Introduction à ROS 2

Qu’est-ce que ROS 2 ?

ROS 2 (Robot Operating System 2) est un framework open-source qui facilite le développement d’applications robotiques. Créé comme une évolution de ROS 1, ROS 2 répond aux limitations de son prédécesseur, notamment en matière de communication distribuée, de performance en temps réel et de compatibilité avec des systèmes variés (comme les microcontrôleurs, les robots autonomes ou les clusters distribués).

Objectifs principaux de ROS 2

1. **Modularité** : Permet aux développeurs de construire des systèmes robotiques à partir de composants réutilisables et interchangeables.
2. **Interopérabilité** : Fonctionne sur divers systèmes d’exploitation (Linux, Windows, macOS) et architectures matérielles.
3. **Fiabilité** : Fournit des mécanismes robustes pour gérer la communication et les défaillances.
4. **Scalabilité** : S’adapte aussi bien aux petits robots qu’aux systèmes robotiques complexes ou distribués.

Applications courantes

- Robotique industrielle (manipulateurs, bras robotiques)
- Robots mobiles (drones, véhicules autonomes)
- Recherche académique et prototypage rapide
- Domotique et robots personnels

Comment fonctionne ROS 2 ?

ROS 2 repose sur une **architecture distribuée**, dans laquelle différents composants, appelés **nœuds**, interagissent les uns avec les autres. Cette interaction est gérée par un middleware appelé **DDS (Data Distribution Service)**, qui garantit une communication performante et fiable.

Principes fondamentaux

1. **Décentralisation** : Il n’y a pas de maître central (contrairement à ROS 1). Tous les nœuds sont autonomes et peuvent s’interconnecter dynamiquement.
2. **Communication inter-processus** : Les nœuds utilisent des mécanismes standardisés comme les **topics**, les **services**, et les **actions** pour échanger des données ou exécuter des tâches.
3. **Compatibilité en temps réel** : Grâce à DDS, ROS 2 peut être utilisé dans des systèmes où des garanties strictes de temps de réponse sont nécessaires.

Avantages par rapport à ROS 1

- **Temps réel** : Support intégré pour les systèmes temps réel, essentiel dans les applications robotiques critiques.

- **Sécurité** : Utilise DDS pour ajouter des fonctionnalités de cryptage et d'authentification.
- **Portabilité** : Fonctionne sur des architectures matérielles et logicielles variées, y compris des plateformes embarquées.

Les Nœuds (Nodes)

Un **nœud** est un processus autonome qui accomplit une tâche spécifique dans un système robotique. Dans ROS 2, les nœuds sont conçus pour être simples et modulaires afin de faciliter le développement et la maintenance.

Exemples de nœuds dans un robot mobile

- **Capteurs** : Un nœud lit les données d'un capteur (comme un LiDAR, une caméra ou un capteur de distance) et publie ces données pour d'autres nœuds.
- **Contrôleur de mouvement** : Un nœud reçoit des commandes de navigation et contrôle les moteurs pour déplacer le robot.
- **Traitement de données** : Un nœud effectue des calculs, comme la fusion de capteurs ou la création d'une carte à partir des données SLAM.

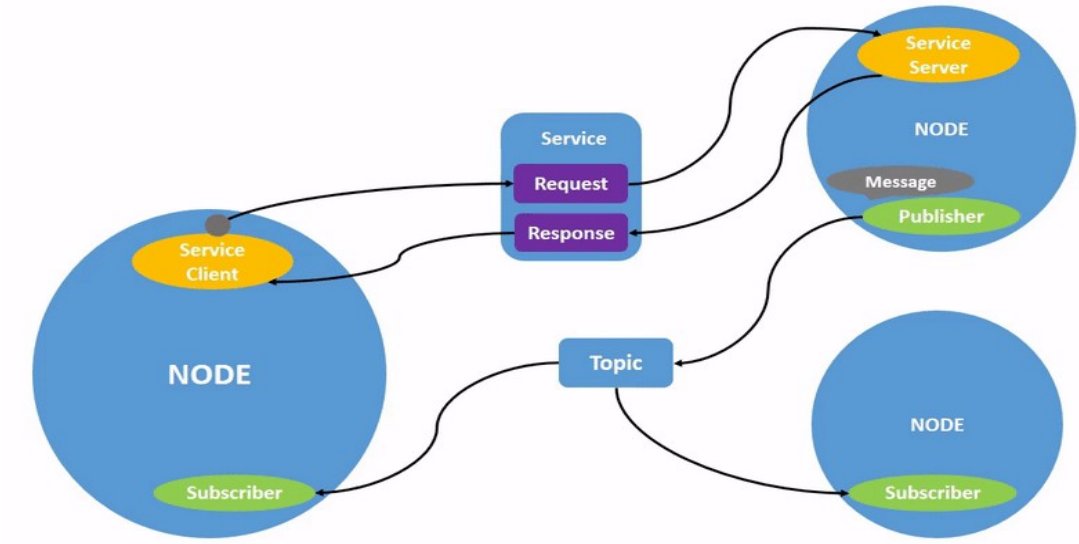
Caractéristiques des nœuds

- Ils peuvent être écrits dans plusieurs langages (C++, Python, etc.).
- Chaque nœud fonctionne indépendamment et peut être redémarré ou remplacé sans affecter le reste du système.
- Les nœuds communiquent entre eux en utilisant des **topics**, des **services**, ou des **actions**.

Commandes utiles

- ros2 node list
- ros2 run <nom_du_package> <nom_de_l'executable>
- ros2 node info <nom_du_nœud>

Relation entre les Nœuds



Les Topics (Sujets)

Les **topics** permettent un échange asynchrone de données entre les nœuds. C'est un mécanisme de **publication-abonnement** :

- Un nœud éditeur publie des messages sur un topic donné.
- Un ou plusieurs nœuds abonnés reçoivent ces messages.

Caractéristiques principales des topics

- **Nom unique** : Chaque topic est identifié par un nom unique dans l'espace de noms ROS 2.
- **Type de message** : Un topic transporte des messages d'un type spécifique, comme des positions, des images ou des commandes.
- **Connexion dynamique** : Les nœuds peuvent se connecter et se déconnecter des topics sans interruption du système.

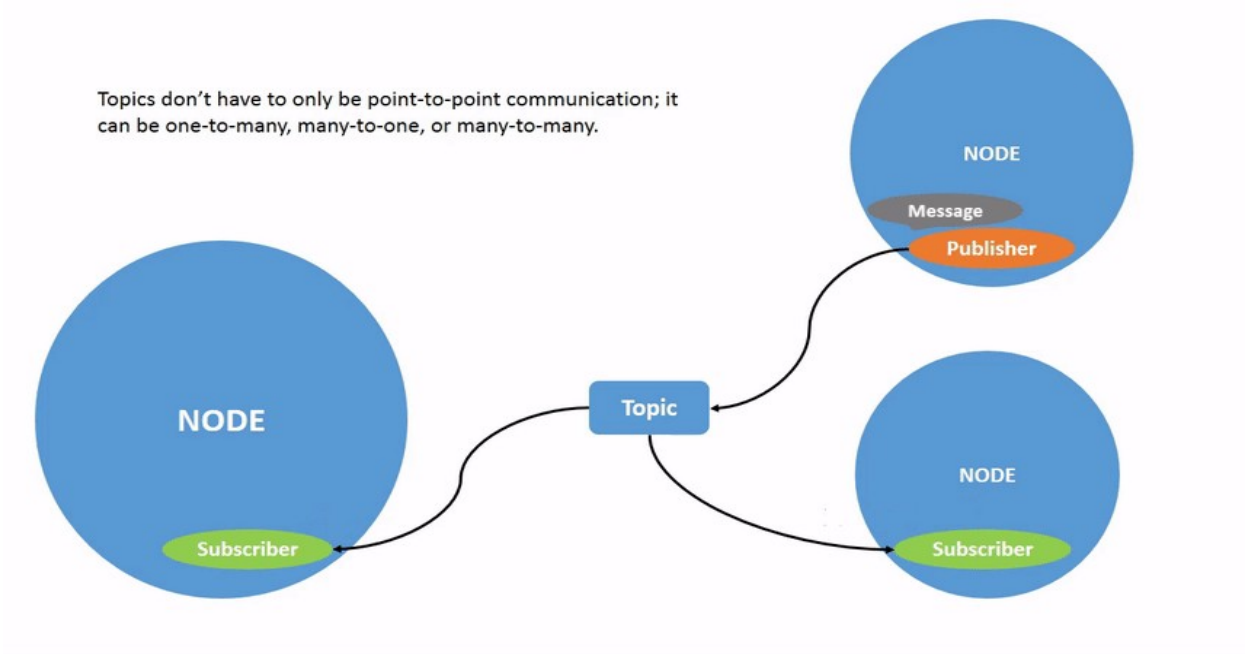
Exemple de communication par topics

1. Un nœud capteur publie les données du LiDAR sur le topic `/scan`.
2. Un nœud de cartographie utilise ces données pour créer une carte en temps réel.
3. Un nœud de navigation utilise la carte pour planifier un chemin.

Commandes utiles

- ros2 topic list
- ros2 topic echo <nom_du_topic>
- ros2 topic info <nom_du_topic>
- ros2 topic pub <nom_du_topic> <type_de message> '<arguments> '
- ros2 topic hz <nom_du_topic>
- ros2 topic find <type_de_topic>

Relation entre les Nœuds et les Topics



Les Services (Services)

Les **services** permettent une interaction synchronisée entre deux nœuds sous le modèle **requête-réponse**.

- Un nœud envoie une requête pour exécuter une tâche ou obtenir des données.
- Un autre nœud traite la requête et renvoie une réponse.

Différences entre services et topics

- **Topics** : Utilisés pour un flux continu de données.
- **Services** : Utilisés pour des interactions ponctuelles nécessitant une réponse immédiate.

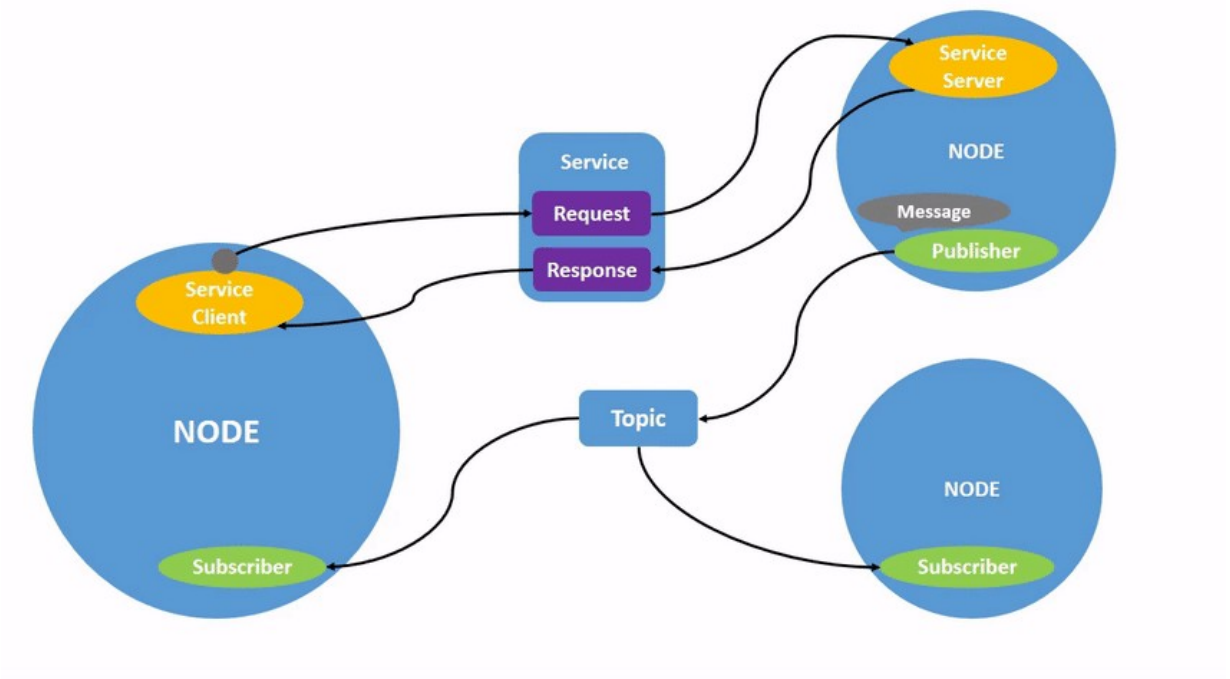
Exemples d'utilisation

- Demander à un bras robotique de saisir un objet.
- Obtenir la température actuelle d'un capteur.
- Redémarrer un nœud défectueux.

Commandes utiles

- ros2 service list
- ros2 service type <nom_du_service>
- ros2 service list -t
- `ros2 service find <nom_de_type>`
- `ros2 service call <service_name> <service_type> <arguments>`

Relation entre les Nœuds et les Services



Les Actions (Actions)

Les **actions** sont une extension des services, conçues pour gérer des tâches longues ou complexes. Elles permettent de :

- Lancer une tâche asynchrone.
- Suivre l’état d’avancement de cette tâche.
- Annuler la tâche en cours si nécessaire.

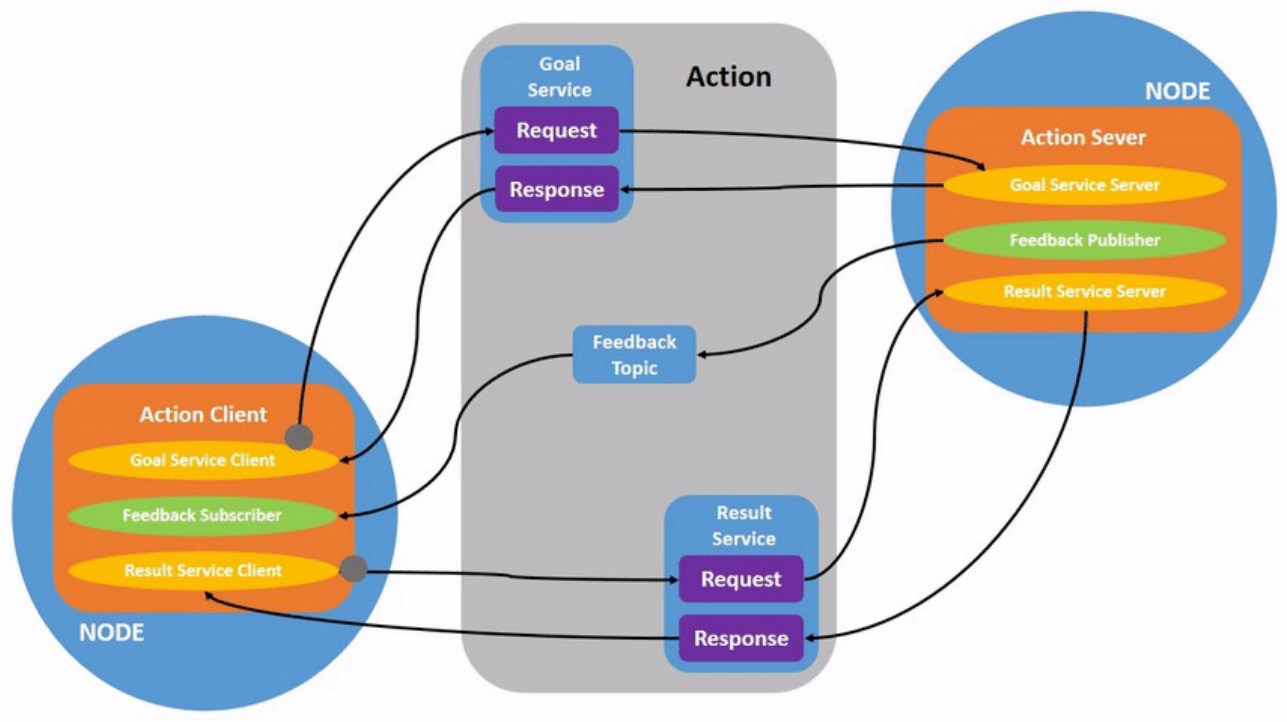
Exemples d’actions

- Planifier et exécuter un chemin complexe pour un robot mobile.
- Manipuler un objet en plusieurs étapes.

Commandes utiles

- ros2 action list
- ros2 action info <nom_de_l’action>
- ros2 action send_goal <nom_de_l’action> <type_de_l’action> <valeurs> (les valeurs en format YAML)

Relation Entre les Nœuds et les Actions



Les Paramètres

Les **paramètres** permettent de configurer les nœuds de manière flexible. Ils offrent des options pour ajuster le comportement d'un nœud sans modifier son code ou le redémarrer.

Exemples de paramètres

- La vitesse maximale d'un robot.
- La fréquence à laquelle un capteur envoie ses données.
- Les seuils pour détecter des obstacles.

Les paramètres peuvent être modifiés en temps réel via des outils comme `rqt` ou des commandes CLI (`ros2 param`).

Commandes utiles

- ros2 param list
- ros2 param get <nom_du_nœud> <nom_de_paramètre>
- ros2 param set <nom_du_nœud> <nom_de_paramètre> <valeur>
- ros2 param dump <nom_de_paramètre>
- ros2 param load <nom_du_nœud> <fichier_du_paramètre>