ROS/ Tutorials/ MultipleMachines

Please ask about problems and questions regarding this tutorial on answers.ros.org. Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Running ROS across multiple machines

Description: This tutorial explains how to start a ROS system using two machines. It explains the use of `ROS_MASTER_URI` to configure multiple machines to use a single master.

Tutorial Level: INTERMEDIATE

Next Tutorial: Defining Custom Messages

Contents

Overview

ROS is designed with distributed computing in mind. A well-written node makes no assumptions about where in the network it runs, allowing computation to be relocated at run-time to match the available resources (there are exceptions; for example, a driver node that communicate with a piece of hardware must run on the machine to which the hardware is physically connected). Deploying a ROS system across multiple machines is easy. Keep the following things in mind:

- You only need one master. Select one machine to run it on.
- All nodes must be configured to use the same master, via `ROS_MASTER_URI`.
- There must be complete, bi-directional connectivity between all pairs of machines, on all ports (see ROS/NetworkSetup).
- Each machine must advertise itself by a name that all other machines can resolve (see ROS/NetworkSetup).

Talker / listener across two machines

Say we want to run a talker / listener system across two machines, named **marvin** and **hal**. These are the machines' hostnames, which means that these are the names by which you would address them when. E.g., to login to **marvin**, you would do:

```
ssh marvin
```

Same goes for **hal**.

1. Start the master

We need to select one machine to run the master; we'll go with **hal**. The first step is start the master:

```
ssh hal
roscore
```

2. Start the listener

Now we'll start a listener on **hal**, configuring `ROS_MASTER_URI` so that we use the master that was just started:

```
ssh hal
export ROS_MASTER_URI=http://hal:11311
rosrun rospy_tutorials listener.py
```

3. Start the talker

Next we'll start a talker on **marvin**, also configuring `ROS_MASTER_URI` so that the master on **hal** is used:

```
ssh marvin
export ROS_MASTER_URI=http://hal:11311
rosrun rospy_tutorials talker.py
```

Voila: you should now see the listener on **hal** receiving messages from the talker on **marvin**.

Note that the sequence of talker / listener startup doesn't matter; the nodes can be started in any order. The only requirement is that you start the master before starting any nodes.

4. Variation: connecting in the other direction

Now let's try it in the other direction. Leaving the master running on **hal**, kill the talker and listener, then bring them up on opposite machines.

First a listener on **marvin**:

```
ssh marvin
export ROS_MASTER_URI=http://hal:11311
rosrun rospy_tutorials listener.py
```

Now a talker on **hal**:

```
ssh hal
export ROS_MASTER_URI=http://hal:11311
rosrun rospy_tutorials talker.py
```

rostopic

For testing you can use the rostopic tool on all machines which are connected to the core.

You get a list of all available topics. If you are not connected to a core there is an error.

```
rostopic list
```

In wireless networks it is sometimes necessary to check if there is a connection and messages still come. For short tests it is handy to print out the messages.

```
rostopic echo /topic_name
```

When something goes wrong

If something in the above sequence didn't work, the cause is likely in your network configuration. See ROS/NetworkSetup and ROS/Troubleshooting for configuration requirements and troubleshooting tips.

One common trap is the missing define of ROS_IP on the machine, where talker.py is running.

check it with: echo $ROS_IP

If you dont't define ROS_IP, then rostopic info will show indeed the proper connections of publisher and listener, but rostopic echo will be empty. You will see no TX-traffic on LAN, on machine with talker. First, after defining ROS_IP with proper IP-address (export ROS_PI=machine_ip_addr) you will see trafic on LAN and the listener.py will show received data.

Brought to you by: Open Source Robotics Foundation