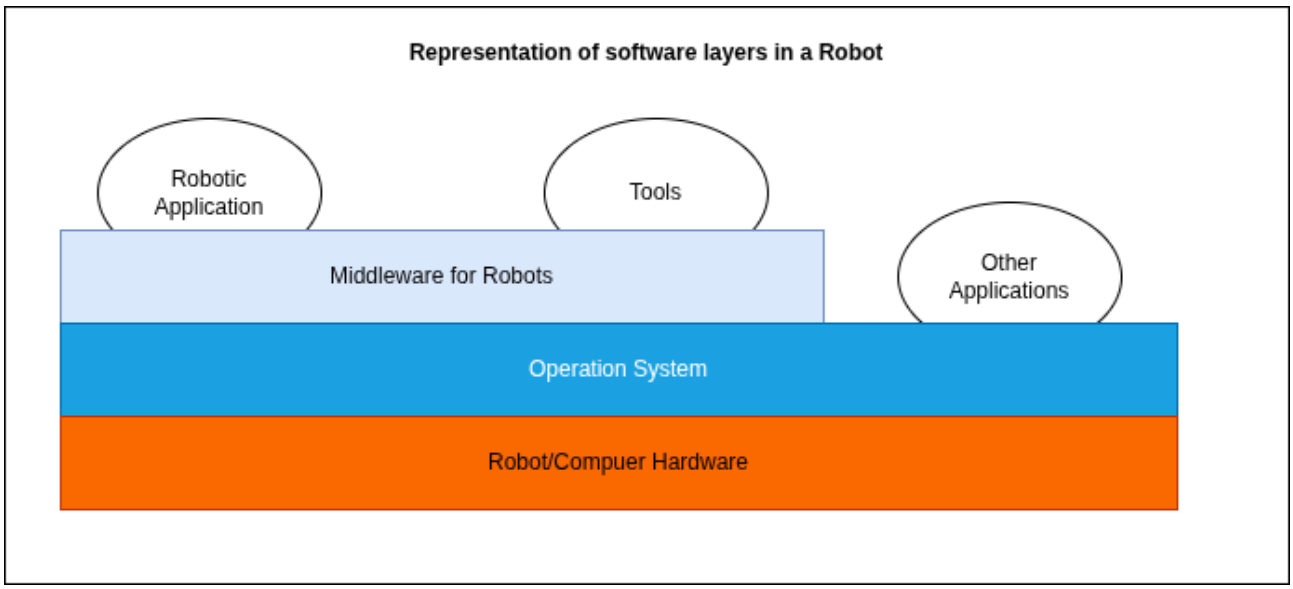
Robot Programming with ROS2



ROS Official Page: https://www.ros.org

Robotics Stack Exchange: https://robotics.stackexchange.com

ROS Discourse: https://discourse.ros.org

Official ROS 2 tutorials: https://docs.ros.org/en/jazzy/Tutorials.html

The Robotics Back-End tutorials: https://roboticsbackend.com/category/ros2

ROS 2 for ROS developers: https://github.com/fmrico/ros_to_ros2_talk_examples

ROS Community: https://www.openrobotics.org

ROS Distro: https://github.com/ros/rosdistro

ROS2 Cli: https://github.com/ros2/ros2cli

Computation Graph: The Computational Graph is a running ROS 2 application. This graph is made up of nodes and arcs. The Node, the primary computing unit in ROS 2, can collaborate with other nodes using several different communication paradigms to compose a ROS 2 application. This dimension also addresses the monitoring tools, which are also nodes that are inserted in this graph.

***Grafo de Computação**: O Grafo de Computação é uma aplicação ROS 2 em execução. Este grafo é composto por nós e arcos. O Nó, a unidade computacional primária no ROS 2, pode colaborar com outros nós usando diversos paradigmas de comunicação diferentes para compor uma aplicação ROS 2. Esta dimensão também abrange as ferramentas de monitoramento, que também são nós inseridos neste grafo.*

The Workspace: The Workspace is the set of software installed on the robot or computer and the programs that the user develops. In contrast to the Computational Graph, which has a dynamic nature, the Workspace is static. This dimension also addresses the development tools to build the elements of the Computational Graph.

O Espaço de Trabalho*: O Espaço de Trabalho é o conjunto de softwares instalados no robô ou computador e os programas que o usuário desenvolve. Em contraste com o Grafo Computacional, que tem natureza dinâmica, o Espaço de Trabalho é estático. Esta dimensão também abrange as ferramentas de desenvolvimento para construir os elementos do Grafo Computacional.*

ROS 2 makes intensive use of Object-Oriented Programming. A node is an object of class Node, in general, whether it is written in C++ or Python. A node can access the Computation Graph and provides mechanisms to communicate with other nodes through three types of paradigms:

Publication/Subscription: It is an asynchronous communication where N nodes publish messages to a topic that reaches its M subscribers. A topic is like a communication channel that accepts messages of a unique type. This type of communication is he most common in ROS 2. A very representative case is the node that contains the driver of a camera that publishes images to a topic. All the nodes in a system needing images from the camera to carry out their function subscribe to this topic.

Publicação/Assinatura*: É uma comunicação assíncrona onde N nós publicam mensagens em um tópico que alcança seus M assinantes. Um tópico é como um canal de comunicação que aceita mensagens de um tipo único. Este tipo de comunicação é o mais comum no ROS 2. Um caso muito representativo é o nó que contém o driver de uma câmera que publica imagens em um tópico. Todos os nós em um sistema que precisam de imagens da câmera para executar sua função se inscrevem neste tópico.*

Services: It is an asynchronous communication in with a node requests another node and expects an immediate response. This communication usually requires an immediate response so as not to affect the control cycle of the node that calls the service. An example could be the request to the mapping service to reset a map, with a response indicating if the call succeeded.

Serviços*: É uma comunicação assíncrona na qual um nó solicita a outro nó e espera uma resposta imediata. Essa comunicação geralmente requer uma resposta imediata para não afetar o ciclo de controle do nó que chama o serviço. Um exemplo poderia ser a solicitação ao serviço de mapeamento para redefinir um mapa, com uma resposta indicando se a chamada foi bem-sucedida.*

Actions: These are asynchronous communications in which a node makes a request to another node. Theses requests usually take time to complete, ant the calling node may periodically receive feedback or the notification that it has finished successfully or with some error. A navigation request is an example of this type of communication. This request is possibly time-consuming, whereby the node requesting the robot to navigate should not be blocked while completing.

Ações*: São comunicações assíncronas nas quais um nó faz uma solicitação a outro nó. Essas solicitações geralmente levam tempo para serem concluídas e o nó que faz a chamada pode receber periodicamente feedback ou a notificação de que a solicitação foi concluída com sucesso ou com algum erro. Uma solicitação de navegação é um exemplo desse tipo de comunicação. Essa solicitação pode ser demorada, portanto, o nó que solicita a navegação do robô não deve ser bloqueado enquanto estiver sendo concluída.*

The function of a node in a computational graph is to perform processing or control. Therefore, they are considered active elements with some alternatives in terms of their execution model:

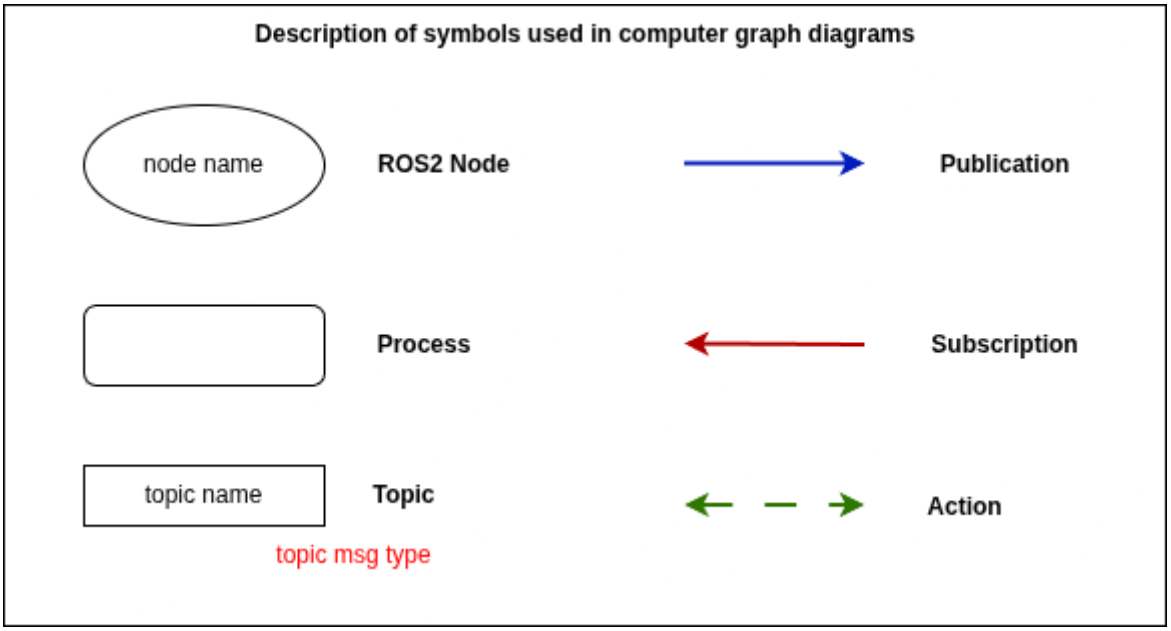
A função de um nó em um grafo computacional é realizar processamento ou controle. Portanto, eles são considerados elementos ativos com algumas alternativas em termos de seu modelo de execução:

Iterative execution: Its is popular in the control software for a node to execute its control cycle at a specific frequency. This approach allows controlling how many computation resources a node requires, and the output flow remains constant. For example, a node calculating motion commands to actuators at 20 Hz based on their status.

***Execução iterativa**: É comum em softwares de controle que um nó execute seu ciclo de controle em uma frequência específica. Essa abordagem permite controlar quantos recursos computacionais um nó requer, e o fluxo de saída permanece constante. Por exemplo, um nó que calcula comandos de movimento para atuadores a 20 Hz com base em seu estado.*

Event-oriented execution: The execution of these nodes is determined by the frequency at which certain events occur, usually the arrival of messages at this node. For example, a node that, for each image it receives, performs detection on it and produces an output. The frequency at which an output occurs depends on the frequency at which images arrive. If no images reach it, it produces no output.

***Execução orientada a eventos**: A execução desses nós é determinada pela frequência com que certos eventos ocorrem, geralmente a chegada de mensagens a esse nó. Por exemplo, um nó que, para cada imagem recebida, realiza uma detecção e produz uma saída. A frequência com que uma saída ocorre depende da frequência com que as imagens chegam. Se nenhuma imagem chegar, ele não produz nenhuma saída.*



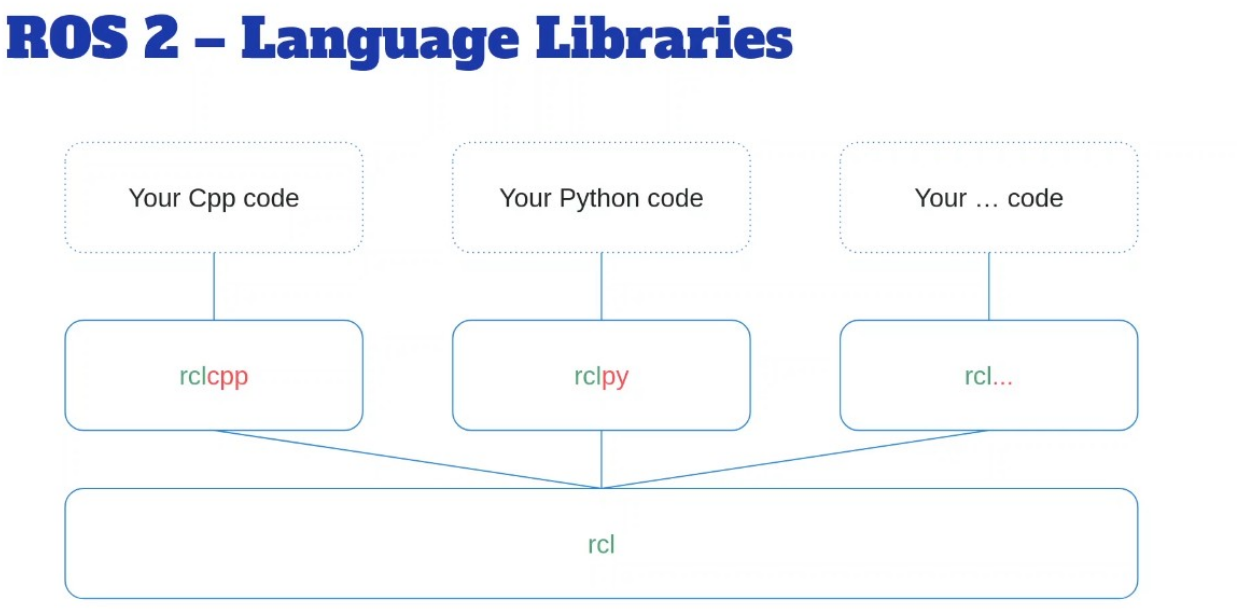
Names of ROS 2

The name of resources in ROS 2 follow a convention very similar to the filesystem in Unix. When creating a resource, for example, a publisher, we can specify its name as relative, absolute (begins with “/”), or private (begins with “~”). Furthermore, we can define a namespace whose objective is to isolate resources from other namespaces by adding the workspace’s name as the first component of tha name. Namespaceds are helpul, for example, in multirobot application. Let’s see an example of the resulting name of a topic based on the node name and the namespace:

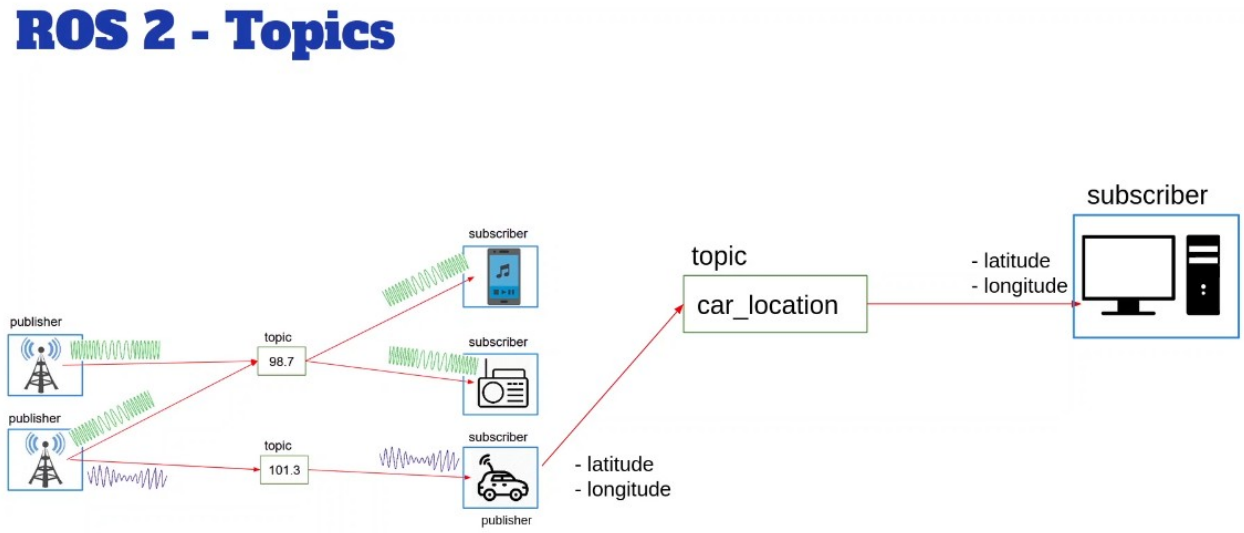
A nomenclatura dos recursos no ROS 2 segue uma convenção muito semelhante à do sistema de arquivos no Unix. Ao criar um recurso, por exemplo, um publicador, podemos especificar seu nome como relativo, absoluto (começando com “/”) ou privado (começando com “~”). Além disso, podemos definir um namespace, cujo objetivo é isolar recursos de outros namespaces, adicionando o nome do espaço de trabalho como o primeiro componente do nome. Namespaces são úteis, por exemplo, em aplicações com múltiplos robôs. Vejamos um exemplo do nome resultante de um tópico com base no nome do nó e no namespace:

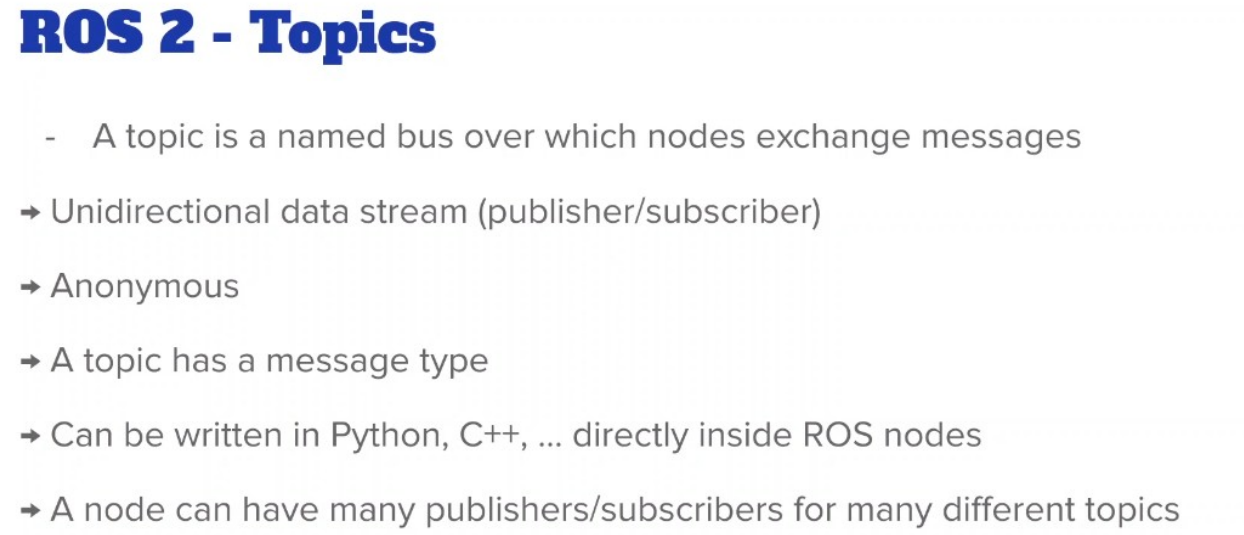
Name	**Result: (node: my_node / ns: none)**	**Result: (node: my_node / ns: my_ns)**
my_topic	/my_topic	/my_ns/my_topic
/my_topic	/my_topic	/my_topic
~my_topic	/my_node/my_topic	/my_ns/my_node/my_topic

ROS2 Language Libraries

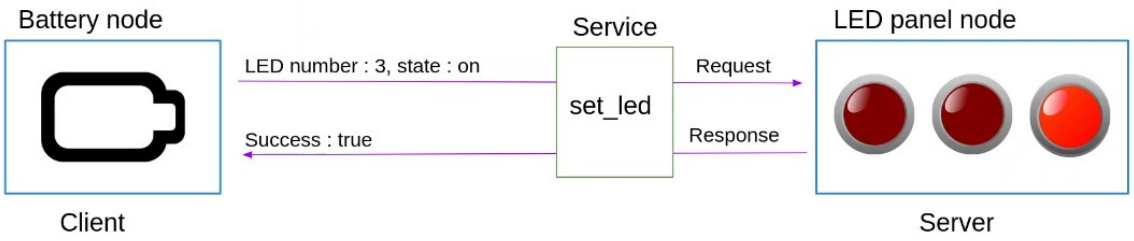


ROS2 Topic Example Comparative (Analogy)





ROS 2 - Services



ROS 2 - Services

- A ROS 2 Service is a client/server system

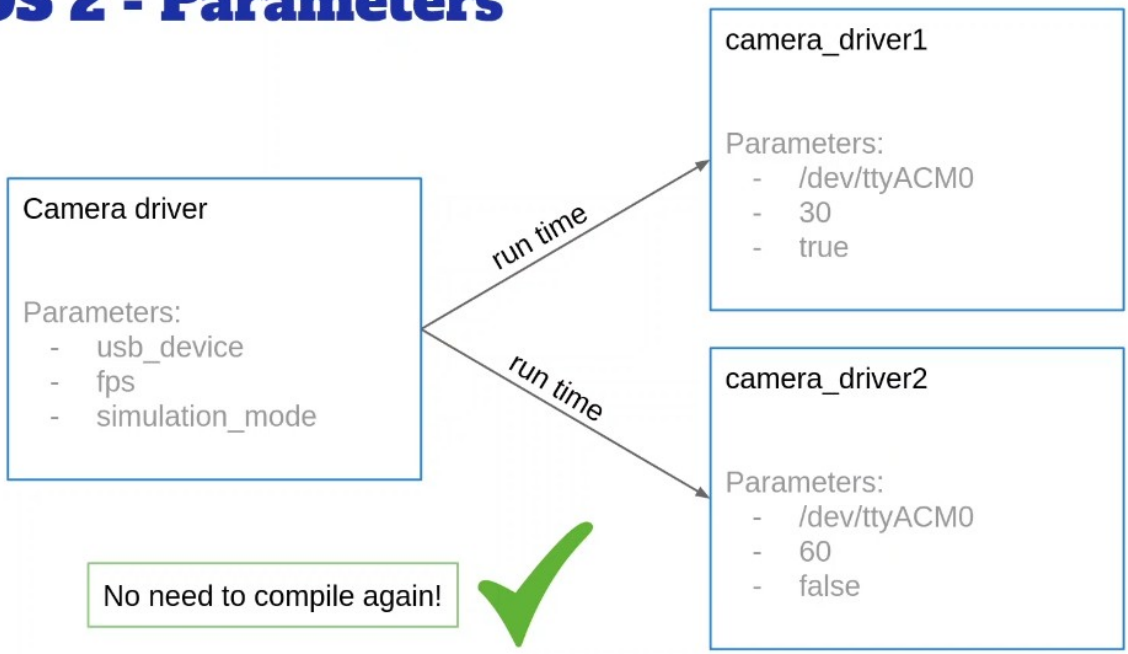
➜ Synchronous or asynchronous

➜ One message type for Request, one message type for Response

➜ Can be written in Python, C++, ... directly inside ROS nodes

➜ A service server can only exist once, but can have many clients

ROS 2 - Parameters



ROS 2 - Parameters

- Settings for your nodes, value set at run time
- A Parameter is specific to a node
- ROS 2 Parameter types:
  - Boolean
  - Int
  - Double
  - String
  - Lists
  - ...