ROS2 Multi-Band Network Optimization for Martian Rover

Administrative Info

Project: ROS2 Multi-Band Network Optimization for Martian Rover
Responsibles: Arno Laurie (Software Systems Engineer) & Giovanni Ranieri (Vice-President Research)
Contact: arno.laurie.@epfl-xplore.ch & giovanni.ranieri@epfl-xplore.ch

1 Context and Motivation

EPFL Xplore is a student-led Swiss robotics association, part of the MAKE initiative at EPFL. Each year the team builds a Martian-like rover to compete in the European Rover Challenge (ERC) in Krakow. The team has around 50 students. The software team is split into navigation, robotic arm, and control station groups.

The rover software is built on **ROS2** (Robot Operating System 2). ROS2 does not talk to the network directly: it uses a middleware layer called the **RMW** (ROS Middleware). Our rover currently uses Cyclone DDS, a DDS implementation that sends data over the network using the **RTPS** protocol (the standard DDS wire protocol) on top of UDP. Note that the default RMW in recent ROS2 versions is Fast DDS; using Cyclone DDS is a deliberate choice on our side.

Today all ROS2 topics share a **single WiFi channel** on the 5 GHz band. At the competition, many teams emit on nearby channels, so this single channel becomes congested. The result is packet loss, connection drops, and higher latency for teleoperation.

DDS already lets us tune each topic through *Quality of Service* (QoS) settings such as reliability, history, deadline, latency budget, and transport priority. So it is not true that we have *no* control. What we are missing is: (1) the ability to send a given topic over a **chosen physical link/band**, and (2) the ability to **change that choice automatically** when the link quality changes.

2 Technical Background

The communication stack has several layers the student will work across:

- **ROS2 application:** nodes and topics. Some topics matter more than others (e.g. emergency stop and teleoperation commands are critical; a camera stream is heavy but less critical).
- **RMW / middleware:** Cyclone DDS (RTPS), or the newer **rmw_zenoh**, an official non-DDS alternative added in ROS2 Jazzy. Zenoh is designed for constrained and lossy networks, which makes it interesting for our case.
- **Transport:** UDP/IP. In RTPS, a participant announces "locators" (interface + address + port); a machine with several interfaces announces several locators. This is the mechanism we can use to steer topics onto a specific interface.
- **Physical links:** the radios themselves. The candidate bands are very different in capacity:
  - 2.4 / 5 GHz WiFi: high bandwidth (Mbps to hundreds of Mbps), but congested at the competition.
  - 433 / 868 MHz (sub-GHz, e.g. LoRa/FSK): very low bandwidth (roughly a few kbps) and limited by EU duty-cycle and power rules. These bands *cannot* carry video, but they are robust and can act as a low-rate lifeline when WiFi fails.

Because the bands are so different, the goal is not to split data evenly, but to match each topic to a suitable link based on how important and how heavy it is, and to adapt this choice when conditions change.

3 Research Question and Objectives

Research question. *Can an online, importance-aware assignment of ROS2 topics to several physical links reduce the latency of critical topics under network congestion, compared to the current single-channel setup?*

To answer this, the student will:

1. Study how ROS2, DDS/RTPS, and QoS work, and how topics can be bound to specific network interfaces (also looking at what rmw_zenoh offers).
2. Formulate an optimization problem that assigns topics to links using topic importance and live link metrics, in order to minimize latency.
3. Implement the assignment (static first, then adaptive as a stretch goal) on the rover.
4. Validate the solution with real experiments on the rover, under controlled congestion.
5. Write a final report with the analysis, the implementation, and the results.

4 Optimization Problem (Sketch)

We describe the problem in a simple form; the student is expected to refine it.

Let each topic t have a rate, a message size, an importance weight w_t, and a latency target. Let each link ℓ have a capacity C_ℓ, a measured latency, and a measured loss. We define a binary variable

$$x_{t\ell} = \begin{cases} 1 & \text{if topic } t \text{ is sent over link } \ell, \\ 0 & \text{otherwise.} \end{cases}$$

A simple version of the objective is to minimize the total importance-weighted latency:

$$\min_x \sum_t \sum_\ell w_t \, L_{t\ell} \, x_{t\ell}$$

subject to

$$\sum_\ell x_{t\ell} = 1 \qquad \text{(each topic on exactly one link)}, \tag{1}$$

$$\sum_t r_t \, x_{t\ell} \le C_\ell \qquad \text{(link capacity not exceeded)}, \tag{2}$$

where $L_{t\ell}$ is the expected latency of topic t on link ℓ and r_t its rate.

Link quality changes over time and other teams act as adversarial interference, so a fixed assignment is not enough. The interesting part is the **online** version, where the assignment is updated as conditions change. Possible directions include a periodic re-solve of the assignment problem, a feedback-control loop, or a learning approach (for example a contextual bandit that picks a link per topic and uses the measured latency as feedback). Related theory the student can build on: network utility maximization, multipath scheduling (as in MPTCP), and backpressure/Lyapunov stability for time-varying wireless links.

5 Scope and Constraints

To keep the project realistic:

- Heavy perception data (camera, LiDAR) stays on WiFi.
- Sub-GHz radios must respect EU duty-cycle and transmit-power limits (ETSI EN 300 220).
- The static, importance-aware assignment plus a solid measurement setup is the **core deliverable**. The online/adaptive policy is a **stretch goal**.

6 Requirements

We are looking for a highly motivated student who wants to tackle a real software and networking challenge. ROS2 is used by most of the robotic community, but few people know what happens behind the scenes in terms of networking. This project digs into harder topics in networking and distributed systems that stay hidden under the simple ROS2 interface we use every day, and that we almost never look at in normal development.

- Hands-on experience with ROS2, including how nodes, topics, and QoS settings actually behave.
- Good understanding of network protocols (TCP/UDP, IP) and of how data moves across a real network; prior exposure to DDS/RTPS or wireless links is a strong plus.
- Comfortable with distributed systems ideas such as latency, reliability, and message loss.
- Strong programming skills in C++, able to work in an existing codebase.
- Confident on the Linux command line and with debugging real systems.
- Strong analytical mindset: able to define a problem clearly, measure it, and reason about the results.
- Clear written and verbal communication; able to document design decisions.
- Able to work both independently and within a student team.

7 Deliverables

- A working multi-band topic-to-link assignment for ROS2 on the rover.
- A measurement setup and reproducible experiments comparing it to the current baseline.
- A final report covering the analysis, the implementation, and the experimental results.

8 Resources

- ROS2 documentation: `https://docs.ros.org/en/humble/index.html`
- ROS2 QoS settings: `https://docs.ros.org/en/humble/Concepts/Intermediate/About-Quality-of-Service-Set html`
- rmw_zenoh (Zenoh RMW for ROS2): `https://github.com/ros2/rmw_zenoh`
- Zenoh + ROS2 integration: `https://zenoh.io/blog/2021-04-28-ros2-integration/`
- Cyclone DDS: `https://github.com/eclipse-cyclonedds/cyclonedds`