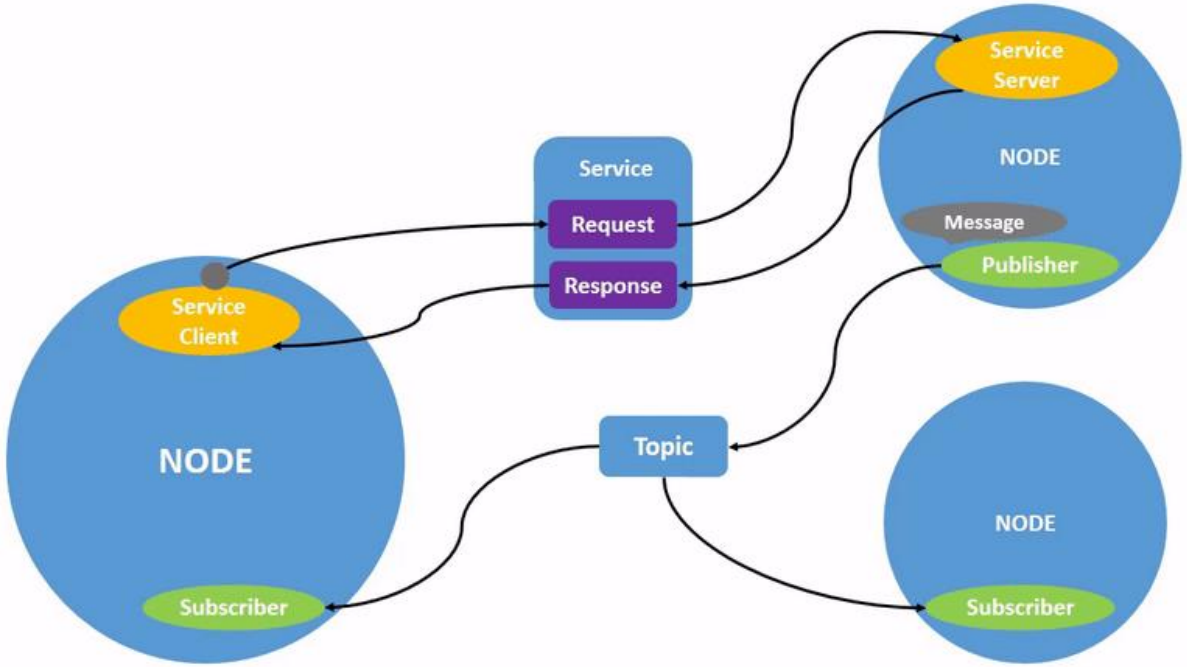
- The Robotic Operating System (ROS) is a set of software libraries and tools for building robot applications
- Two different OSes: ROS and ROS2
- Similar core functionality between the two, but they differ in execution
- Both can be viewed as the infrastructure behind nodes and message passing
- ROS2 has new features, such as using a Data Distribution Service (DDS) for publishing and subscribing and acts more as a middleware

ROS2 Languages

- C++ and Python client libraries are maintained by core ROS2 team
- Community-maintained libraries:
  - Ada
  - C
  - JVM and Android
  - .NET Core, UWP
  - Node.js
  - Rust
- Unmaintained client libraries:
  - C#
  - Objective C and iOS

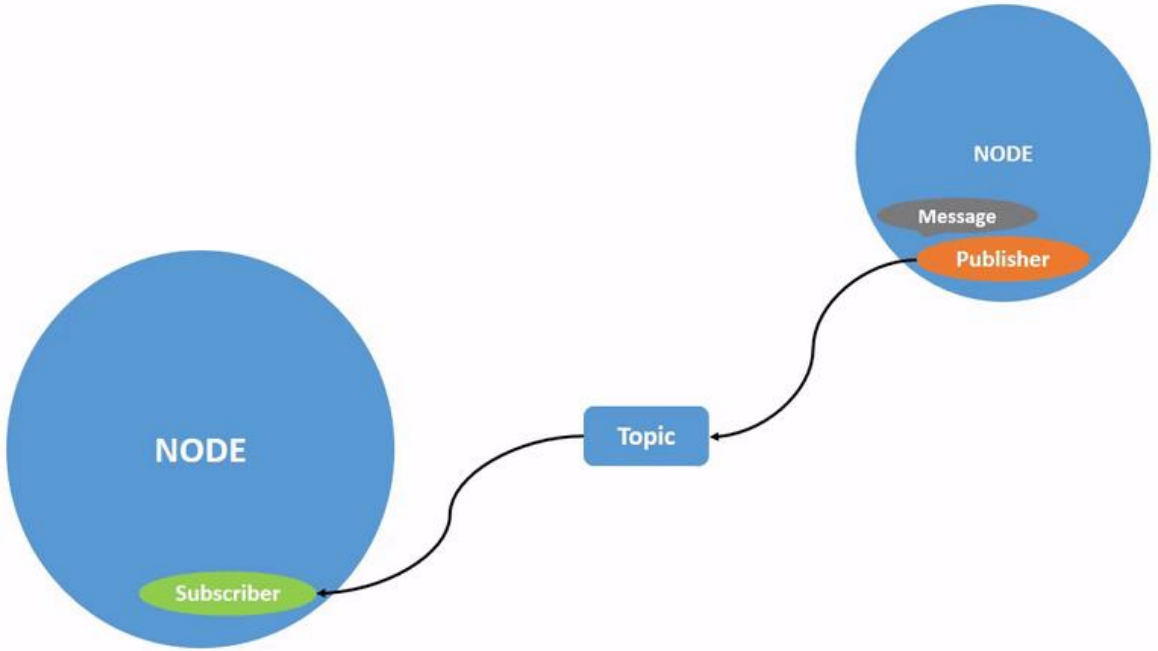
ROS2 Nodes

- Each node should be responsible for a single, modular purpose
- Each node can send data to and receive data from other nodes
  - Can use topics, services, actions, or parameters



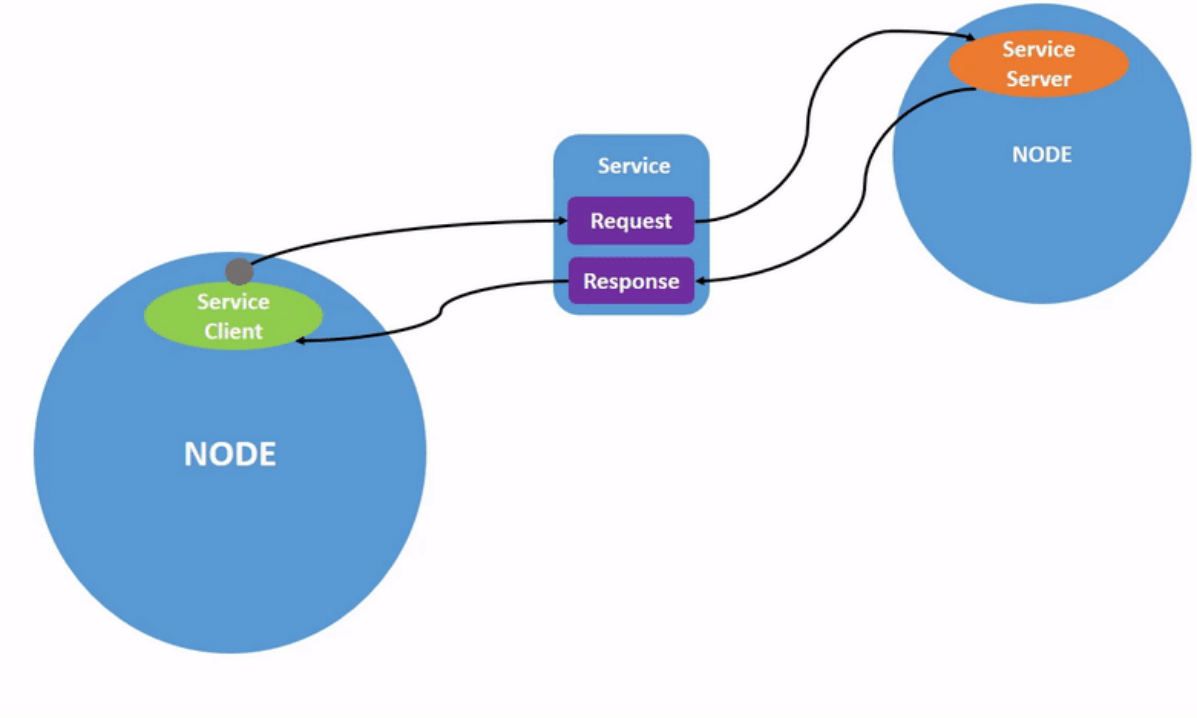
ROS2 Topics

- Allow nodes to pass data
- Nodes can publish data to any number of topics and have subscriptions to any number of topics
- Can be one-to-one, one-to-many, or many-to-many



ROS2 Services

- Allow nodes to pass data
- Call and response model, instead of publisher and subscriber
- Can have many service clients, but only one service server
- Only provides data when called by a client



Run An Example

Open a terminal and type the following command:

source /opt/ros/foxy/setup.bash

ros2 run demo_nodes_cpp talker

Open a second terminal and type the following command:

source /opt/ros/foxy/setup.bash

ros2 run demo_nodes_cpp listener

Run An Example

To add the setup file to the shell startup script, type

echo "source /opt/ros/foxy/setup.bash" >> ~/.bashrc

Creating a Publisher

Create a folder to hold your ROS2 code

mkdir ROS2_Introduction

Move into the folder

cd ROS2_Introduction

Create a new subfolder called src

mkdir src

Move into the subfolder

cd src

Type ros2 pkg create --build-type ament_cmake cpp_pubsub

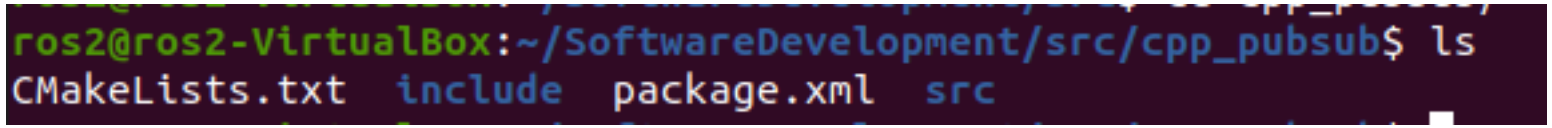
```
ros2@ros2-VirtualBox:~/SoftwareDevelopment/src$ ros2 pkg create --build-type am
ent_cmake cpp_pubsub
going to create a new package
package name: cpp_pubsub
destination directory: /home/ros2/SoftwareDevelopment/src
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['ros2 <ros2@todo.todo>']
licenses: ['TODO: License declaration']
build type: ament_cmake
dependencies: []
creating folder ./cpp_pubsub
creating ./cpp_pubsub/package.xml
creating source and include folder
creating folder ./cpp_pubsub/src
creating folder ./cpp_pubsub/include/cpp_pubsub
creating ./cpp_pubsub/CMakeLists.txt
ros2@ros2-VirtualBox:~/SoftwareDevelopment/src$ ls
cpp_pubsub
```

Creating a Publisher

Move inside the cpp_pubsub folder and type ls

cd cpp_pubsub

ls



Items of note:

CMakeLists.txt – Describes how to build the code within the package

package.xml – Contains meta information about the package

include – Contains header files

src – Contains source code and files for the node

Creating a Publisher

Change to the cpp_pubsub/src folder

cd cpp_pubsub/src

Download the publisher code

wget -O publisher_member_function.cpp https://raw.githubusercontent.com/ros2/examples/foxy/rclcpp/topics/minimal_publisher/member_function.cpp

Creating a Publisher

```
#include <chrono>
#include <functional>
#include <memory>
#include <string>
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"
using namespace std::chrono_literals;

class MinimalPublisher : public rclcpp::Node
{
  public:
    MinimalPublisher()
    : Node("minimal_publisher"), count_(0)
    {
      publisher_ = this->create_publisher<std_msgs::msg::String>("topic", 10);
      timer_ = this->create_wall_timer(
      500ms, std::bind(&MinimalPublisher::timer_callback, this));
    }
```

Creating a Publisher

```
 private:
   void timer_callback()
   {
     auto message = std_msgs::msg::String();
     message.data = "Hello, world! " + std::to_string(count_++);
     RCLCPP_INFO(this->get_logger(), "Publishing: '%s'", message.data.c_str());
     publisher_->publish(message);
   }
   rclcpp::TimerBase::SharedPtr timer_;
   rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
   size_t count_;
};
```

Creating a Publisher

```
int main(int argc, char * argv[])
{
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<MinimalPublisher>());
  rclcpp::shutdown();
  return 0;
}
```

Creating a Publisher

Navigate back to the cpp_pubsub folder and open package.xml

cd ..

vi package.xml

Change the <description>, <maintainer>, and <license> tags to the correct values

Below the line <build_type>ament_cmake</build_type>, enter the following

<depend>rclcpp</depend>

<depend>std_msgs</depend>

Save and close the file

Creating a Publisher

Open the CMakeLists.txt

vi CMakeLists.txt

Below find_package(ament_cmake REQUIRED), enter

```
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)

add_executable(talker src/publisher_member_function.cpp)
ament_target_dependencies(talker rclcpp std_msgs)

install(TARGETS
  talker
  DESTINATION lib/${PROJECT_NAME})
```

Creating a Subscriber

Change to the cpp_pubsub/src folder

cd src

Download the subscriber code

wget -O subscriber_member_function.cpp https://raw.githubusercontent.com/ros2/examples/foxy/rclcpp/topics/minimal_subscriber/member_function.cpp

Creating a Subscriber

```
#include <memory>

#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"
using std::placeholders::_1;

class MinimalSubscriber : public rclcpp::Node
{
  public:
    MinimalSubscriber()
    : Node("minimal_subscriber")
    {
      subscription_ = this->create_subscription<std_msgs::msg::String>("topic", 10,
std::bind(&MinimalSubscriber::topic_callback, this, _1));
    }
```

Creating a Subscriber

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

int main(int argc, char * argv[])
{
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<MinimalSubscriber>());
  rclcpp::shutdown();
  return 0;
}
```

Creating a Subscriber

```
#include <memory>
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp“
rclcpp::Node::SharedPtr nodeHandle;

void topic_callback(const std_msgs::msg::String::SharedPtr msg){
   RCLCPP_INFO(nodeHandle->get_logger(), “I heard: ‘%s’”, msg->data.c_str());
}

int main(int argc, char** argv){
   rclcpp::init(argc, argv);
   nodeHandle = rclcpp::Node::make_shared(“minimal_subscriber”);
   auto subscription = nodeHandle->create_subscription<std_msgs::msg::String>(“topic”, 10, topic_callback>
   while(rclcpp::ok()){
      rclcpp::spin_some(nodeHandle);
   }
}
```

Creating a Subscriber

Move back to the cpp_pubsub folder
cd ..
Edit the CMakeLists.txt file
vi CMakeLists.txt

Add the following lines below the publisher executable
add_executable(listener src/subscriber_member_function.cpp)
ament_target_dependencies(listener rclcpp std_msgs)

In the install parentheses, add the listener
install(TARGETS
  talker
  listener
  DESTINATION lib/${PROJECT_NAME})

Running the Nodes

Navigate back to the root of the workspace, SoftwareDevelopment
cd ROS2_Introduction

Build the project with colcon
colcon build --packages-select cpp_pubsub

Type . install/setup.bash

Open a second terminal and navigate to the root, SoftwareDevelopment
cd ROS2_Introduction
Type . install/setup.bash

Running the Nodes

In the first terminal, type the following

ros2 run cpp_pubsub talker

In the second terminal, type the following

ros2 run cpp_pubsub listener

Creating Custom Messages

In folder ROS2_Introduction /src

ros2 pkg create --build-type ament_cmake messages

cd messages

mkdir msg

cd msg

touch Test.msg

nano Test.msg

Add the line following line to Test.msg

int64 num

Creating Custom Messages

cd ..

nano CMakeLists.txt

Add the following lines to CMakeLists.txt

```
find_package(rosidl_default_generators REQUIRED)

rosidl_generate_interfaces(${PROJECT_NAME}
  "msg/Test.msg"
)
```

Creating Custom Messages

Add the following lines to package.xml

<build_depend>rosidl_default_generators</build_depend>

<exec_depend>rosidl_default_runtime</exec_depend>

<member_of_group>rosidl_interface_packages</member_of_group>

Creating Custom Messages

Navigate back to the root, SoftwareDevelopment

Build the messages package

colcon build --packages-select messages

. install/setup.bash

ros2 interface show messages/msg/Test.msg

Modifying the Publisher

```
#include <chrono>
#include <functional>
#include <memory>
#include <string>
#include "rclcpp/rclcpp.hpp"
#include “messages/msg/test.hpp"
using namespace std::chrono_literals;

class MinimalPublisher : public rclcpp::Node
{
  public:
    MinimalPublisher()
    : Node("minimal_publisher"), count_(0)
    {
      publisher_ = this->create_publisher<messages::msg::Test>("topic", 10);
      timer_ = this->create_wall_timer(
      500ms, std::bind(&MinimalPublisher::timer_callback, this));
    }
```

Modifying the Publisher

```
 private:
   void timer_callback()
   {
     auto message = messages::msg::Test();
     message.num = this->count_++;
     RCLCPP_INFO(this->get_logger(), "Publishing: ‘%d'", message.num);
     publisher_->publish(message);
   }
   rclcpp::TimerBase::SharedPtr timer_;
   rclcpp::Publisher<messages::msg::Test>::SharedPtr publisher_;
   size_t count_;
};
```

Modifying the Subscriber

```
#include <memory>

#include "rclcpp/rclcpp.hpp"
#include “messages/msg/test.hpp"
using std::placeholders::_1;

class MinimalSubscriber : public rclcpp::Node
{
  public:
    MinimalSubscriber()
    : Node("minimal_subscriber")
    {
      subscription_ = this->create_subscription<messages::msg::Test>("topic", 10, std::bind(&MinimalSubscriber::topic_callback, this, _1));
    }
```

Modifying the Subscriber

```
 private:
   void topic_callback(const messages::msg::Test::SharedPtr msg) const
   {
     RCLCPP_INFO(this->get_logger(), "I heard: ‘%d'", msg->num);
   }
   rclcpp::Subscription<messages::msg::Test>::SharedPtr subscription_;
};

int main(int argc, char * argv[])
{
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<MinimalSubscriber>());
  rclcpp::shutdown();
  return 0;
}
```

Modifying CMakeLists.txt

```
#...
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(messages REQUIRED)

add_executable(talker src/publisher_member_function.cpp)
ament_target_dependencies(talker rclcpp messages)

add_executable(listener src/subscriber_member_function.cpp)
ament_target_dependencies(listener rclcpp messages)
```

Modifying package.xml

Remove the following line:

<depend>std_msgs</depend>

Replace it with the following:

<depend>messages</depend>

Running the Nodes

In the first terminal, type the following

colcon build --packages-select cpp_pubsub

. install/setup.bash

ros2 run cpp_pubsub talker

In the second terminal, type the following

. install/setup.bash

ros2 run cpp_pubsub listener

Creating Launch Files

Navigate to the workspace root, ROS2_Introduction

Make a new directory called launch

mkdir launch

cd launch

touch launch.py

Editing launch.py

Enter the following code into launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='cpp_pubsub',
            executable='talker'
        ),
        Node(
            package='cpp_pubsub',
            executable='listener'
        )
    ])
```

Launching the Nodes

To launch the nodes, type the following command:

ros2 launch launch.py

Editing launch_pub.py

Enter the following code into launch_pub.py

```
from launch import LaunchDescription
from launch_ros.actions import Node


def generate_launch_description():
    return LaunchDescription([
        Node(
            package='cpp_pubsub',
            executable='talker'
        )
])
```

Launching launch_pub.py

To launch the nodes, type the following command:

ros2 launch launch_pub.py

Editing launch_sub.py

Enter the following code into launch_sub.py

```
from launch import LaunchDescription
from launch_ros.actions import Node


def generate_launch_description():
    return LaunchDescription([
        Node(
            package='cpp_pubsub’,
            executable=‘listener'
        )
])
```

Launching launch_sub.py

To launch the nodes, type the following command:

ros2 launch launch_sub.py

Closing Remarks

Google is your friend. When in doubt, Google it.

This is not intended to be a comprehensive introduction. Please look at the ROS2 tutorials found at https://docs.ros.org/en/foxy/Tutorials.html.