ChatGPT

Troubleshooting Gazebo Sim IMU Sensor Bridge to ROS 2

Problem Overview

You have set up a Gazebo Sim (Ignition Gazebo) environment in a non-admin (Aalto) Linux machine using a Micromamba environment. The goal is to simulate a drone with an IMU sensor and bridge its data to ROS 2 (Humble). You followed various steps (running `gz sim` headlessly with a world SDF, launching `ros_gz_bridge` with YAML configs, setting environment variables, etc.), but **no IMU data is being received on the ROS 2 topic**. Specifically, the `/sim/imu` ROS topic exists (created by the bridge), but no messages are arriving, and even Gazebo's own topic list (`gz topic -l`) showed no IMU topics. You also saw **"Exception sending a multicast message: Network is unreachable"** errors in Gazebo logs, and had issues with the Gazebo GUI (scene not rendering, plugin errors).

In summary, the IMU sensor appears to not be publishing, and the ROS bridge is consequently relaying nothing. We will break down the likely causes and provide solutions.

Likely Causes and Observations

1. IMU Sensor Not Publishing in Gazebo

The primary cause of no data is likely that **the IMU sensor plugin is not active in the Gazebo simulation**. In Gazebo Sim (Ignition), simply defining a `<sensor type='imu'>` in SDF is not enough – you also need to have the corresponding **Imu system plugin** loaded so that the sensor actually generates data. If this plugin is missing, the IMU topic will never publish (it won't even appear in `gz topic -l`). This is a common oversight when moving from classic Gazebo to Ignition Gazebo. In fact, a similar issue was reported by another user: *"I have an IMU in my SDF and no data is being published"*, and the answer was: *"You're probably missing the Imu world plugin in the world SDF"* [1]. In your case, your world SDF did load some plugins (Physics, SceneBroadcaster, etc.), but **did you include the IMU system plugin?**

- **Symptoms:** `gz topic -l | grep -i imu` shows nothing, and `gz topic -e -t <imu_topic>` times out. The IMU sensor is defined in SDF with `<always_on>1</always_on>`, `<update_rate>100</update_rate>`, etc., but no messages. Gazebo log shows no specific sensor errors – meaning the sensor probably never initialized.

- **Cause:** The Gazebo Sim `Imu` **system plugin** (responsible for simulating IMU measurements) is not loaded. Gazebo's new modular system requires explicitly loading sensor plugins. If the IMU plugin isn't loaded, the sensor will be ignored. (By contrast, sensors like cameras or lidars often require the generic "Sensors" plugin with a rendering engine; IMUs require the `Imu` plugin.)

2. Gazebo Transport Discovery / Network Issues

The "Network is unreachable" error suggests that Gazebo's transport was unable to use the default multicast discovery mechanism (likely blocked by the network or firewall on the Aalto PC). This can prevent Gazebo components from discovering each other's topics. Initially, you saw **no topics at all** in `gz topic -l` (not even `/clock`), which indicates a discovery failure. You mitigated this by setting environment variables like `IGN_IP=127.0.0.1` and `ROS_LOCALHOST_ONLY=1`. Indeed, it's recommended to restrict Gazebo to localhost in such cases: *"try setting* `IGN_IP=127.0.0.1` *on all terminals to limit discovery to localhost"* [2]. After this, Gazebo's transport should function on the local interface, and `gz topic -l` should list topics.

- **Symptoms:** Gazebo runs but `gz topic -l` returns empty, and log repeats *"Exception sending a multicast message: Network is unreachable"*. This means the simulator couldn't broadcast its presence on the network (common in restricted or VM environments with no multicast). As a result, the `ros_gz_bridge` and other tools might not see any Gazebo topics.

- **Cause:** Network/multicast discovery issues due to the restricted environment. By default, Gazebo transport uses UDP multicast for discovery, which is failing. Without discovery, topics won't be visible to `gz topic` or bridges.

3. ROS 2 Bridge Configuration Mismatch

Another potential issue is **message type mismatches in the ROS 2 bridge configuration**. You tried two YAML configs: one mapping `ignition.msgs.IMU` to `sensor_msgs/Imu`, and another updated to `gz.msgs.IMU` to `sensor_msgs/Imu`. It's crucial to use the correct message namespace corresponding to your Gazebo version:

- Gazebo **Fortress (Ignition Gazebo)** uses `ignition.msgs.Type`.
- Gazebo **Garden (Gazebo Sim 7+)** uses `gz.msgs.Type`.

Your environment (libgz-sim7, gz-transport12, gz-msgs9 with ROS 2 Humble) suggests **Gazebo Garden** or later. In that case, the correct message name for an IMU is `gz.msgs.IMU`. Using the wrong one means the bridge won't actually subscribe to the Gazebo topic. For example, a ROS 2 integration snippet for Fortress uses `... Imu[sensor_msgs/msg/Imu[ignition.msgs.IMU` [3]. For Garden, it should be `[gz.msgs.IMU`. If your bridge was configured with the wrong type at any point, it would create the ROS topic but not connect to the Gazebo publisher.

- **Symptoms:** The ROS topic `/sim/imu` exists (so the bridge is running and created it), but `ros2 topic echo /sim/imu` shows nothing even though the simulation is running. Meanwhile, if you inspect Gazebo transport (once discovery is fixed), you might see the IMU topic is present and publishing on the Gazebo side.

- **Cause:** Using `ignition.msgs.IMU` vs `gz.msgs.IMU` inconsistently with the Gazebo version. If ros_gz_bridge was installed for Fortress by default, it might expect Ignition message types. Using the correct branch or specifying the type fixes this. (It sounds like you tried both in separate attempts, so this might be secondary to the sensor not publishing at all. Still, it's worth ensuring consistency.)

4. Gazebo GUI and Rendering Issues (Secondary)

You also noted GUI problems (Gazebo GUI not rendering the scene, QML errors like missing Grid3D, etc.). These likely do not affect the headless sensor data publishing, but they indicate your GUI isn't fully functional – possibly due to missing plugins in the conda environment or OpenGL issues (software rendering needed). For example, running the GUI without specifying `--render-engine ogre2` might default to OGRE2 anyway (which requires GPU/GL support). The errors about "WorldControl service missing" might be related to using a custom partition or not launching the world control plugin. Since your focus is the headless simulation and ROS bridge, the GUI issues can be addressed separately (e.g., by ensuring `gz-gui` plugins are installed and by using `LIBGL_ALWAYS_SOFTWARE=1` for software rendering if no GPU). This is less critical for getting IMU data to ROS, so we will focus on the core headless pipeline.

Solutions and Next Steps

Addressing the above points, here are the steps to get the IMU data flowing from Gazebo to ROS 2:

1. Enable the IMU Sensor Plugin in SDF

Make sure your world or model SDF loads the IMU system plugin. There are two ways to do this:

- **As a World plugin:** Add the plugin under the `<world>` tag. For example:

```
<plugin filename="gz-sim-imu-system" name="gz::sim::systems::Imu"/>
```

This globally enables simulation of all IMU sensors in that world.

- **As a Sensor plugin:** Alternatively, within your `<sensor type="imu" ...>` element, include the plugin:

```
<sensor name="simple_imu" type="imu">
  ...
  <plugin filename="gz-sim-imu-system" name="gz::sim::systems::Imu"/>
</sensor>
```

Either method will load Gazebo's IMU system. This plugin is what actually publishes `gz.msgs.IMU` data on the Gazebo topic for your sensor. *Without it, the IMU will remain idle.* This is a known requirement in Ignition/Gazebo Sim: another user's IMU topic was empty until they "added the Imu world plugin" to their SDF, after which the IMU data began publishing [1].

Double-check the SDF you used (`simple_world.sdf` and `drone_world_new.sdf`) for the presence of an Imu plugin. If it's missing, add it as above. After adding, run `gz sim` again (with `-v 4` for verbose output) and watch the log; you should see that the `Imu` system is loaded. Then use `gz topic -l | grep imu` – you should now see a topic like `/world/<worldname>/model/<modelname>/link/<link>/sensor/<sensor>/imu` being advertised. You can further inspect it with

`gz topic -i -t <topicName>` to confirm it's of type `gz.msgs.IMU`. This will confirm the sensor is live.

2. Fix Gazebo Transport Networking

Since you're on a restricted network, continue using the workaround to force local communication:

- **Set** `IGN_IP=127.0.0.1` in every terminal that runs Gazebo or its tools (including the bridge). This ensures Gazebo transport uses the loopback interface and avoids multicast on unreachable networks. As an Open Robotics engineer suggested, this limits discovery to localhost and circumvents the multicast issue [2]. You've done this, but ensure it's exported in the environment before launching both `gz sim` and `ros2` bridge commands.

- You can keep `ROS_LOCALHOST_ONLY=1` for ROS 2 as well (this restricts DDS to loopback, which is fine here). It might not directly affect Gazebo, but it doesn't hurt given your scenario.

- **Consistent GZ/IGN Partition:** You set `GZ_PARTITION=sim_cli` (and `IGN_PARTITION=sim_cli`). In Gazebo Garden, the `GZ_PARTITION` env var controls the transport partition (Ignition used `IGN_PARTITION`). Using a custom partition is okay, but ensure *all* processes use the same value – the Gazebo server, GUI (if used), and ros_gz_bridge must have matching partition to talk to each other. If in doubt, you can also stick to the default (unset) partition for simplicity. The key is consistency. If after launching everything you still see no topics, a partition mismatch could be why, so double-check this.

After these network settings, verify that `gz topic -l` (or `gz topic -l --timeout <sec>`) shows the list of topics. You should at least see `/world/<name>/clock` and your IMU topic, etc. If not, there's still a discovery issue.

3. Use the Correct ROS 2 Bridge Configuration

Now that Gazebo is publishing an IMU topic, the ROS 2 bridge needs to relay it. Make sure the bridge is configured for the **exact Gazebo message type and topic name**:

- If using a YAML config, set the Gazebo message type to `gz.msgs.IMU` (for Garden and later) or `ignition.msgs.IMU` (for Fortress). In your case with libgz-sim7, it should be `gz.msgs.IMU`. Your updated `bridge.yaml` was on the right track. For example, a YAML entry might look like:

```
- topic_name: "/world/simple_world/model/simple_drone/link/base_link/sensor/
simple_imu/imu"
  ros_type_name: "sensor_msgs/msg/Imu"
  gz_type_name: "gz.msgs.IMU"
  direction: GZ_TO_ROS
```

Ensure the `topic_name` matches exactly what `gz topic -l` showed for the IMU. (You can also give the IMU sensor a simpler topic in SDF, like `<topic>/imu</topic>`, to avoid the long default name.)

- If you launch the bridge via CLI arguments (as you also attempted), use the `@` separator for ROS and `[` for Gazebo type. For example:

  ```
  ros2 run ros_gz_bridge parameter_bridge /imu/data@sensor_msgs/msg/
  Imu[gz.msgs.IMU]
  ```

  This would bridge a Gazebo topic `/imu/data` if your sensor was configured to use that topic name.

- **Verify Bridge Subscription:** After launching `ros_gz_bridge`, you can confirm it connected by running `gz topic -i -t <imu_topic>` again. It should list a subscriber (the bridge) on that Gazebo topic. Similarly, `ros2 topic info /sim/imu` should show a publisher (the bridge) on the ROS side. If those are present, the bridge is correctly set up.

- **RMW Implementation:** You experimented with `RMW_IMPLEMENTATION=rmw_fastrtps_cpp` vs CycloneDDS. Either should work for bridging, but FastRTPS (the default in Humble) is fine. The empty ROS topic issue was likely not DDS-related but due to no Gazebo data. So this is likely not a factor once the sensor is fixed. Still, after everything, if you suspect ROS2 DDS issues, you can stick with FastRTPS and ensure the ROS Domain ID matches default (usually 0) – typically not a problem unless changed.

At this point, if the IMU plugin is loaded and the bridge is subscribed, you *should* start seeing data on `/sim/imu`. Try running `ros2 topic echo /sim/imu -n 5` to see a few messages. They should contain orientation (quaternion), angular velocity, and linear acceleration readings. If your drone is just sitting, you might see orientation constant and angular vel/accel near zero (maybe just gravity on accel Z). That still counts as valid data. If you see those, congrats – the end-to-end path is working!

4. Testing with a Known Good World (Optional)

If issues persist, it can help to isolate the problem by testing a known working simulation:

- **Gazebo Example World:** Gazebo Sim comes with example worlds (in `gz-sim/examples/worlds`) that include sensors. For instance, there are worlds demonstrating IMU, LIDAR, etc. You could try running one of those to see if the IMU publishes in your setup. For example, the Gazebo tutorial's final world in the Sensors demo or any world with an `<imu>` sensor and proper plugin. Launch it with `gz sim` and check topics. This can rule out any mistake in your custom SDF. (Ensure you use `-r` to run the simulation unpaused, or hit the play button if using GUI, since a paused simulation won't produce sensor data.)

- **ROS 2 Demo:** There is a package `ros_gz_sim_demos` [4] which includes launch files that spawn a robot with sensors and bridges them. Since you might not easily install that in micromamba, manually doing the above is fine. But the idea is to confirm that, for example, a simple IMU on a box publishes data in your environment.

If the example world's IMU also doesn't publish, it points to an environment issue (e.g., missing Gazebo sensor components in your conda environment). You might then need to verify that `libgz-sensorsX` is installed in the environment and matches the version. Conda-forge's `libgz-sim7` should have pulled in `libgz-sensors` dependency. If something is missing, you could try installing `libgz-sensors9` (or appropriate version) via micromamba. The fact that there were no explicit plugin load errors in the log is a good sign that the libraries are there – just possibly not invoked until you add the plugin tag.

5. Considering Container or Different Setup (Last Resort)

In case you still hit roadblocks (especially due to the restricted PC environment), consider using a container or alternate installation that's known to work:

- **Docker/Podman:** If you can use Docker or Podman, Open Robotics provides Docker images for Gazebo and ROS. You could run a container with Gazebo Garden and ROS 2 Humble that has everything pre-installed. Then use bridge inside the container or network it with your ROS nodes. This avoids the need for admin privileges on the host (except for Docker itself) and bypasses the conda setup. However, not all restricted systems allow Docker; if not, try Apptainer/Singularity which can often run without root. Your mention of Apptainer on Triton is a good approach if available.

- **Apt installation with admin help:** If there's any way to get the IT admin to install Gazebo Fortress/Garden via apt or allow you to use `ros_gz`, that might simplify things. For example, ROS Humble's default ignition is Fortress (gazebo11 classic or Fortress new) which might be easier to use out-of-the-box with `ros_ign_bridge` (the older bridge name). But since you got pretty far with conda, this is only if you suspect the conda environment is the culprit.

- **Verify Versions Alignment:** If using a container or new setup, use a recommended pairing (e.g., ROS Humble with Gazebo Fortress + `ros_ign_bridge`, or ROS Humble with Gazebo Garden + `ros_gz_bridge` built from source). Mixing versions can cause subtle issues. In your logs, you mentioned using ros_gz with Humble and pinning libgz-sim7 (Garden). That's okay as long as you checked out the Humble branch of ros_gz which supports Garden. If not, that could also cause the bridge to malfunction. The ros_gz README and ROS 2 Docs list which combination is supported.

6. Gazebo GUI Tips (Optional)

Though not critical for the headless operation, a couple of notes on the GUI problems you faced:

- Launching `gz gui` with `--render-engine ogre2` (or setting it in your GUI config) can sometimes help if the default render engine didn't initialize. The error about missing `Grid3D.qml` is often harmless (just the grid overlay missing). The "WorldControl service not found" might be because the GUI couldn't connect to the simulation (e.g., partition mismatch or not running). Once the server is running with the correct partition and IGN_IP, starting `gz gui` in the same environment should connect and allow play/pause etc. Make sure the GUI is using the same `GZ_PARTITION` / `IGN_IP` environment.

- If the GUI still shows rendering issues, you can try running with `LIBGL_ALWAYS_SOFTWARE=1 gz gui` to force software rendering (useful on machines without proper OpenGL support or when using X forwarding). This can be slow, but at least you'll see something.

- You mentioned manually adding plugins like Scene3D, EntityTree, etc. That's fine. The "render plugin load issues" might indicate missing rendering plugin libraries – but if headless works, you can defer this. You have already created simplified GUI config files; use those to test incrementally.

In summary, **focus on getting the IMU to publish and bridging that data**. Once you see actual IMU messages on the ROS side, you have a working simulation pipeline. From there, you can feed those IMU readings into your EKF for the drone. The key adjustments were enabling the IMU plugin in Gazebo and configuring networking and message types correctly. After applying these fixes, multiple users in similar situations have reported that their IMU data started flowing [1]. Good luck, and happy simulating!

Sources

- Gazebo transport network issue and solution (use loopback for discovery) [2].
- Resolution for missing IMU data by adding the IMU plugin to SDF [1].
- Example of adding `gz::sim::systems::Imu` plugin in SDF [5].
- ROS 2 bridge configuration snippet (Fortress example with IMU) [3].

---

[1] Can't see IMU plugin data - Gazebo Help - Open Robotics Discourse
https://discourse.openrobotics.org/t/cant-see-imu-plugin-data/49039

[2] ros - Gazebo simulation performance problem on long run - Robotics Stack Exchange
https://robotics.stackexchange.com/questions/113901/gazebo-simulation-performance-problem-on-long-run

[3] ros - Gazebo Bridge is not working correctly in Gazebo Sim - Robotics Stack Exchange
https://robotics.stackexchange.com/questions/111988/gazebo-bridge-is-not-working-correctly-in-gazebo-sim

[4] ros_gz_sim_demos - ROS documentation
https://docs.ros.org/en/jazzy/p/ros_gz_sim_demos/

[5] How to import plugins? - PX4 Autopilot - Discussion Forum for PX4, Pixhawk, QGroundControl, MAVSDK, MAVLink
https://discuss.px4.io/t/how-to-import-plugins/37057