Une première mise en œuvre de ROS (Robot Operating System)

Génération Robots / Humarobotics
Novembre 2012

Fondation unit
Université Numérique Ingénierie et Technologie

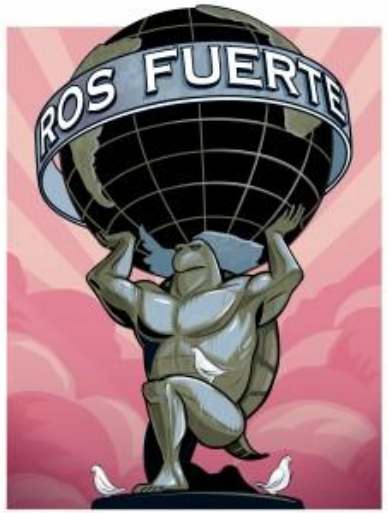
GDR ROBOTIQUE

Objectifs de cette présentation

Objectifs

- A partir d'une article de présentation en ligne
- Utiliser ROS concrètement
- Comprendre les concepts à l'œuvre dans ROS
- Comprendre comment les différents processus communiquent

ROS.org



Présentation des notions de ROS

Qu'est ce que ROS?

A quoi sert ROS ?

Quelle est l'histoire de ROS ?

> Une présentation des concepts à l'œuvre et la réponse à toutes ces questions se trouve dans l'article d'introduction : « **ROS, Système d'exploitation open-source pour robot »**

http://www.generationrobots.com/ros-robot-operating-system,fr,8,74.cfm

Mise en application des concepts

Dans la suite de cette présentation :

> Description concrète des concepts en se basant sur l'utilisation d'un exemple pas à pas

> Hypothèse de démarrage : ROS est installé sur votre PC (Linux)

Présentation de ROS

Démarrer ROS

- `$ roscore`
- `Cette commande lance deux choses:`
  - Le MASTER
  - Le serveur de paramètres

Présentation de ROS

Le Master

- C'est le processus cœur
- Il comprend un serveur de noms
- Les différents nœuds vont se présenter à lui
- Il va mettre les nœuds en contact
  - Une fois que les nœuds sont en contact, ils communiquent directement → architecture distribuée

Le serveur de paramètres

- Il permet de définir et de gérer les paramètres globaux du robot
  - Les paramètres peuvent être des entiers, des flottants, des booléens, des strings, des listes et dictionnaires
- Le serveur de paramètres est une mémoire partagée
  - Chaque nœud peut lire et modifier ces paramètres
- Modifier un paramètre en ligne de commande :

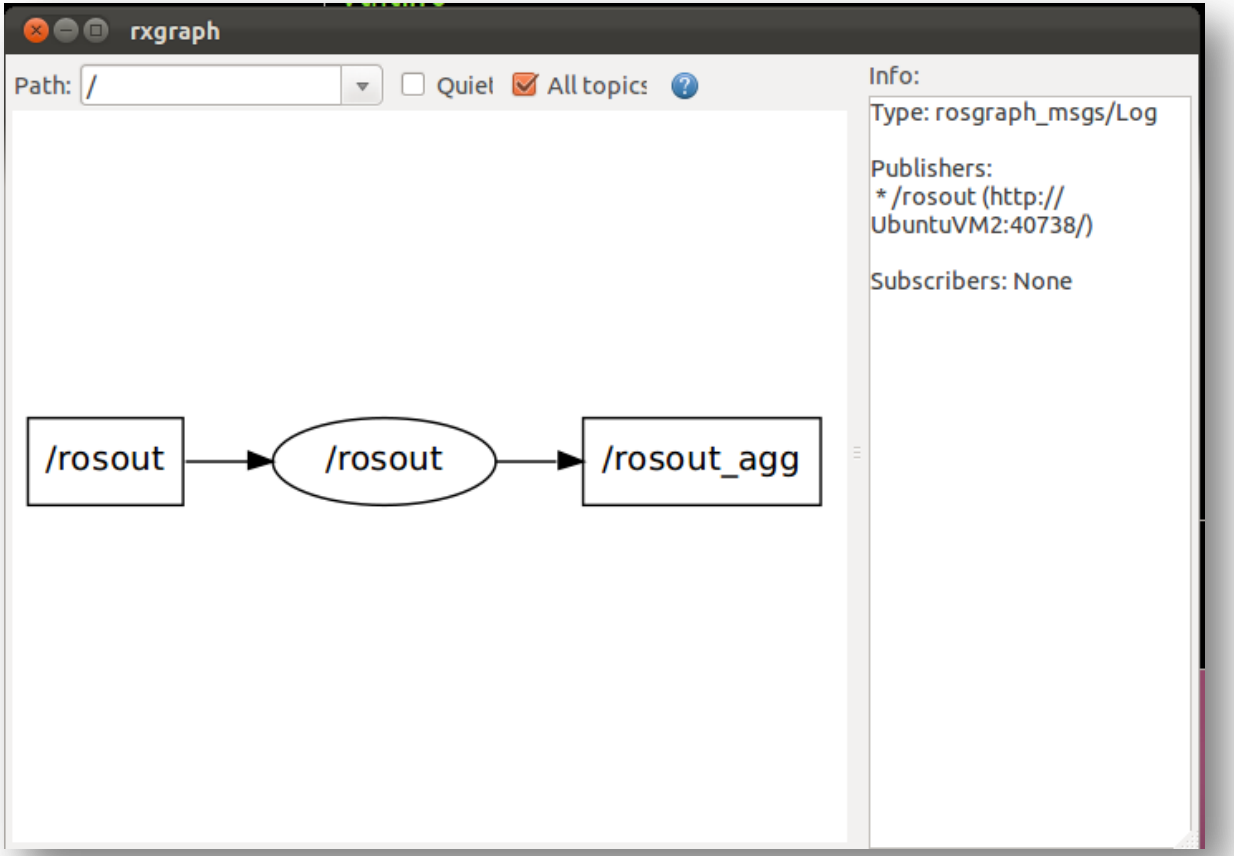
```
$ rosparam set parameter_name value
```

Nœuds, messages, topics

- Le MASTER démarre un nœud, qui démarre un *topic*
- `$ rosnode list`
  - Le nœud s'appelle */rosout*
- `$ nosnode info /rosout`
  - Il publie sur le topic */rosout_agg*
  - Il est abonné au topic */rosout*
  - Il fournit les services */rosout/set_logger_level* et */rosout/get_loggers*

Présentation de ROS

$ rxgraph



Messages

- Un message est une structure de données
  - Il peut contenir des entiers, flottants, booléens, strings ou d'autres messages
  - On crée un message avec un fichier texte dans le dossier msg
  - Ex : pose_and_twist.msg

```
Header header
string child_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist
```

Topics

- Un topic est un système de transport de l'information many-to-many
  - Typé : il faut préciser quel type de messages on transporte
  - Basé sur le système de l'abonnement / publication (subscribe / publish)
- On peut écouter un topic en ligne de commandes

```
$ rostopic list
$ rostopic echo /topic_name
```

Présentation de ROS

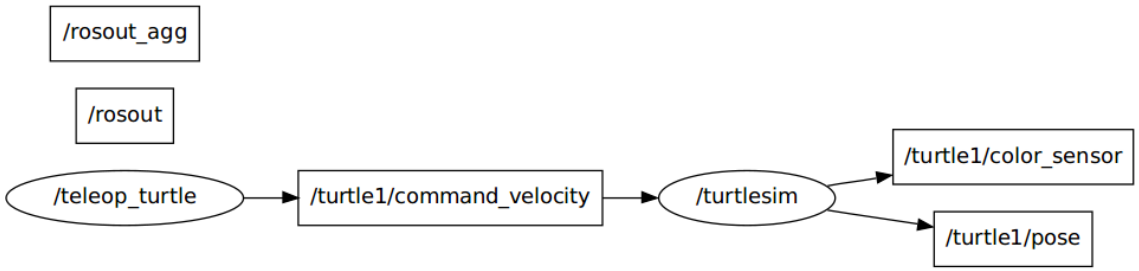
Les nœuds

- Un nœud est une instance d'un exécutable
  - Il peut être lié à un moteur, un capteur, ou purement logiciel...
- Un nœud peut publier des messages sur un topic ou souscrire aux messages d'un topic
  - Les topics sont automatiquement créés
- Les nœuds sont des processus indépendants
  - Le crash d'un nœud ne fait pas crasher tout ROS (Notion de micro-kernel sur laquelle est basée ROS)

Présentation de ROS

En pratique :

- Chaque sous programme sera dans un nœud
- Les nœuds communiquent entre eux de manière événementielle en utilisant les topics
- Ils partagent une mémoire dans le serveur de paramètres



Présentation de ROS

Exemple :

- Le nœud /obst_detect lit les capteurs de distance
  - Il publie dans les topics obst_droit et obst_gauche
- Le nœud /robot_size
  - Il fournit le service /robot_size/motSpeed pour calculer la vitesse des moteurs en fonction de la vitesse de rotation souhaitée et de la taille des roues

Présentation de ROS

Exemple :

- Le nœud /nav calcule la direction
  - Il souscrit à obst_droit et obst_gauche pour éviter les obstacles
  - Il lit le paramètre destination dans le serveur de paramètres
  - Il utilise le service /robot_size/motSpeed
  - Il publie dans le topic vitesse_moteurs
- Le nœud /ctrl commande les moteurs des roues
  - Il souscrit à vitesse_moteurs

Les services

- Un service est une fonction
  - Implémentée dans un nœud
  - Accessible aux autres nœuds
- Un service doit être déclaré dans un fichier .srv
  - En précisant les types des entrées et des sorties
  - Exemple : add_two_ints.srv

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

- `$ rosservice list`
- `$ rosservice call /service_name service_args`

Présentation de ROS

Les packages :

- Un package est un projet ROS. Il peut contenir :
  - Des nœuds dans le répertoire nodes
  - Des messages dans le répertoire msg
  - Des services dans le répertoire srv
  - Des scripts de lancement dans le répertoire launch
- Un stack est une ensemble de packages

Présentation de ROS

Les packages

- Un package contient obligatoirement un fichier manifest.xml qui décrit
  - Les licences
  - Les dépendances
- **`$ rosmake`** utilise le manifeste pour compiler le package et ses dépendances

```
<package>
 <description brief="one line of text"/>
  <author>Alice/alice@somewhere.bar</author>
  <license>BSD</license>
  <url>http://pr.willowgarage.com/</url>
  <depend package="pkgname"/>
  <depend package="common"/>
  <rosdep name="python" />
</package>
```

Présentation de ROS

Quelques robots programmables à l'aide de ROS :

Q.bo

Darwin-Op

Aisoy1 II

Eddie

Merci !

Que faire ensuite ?

- Consulter nos TPs ROS sur ce même site
  - TurltleSim with ROS
  - Piloter le robot mobile Eddie avec ROS
- Nous contacter pour toute question : contact@humarobotics.com