Communication Infrastructure for CubeSat FlatSat Testbed

Lab: Ω-Space Group, ORION Lab, National Technical University of Athens

Duration: 3-6 months

Level: Master Thesis

Background

Reliable inter-subsystem communication is critical for CubeSat operations. This thesis focuses on implementing a hybrid communication infrastructure for the FlatSat testbed that combines modern Space ROS (ROS2 DDS) over Gigabit Ethernet with traditional satellite protocols (CubeSat Space Protocol over CAN bus). This dual-layer approach enables both high-bandwidth data transfer and flight-like control messaging, while exploring the integration of emerging (Space ROS) and established (CSP/CAN) satellite communication paradigms.

Objectives

Develop a communication infrastructure that:

1. Implements Space ROS (ROS2 DDS) networking over Gigabit Ethernet for high-bandwidth communication
2. Adds CubeSat Space Protocol (CSP) over CAN bus for control messaging
3. Creates bridge nodes between ROS2 and CSP domains
4. Provides reliable message routing across both communication layers
5. Integrates with C&DH (Raspberry Pi 4) and Payload (Jetson) subsystems
6. Includes monitoring, debugging, and diagnostic capabilities for both networks

Scope of Work

Phase 1: Space ROS Network Architecture

Network Design & Setup

- Configure Gigabit Ethernet network topology (switch, IP addressing scheme)
- Setup Space ROS (ROS2 Humble or Iron) on all nodes (RPi4, Jetson, any STM32 with micro-ROS if included)
- Configure ROS2 DDS middleware (FastDDS or CycloneDDS)
- Define Quality of Service (QoS) policies for different data types (telecommands: reliable delivery, telemetry: best-effort with history, bulk data: reliable with large buffers, housekeeping: best-effort periodic)

ROS2 Communication Patterns

- Topic-based publish/subscribe for telemetry and data streams
- Service-based request/reply for synchronous operations
- Action-based for long-running commands
- Parameter server for configuration management

Network Services

- Message routing and namespace management
- Node discovery and health monitoring
- Network diagnostics (latency measurement, packet loss detection)
- Performance profiling tools (topic bandwidth, message frequency)

C&DH Integration (Primary)

- Full ROS2 integration with C&DH subsystem
- Telecommand routing from C&DH to payload
- Telemetry collection from payload to C&DH
- End-to-end validation of command/telemetry flow

Payload Integration (Secondary)

- Camera data transfer (high-bandwidth image topics)
- AI results and status reporting
- Resource telemetry collection

Phase 2: CAN Bus + CSP Integration

Physical Layer Setup

- Configure CAN bus interfaces on Raspberry Pi 4, Jetson, and any STM32 modules
- Hardware verification: CAN transceivers, termination resistors, signal quality
- CAN bus configuration (bitrate, error handling)

CSP Implementation

- Deploy libcsp (CubeSat Space Protocol library) on all modules
- Implement CSP over CAN driver
- Configure CSP addressing scheme and routing tables
- CSP services: ping, buffer management, connection handling

Communication Layer Selection

- **High-bandwidth data** (camera images, bulk telemetry): ROS2 DDS over Ethernet
- **Control messages** (mode changes, critical commands): CSP over CAN
- Define message type routing policy

Bridge Nodes (ROS2 ↔ CSP)

- Develop ROS2 nodes that bridge between ROS2 topics and CSP messages
- Example: Subscribe to /telecommand topic → send as CSP packet over CAN
- Example: Receive CSP packet from CAN → publish to /telemetry/from_can topic
- Bidirectional message translation
- Handle different message formats and serialization

Testing & Validation

- CAN bus stress testing and reliability analysis
- CSP protocol compliance testing
- Bridge node correctness and performance
- Hybrid network scenarios (simultaneous ROS2 and CSP traffic)

Integration & Testing

End-to-End Scenarios

- Telecommand via ROS2 → C&DH execution → response via CSP
- Camera image via ROS2/Ethernet + control via CSP/CAN
- Multi-path redundancy testing

Performance Analysis

- Latency comparison: ROS2 DDS vs. CSP/CAN
- Throughput testing for both networks
- Resource overhead (CPU, memory) of dual-stack approach
- Reliability metrics (packet loss, error rates)

Monitoring & Diagnostic Tools

- ROS2 introspection tools (rqt, ros2 topic/node/service commands)
- CAN bus analyzer integration
- Custom dashboard for network health visualization
- Logging and rosbag recording for both domains

Technical Requirements

- **Operating Systems:** Ubuntu 22.04 on Raspberry Pi 4 and Jetson
- **Middleware:** Space ROS (ROS2 Humble or Iron)
- **Protocols:** ROS2 DDS (over Ethernet), CubeSat Space Protocol (over CAN)
- **Programming Languages:** C/C++ for CSP and performance-critical components, Python for ROS2 nodes and tools
- **Hardware:** CAN transceivers, Ethernet switch, all processing modules
- **Libraries:** libcsp, ROS2 client libraries
- **Version Control:** All code contributed to project GitHub repository
- **Testing:** Protocol compliance tests, performance benchmarks, integration tests

Deliverables

1. Working Space ROS network over Gigabit Ethernet (all testbed modules)
2. CSP over CAN bus implementation (all testbed modules)
3. ROS2↔CSP bridge nodes with bidirectional communication
4. Full integration with C&DH subsystem (primary) and Payload (secondary)
5. Network monitoring and diagnostic tools (ROS2 and CAN)
6. Example applications demonstrating hybrid communication patterns
7. Network architecture documentation and integration guides
8. Source code with documentation in GitHub repository
9. Master's thesis report documenting design, implementation, validation and performance analysis for both communication layers.

Prerequisites

- Programming skills in C/C++ and Python
- Understanding of network protocols and communication systems
- Familiarity with embedded systems and real-time constraints
- ROS2 knowledge beneficial but not required
- Knowledge of CAN bus or industrial protocols beneficial but not required
- Experience with Linux networking beneficial but not required

Why This Thesis?

- **Dual Expertise:** Learn both modern (Space ROS) and traditional (CSP/CAN) satellite protocols
- **Network Architecture:** Design and implement complete communication infrastructure
- **Real Hardware:** Work with actual CAN bus, Ethernet, multiple processing modules
- **Protocol Development:** Create bridge nodes and integration patterns
- **ESA Alignment:** Space ROS is a research focus for European Space Agency
- **Open Source:** Contribute to open-source CubeSat community (GPL-3.0)
- **Industry Relevance:** Skills applicable to space industry, industrial automation, robotics

Management

- Weekly coordination meetings ensure project tracking and knowledge sharing.

- All work is documented and shared via GitHub, following open-source guidelines and contributing to the open-source community.

- Master thesis can overlap with other theses and/or ongoing developments in the working group. Collaboration is encouraged but all topics have been designed to not introduce blocking points.

Contact

Simon Vellas: svellas@mail.ntua.gr

Alexis Apostolakis: alexis.apostolakis@gmail.com

Giorgos Athanasiou: georgios.athanasiou.ntua@gmail.com

Expected Start: January 2026