CCSDS-Protocol-Testbed

Overview

This repository is dedicated to implementing and testing the CCSDS (Consultative Committee for Space Data Systems) protocol stack on an open-source platform. The primary goal is to create a functional testbed that allows the analysis of communication mechanisms, protocol stack behavior, and associated overheads, particularly in interplanetary communication scenarios.

Objectives

- Develop a detailed understanding of the CCSDS protocol stack and implement its layers using an open-source platform.
- Analyze communication mechanisms, protocol efficiency, and overheads in a simulated environment.
- Ensure the testbed aligns with CCSDS standards while allowing modifications to evaluate different scenarios.

Selected Open-Source Platform

Repository: OpenSatKit

- **Purpose:** Satellite command and control system with CCSDS protocol support.
- **Programming Language:** C, Lua.
- **Key Features:**
  - Detailed CCSDS protocol implementation.
  - Active community and consistent updates.
  - Comprehensive documentation.
  - Compatible with software-only environments.
- **Rationale for Selection:** OpenSatKit is robust, feature-rich, and has a strong focus on CCSDS compliance. It provides a good foundation for implementing and testing communication protocols.

Repository Structure

```
ccsds_protocol_testbed/
├── docs/                    # Documentation for CCSDS protocols and setup
├── src/                     # Source code for protocol implementation
│   ├── layer1/              # Physical layer implementation
│   ├── layer2/              # Data link layer
│   ├── layer3/              # Network layer
│   └── utils/               # Utility scripts and tools
```

```
├── test/                    # Test cases for protocol validation
├── tools/                   # Tools for simulation and data visualization
├── README.md                # Repository overview and setup instructions
└── LICENSE                  # License details
```

Setup Instructions

1. **Clone the Repository:**

   ```
   git clone https://github.com/your-username/ccsds_protocol_testbed.git
   cd ccsds_protocol_testbed
   ```

2. **Install Dependencies:** Ensure the necessary compilers, libraries, and dependencies for C and Lua are installed.

   ```
   sudo apt-get update
   sudo apt-get install gcc make lua5.3
   ```

3. **Build the Testbed:** Run the build script to compile the source code.

   ```
   make all
   ```

4. **Run Simulations:** Use provided scripts to initiate test cases and observe protocol behavior.

   ```
   ./run_tests.sh
   ```

Roadmap

- ☒ Select an open-source platform.
- ☒ Define repository structure and objectives.
- ☐ Implement physical and data link layers.
- ☐ Test and validate CCSDS protocol behavior.
- ☐ Add support for higher layers and end-to-end simulation.
- ☐ Document findings and publish results.

References

- CCSDS Standards Documentation: https://public.ccsds.org
- OpenSatKit GitHub Repository: https://github.com/OpenSatKit/OpenSatKit

Weekly Update: CCSDS Protocol Project

This week's progress on the CCSDS Protocol Project includes the following developments:

1. Objective Review

The primary objective of the project is to understand and implement the CCSDS (Consultative Committee for Space Data Systems) protocol. The key focus areas are: - Studying the CCSDS protocol stack and analyzing communication overheads. - Identifying and evaluating open-source platforms that align with CCSDS implementation requirements. - Establishing a testbed to simulate and test CCSDS protocol functionality.

2. Platform Exploration

We identified and evaluated several open-source platforms for potential CCSDS protocol implementation. Each platform was assessed based on the following criteria: - GitHub activity and support. - Programming language compatibility with project requirements. - Ease of installation and setup. - Regular maintenance and updates. - Features supporting CCSDS communication standards.

Shortlisted Platforms:

1. Cosmos (COSMOS Open Source Mission Operations System)

- **GitHub Repository:** COSMOS
- **Description:** Provides a complete suite of tools for telemetry and command in space systems.
- **Features:**
  - Integrated tools for data parsing and visualization.
  - Extensive documentation and active community support.
- **Language:** Ruby.
- **Maintenance:** Actively maintained.
- **Hardware Requirements:** Software-only, no specialized hardware required.

2. OpenSatKit

- **GitHub Repository:** OpenSatKit
- **Description:** A development and operations environment for satellite systems.
- **Features:**
  - CCSDS protocol support for telemetry and commanding.
  - Compatibility with hardware and simulation tools.
- **Language:** C and Python.
- **Maintenance:** Actively maintained.
- **Hardware Requirements:** Software-based testbeds supported.

3. CLTU Simulator

- **GitHub Repository:** CLTU Simulator

- **Description:** Command Link Transmission Unit Simulator for CCSDS uplink protocols.
- **Features:**
  - High-fidelity simulation of CCSDS uplink protocols.
  - Lightweight, easy deployment.
- **Language:** Python.
- **Maintenance:** NASA-supported, actively maintained.
- **Hardware Requirements:** None.

4. GSWS (Generic Spacecraft Workbench Simulator)

- **GitHub Repository:** GSWS
- **Description:** A modular spacecraft simulation framework.
- **Features:**
  - Protocol simulation with CCSDS-compliant interfaces.
  - Modular architecture for flexible usage.
- **Language:** Java.
- **Maintenance:** Actively maintained by ESA.
- **Hardware Requirements:** None.

3. Research Materials

The following CCSDS documents were consulted to guide the platform evaluation and protocol analysis: - **CCSDS 131.0-B-3**: TM Synchronization and Channel Coding. Available here - **CCSDS 232.0-B-1**: Telecommand Part 1 - Communications Protocol. Available here - **CCSDS 734.1-B-1**: Proximity-1 Space Link Protocol - Data Link Layer. Available here

4. Next Steps

- Finalize the platform selection based on the project's technical requirements and platform features.
- Begin setting up the testbed using the chosen platform.
- Document findings and prepare initial protocol simulation experiments.

5. Challenges

- Ensuring platform compatibility with the specific requirements of CCSDS protocols.
- Identifying solutions for interplanetary communication challenges, especially for Mars-Earth scenarios.

6. Upcoming Tasks

- Deepen understanding of CCSDS protocols through further study of Blue Books.
- Perform hands-on testing with the shortlisted platforms to determine feasibility.
- Develop a preliminary demonstration to showcase early results.

Feedback Requested:

- Additional criteria for platform evaluation.
- Recommendations for alternative tools or repositories.
- Advice for addressing interplanetary communication challenges in CCSDS implementation.