ROSbridge communication between ROS1 and ROS2

Hammoudeh Garcia Nadia, Eram Arfa

December 2021

1 Introduction

Rosbridge provides a JSON API to ROS functionality for non-ROS programs. There are a variety of front ends that interface with rosbridge, including a WebSocket server for web browsers to interact with. For communication between ROS1 and ROS2, the bridge has pre built ROS2 binaries that include support for common ROS interfaces.
Topics are bridged when matching publisher and subscriber are active for a topic on either side of the bridge. To specify the topic type explicitly for ROS2 **ros2 topic echo topic-name topic-type** command is used. On the ROS1 side one can make use of −**bridge-all-2to1-topics** option to bridge all ROS 2 topics to ROS1 so that tools such as rostopic echo, rostopic list and rqt will see the topics even if there are no matching ROS 1 subscribers.

2 Steps to build the bridge

The first step before setting up ROSbridge is to have ROS1 and ROS2 up and running on the Ubuntu. In this case, the version of Ubuntu is 20.04 LTS, ROS1 Noetic and ROS2 foxy is installed. The workspaces are created and built beforehand. To install and set up ROS1 and ROS2 the following references can be used. ROS1 installation steps and ROS2 installation steps
There are also some prerequisites that need to be fulfilled before starting to run the bridge. The bridge needs to be build before running with other ROS2 packages from the source. One needs to source the environment of the install space where the bridge was built or unpacked to. Additionally **ROS1** needs to be sourced to the correct environment and start **roscore**. Also some of the ros1 packages need to be installed and built to use the bridge. The list is as followed

- catkin
- roscpp
- roslaunch

- rosmsg
- std_msgs
- rospkg
- rosbash
- roscpp_tutorials
- rospy_tutorials
- rostopic
- rqt_image_view

ROS2 has to be built from the source following the instructions given here.
The next step is to install ros bridge package into the workspace by following the steps given below:

- Navigate to the workspace by using the following command **cd ros2_example_ws/src**
- Clone the repository **git clone https://github.com/ros2/ros1_bridge.git**
- You should first build everything but the ROS 1 bridge with normal colcon arguments. It is not recommended having ROS 1 environment sourced during this step as it can add other libraries to the path.
  colcon build –symlink-install –packages-skip ros1_bridge
- Next source the ROS 1 environment, for Linux and ROS Noetic that would be: **source /opt/ros/noetic/setup.bash**
  Or **. /ros_catkin_ws/install_isolated/setup.bash**
- The bridge will be built with support for any message/service packages that are on your path and have an associated mapping between ROS 1 and ROS 2. Therefore you must add any ROS 1 or ROS 2 workspaces that have message/service packages that you want to be bridged to your path before building the bridge. This can be done by adding explicit dependencies on the message/service packages to the package.xml of the bridge, so that colcon will add them to the path before it builds the bridge.
- Another way of doing it is by sourcing the relevant workspaces ourselves like below
  - You have already sourced your ROS installation above.
  - Source your ROS 2 installation: . install-space-with-ros2/local_setup.bash
  - And if you have a ROS 1 overlay workspace, something like: . install-space-to-ros1-overlay-ws/setup.bash
  - And if you have a ROS 2 overlay workspace, something like: . install-space-to-ros2-overlay-ws/local_setup.bash

- Then build just the ROS 1 bridge:
  colcon build –symlink-install –packages-select ros1_bridge –cmake-force-configure

3 Examples of ROS1 and ROS2 communication using ROSBridge

3.1 ROS1 talker and ROS2 listener

- First, start a ROS1 core
  - **SHELL A**
    (ROS 1 only): source /opt/ros/noetic/setup.bash
    Or: source home/catkin_ws/devel/setup.bash
    roscore
- - **SHELL B**
    (ROS 1 and ROS2):
    Source ROS 1 first: source /opt/ros/noetic/setup.bash
    Or: source home/catkin_ws/devel/setup.bash
    Source ROS 2 next:
    source home/ros2_example_ws/install/localsetup.bash
    export ROS_MASTER_URI=http://localhost:11311
    ros2 run ros1_bridge dynamic_bridge
    The program will start outputting the currently available topics in ROS 1 and ROS 2 in a regular interval.
- Now, start the ROS1 talker
  - **SHELL C**
    (ROS 1 only): source home/catkin_ws/devel/setup.bash
    rosrun ros_basics talker.py
- Next, start the ROS 2 listener from the demo_nodes_cpp ROS 2 package.
  - **SHELL D**
    (ROS 2 only): source home/ros2_example_ws/install/localsetup.bash
    OR
    source /opt/ros/foxy/setup.bash
    ros2 run demo_nodes_cpp listener
- When looking at the output in shell B there will be a line stating that the bridge for this topic has been created.
- At the end stop all programs with Ctrl-C. Once you stop either the talker or the listener in shell B a line will be stating that the bridge has been torn down.

3.2 ROS2 talker and ROS1 listener

- First, start a ROS1 core
  - **SHELL A**
    (ROS 1 only): source /opt/ros/noetic/setup.bash
    Or: source home/catkin_ws/devel/setup.bash
    roscore
- - **SHELL B**
    (ROS 1 and ROS2):
    Source ROS 1 first: source /opt/ros/noetic/setup.bash
    Or: source home/catkin_ws/devel/setup.bash
    Source ROS 2 next:
    source home/ros2_example_ws/install/localsetup.bash
    export ROS_MASTER_URI=http://localhost:11311
    ros2 run ros1_bridge dynamic_bridge
    The program will start outputting the currently available topics in ROS 1 and ROS 2 in a regular interval.
- Now, start the ROS1 talker
  - **SHELL C**
    (ROS 1 only): source home/catkin_ws/devel/setup.bash
    ros2 run demo_nodes_cpp talker
- Next, start the ROS 2 listener from the demo_nodes_cpp ROS 2 package.
  - **SHELL D**
    (ROS 2 only): source home/ros2_example_ws/install/localsetup.bash
    OR
    source /opt/ros/foxy/setup.bash
    rosrun ros_basics listener.py
- When looking at the output in shell B there will be a line stating that the bridge for this topic has been created.
  Passing message from ROS 2 std_msgs/msg/String to ROS 1 std_msgs/String (showing msg only once per type) removed 2to1 bridge for topic '/chatter' created 2to1 bridge for topic '/image' with ROS 2 type 'sensor_msgs/msg/Image' and ROS 1 type "

3.3 Run the bridge to exchange images

- Launch ROS master
  - **SHELL A**
    (ROS 1 only): source /opt/ros/noetic/setup.bash
    Or: source home/catkin_ws/devel/setup.bash
    roscore

- Launch the bridge
  - **SHELL B**
    Source ROS 1 first: source /opt/ros/noetic/setup.bash
    Or: source home/catkin_ws/devel/setup.bash
    Source ROS 2 next:
    source home/ros2_example_ws/install/localsetup.bash
    export ROS_MASTER_URI=http://localhost:11311
    ros2 run ros1_bridge dynamic_bridge
    The program will start outputting the currently available topics in ROS 1 and ROS 2 in a regular interval.
- Start ROS1 GUI
  - **SHELL C**
    (ROS 1 only): source home/catkin_ws/devel/setup.bash
    rqt_image_view /image
- Now we start the ROS 2 image publisher from the image_tools ROS 2 package:
  - **SHELL D**
    (ROS 2 only): source home/ros2_example_ws/install/localsetup.bash
    OR
    source /opt/ros/foxy/setup.bash
    ros2 run image_tools cam2image
- When looking at the output in shell B there will be a line stating that the bridge for this topic has been created.
  Passing message from ROS 2 std_msgs/msg/String to ROS 1 std_msgs/String (showing msg only once per type) removed 2to1 bridge for topic '/chatter' created 2to1 bridge for topic '/image' with ROS 2 type 'sensor_msgs/msg/Image' and ROS 1 type ''
- To exercise the bridge in the opposite direction at the same time you can publish a message to the ROS 2 node from ROS 1. By publishing either true or false to the flip_image topic, the camera node will conditionally flip the image before sending it. You can either use the Message Publisher plugin in rqt to publish a std_msgs/Bool message on the topic flip_image, or run one of the two following rostopic commands:
  - **SHELL E** (ROS 1 only):
    source home/catkin_ws/devel/setup.bash
    rostopic pub -r 1 /flip_image std_msgs/Bool "data: true"
    rostopic pub -r 1 /flip_image std_msgs/Bool "data: false"

3.4 Run the bridge for AddTwoInts service

- Launch ROS master
  - **SHELL A**
    (ROS 1 only): source /opt/ros/noetic/setup.bash
    Or: source home/catkin_ws/devel/setup.bash
    roscore -p 11311
- - **SHELL B**
    Launch the dynamic bridge
    Source ROS 1 first: source /opt/ros/noetic/setup.bash
    Or: source home/catkin_ws/devel/setup.bash
    Source ROS 2 next:
    source home/ros2_example_ws/install/localsetup.bash
    export ROS_MASTER_URI=http://localhost:11311
    ros2 run ros1_bridge dynamic_bridge
    The program will start outputting the currently available topics in ROS 1 and ROS 2 in a regular interval.
- Launch TwoInts server
  - **SHELL C**
    (ROS 1 only): source home/catkin_ws/devel/setup.bash
    ros2 run ros1_bridge dynamic_bridge
- Launch AddTwoInts client.
  - **SHELL D**
    (ROS 2 only): source home/ros2_example_ws/install/localsetup.bash
    OR
    source /opt/ros/foxy/setup.bash
    ros2 run demo_nodes_cpp add_two_ints_client