Curso Básico de ROS (Robot Operating System)

Curso #1

Thibaud Hiltenbrand[1]

[1]Pasante en robótica y mecatrónica
SIGMA Clermont, Francia

28/10/2019

Índice

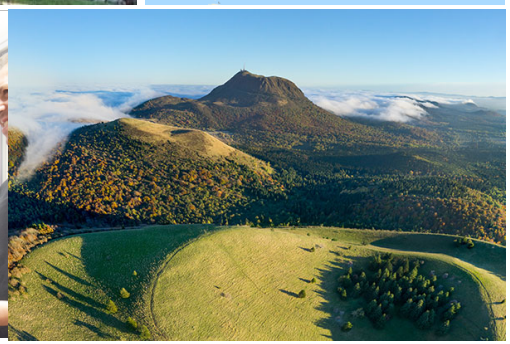
Índice

Presentaciones

- Thibaud, 23 años (\ti.bo\)
- SIGMA Clermont (Clermont-Ferrand, Francia)
- MSc en Maquinas, Mecanismos y Sistemas + Robótica
- Parte de un año opcional al extranjero

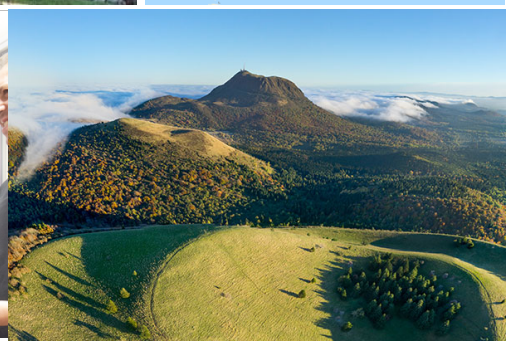
Presentaciones

- Thibaud, 23 años (\ti.bo\)
- SIGMA Clermont (Clermont-Ferrand, Francia)
- MSc en Maquinas, Mecanismos y Sistemas + Robótica
- Parte de un año opcional al extranjero



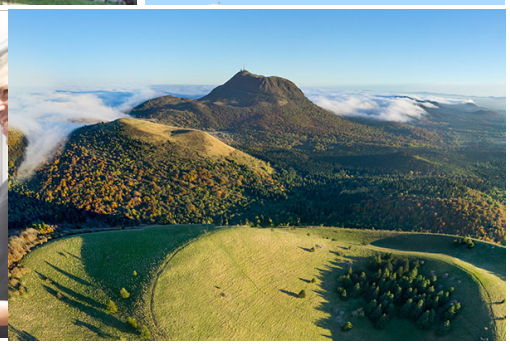
Presentaciones

- Thibaud, 23 años (\ti.bo\)
- SIGMA Clermont (Clermont-Ferrand, Francia)
- MSc en Maquinas, Mecanismos y Sistemas + Robótica
- Parte de un año opcional al extranjero



Presentaciones

- Thibaud, 23 años (\ti.bo\)
- SIGMA Clermont (Clermont-Ferrand, Francia)
- MSc en Maquinas, Mecanismos y Sistemas + Robótica
- Parte de un año opcional al extranjero

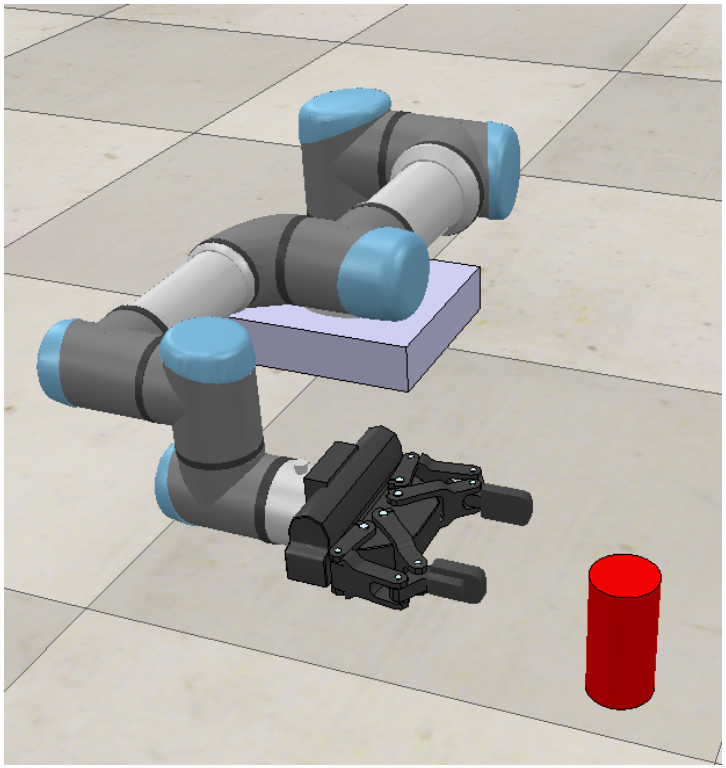


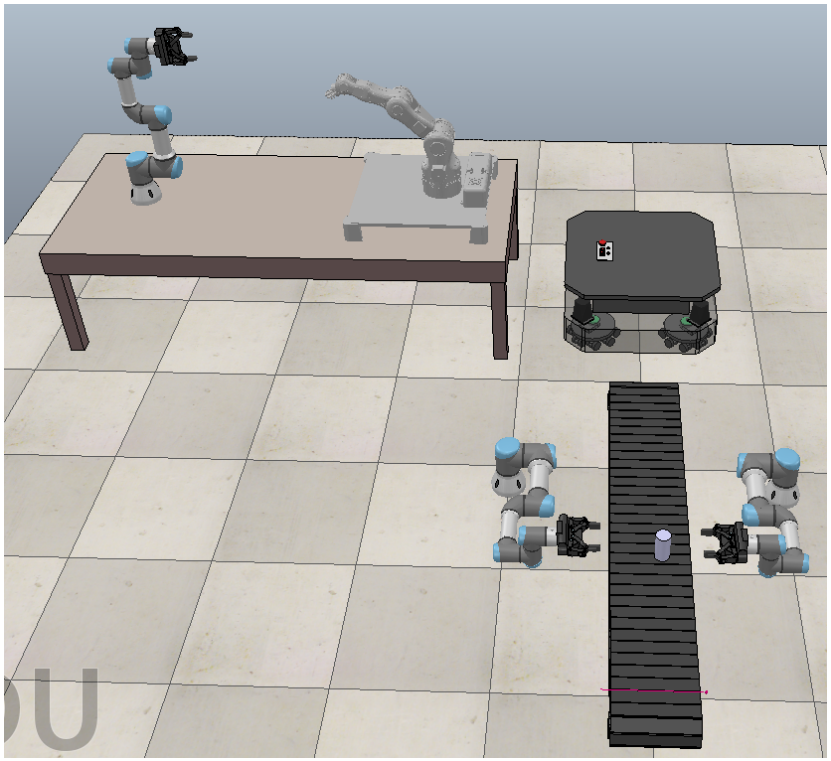
Presentaciones

- Pasantía en el ROSETTA Lab (4 meses)
- Estrategias de colaboración robot-robot en simulación

Presentaciones

- Pasantía en el ROSETTA Lab (4 meses)
- Estrategias de colaboración robot-robot en simulación





Índice

Motivación - Definición

- **Progresos** de la robótica en los últimos años
  - Fiable
  - Económica
  - Aplicaciones múltiples
- **Retos** significativos para el **desarrollo** y la **integración**
- **ROS (Robot Operating System)**

Definición

ROS es un **conjunto de bibliotecas de software y herramientas** que ayudan a crear aplicaciones de robots. Desde drivers a algoritmos actuales, y con potentes herramientas de desarrollo, ROS tiene lo que necesitas para tu próximo proyecto de robótica. Y todo es de **código abierto**.

Motivación - Definición

- **Progresos** de la robótica en los últimos años
  - Fiable
  - Económica
  - Aplicaciones múltiples
- **Retos** significativos para el **desarrollo** y la **integración**
- **ROS (Robot Operating System)**

Definición

ROS es un **conjunto de bibliotecas de software y herramientas** que ayudan a crear aplicaciones de robots. Desde drivers a algoritmos actuales, y con potentes herramientas de desarrollo, ROS tiene lo que necesitas para tu próximo proyecto de robótica. Y todo es de **código abierto**.

Motivación - Definición

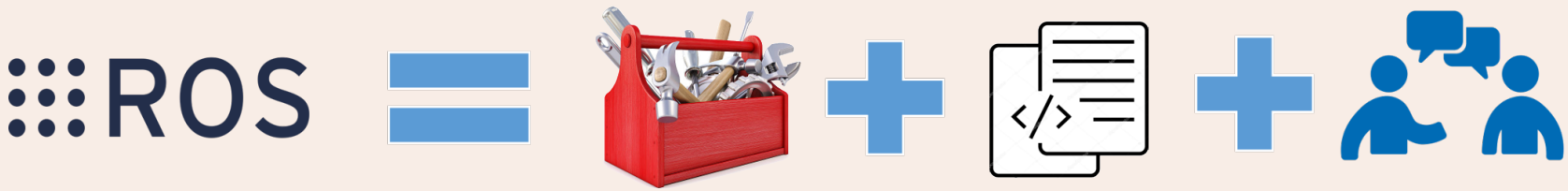
- **Progresos** de la robótica en los últimos años
  - Fiable
  - Económica
  - Aplicaciones múltiples
- **Retos** significativos para el **desarrollo** y la **integración**
- **ROS (Robot Operating System)**

Definición

ROS es un **conjunto de bibliotecas de software y herramientas** que ayudan a crear aplicaciones de robots. Desde drivers a algoritmos actuales, y con potentes herramientas de desarrollo, ROS tiene lo que necesitas para tu próximo proyecto de robótica. Y todo es de **código abierto**.

ROS en una sola frase

La ecuación de ROS



Índice

Algunos proyectos utilizando ROS

- PR2: el principio de ROS (2009)
- iRobot Rumba (2011)
- SoftbanksRobotics Nao (2011)
- RethinkRobotics Baxter (2012)
- ClearpathRobotics Ridgeback (2015)
- Franka Emika Panda (2018)

Algunos proyectos utilizando ROS

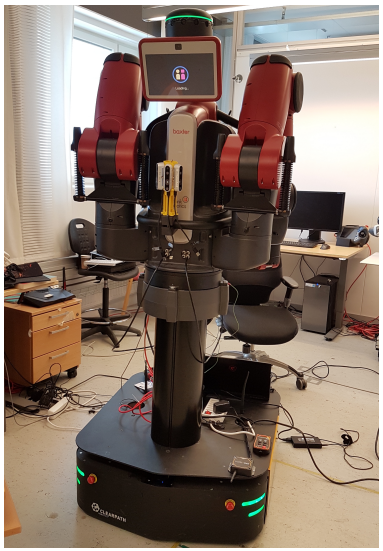
- PR2: el principio de ROS (2009)
- iRobot Rumba (2011)
- SoftbanksRobotics Nao (2011)
- RethinkRobotics Baxter (2012)
- ClearpathRobotics Ridgeback (2015)
- Franka Emika Panda (2018)

Algunos proyectos utilizando ROS

- PR2: el principio de ROS (2009)
- iRobot Rumba (2011)
- SoftbanksRobotics Nao (2011)
- RethinkRobotics Baxter (2012)
- ClearpathRobotics Ridgeback (2015)
- Franka Emika Panda (2018)

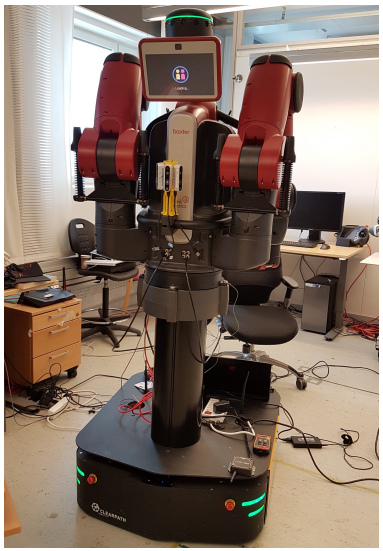
Algunos proyectos utilizando ROS

- PR2: el principio de ROS (2009)
- iRobot Rumba (2011)
- SoftbanksRobotics Nao (2011)
- RethinkRobotics Baxter (2012)
- ClearpathRobotics Ridgeback (2015)
- Franka Emika Panda (2018)



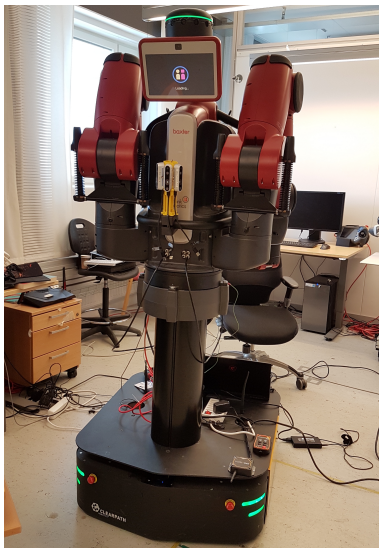
Algunos proyectos utilizando ROS

- PR2: el principio de ROS (2009)
- iRobot Rumba (2011)
- SoftbanksRobotics Nao (2011)
- RethinkRobotics Baxter (2012)
- ClearpathRobotics Ridgeback (2015)
- Franka Emika Panda (2018)



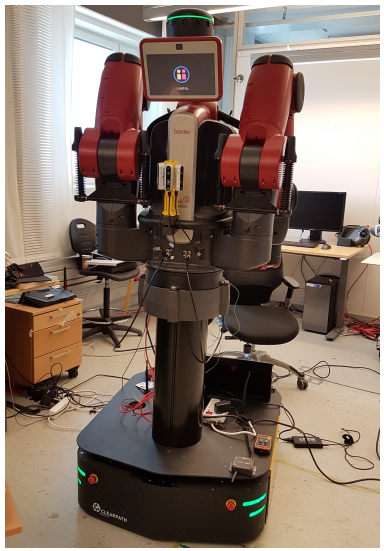
Algunos proyectos utilizando ROS

- PR2: el principio de ROS (2009)
- iRobot Rumba (2011)
- SoftbanksRobotics Nao (2011)
- RethinkRobotics Baxter (2012)
- ClearpathRobotics Ridgeback (2015)
- Franka Emika Panda (2018)



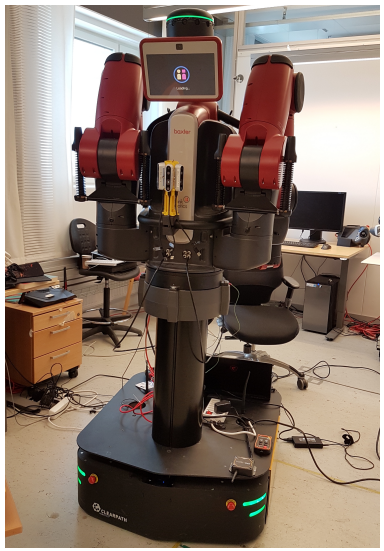
Algunos proyectos utilizando ROS

- PR2: el principio de ROS (2009)
- iRobot Rumba (2011)
- SoftbanksRobotics Nao (2011)
- RethinkRobotics Baxter (2012)
- ClearpathRobotics Ridgeback (2015)
- Franka Emika Panda (2018)



Algunos proyectos utilizando ROS

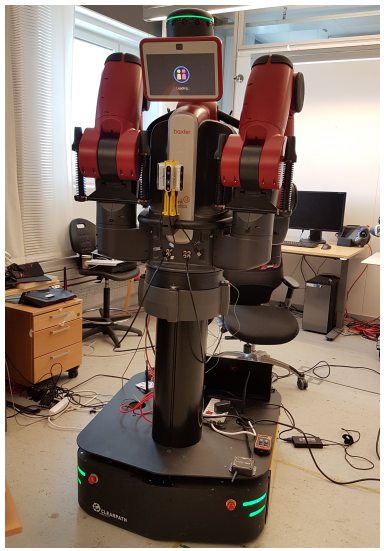
- PR2: el principio de ROS (2009)
- iRobot Rumba (2011)
- SoftbanksRobotics Nao (2011)
- RethinkRobotics Baxter (2012)
- ClearpathRobotics Ridgeback (2015)
- Franka Emika Panda (2018)



GAZEBO

Algunos proyectos utilizando ROS

- PR2: el principio de ROS (2009)
- iRobot Rumba (2011)
- SoftbanksRobotics Nao (2011)
- RethinkRobotics Baxter (2012)
- ClearpathRobotics Ridgeback (2015)
- Franka Emika Panda (2018)



GAZEBO

YOLO

Índice

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS puede ayudarte en...

- **Computación distribuida**
  - Varios ordenadores
  - Dividir el software del robot
  - Cooperación entre robots
  - Interfaz con los usuarios
- **Reutilización de software**
  - Algoritmos para tareas comunes
  - Paquetes estándar de ROS
  - Interfaz de paso de mensajes de ROS
- **Pruebas funcionales**
  - Simulación bajo nivel / desarrollo alto nivel
  - Repetición y pruebas con datos reales
  - Similitud entre simulación y sistema real

ROS no es...

- ROS no es un lenguaje de programación.
- ROS no es (sólo) una biblioteca.
- ROS no es un entorno de desarrollo (IDE).

ROS no es...

- ROS no es un lenguaje de programación.
- ROS no es (sólo) una biblioteca.
- ROS no es un entorno de desarrollo (IDE).

ROS no es...

- ROS no es un lenguaje de programación.
- ROS no es (sólo) una biblioteca.
- ROS no es un entorno de desarrollo (IDE).

Ventajas e inconvenientes

Ventajas

- Paralelismo
- Ubicuidad
- Modularidad
- Lenguaje de programación

Inconvenientes

- Filosofía de desarrollo
- Desorden en el desarrollo
- Linux centrado
- Aprendizaje

Ventajas e inconvenientes

Ventajas

- Paralelismo
- Ubicuidad
- Modularidad
- Lenguaje de programación

Inconvenientes

- Filosofía de desarrollo
- Desorden en el desarrollo
- Linux centrado
- Aprendizaje

Índice

Entorno de desarollo

- ROS Kinetic // Ubuntu 16.04 (~~Windows~~)
  - Sistema operativo de código abierto
  - Basado en Linux
- Herramientas esenciales:
  - Terminal
  - Editor de texto (**VSCode**, Gedit, Eclipse, etc.)

Entorno de desarollo

- ROS Kinetic // Ubuntu 16.04 (~~Windows~~)
  - Sistema operativo de código abierto
  - Basado en Linux
- Herramientas esenciales:
  - Terminal
  - Editor de texto (**VSCode**, Gedit, Eclipse, etc.)

Comandos básicos - Presentación

Comandos básicos de Linux y ROS

Comando (Ubuntu)	Comando (ROS)	Descripción / Uso
`ls`	`rosls`	Enumera archivos y carpetas
`cd <carpeta>`	`roscd <carpeta>`	Cambia el directorio de trabajo a <carpeta>
`mkdir <carpeta>`		Crea un directorio
`sudo`		Ejecuta el comando como usuario root (admin)
`sudo apt install <paquete>`		Instala un paquete
`<Tab>`	`<Tab>`	Tab-completion
	`rospack find <package>`	Obtener información sobre los paquetes (acá, ruta al paquete)

Comandos básicos - Práctica (1/7)

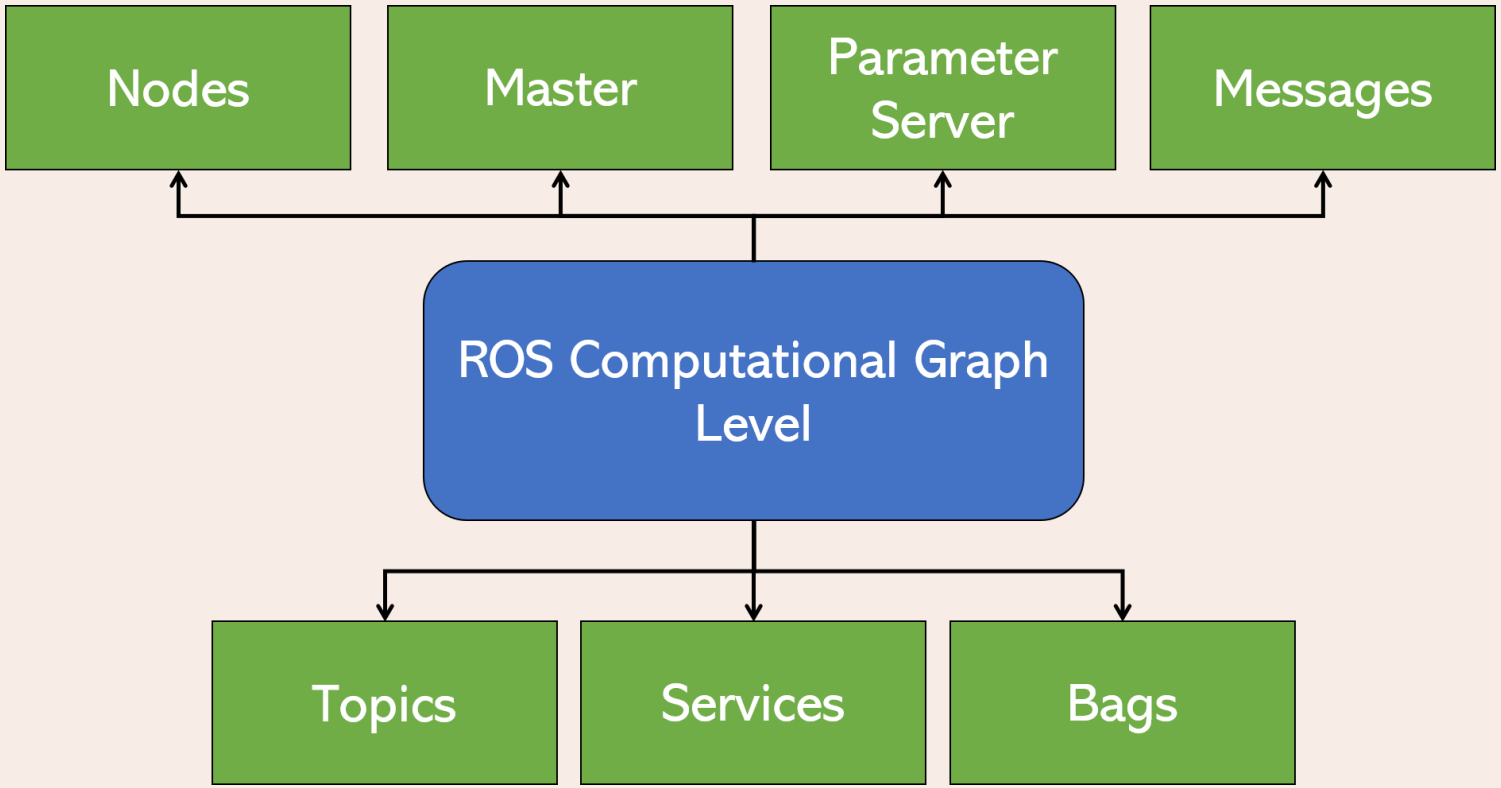
¡Tiempo de práctica!

- Acceder a la maquina virtual
- Primero contacto con Ubuntu
- Sistema de archivos de ROS

Índice

Arquitectura de ROS - Vista general

Concepto de *Graphs* en ROS

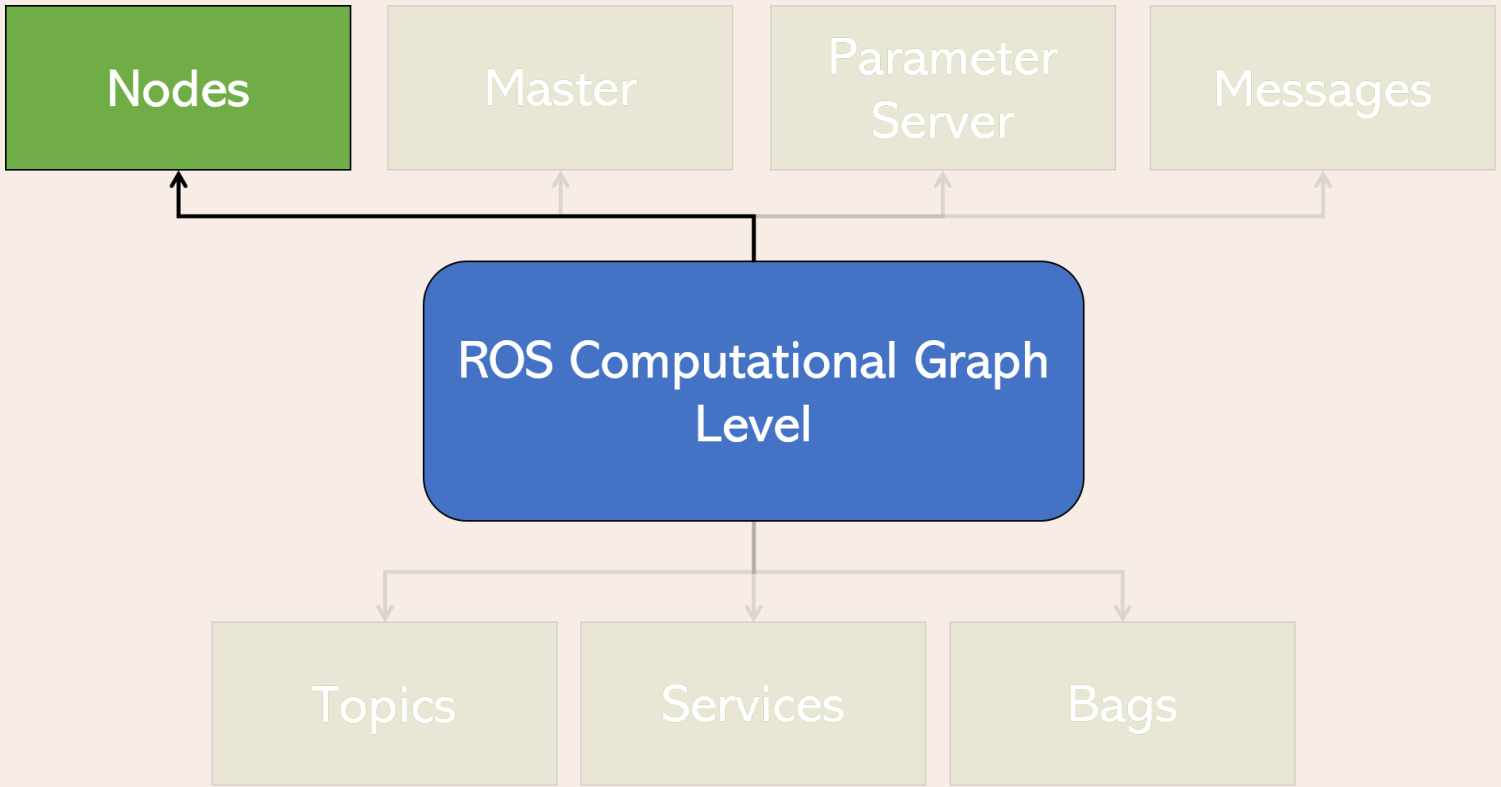


Planteo del problema en general

- Quiero...
  - comunicar entre **varios sensores y robots**
  - en un **entorno heterogéneo**
  - sin conocer las **particulares de comunicación** entre cada elementos

Arquitectura de ROS - *Nodes*

Arquitectura de *Graphs* en ROS (1/7)



Arquitectura de ROS - *Nodes*

Definición

Entidad esencial de ROS, un *node* es un **programa ejecutable** con un solo propósito. Interactúan entre sí para formar un **sistema robótico**.

Ejemplos

Controlo de un **telémetro láser**, de los **motores** de rueda del robot, **localización**, **planificación** de rutas, **vista gráfica** del sistema, etc.

Arquitectura de ROS - *Nodes*

Definición

Entidad esencial de ROS, un *node* es un **programa ejecutable** con un solo propósito. Interactúan entre sí para formar un **sistema robótico**.

Ejemplos

Controlo de un **telémetro láser**, de los **motores** de rueda del robot, **localización**, **planificación** de rutas, **vista gráfica** del sistema, etc.

Arquitectura de ROS - Práctica (2/7)

¡Tiempo de práctica!

- Primera ejecución de ROS
  - `roscore (ros + core)`
- Comprensión de los nodos
  - `rosnode (ros + node)`
- Lanzamiento de un node
  - rosrun (`ros + run`)

Arquitectura de ROS - `roscore`

`roscore` = Master + `rosout` + parameter server

Master

- Nombre del servicio por ROS
- Ayuda a los nodos a encontrarse entre sí

`rosout`

- Salida de consola

Parameter server

- Diccionario para almacenar y recuperar parámetros

Arquitectura de ROS - `roscore`

`roscore` = Master + `rosout` + parameter server

Master

- Nombre del servicio por ROS
- Ayuda a los nodos a encontrarse entre sí

`rosout`

- Salida de consola

Parameter server

- Diccionario para almacenar y recuperar parámetros

Arquitectura de ROS - `roscore`

`roscore` = Master + `rosout` + parameter server

Master

- Nombre del servicio por ROS
- Ayuda a los nodos a encontrarse entre sí

`rosout`

- Salida de consola

Parameter server

- Diccionario para almacenar y recuperar parámetros

Arquitectura de ROS - `roscore`

`roscore` = Master + `rosout` + parameter server

Master

- Nombre del servicio por ROS
- Ayuda a los nodos a encontrarse entre sí

`rosout`

- Salida de consola

Parameter server

- Diccionario para almacenar y recuperar parámetros

Arquitectura de ROS - *Nodes*

- Compilado, ejecutado y gestionado **individualmente**
- Se escriben utilizando una **biblioteca cliente ROS**
  - `roscpp` (biblioteca `C++`)
  - `rospy` (biblioteca `Python`)

Arquitectura de ROS - *Nodes*

- Compilado, ejecutado y gestionado **individualmente**
- Se escriben utilizando una **biblioteca cliente ROS**
  - **`roscpp`** (biblioteca `C++`)
  - **`rospy`** (biblioteca `Python`)

Arquitectura de ROS - Comunicación entre *Nodes*

- Dos posibilidades:

Topic

Publicación o subscripción a un *Topic*

Service / Action

Proporcionar o utilizar un *Service* o *Action*

Arquitectura de ROS - Comunicación entre *Nodes*

- Dos posibilidades:

Topic

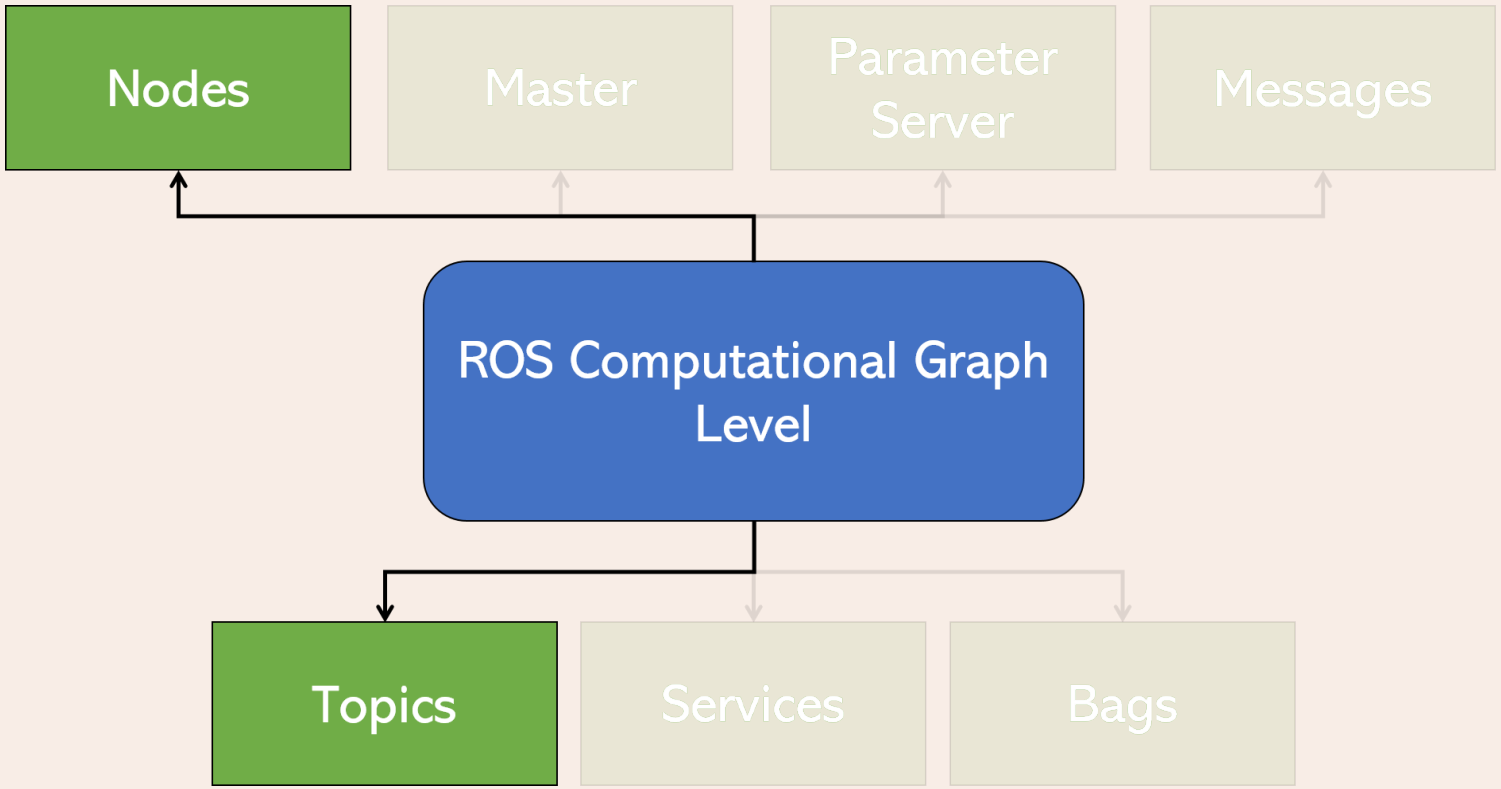
Publicación o subscripción a un *Topic*

Service / Action

Proporcionar o utilizar un *Service* o *Action*

Arquitectura de ROS - *Topics*

Arquitectura de *Graphs* en ROS (2/7)



Arquitectura de ROS - *Topics*

Definición

Flujo de mensajes con un **nombre** y un **tipo definido**.

Ejemplo

Envio de los datos de un **telémetro laser** a un ***Topic*** llamado **'scan'**, con un mensaje del tipo **'LaserScan'**

Arquitectura de ROS - *Topics*

Definición

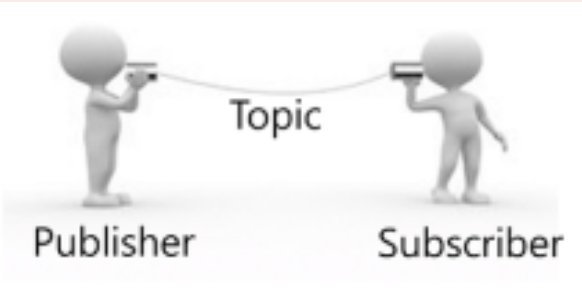
Flujo de mensajes con un **nombre** y un **tipo definido**.

Ejemplo

Envio de los datos de un **telémetro laser** a un ***Topic*** llamado **'scan'**, con un mensaje del tipo **'LaserScan'**

Arquitectura de ROS - *Topics*

Modelo *Publish/Subscribe*: *1-to-1 broadcasting*



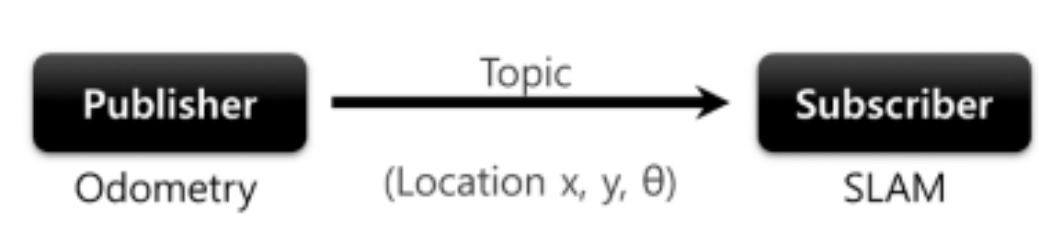
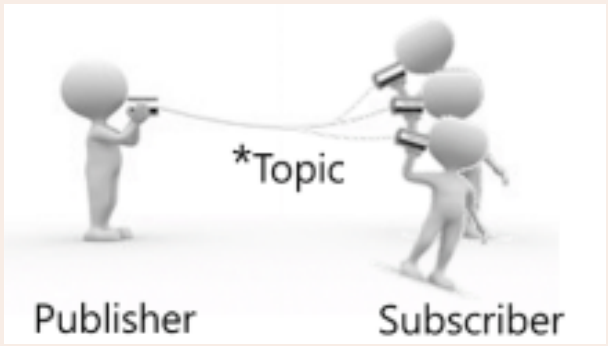


Figura: Comunicación *Topic-Message*

Arquitectura de ROS - *Topics*

Modelo *Publish/Subscribe*: *1-to-N broadcasting*



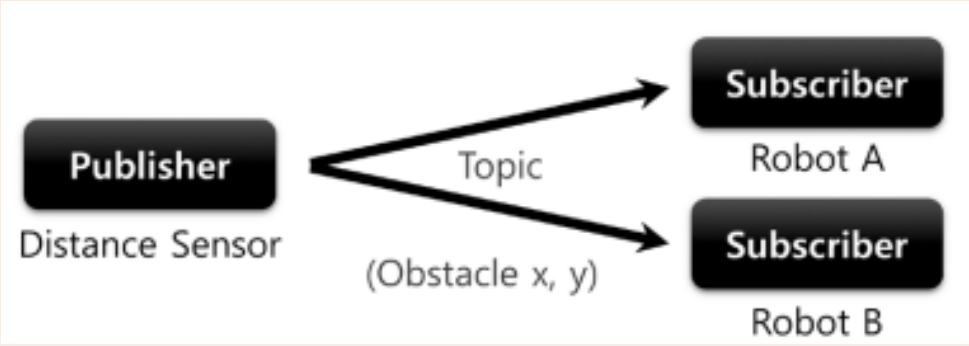


Figura: Comunicación *Topic-Message*

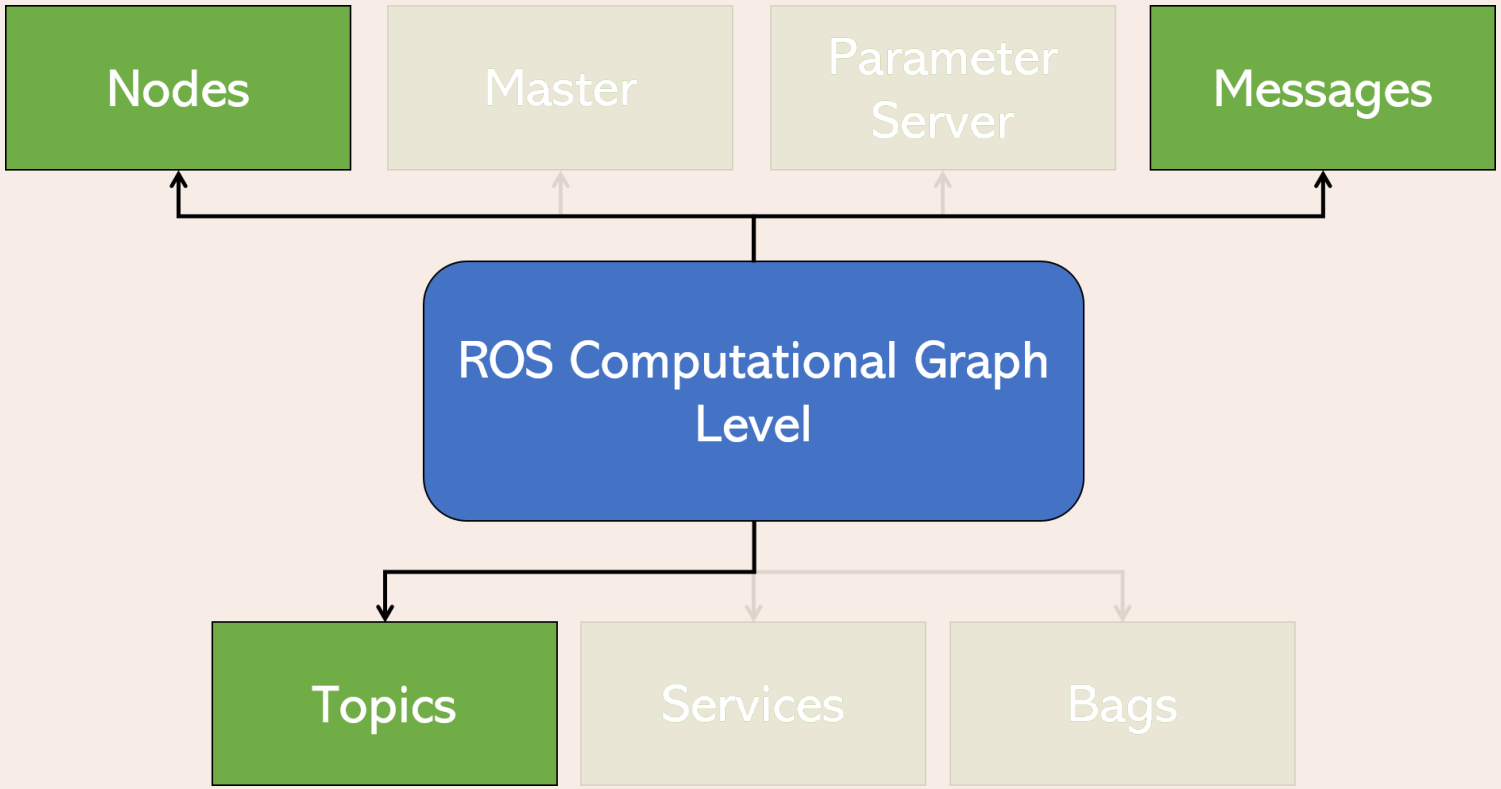
Arquitectura de ROS - Práctica (3/7)

¡Tiempo de práctica!

- Comunicación entre nodos
  - `rostopic (ros + topic)`

Arquitectura de ROS - *Messages*

Concepto de *Graphs* en ROS (3/7)



Arquitectura de ROS - *Messages*

Definición

Estructuras de datos de un **tipo especifico** para la **comunicación entre *Nodes***

Ejemplo

geometry_msgs/Twist usado para expresar comandos de velocidad:

geometry_msgs/Twist
Vector3 linear
Vector3 angular

donde

Vector3
float64 x
float64 y
float64 z

Arquitectura de ROS - *Messages*

Definición

Estructuras de datos de un **tipo especifico** para la **comunicación entre *Nodes***

Ejemplo

geometry_msgs/Twist usado para expresar comandos de velocidad:

geometry_msgs/Twist
Vector3 linear
Vector3 angular

donde

Vector3
float64 x
float64 y
float64 z

Arquitectura de ROS - *Messages*

Ejemplo

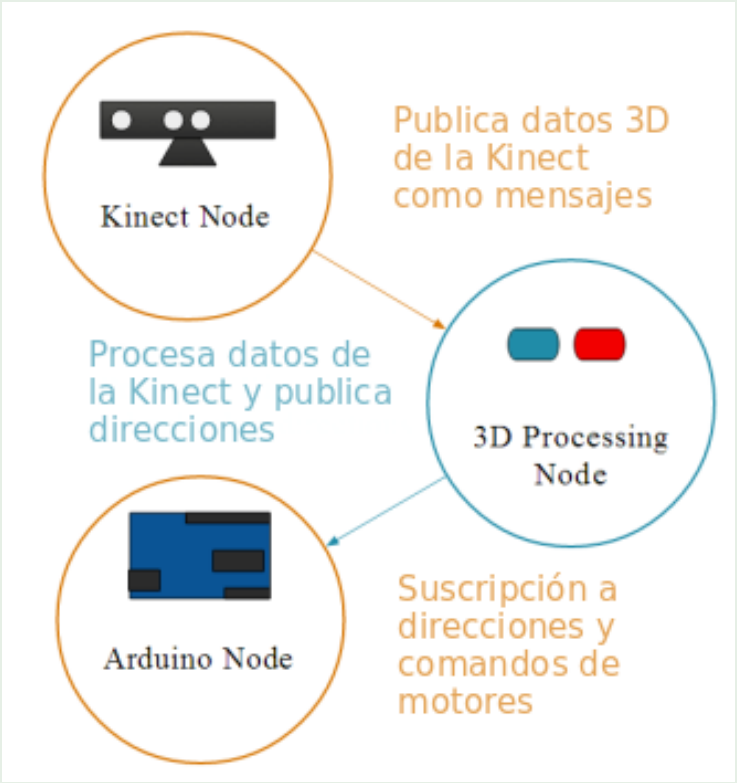


Figura: Ejemplo de tratamiento de datos de una camera Kinect

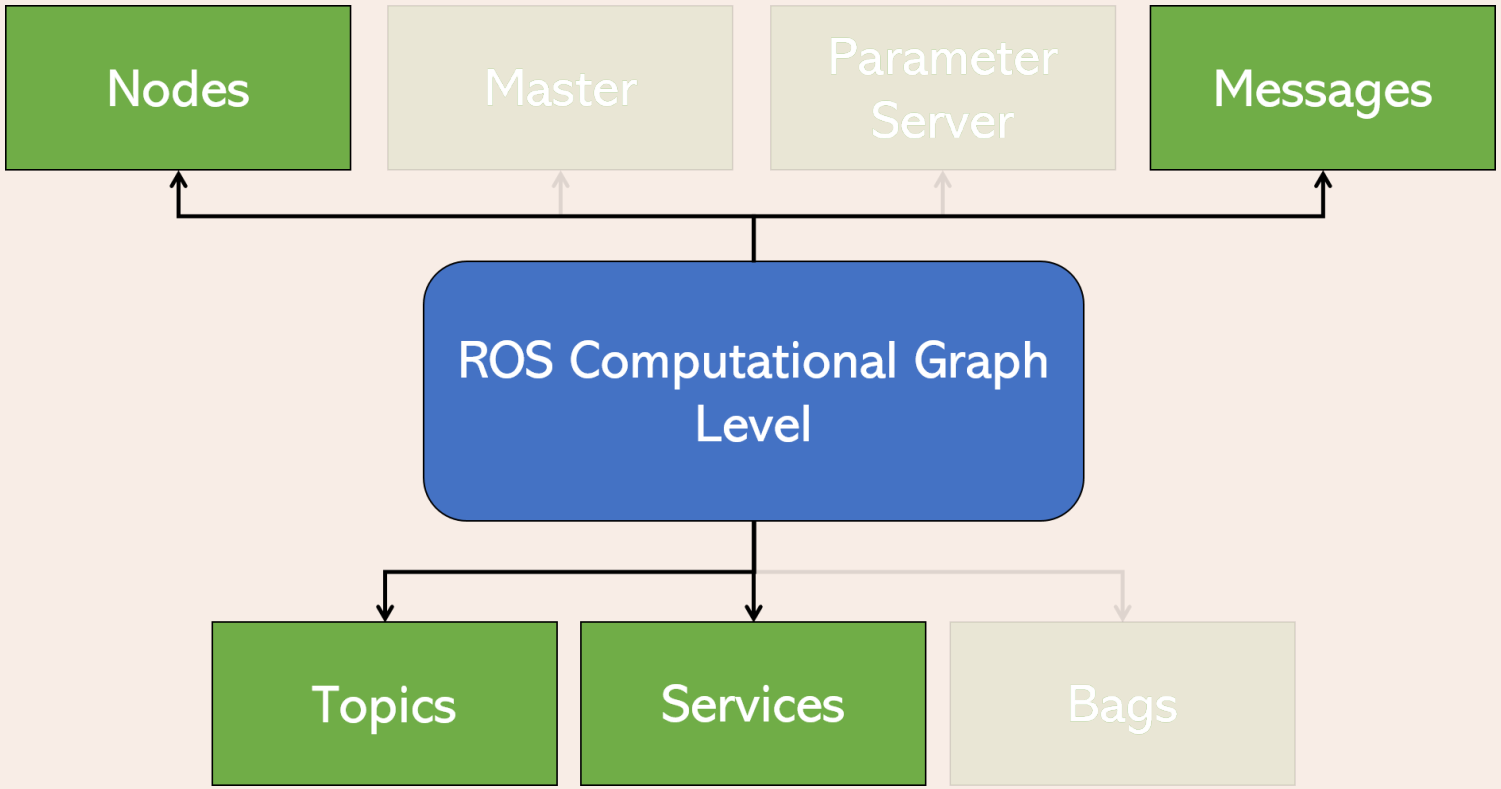
Arquitectura de ROS - Práctica (4/7)

¡Tiempo de práctica!

- Topics y Messages
  - `rostopic (ros + topic)`

Arquitectura de ROS - *Services*

Concepto de *Graphs* en ROS (4/7)



Arquitectura de ROS - *Services*

Definición

Transacciones síncronas entre *Nodes*: pedir algo (*request*) y esperar la respuesta (*response*).

Ejemplo

Acciones discretas: encender un sensor, tomar una fotografía de alta resolución con una cámara, etc.

Arquitectura de ROS - *Services*

Definición

Transacciones síncronas entre *Nodes*: pedir algo (*request*) y esperar la respuesta (*response*).

Ejemplo

Acciones discretas: encender un sensor, tomar una fotografía de alta resolución con una cámara, etc.

Arquitectura de ROS - *Services*

Modelo Service/Client: 1-to-1 request-response

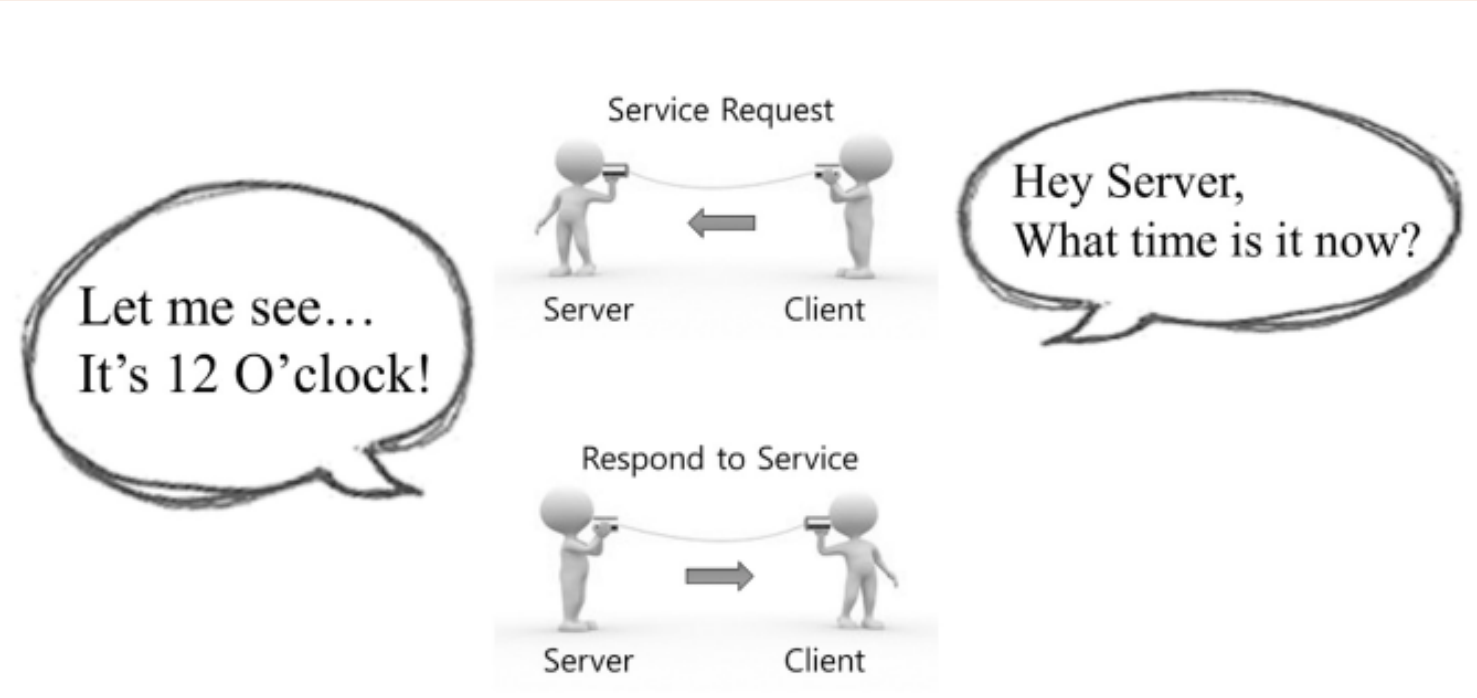


Figura: Comunicación Service-Message

Arquitectura de ROS - *Action*

Definición

Mismo método de transmisión de mensajes que el *topic*, añadiendo información sobre la realización de la tarea (*feedback*)

Ejemplo

Comandar tareas complejas: cancelación del objetivo de navegación mientras la operación está en curso

Arquitectura de ROS - *Action*

Definición

Mismo método de transmisión de mensajes que el *topic*, añadiendo información sobre la realización de la tarea (*feedback*)

Ejemplo

Comandar tareas complejas: cancelación del objetivo de navegación mientras la operación está en curso

Arquitectura de ROS - *Action*

Modelo Action/Client: 1-to-1 goal-feedback-result

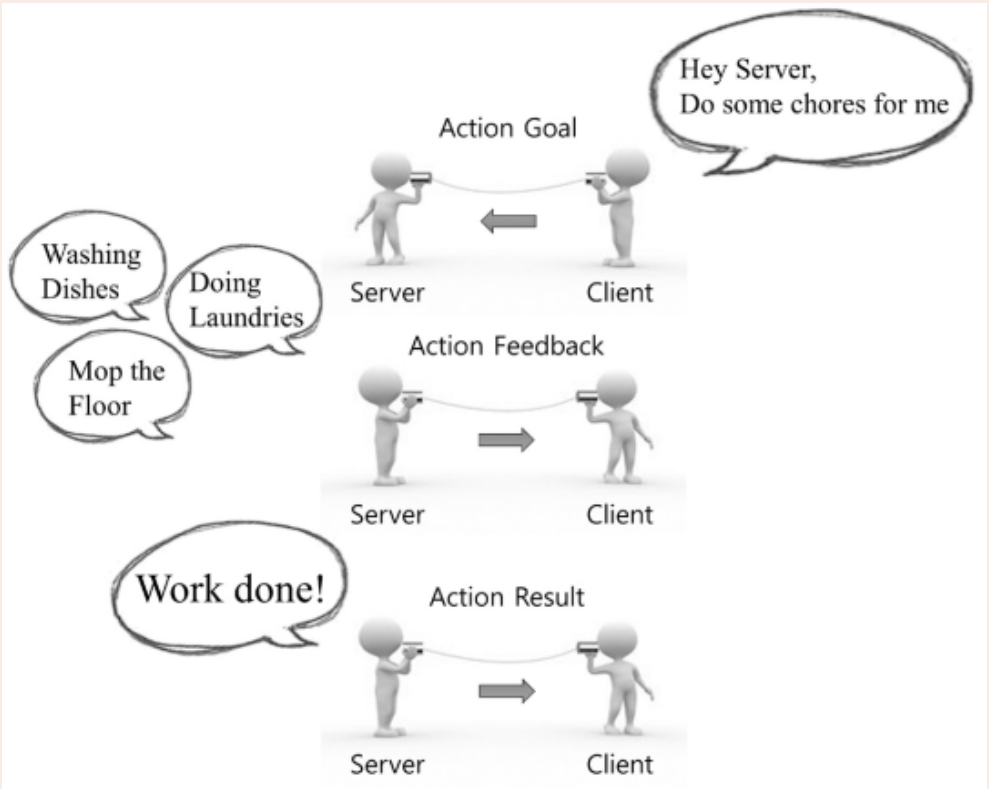


Figura: Comunicación Action-Message

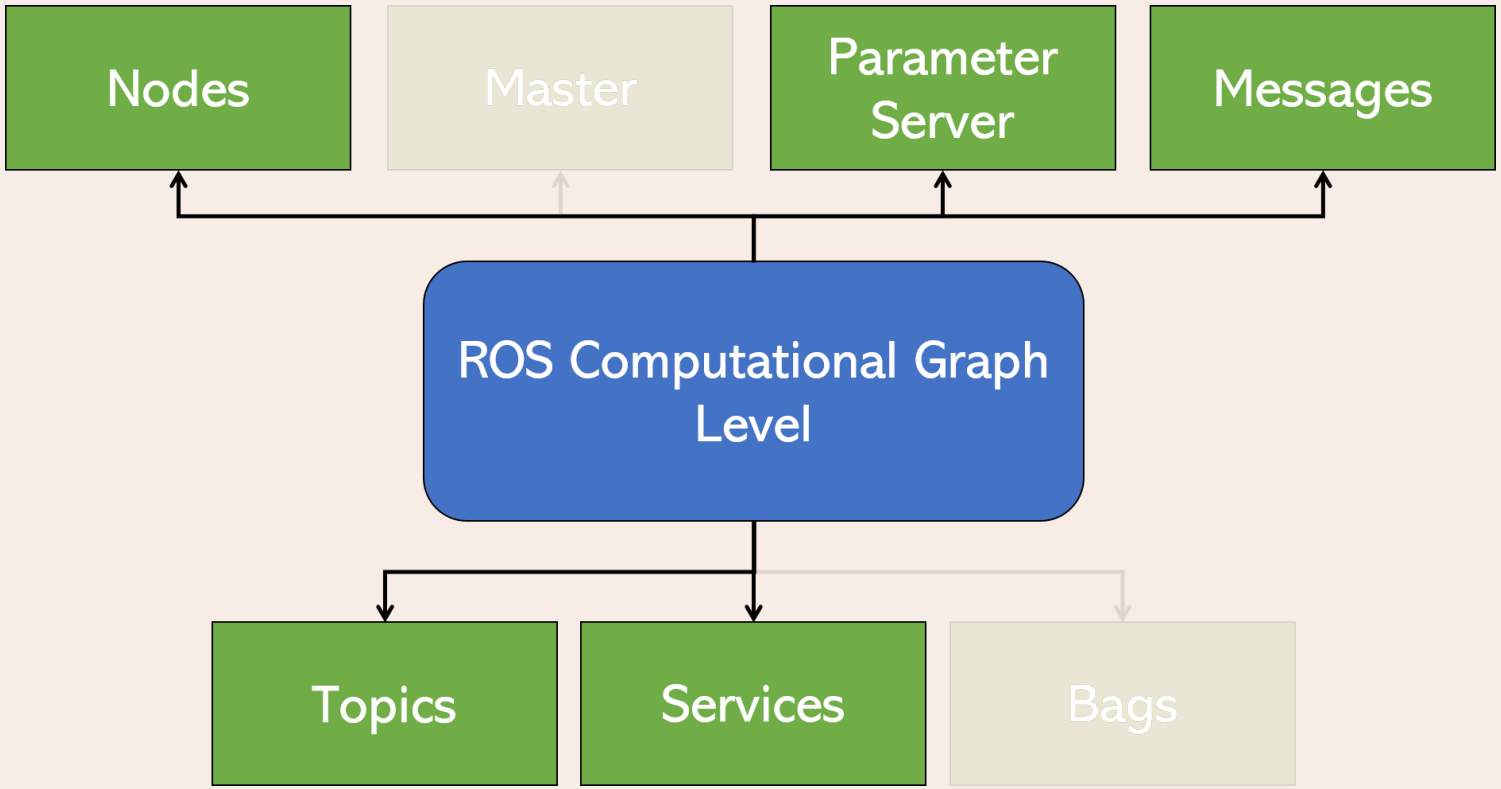
Arquitectura de ROS - Práctica (5/7)

¡Tiempo de práctica!

- ROS Services
  - `rosservice (ros + service)`

Arquitectura de ROS - *Parameters server*

Concepto de *Graphs* en ROS (5/7)



Arquitectura de ROS - *Parameters server*

Definición

Diccionario compartido y multivariable al que se puede acceder a través de ROS (gracias al ROS Master)

Ejemplo

Se utiliza mejor para datos estáticos y no binarios, como parámetros de configuración (longitud, masas, etc.)

Arquitectura de ROS - *Parameters server*

Definición

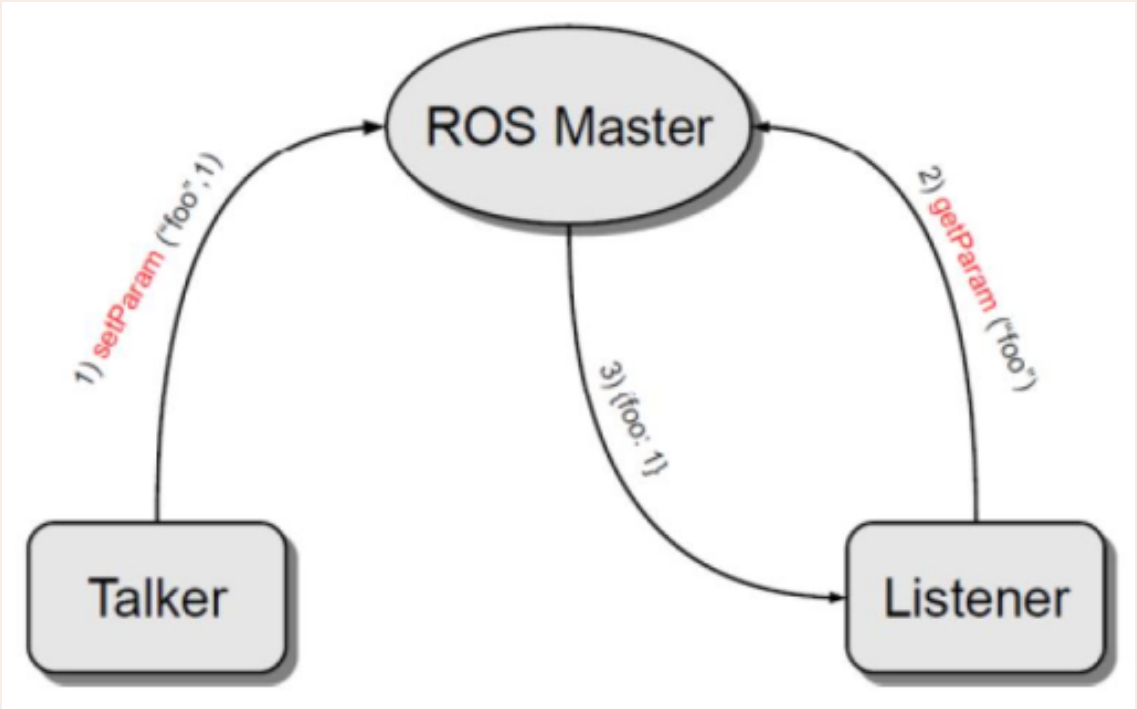
Diccionario compartido y multivariable al que se puede acceder a través de ROS (gracias al ROS Master)

Ejemplo

Se utiliza mejor para datos estáticos y no binarios, como parámetros de configuración (longitud, masas, etc.)

Arquitectura de ROS - *Parameters server*

Funcionamiento del ROS Param Server



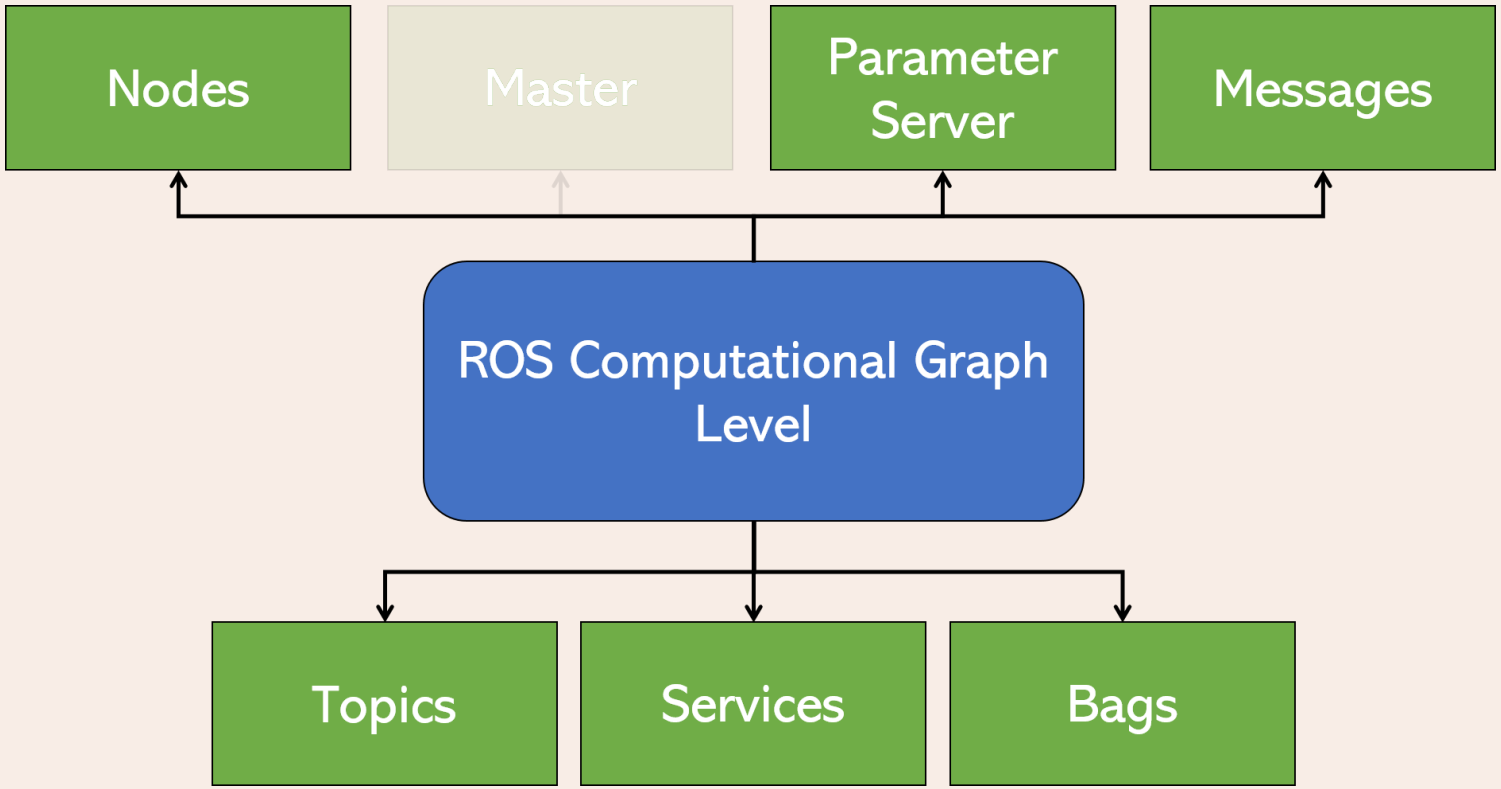
Arquitectura de ROS - Práctica (6/7)

¡Tiempo de práctica!

- ROS Parameter Server
  - `rosparam (ros + param)`

Arquitectura de ROS - *Bags*

Concepto de *Graphs* en ROS (6/7)



Arquitectura de ROS - *Bags*

Definición

Formato para guardar y reproducir datos de mensajes de ROS.

Ejemplo

Almacenar datos, como datos de sensores, para desarrollar y probar algoritmos.

Arquitectura de ROS - *Bags*

Definición

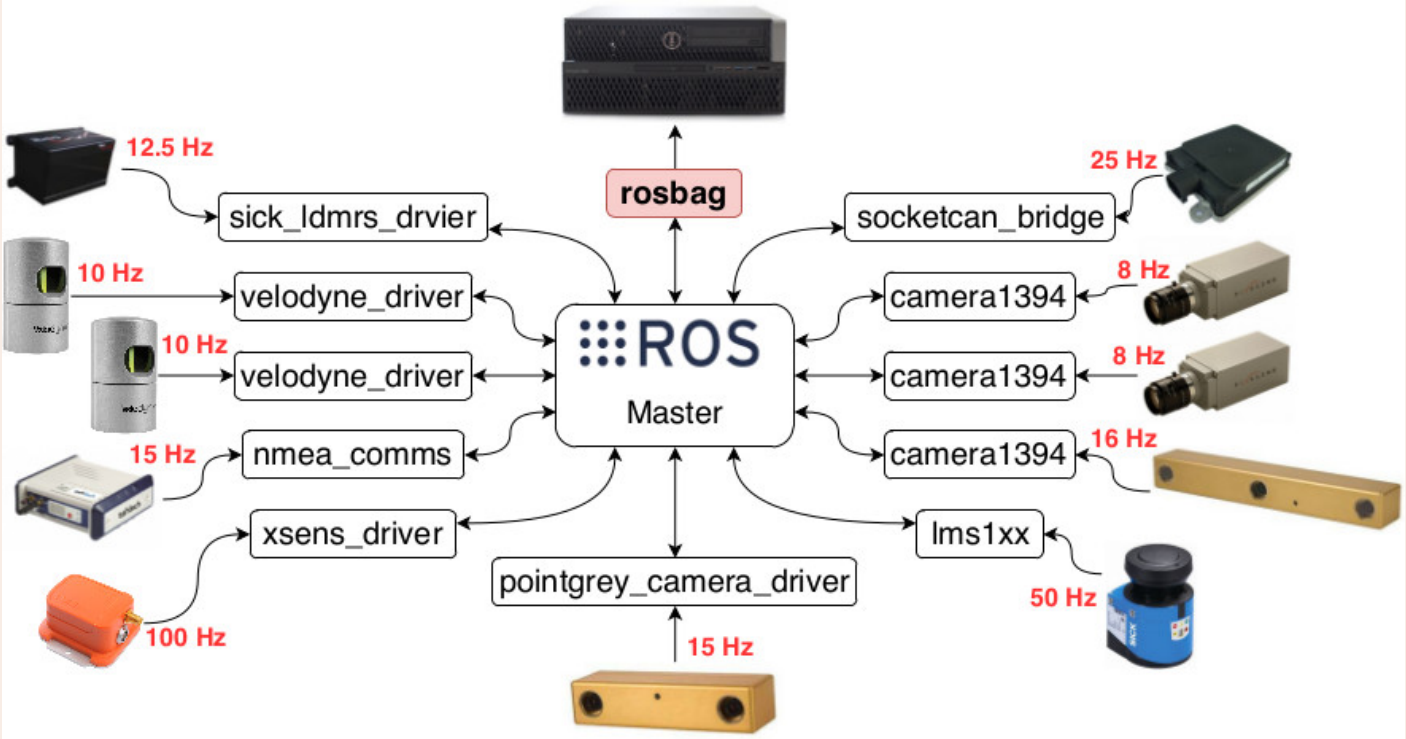
Formato para guardar y reproducir datos de mensajes de ROS.

Ejemplo

Almacenar datos, como datos de sensores, para desarrollar y probar algoritmos.

Arquitectura de ROS - *Bags*

Funcionamiento



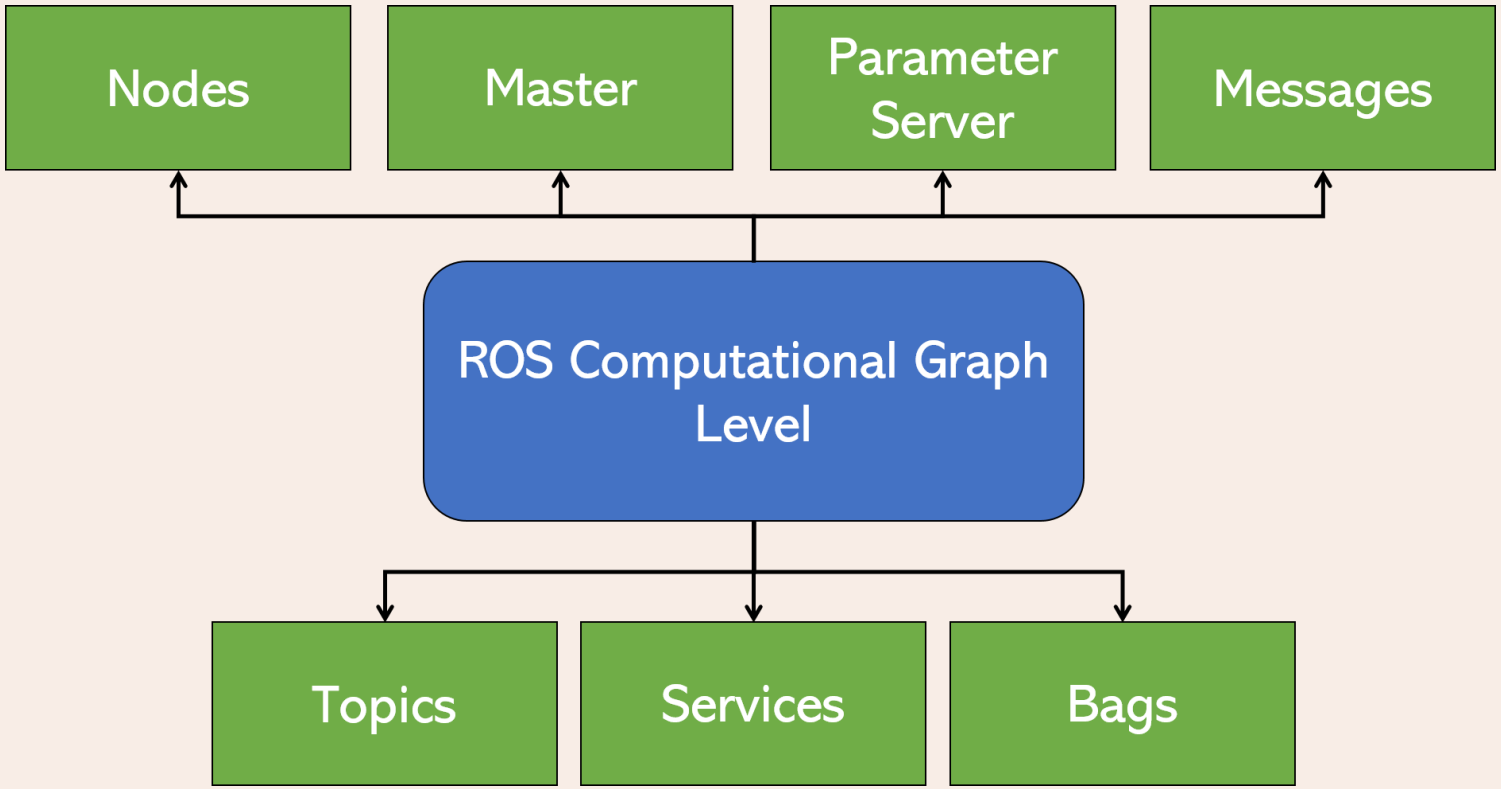
Arquitectura de ROS - Práctica (7/7)

¡Tiempo de práctica!

- Grabación y reproducción de datos
  - `rosbag (ros + bag)`

Arquitectura de ROS - *Master*

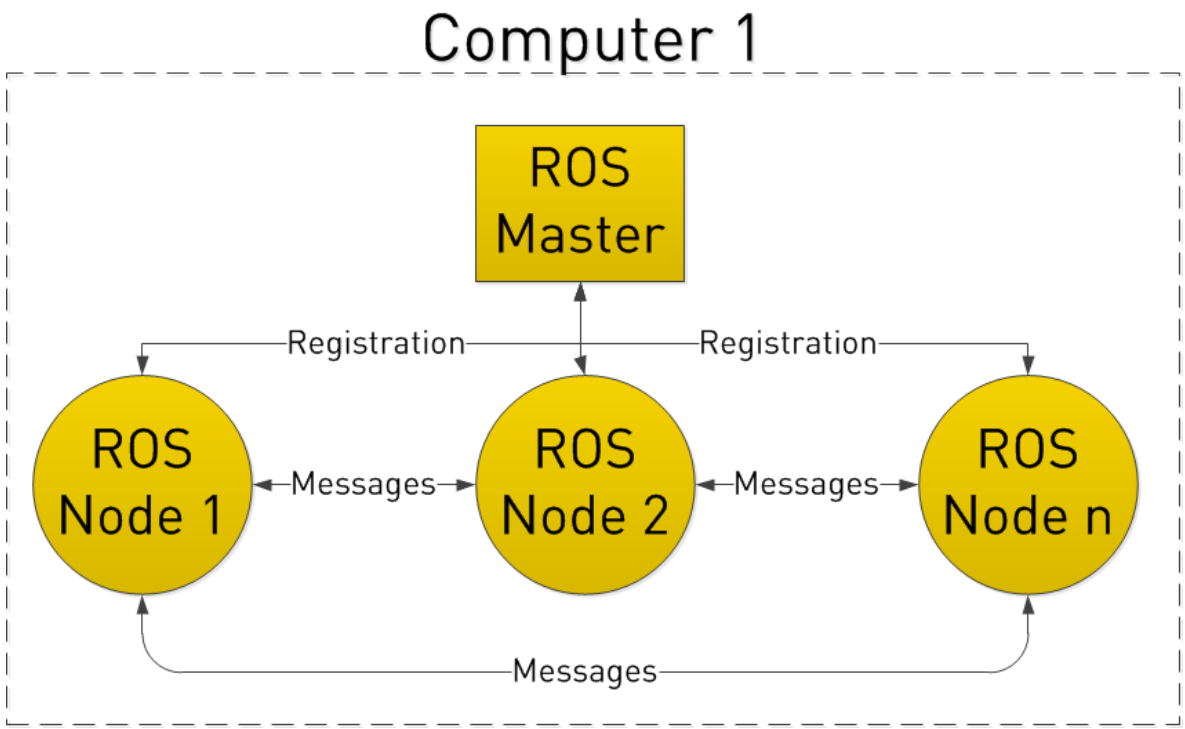
Concepto de *Graphs* en ROS (7/7)



Arquitectura de ROS - *Master*

Definición

Brinda información de conexión a los nodos para que puedan transmitirse mensajes entre sí.



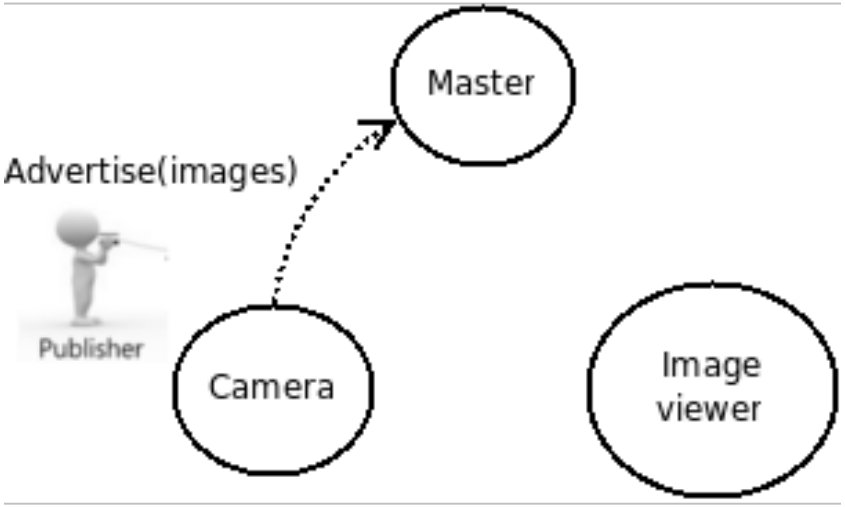
Arquitectura de ROS - *Master*

Ejemplo

Dos Nodes: Camera + Image_viewer

1. Advertise

Camera notifica al Master que quiere publicar imágenes



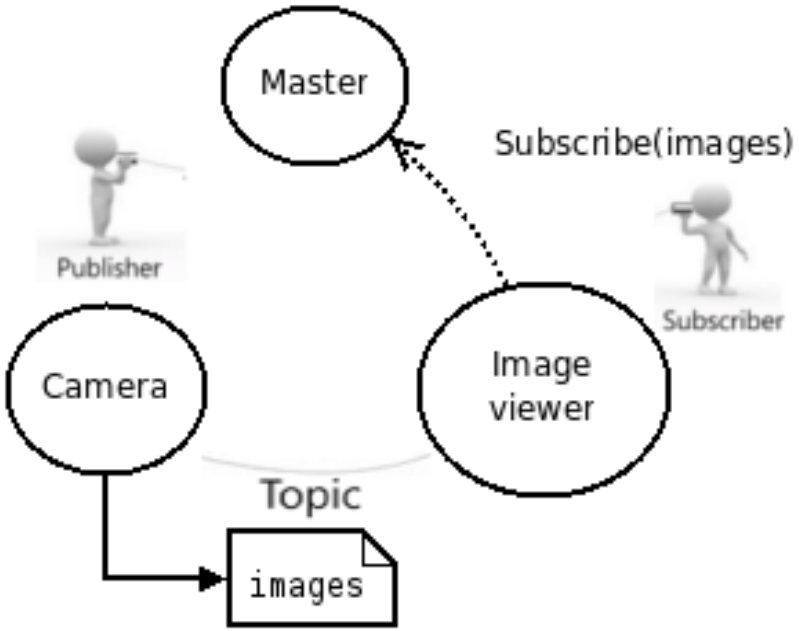
Arquitectura de ROS - *Master*

Ejemplo

Dos Nodes: Camera + Image_viewer

2. Publicación

Camera publica en el Topic "images"



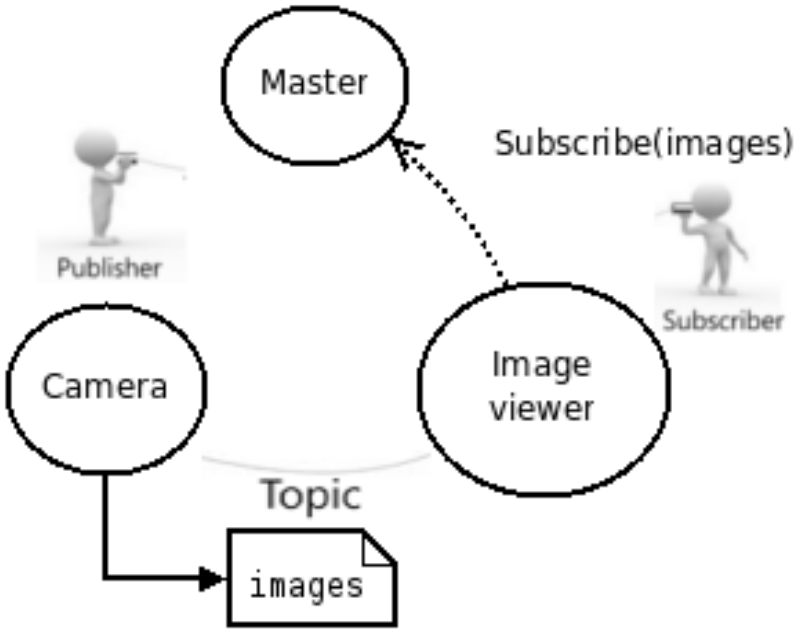
Arquitectura de ROS - *Master*

Ejemplo

Dos Nodes: Camera + Image_viewer

3. Subscriber

“image_viewer” quiere suscribirse al Topic ”images”



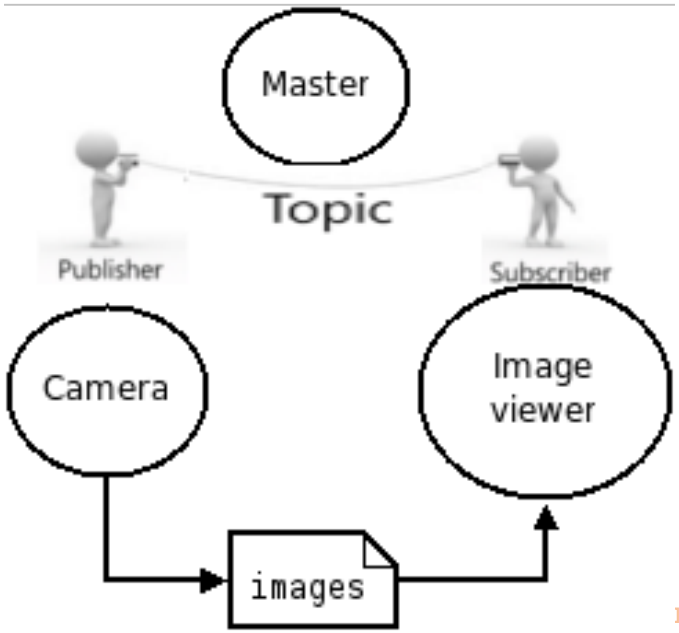
Arquitectura de ROS - *Master*

Ejemplo

Dos Nodes: Camera + Image_viewer

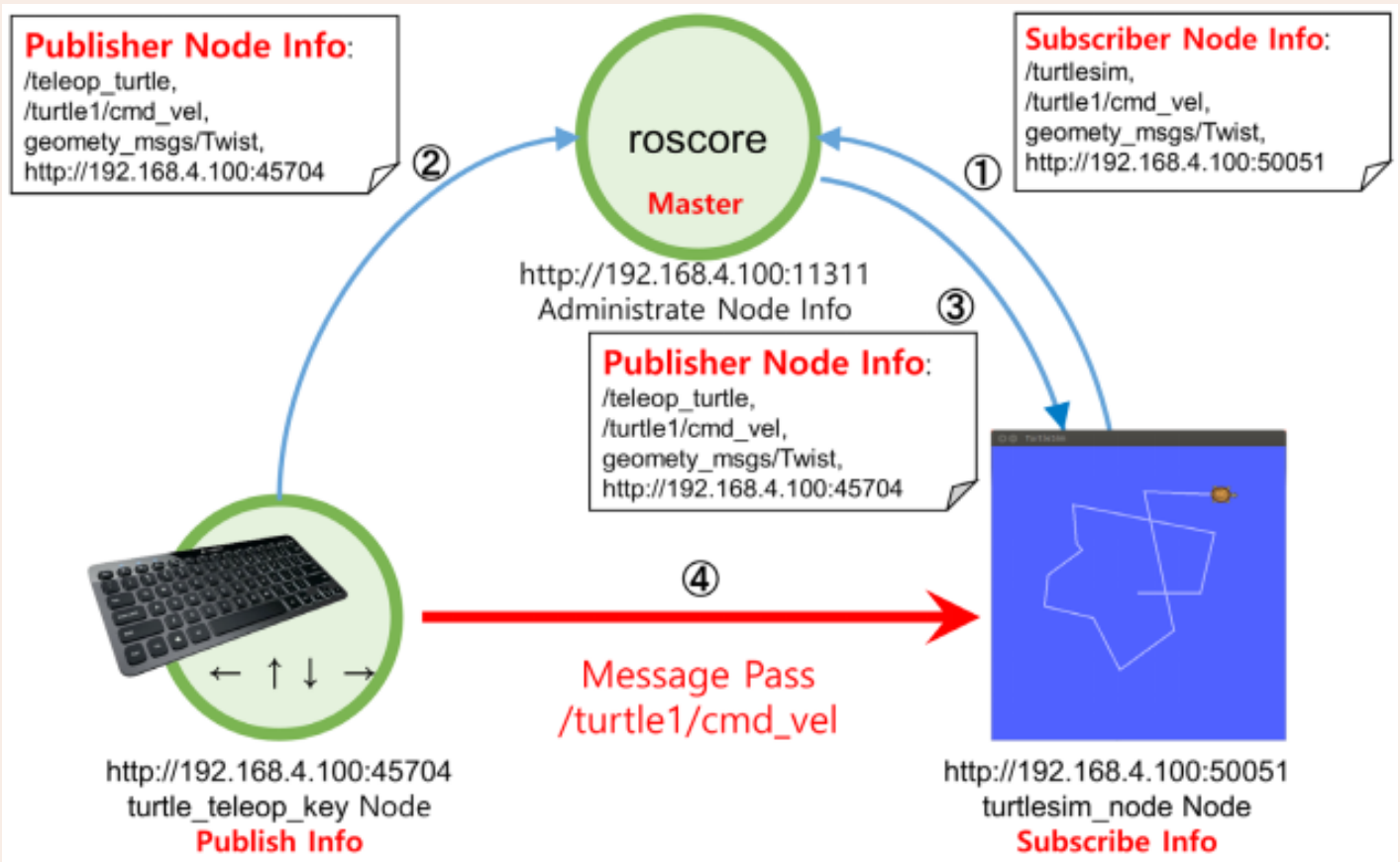
4. Comunicación

Nodes pueden ahora comunicar entre sí



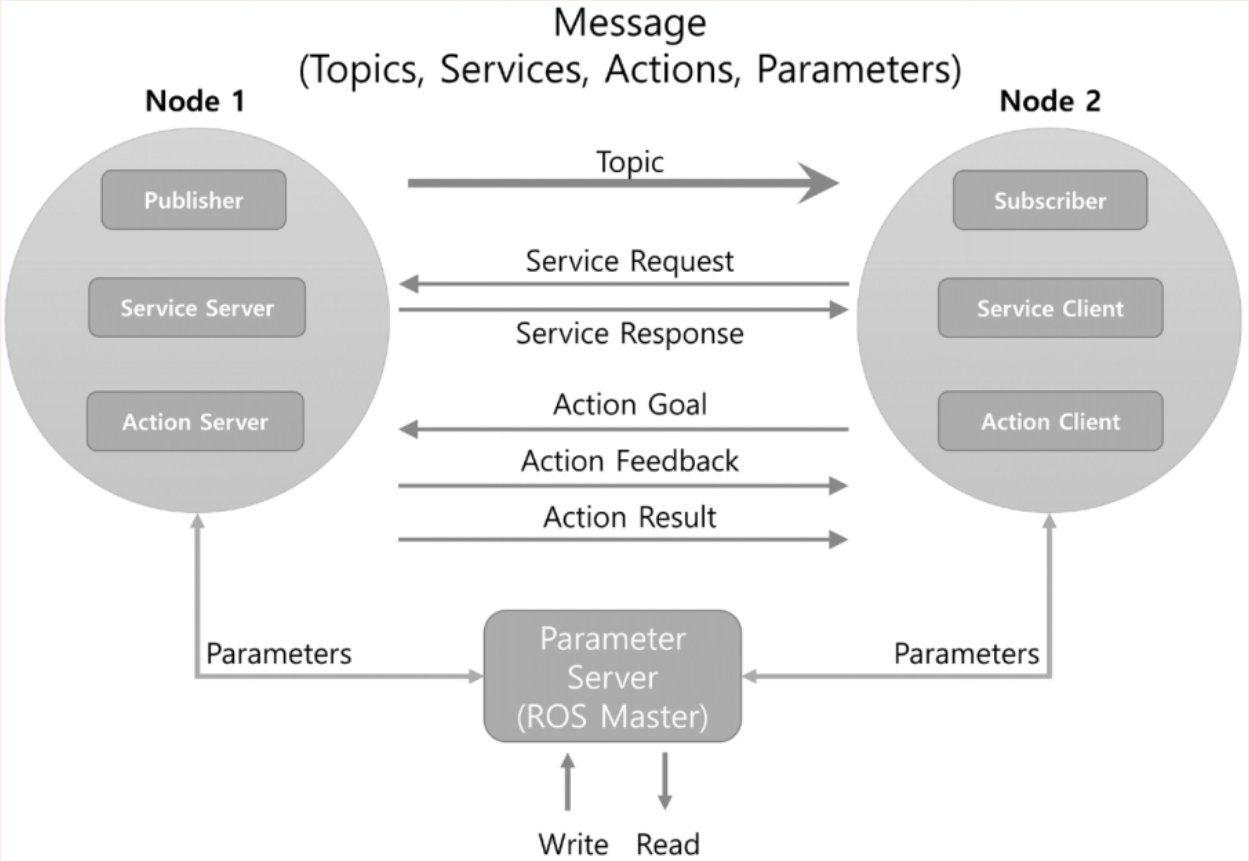
Arquitectura de ROS - Práctica (7bis)

"turtlesim": comunicación entre *Nodes*



Arquitectura de ROS - Resumen

Resumen - Comunicación entre Nodes



Arquitectura de ROS - Resumen

Resumen - Comunicación entre Nodes

Tipo	Caract.		Descripción / Uso
Topic	Asíncrono	Unidireccional	Intercambio de datos de forma continua
Service	Síncrono	Bi-direccional	Proceso de las Requests y se responde a ellas
Action	Asíncrono	Bi-direccional	Request con largo tiempo de respuesta o cuando se necesita un valor intermedia (feedback)

Arquitectura de ROS - ¿Has entendido todo?

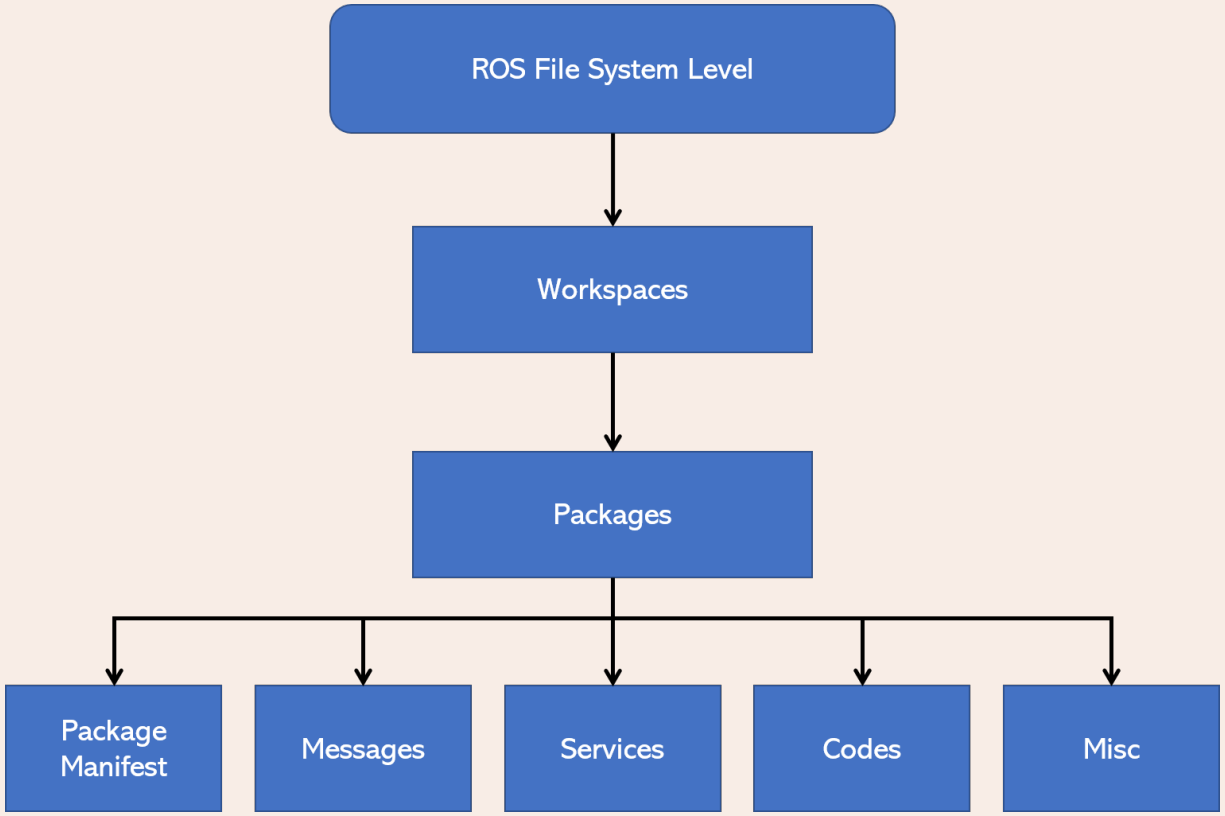
¡Tiempo de práctica!

- Verificación de la comprensión

Índice

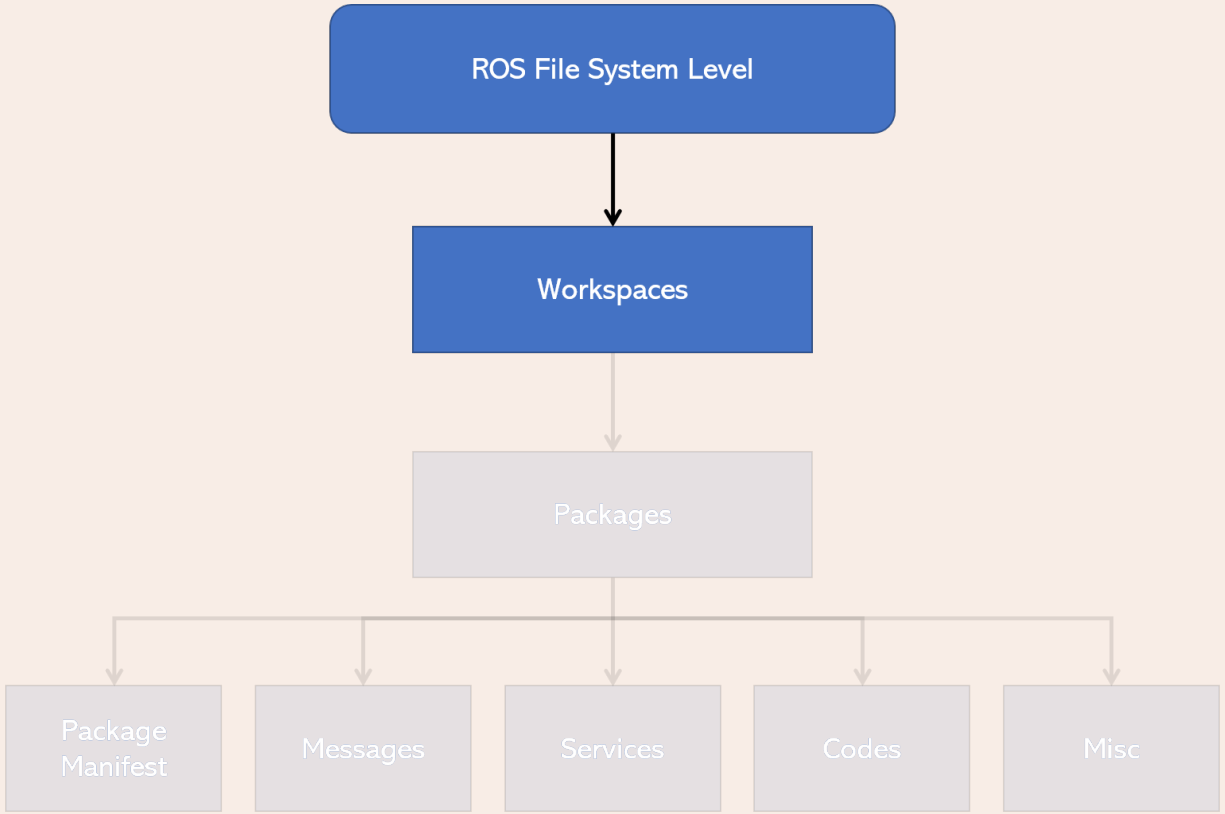
Organización de ROS - Descripción general

Estructura del sistema de archivos de ROS



Organización de ROS - *Workspaces*

Estructura del sistema de archivos de ROS (1/7)



Organización de ROS - *Workspaces*

Definición

Un *catkin workspace* es un conjunto de directorios en los que vive un conjunto de códigos/paquetes de ROS relacionados.

Ejemplo

Cada proyecto tiene su (o sus) workspaces, reconocibles al '`_ws`' en el nombre de la carpeta. Permite compilar múltiples *packages* interdependientes todos juntos a la vez.

Organización de ROS - *Workspaces*

Definición

Un *catkin workspace* es un conjunto de directorios en los que vive un conjunto de códigos/paquetes de ROS relacionados.

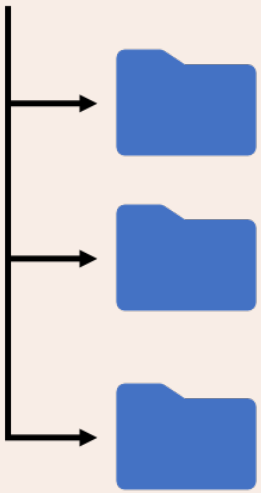
Ejemplo

Cada proyecto tiene su (o sus) workspaces, reconocibles al '`_ws`' en el nombre de la carpeta. Permite compilar múltiples *packages* interdependientes todos juntos a la vez.

Organización de ROS - *Workspaces*

Estructura típica del *catkin workspace* - Global

NameOfTheWorkspace_ws/



src/

Extract/Checkout/Clone el código fuente de los paquetes que desea crear (todo pasa acá)

build/

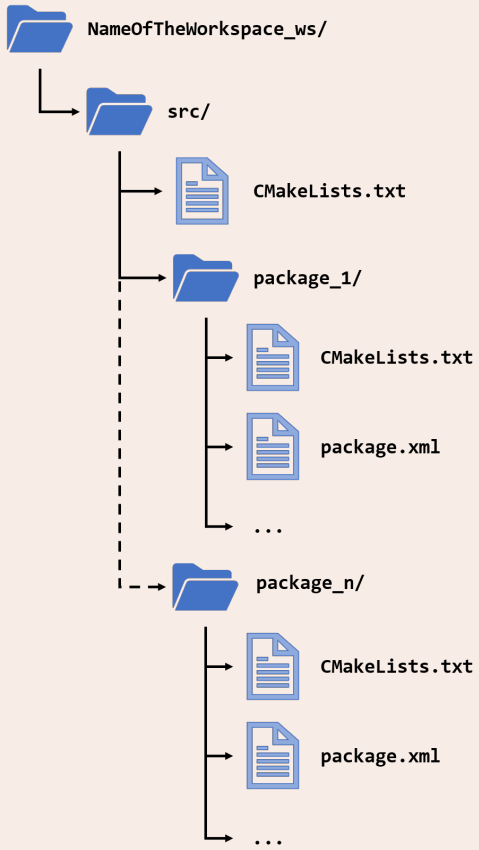
CMake and catkin mantienen su información de caché y otros archivos intermedios aquí

devel/

Donde los scripts compilados se colocan antes de ser instalados

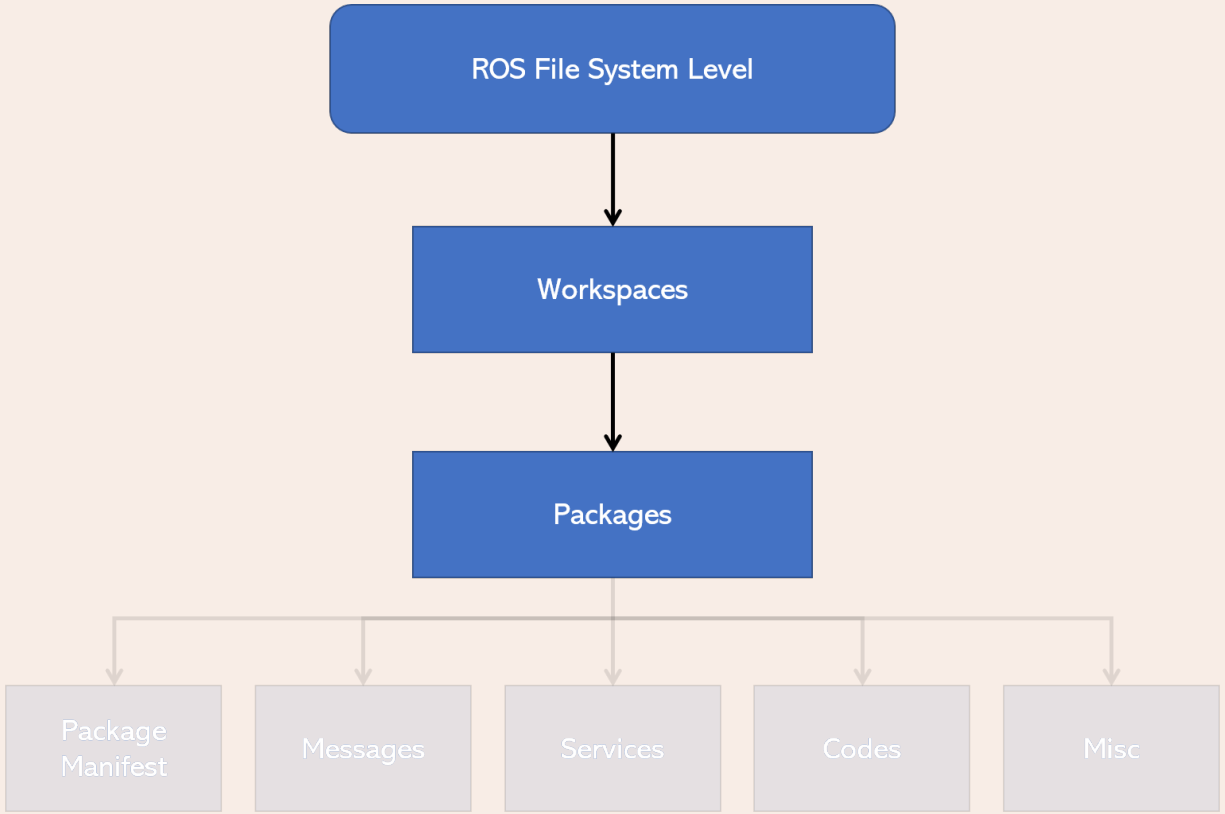
Organización de ROS - *Workspaces*

Estructura típica del *catkin workspace* - Detalles



Organización de ROS - *Workspaces*

Estructura del sistema de archivos de ROS (2/7)



Organización de ROS - *Packages*

Definición

Los *Packages* son la unidad de organización de software del código de ROS. Cada *Package* puede contener bibliotecas, ejecutables, scripts u otros elementos.

Ejemplo

Un *Package* puede ser desarrollado para realizar una tarea colaborativa, y contiene todos los scripts necesarios para realizarla.

Organización de ROS - *Packages*

Definición

Los *Packages* son la unidad de organización de software del código de ROS. Cada *Package* puede contener bibliotecas, ejecutables, scripts u otros elementos.

Ejemplo

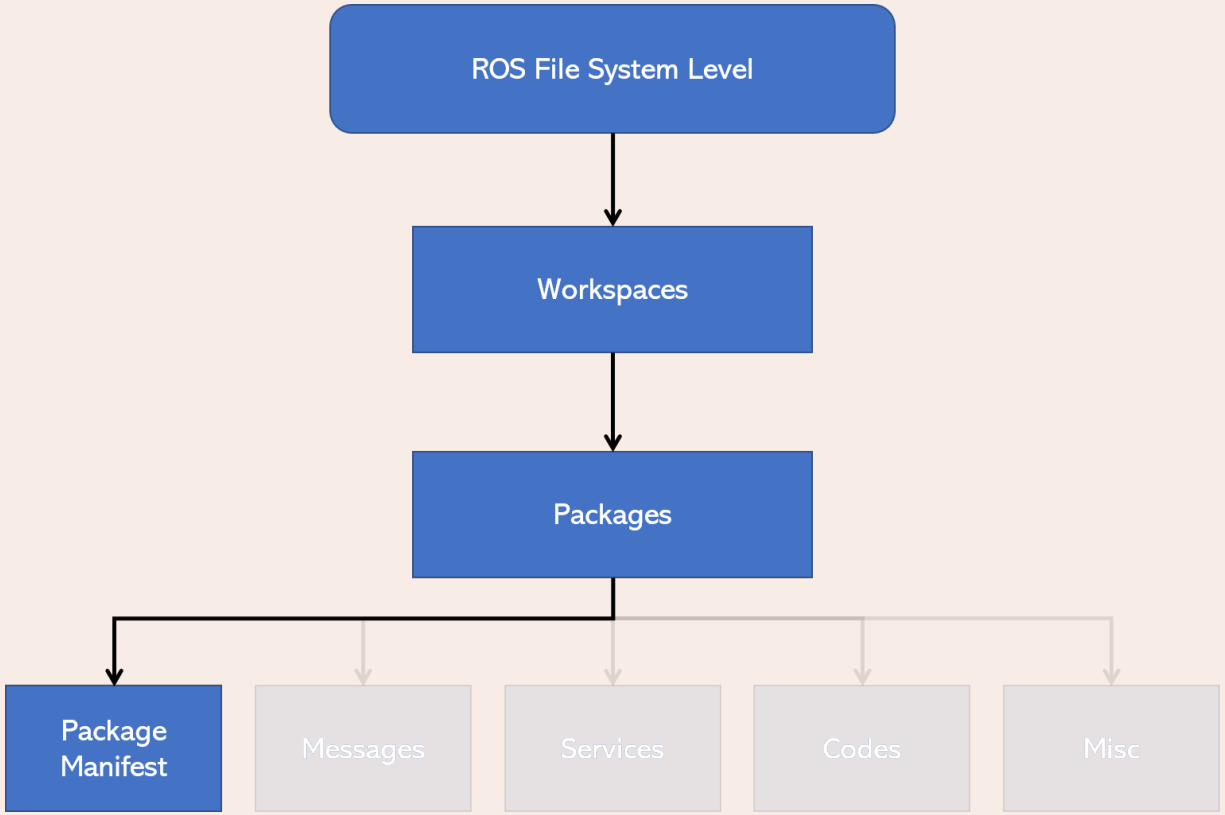
Un *Package* puede ser desarrollado para realizar una tarea colaborativa, y contiene todos los scripts necesarios para realizarla.

Organización de ROS - *Packages*

Estructura típica de los *Packages*

Organización de ROS - *Packages* > *Manifests*

Estructura del sistema de archivos de ROS (3/7)



Organización de ROS - *Packages* > *Manifests*

Definición

Un *Manifest* es la descripción de un paquete. Sirve para definir dependencias entre *Packages* y para capturar meta información sobre el paquete.

Ejemplo

Meta información contiene, entre otros, versión, mantenedor, licencia, etc. Está definida en el archivo `packages.xml`

Organización de ROS - *Packages* > *Manifests*

Definición

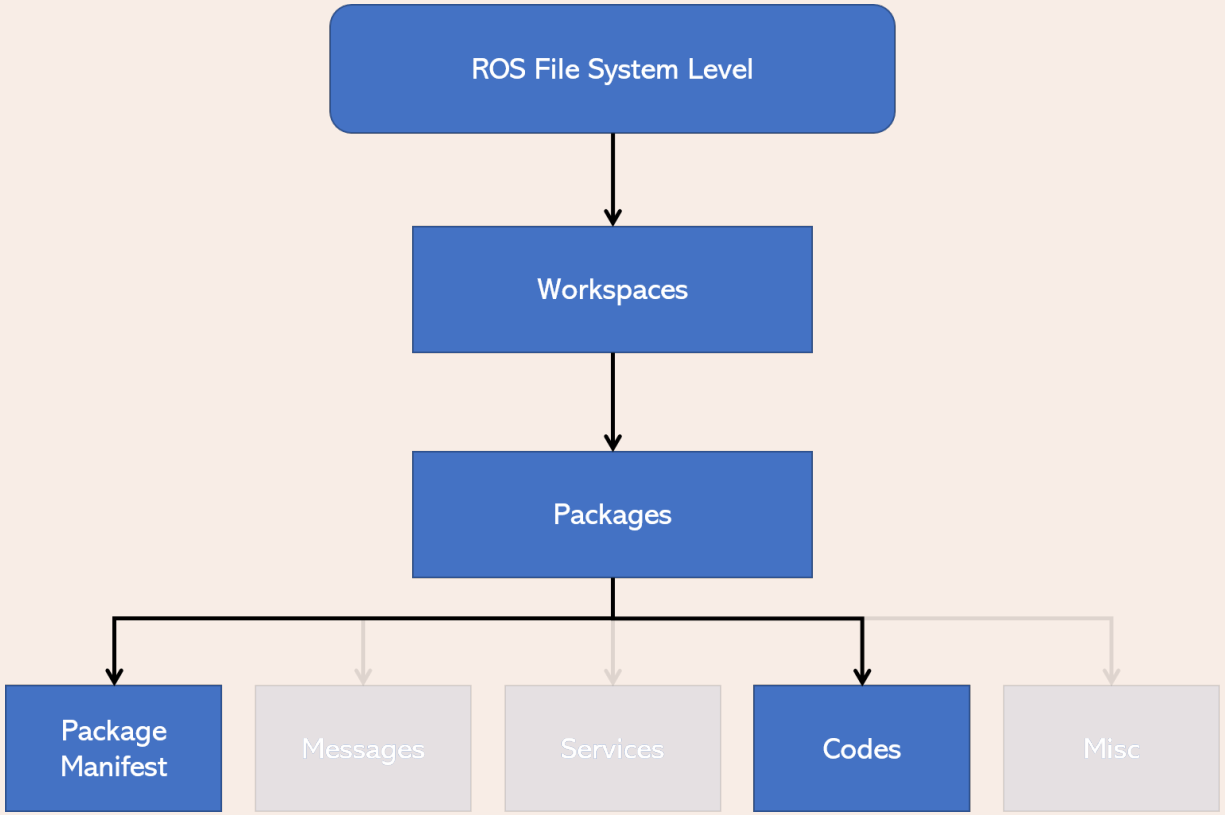
Un *Manifest* es la descripción de un paquete. Sirve para definir dependencias entre *Packages* y para capturar meta información sobre el paquete.

Ejemplo

Meta información contiene, entre otros, versión, mantenedor, licencia, etc. Está definida en el archivo `packages.xml`

Organización de ROS - *Packages* > *Scripts*

Estructura del sistema de archivos de ROS (4/7)



Organización de ROS - *Packages* > *Scripts*

Definición

Los *Scripts* contienen el código fuente de un *Node*.

Ejemplo

Pueden ser escritos en cualquier lenguaje de programación (C++ o Python).

Organización de ROS - *Packages* > *Scripts*

Definición

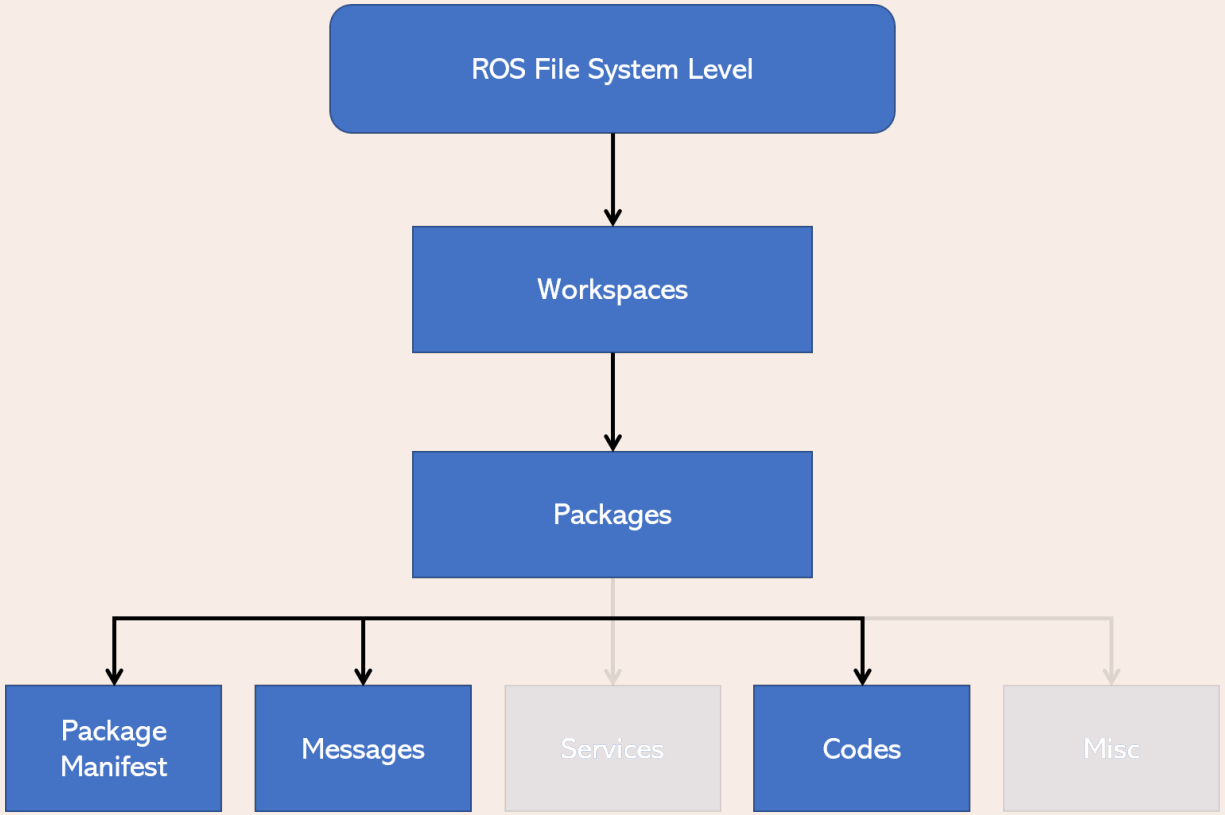
Los *Scripts* contienen el código fuente de un *Node*.

Ejemplo

Pueden ser escritos en cualquier lenguaje de programación (C++ o Python).

Organización de ROS - *Packages* > *Messages*

Estructura del sistema de archivos de ROS (5/7)



Organización de ROS - *Packages* > *Messages*

Definición

ROS Client Libraries implementan generadores de mensajes que traducen archivos `.msg`, que contienen descripciones de los mensajes, en código fuente.

Ejemplo

Clase de mensaje personalizada `block_poses.msg`, para obtener la posición de un objeto gracias a una camera:

```
float64[] x # x coordinate in the world
float64[] y # y coordinate in the world
float64[] z # z coordinate in the world
```

Organización de ROS - *Packages* > *Messages*

Definición

ROS Client Libraries implementan generadores de mensajes que traducen archivos `.msg`, que contienen descripciones de los mensajes, en código fuente.

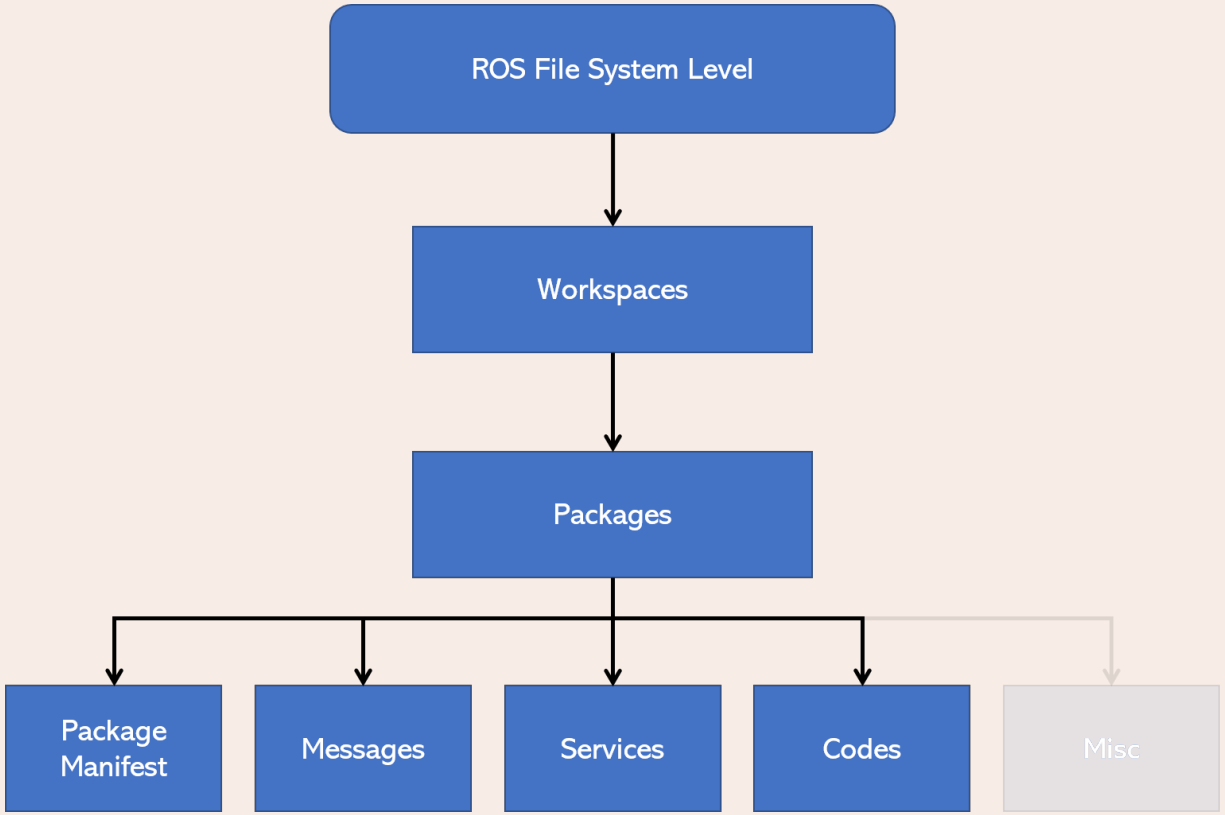
Ejemplo

Clase de mensaje personalizada `block_poses.msg`, para obtener la posición de un objeto gracias a una camera:

```
float64[] x # x coordinate in the world
float64[] y # y coordinate in the world
float64[] z # z coordinate in the world
```

Organización de ROS - *Packages* > *Services*

Estructura del sistema de archivos de ROS (6/7)



Organización de ROS - *Packages* > *Services*

Definición

Se basa directamente en el formato *ROS msg* para permitir la comunicación *request/response* entre nodos. Las descripciones de los servicios se almacenan en archivos `.srv` en el subdirectorio `srv/` del paquete.

Ejemplo

Un archivo de descripción de servicio consiste en un *request* y un *response* (de tipo *msg*), separados por `'---'`

```
string str
---
string str
```

Organización de ROS - *Packages* > *Services*

Definición

Se basa directamente en el formato *ROS msg* para permitir la comunicación *request/response* entre nodos. Las descripciones de los servicios se almacenan en archivos `.srv` en el subdirectorio `srv/` del paquete.

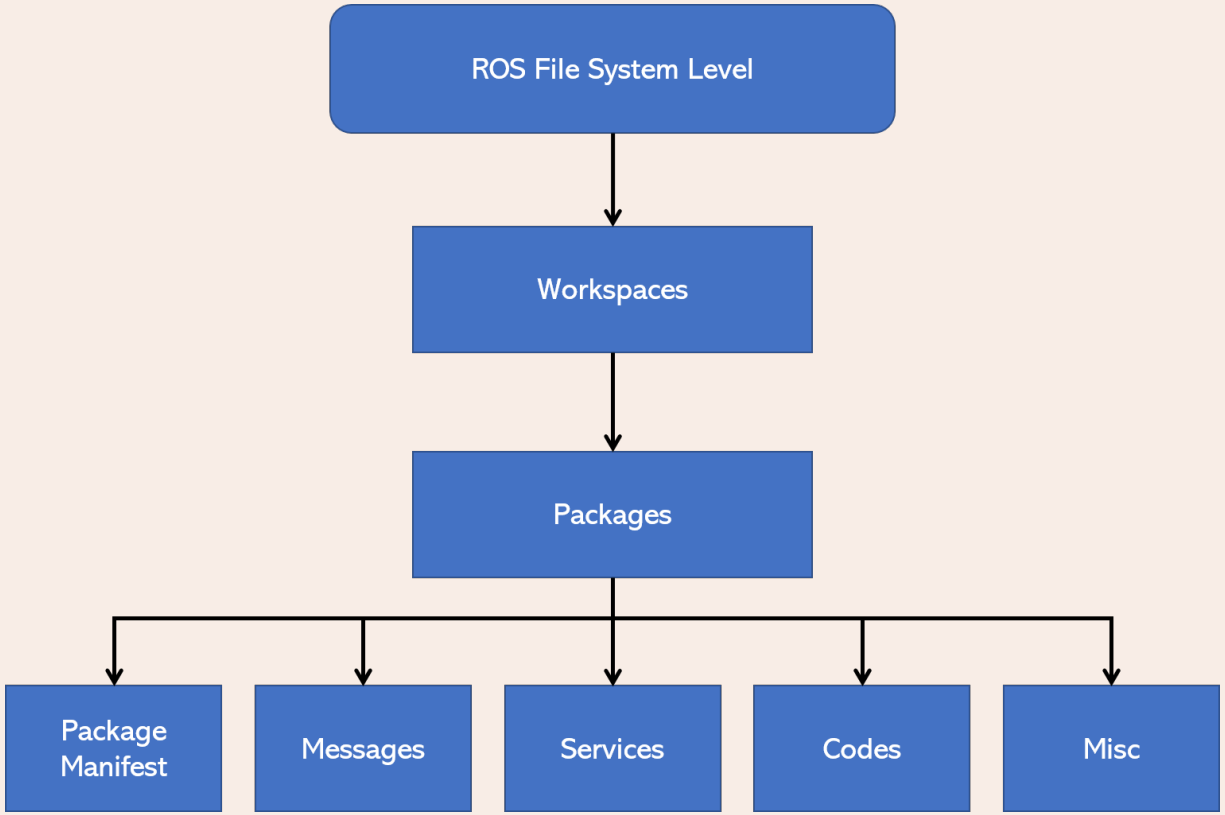
Ejemplo

Un archivo de descripción de servicio consiste en un *request* y un *response* (de tipo *msg*), separados por `'---'`

```
string str
---
string str
```

Organización de ROS - *Packages* > *Misc*

Estructura del sistema de archivos de ROS (7/7)



Organización de ROS - *Packages* > *Misc*

Definición

Se puede añadir, además de los paquetes requisitos, otras carpetas conteniendo archivos especiales, o para una mejor visibilidad.

Ejemplo

Archivos para simulaciones (`/worlds`, `/meshes`, `/gazebo`), Python scripts (`/scripts`), archivos de configuración de `Rviz`, etc.

Organización de ROS - *Packages* > *Misc*

Definición

Se puede añadir, además de los paquetes requisitos, otras carpetas conteniendo archivos especiales, o para una mejor visibilidad.

Ejemplo

Archivos para simulaciones (`/worlds`, `/meshes`, `/gazebo`), Python scripts (`/scripts`), archivos de configuración de `Rviz`, etc.

Índice

Lo más importante: saber buscar

- Recursos disponibles:
  - Distribuciones
  - Repositorios
  - ROS Wiki
  - ROS Answers
  - Git
  - Youtube
  - Forums
  - Tutorials

Lo más importante: saber buscar

- Recursos disponibles:
  - Distribuciones
  - Repositorios
  - ROS Wiki
  - ROS Answers
  - Git
  - Youtube
  - Forums
  - Tutorials

Lo más importante: saber buscar

- Recursos disponibles:
  - Distribuciones
  - Repositorios
  - ROS Wiki
  - ROS Answers
  - Git
  - Youtube
  - Forums
  - Tutorials

Lo más importante: saber buscar

- Recursos disponibles:
  - Distribuciones
  - Repositorios
  - ROS Wiki
  - ROS Answers
  - Git
  - Youtube
  - Forums
  - Tutorials

Lo más importante: saber buscar

- Recursos disponibles:
  - Distribuciones
  - Repositorios
  - ROS Wiki
  - ROS Answers
  - Git
  - Youtube
  - Forums
  - Tutorials

Lo más importante: saber buscar

- Recursos disponibles:
  - Distribuciones
  - Repositorios
  - ROS Wiki
  - ROS Answers
  - Git
  - Youtube
  - Forums
  - Tutorials

Lo más importante: saber buscar

- Recursos disponibles:
  - Distribuciones
  - Repositorios
  - ROS Wiki
  - ROS Answers
  - Git
  - Youtube
  - Forums
  - Tutorials

Lo más importante: saber buscar

- Recursos disponibles:
  - Distribuciones
  - Repositorios
  - ROS Wiki
  - ROS Answers
  - Git
  - Youtube
  - Forums
  - Tutorials

Lo más importante: saber buscar

- Recursos disponibles:
  - Distribuciones
  - Repositorios
  - ROS Wiki
  - ROS Answers
  - Git
  - Youtube
  - Forums
  - Tutorials

Sumario - Lo que aprendiste hoy

- Posibles retos y aplicaciones de ROS
- Los conceptos esenciales de ROS
- Aplicaciones concretas con ejemplos y proyecto

Sumario - Lo que aprendiste hoy

- Posibles retos y aplicaciones de ROS
- Los conceptos esenciales de ROS
- Aplicaciones concretas con ejemplos y proyecto

Sumario - Lo que aprendiste hoy

- Posibles retos y aplicaciones de ROS
- Los conceptos esenciales de ROS
- Aplicaciones concretas con ejemplos y proyecto