Turtlebot3 docs

Tartalom

1 ROS

1.1 Ismertető

A ROS-ban a számítások elvégzéséhez ROS csomópontok néven ismert szoftverkomponenseket használják. A csomópontok, a mester, a paraméterkiszolgáló, az üzenetek, a témák, a szolgáltatások és a táskák a számítási gráf központi elemei, melyek mind más-más módon járulnak hozzá ehhez a ROS gráfhoz.

A ros_comm nevű verem tartalmazza a ROS kommunikációval kapcsolatos csomagjait, beleértve az olyan alapvető klienskönyvtárakat, mint a roscpp és a rospy, valamint az olyan fogalmak megvalósítását, mint a témák, csomópontok, paraméterek és szolgáltatások. Ezek a fogalmak további megismeréséhez a verem olyan eszközöket is tartalmaz, mint a rostopic, rosparam, rosservice és rosnode.

A ROS kommunikációs middleware csomagjai együttesen alkotják a ROS gráf réteget, a ros_comm stack tartalmazza ezeket a csomagokat.

A ROS gráfok fogalmai:

- Csomópontok: A számításokat végző folyamatokat csomópontoknak nevezzük, melyek létrehozásához a roscpp és rospy klienskönyvtárakat használjuk. Ezekkel a kommunikáció több formáját is megvalósíthatjuk az API-k segítségével. Egy robotban számos csomópont lehet a különböző feladatok elvégzésére, valamint ROS kommunikációs protokollok segítségével kommunikálhatnak egymással és oszthatnak meg adatokat. A csomópontok egyik célja, hogy inkább kis folyamatokat építsünk ki minimális funkcionalitással, mint egy komplexet, mivel egyszerűbb a hibakeresés.

- Mester: A mester névkeresést és regisztrációt kínál a többi csomópont számára, enélkül a csomópontok nem tudják megtalálni egymást, nem tudnak kommunikálni egymással, és nem tudnak szolgáltatásokat használni. Egy elosztott rendszerben a mester egy számítógépen fut, majd a csomópontok ehhez a mesterhez tudnak csatlakozni, hogy megtalálják egymást.

- Paraméter szerver: A paraméter szerver használatával egy helyen tarthatja az adatokat, ennek segítségével minden csomópont hozzáférhet ezekhez az

értékekhez és ellenőrizheti azokat. A mester tartalmaz egy paraméter szervert.

- Üzenetek: Az üzenetek a csomópontok közötti kommunikáció elsődleges eszközei. Egyszerűen fogalmazva, az üzenet egy olyan adatszerkezet, amely egy tipizált mezővel rendelkezik, amely egy adatgyűjteményt tartalmazhat, és továbbítható egy másik csomópontnak. Az üzenetek egész, lebegőpontos, Boolean és más általános primitív típusokat támogatnak, de szerkeszthetőek saját üzenettípusok.

- Témák: A ROS-ban minden kommunikáció egyedi névvel rendelkező témákon, azaz csatornákon keresztül történik. Azt mondhatjuk, hogy egy csomópont egy témát publikál, amikor egy témán keresztül üzenetet küld, vagy feliratkozik egy témára, amelyen keresztül üzeneteket kap. A feliratkozó csomópont és a publikáló csomópont nem tudnak egymásról, így olyan témákra is lehet feliratkozni, amelyeknek nincs kiadója. Más szóval az információ létrehozása és fogyasztása különálló folyamat. Amíg egy csomópont rendelkezik a megfelelő üzenettípussal, addig hozzáférhet a témához és küldhet rajta keresztül adatokat.

- Szolgáltatások: Ha egy robotalkalmazás kérés-válasz interakciót igényel, a publikálás/feliratkozás megközelítés nem biztos elegendő. A kérés/válasz típusú interakcióra akkor lehet szükség, ha elosztott rendszerrel dolgozunk, mivel a publikálás/feliratkozás architektúra egy egyirányú szállítási rendszer. Ezekben a helyzetekben szolgáltatásokat használnak. Létezik olyan szolgáltatásdefiníció, amely két részből áll, egy a kéréseknek és egy a válaszoknak. A szolgáltatások segítségével létrehozhatunk egy szerver csomópontot és egy kliens csomópontot. Amikor a kliens csomópont kérelmez a szervertől, az válaszol, és az eredményt átadja a kliensnek. A szerver csomópont egy név alatt nyújtja a szolgáltatást és lehetséges, hogy az ügyfélnek várnia kell, amíg a szerver válaszol.

- Táskák: A ROS kommunikációs adatok tárolása és lejátszása táskákban történik, melyek kulcsfontosságú adattárolási mechanizmust jelentenek a szenzoradatok számára. Ezek megszerzése kihívást jelenthet, de különböző projektek létrehozásához és teszteléséhez esedékesen szükségesek.

Az ilyen típusú gráfok az rqt_graph nevű eszközzel generálhatók, a csomópontok közti kommunikációs folyamát ábrázolja.

1.2 ROS csomópontok

A ROS kliens könyvtárak, például a roscpp és a rospy használatával hozhatunk létre csomópontokat, amelyek számításokat végeznek. A témákat, szolgáltatásokat és paramétereket egy csomópont használhatja más csomópontokkal való kommunikációhoz. Egy robotban számos csomópont lehet jelen, amelyek feldolgozzák a kameraképeket, kezelik a robot soros kommunikációját stb. A rendszer hibatűrővé tehető a csomópontok használatával. Egy teljes robotrendszer akkor

is tovább működhet, ha egy csomópont megszűnik működni. Mivel minden egyes csomópont csak egy funkciót kezel, a csomópontok megkönnyítik a hibakeresést. A rosbash eszközzel csomópontokat vizsgálhatunk.

```
$ rosnode info [node_name]
$ rosnode kill [node_name]
$ rosnode list
$ rosnode machine [machine_name]
$ rosnode ping
$ rosnode cleanup
```

1.3 ROS üzenetek

A csomópontok üzeneteket cserélnek egymással egy témába való publikálással/feliratkozással. Az üzenetek egyszerű adatstruktúrák, amelyek különböző mezőtípusokkal rendelkeznek, ahogyan fennebb említettük. Az üzenetek a szabványos primitív adattípusokat és a primitív típusokból álló tömböket támogatják.

A következő technikával hozzáférhetünk az üzenetspecifikációhoz. Például az

```
st\_msgs/String
```

használatával megkaphatjuk az

```
std\_msgs/msg/String.msg
```

fájlt. A string üzenet definíciójához a roscpp kliens használata esetén be kell importálni az

```
std\_msgs/String.h
```

állományt, hasonlóan a rospy esetén a String modult. Az üzenetekkel kapcsolatos információkért létezik a rosmsg. A következő parancsokat használhatjuk:

```
$ rosmsg show [message]
$ rosmsg list
$ rosmsg md5 [message]
$ rosmsg package [package_name]
$ rosmsg packages [package_1] [package_2]
```

1.4 ROS témák

Csomóponti kommunikációt szolgálnak. A topikok anonim publikálási és feliratkozási képessége azt jelenti, hogy az üzenetek létrehozása és fogyasztása különálló folyamatok. A ROS-csomópontok csak a téma nevét ellenőrzik, valamint azt, hogy a kiadó és a feliratkozó üzenettípusa megegyezik-e; nem érdekli őket, hogy melyik csomópont publikálja vagy iratkozik fel a témákra.

A topikok csak egyirányú kommunikációt tesznek lehetővé; ha kérés-válasz kommunikációt akarunk kiépíteni, akkor ROS-szolgáltatásokat kell használnunk.

A ROS-csomópontok a TCP/IP-alapú TCPROS transzportot használják a topikokhoz való csatlakozáshoz. A ROS-ban ezt a technikát használják alapértelmezett szállítási mechanizmusként. A kommunikáció másik formája az UDPROS, amely kizárólag a távműködtetésre alkalmas, és alacsony késleltetésű, laza szállítást biztosít.

A ROS téma eszközzel a ROS témákról kaphatunk információkat. Példa:

```
$ rostopic bw /topic
$ rostopic echo /topic
$ rostopic find /message_type
$ rostopic hz /topic
$ rostopic info /topic
$ rostopic list
$ rostopic pub /topic message_type args
$ rostopic type /topic
```

1.5 ROS szolgáltatások

A szolgáltatásokat minden olyan esetben használni kell, amikor kérés/válasz típusú kommunikációra van szükség. A témák egyirányú jellege miatt ez a fajta kommunikáció nem lehetséges. Az elosztott rendszerek az esetek többségében szolgáltatásokat alkalmaznak. A szolgáltatásokat egy üzenetpár határozza meg. Egy srv fájlban meg kell adnunk a kérés adattípusát és a visszatérés adattípusát. Egy csomag belső srv alkönyvtárában találhatók az srv fájlok.

Egy csomópont egyszerre szolgál kliensként és szerverként a szolgáltatások számára, lehetővé téve, hogy a kliensek szolgáltatásokat kérjenek a szerverektől. Az eredmények akkor kerülnek elküldésre a szolgáltatás kliensének, ha a szerver sikeresen végrehajtja a szolgáltatási eljárást.

A következő megközelítéssel például elérhetjük egyes szolgáltatások definíciót, például ha a

```
my\_package/Image
```

elérheti a

```
my\_package/srv/Image.srv
```

állományt.

Van egy MD5 ellenőrző összeg, amely a szolgáltatások csomópontjait is ellenőrzi. Csak akkor válaszolhat a szerver az ügyfélnek, ha az összeg megegyezik.

Két ROS eszköz áll rendelkezésre a szolgáltatások megismeréséhez. A különböző szolgáltatástípusok megismeréséhez használható az első eszköz, a rossrv-t, amely megegyezik a rosmsg-gel. A következő parancs, a rosservice, az aktuálisan aktív szolgáltatások listázására és lekérdezésére szolgál.

Az alábbiakban néhány utasítást olvashat, hogyan használhatja a rosservice eszközt, hogy többet tudjon meg a jelenleg aktív szolgáltatásokról:

```
$ rosservice call /service args
$ rosservice find service_type
$ rosservice info /services
$ rosservice list
$ rosservice type /service
$ rosservice uri /service
```

1.6 ROS táskák

A ROS-ban a témák és szolgáltatások üzenetadatait bag-fájlokban tárolják. A bag fájlokat a.bag kiterjesztés jelöli.

A rosbag parancs egy vagy több témára való feliratkozással és az üzenet adatainak a fogadáskor történő elmentésével hoz létre bag-fájlokat. Ez a fájl újra lejátszhatja ugyanazokat a témákat, amelyekből rögzítették őket, vagy újra leképezheti az aktuális témákat.

Az adatnaplózás a rosbag elsődleges felhasználási területe. A robotadatok offline is megjeleníthetők és feldolgozhatók, valamint naplózhatók.

A rosbag fájlokkal való munkához a rosbag parancsra van szükség. A bag fájl rögzítésére és lejátszására szolgáló parancsok a következők:

```
$ rosbag record [topic_1] [topic_2] -o [bag_name]
$ rosbag play [bag_name]
```

A GUI eszköz amivel táskákkal dolgozhatunk az rqt_bag.

1.7 ROS mester

A DNS-kiszolgáló hasonló a ROS Masterhez. Minden induló ROS-csomópont elkezdi megkeresni a ROS Master-t és regisztrálni a nevét. Ennek eredményeképpen a ROS Master minden olyan csomópontról tud, amely jelenleg aktív a ROS-rendszerben. Visszahívást generál, és frissíti a legfrissebb információkkal, amikor bármelyik csomópont adatai megváltoznak. Az egyes csomópontokhoz való csatlakozáshoz ezek a csomópontadatok hasznosak.

Amikor egy csomópont elkezd közzétenni egy témát, a csomópont értesíti a ROS Master-t a téma nevéről és adattípusáról. Ha vannak további csomópontok, amelyek feliratkoztak ugyanarra a témára, a ROS Master ellenőrzi őket. A kiadó csomópont információit a ROS Master megosztja az előfizető csomóponttal, ha ugyanarra a témára több csomópont is előfizetett. A két csomópont a csomópontadatok átvétele után a TCP/IP-alapú TCPROS protokoll segítségével csatlakozik. A ROS Master nem játszik szerepet a két csomópont kezelésében, miután azok összekapcsolódtak. A preferenciáinktól függően megállíthatjuk akár a kiadó csomópontot, akár az előfizető csomópontot. Újra ellenőrzi a ROS Masterrel, ha bármelyik csomópont leállt. A ROS-szolgáltatások ugyanezt az eljárást alkalmazzák.

A ROS kliens könyvtárak, köztük a roscpp és a rospy a csomópontok létrehozására szolgálnak. Ezek a kliensek XMLRPC-alapú API-kon keresztül kommunikálnak a ROS Masterrel, amelyek a ROS rendszer API-k rendszerhátttereként szolgálnak.

A ROS Master IP-címe és portja a ROS_MASTER_URI környezeti változóban található. Ez a változó lehetővé teszi, hogy a ROS-csomópontok megtalálják a ROS Master-t. A csomópontok közötti kommunikáció nem fog megtörténni, ha ez a változó ki van kapcsolva. A localhost IP-cím vagy név használható, ha a ROS-t egyetlen rendszerben használjuk. Azonban egy elosztott hálózatban, ahol a feldolgozás sok fizikai gépen zajlik, helyesen kell definiálnunk a ROS_MASTER_URI-t. Csak így lesznek képesek a távoli csomópontok megtalálni és kommunikálni egymással. Egy elosztott rendszerben csak egy Masterre van szükségünk, és ahhoz, hogy a távoli ROS-csomópontok elérjék a Master-t, annak olyan számítógépen kell futnia, amelyet az összes többi számítógép sikeresen tud pingelni. [**joseph:ros**]

2 OpenCR

OpenCR: - A motrokat irányító modul, a motorkontroller. Itt fontos megemlíteni, hogy több könyvtár is van mellyel a motrokat irányíthatjuk. A motrokat tudjuk konfigurálni különböző vezérlési módra, állíthatunk baud rate-t, stb.:

1. Dynamixel2Arduino - gyors és egyszerű vezérlése a motroknak Arduino IDE-ben
2. Dynamixel sdk - ezt a könyvtárat használja a ROS

3 Dynamixel

Dynamixel motrok:

1. Motorleírás

4 Rugós model kód