MOOS/ROS Bridge

2012

Table of Contents

1. Introduction

This document describes the basic requirements for the MOOS/ROS Bridge software application, how to install the Bridge, and basic Bridge configuration. The MOOS/ROS Bridge is an application that allows for variables (numerical data or string data) to be transferred between a ROS Core and a MOOS Database. The Bridge is configured via XML files that designate which variables are passed between the two databases and the variable types.

2. Requirements

The MOOS/ROS Bridge has been designed to be run on UNIX operating systems. All initial tests were executed on Ubuntu 10.04.

The two main requirements for the MOOS/ROS Bridge are working installations of:

- The Mission Oriented Operating Suite (MOOS) - http://oceanai.mit.edu/moos-ivp/
- The Robot Operating System (ROS) - http://www.ros.org

MOOS 12.2 was used for initial testing of the Bridge. The user should ensure that the appropriate MOOS folders are added to the system path:

```
export PATH=~/moos-ivp/MOOS/MOOSBin:~/moos-ivp/ivp/bin:$PATH
```

The Bridge was tested with ROS Fuerte. Ensure that the following line is added to the user's .bashrc or similar start up script.

```
source /opt/ros/fuerte/setup.bash
```

3. Installation

The compilation of the MOOS/ROS Bridge is done through the ROS build system. The CMakeLists.txt file in the MOOS/ROS Bridge source directory uses cmake to search for MOOS installation. Unzip the MOOS/ROS source code (moosrosbridge.5.25.2012.zip) into a directory that lies within the user's $ROS_PACKAGE_PATH. Then change into the moosrosbridge directory and use the ROS tool chain to build the MOOS/ROS Bridge.

```
$ cd moosros
$ rosmake
```

4. MOOS/ROS Bridge Configuration

The MOOS/ROS Bridge is configured via an XML file. Each variable (or message) that is passed between MOOS and ROS is defined as a message tag in the configuration XML file. This allows the user to change which messages are passed over the Bridge without having to recompile the Bridge. An example of the XML file is given in the MOOS/ROS Bridge source folder, called moosrosconfig.xml. Examining the first message in moosrosconfig.xml shows a variable called Counter0. When the Counter0 variable is updated in the ROS system, the Bridge forwards the new value for Counter0 to the MOOS database. The direction of the transfer is designated by the direction element in the message. The direction element in the message can only have two values: toMOOS or toROS. The moosname and rosname elements define the respective names of the variable in each database. While these names could be different, they are almost always the same. Finally, moostype and rostype elements define the types for the message in their respective databases. MOOS only supports two types at the time this document was written: Double and String. However, the ROS side supports a number of types. Thus, it is up to the system designer to ensure that the type conversions are realistic. The MOOS/ROS Bridge supports almost all of the ROS std_msgs at the time this document was written. Additional support for new ROS types will mean that a programmer will have to modify the src/Bridge.cpp file to provide accurate type conversions. While the MOOS/ROS Bridge was designed with a goal of minimal latency, it is not the intent of the MOOS/ROS Bridge to stream large amounts of data such as 3D point clouds or image data. Instead, the system designer should use software nodes to process the large data structures on the database side that generated the data and then pass higher-level information about the data structures over the Bridge.

5. MOOS/ROS Bridge Initial Test

In order to test the installation of the MOOS/ROS Bridge the user should setup a simple scenario where simple counter information is passed between the MOOS and ROS databases. A counter called CounterFromROS is generated on the ROS side and a counter called CounterFromMOOS is generated on the MOOS side. An XML file called counters.xml (found in the MOOS/ROS Bridge source files) sets up the Bridge to transfer the counters between the two databases. In order to test the Bridge execute the following steps:

1. Build the moosros_tester ROS program that will increment a counter once a second and publish it to the ROS Core:

```
$ cd moosros_tester
$ rosmake
```

2. In a separate terminal, start up the MOOS Database:

```
$ MOOSDB
```

3. In a separate terminal, start up ROS Core:

```
$ roscore
```

4. In a separate terminal, start up the MOOS/ROS Bridge from within the MOOS/ROS directory in order to be in the same directory as counters.xml and bridge.moos:

```
$ roscd moosros
$ rosrun moosros Bridge counters.xml bridge.moos
```

5. In a separate terminal, start up the tester program:

```
$ rosrun moosros_tester counter
```

6. In a separate terminal, start up the MOOS uMS debug program:

```
$ uMS
```

7. In a separate terminal, type

```
rostopic echo /CounterFromMOOS
```

8. Hold Control and Left-Click in the uMS window just below the "CounterFromROS" box and type in "CounterFrom-MOOS". Click OK and then choose "numeric." Enter a new value for "CounterFromMOOS" and click OK. The user should see the updated value print in the terminal that is monitoring the "CounterFromMOOS" variable in the previous step.

6. Known Issues / Concerns

As of ROS cturtle and Fuerte, the topic name is not always transmitted in the header. In order to work around this, the ros::TransportHints().unreliable() was used when setting up the ROS subscribers. This works well with this modification, but the functionality should be monitored since Willow Garage seems to change their mind about this functionality often. There is another way to remove the unreliable() logic, but it is quite difficult with the callback functions. See the Bridge.cpp code for a more detailed explanation.