eng

MDPI

Article

The apex MCC: Blueprint of an Open-Source, Secure, CCSDS-Compatible Ground Segment for Sounding Rockets, CubeSats, and Small Lander Missions

Nico Maas [1,*], Sebastian Feles [2] and Jean-Pierre de Vera [1]

[1] Microgravity User Support Center (MUSC), German Aerospace Center (DLR), 51147 Köln, Germany; jean-pierre.devera@dlr.de
[2] Institute of Aerospace Medicine, Aeromedical FabLab, German Aerospace Center (DLR), 51147 Köln, Germany
* Correspondence: nico.maas@dlr.de; Tel.: +49-2203-601-2355

Abstract

The operation of microgravity research missions, such as sounding rockets, CubeSats, and small landers, typically relies on proprietary mission control infrastructures, which limit reproducibility, portability, and interdisciplinary use. In this work, we present an open-source blueprint for a distributed ground-segment architecture designed to support telemetry, telecommand, and mission operations across institutional and geographic boundaries. The system integrates containerized services, broker bridging for publish–subscribe communication, CCSDS-compliant telemetry and telecommand handling, and secure virtual private networks with two-factor authentication. A modular mission control system based on Yamcs was extended with custom plug-ins for CRC verification, packet reassembly, and command sequencing. The platform was validated during the MAPHEUS-10 sounding rocket mission, where it enabled uninterrupted remote commanding between Sweden and Germany and achieved end-to-end command–response latencies of ~550 ms under flight conditions. To the best of our knowledge, this represents the first open-source ground-segment framework deployed in a space mission. By combining elements from computer science, aerospace engineering, and systems engineering, this work demonstrates how interdisciplinary integration enables resilient, reproducible, and portable mission operations. The blueprint offers a practical foundation for future interdisciplinary research missions, extending beyond sounding rockets to CubeSats, ISS experiments, and planetary landers. This study is part two of a three-part series describing the apex Mk.2/Mk.3 experiments, open-source ground segment, and service module simulator.

Keywords: commercial off-the-shelf; mission control systems; ground segment; open source; CCSDS; sounding rockets

Academic Editor: Antonio Gil Bravo

Received: 5 August 2025
Revised: 12 September 2025
Accepted: 15 September 2025
Published: 17 September 2025

Citation: Maas, N.; Feles, S.; de Vera, J.-P. The apex MCC: Blueprint of an Open-Source, Secure, CCSDS-Compatible Ground Segment for Sounding Rockets, CubeSats, and Small Lander Missions. *Eng* **2025**, *6*, 246. https://doi.org/10.3390/eng6090246

1. Introduction

"It's rocket science!"—the idiom to label something inherently difficult—might be a bit clichéd; however, it still bears a lot of wisdom about the inner workings of the aerospace industry. Space is hard, so many specialists are needed to land a probe on an asteroid, for example. These specialists, working in their narrow fields, design harnesses, on-board computers, power sources, and thermal solutions, all with the same goal: to create a small package that fulfills all the requirements to successfully deliver and execute an experiment at a certain location and send back its findings. To allow for such a puzzle to be completed

in the first place, everyone needs to follow the same rules; otherwise, the multitude of complex parts will not fit together. Such rules are also known as standards and are the backbone of every major space project; for example, the Consultative Committee for Space Data Systems (CCSDS) has standards for all kinds of data formats (e.g., space packets [1]).

One subset of these challenges includes the remote control and monitoring of an experiment or complete spacecraft in flight, which is usually achieved using a monitoring and control system [2] or a mission control system (MCS). In the two standard scenarios shown in Figure 1, we can see the difference between controlling the experiment in flight or on the ground, as differentiated by the ground segments and hardware present. Nevertheless, the same basic principles apply: the experiment generates some type of data (e.g., temperature values), which are transmitted to the operator in the form of telemetry (TM) data. In return, the human operator can control elements within the experiment (e.g., a heater) in the form of telecommand (TC) instructions. As bandwidth for data transfer from space to the ground and back is limited, using American Standard Code for Information Interchange (ASCII) encoding, for example, to show data or commands as normal text strings would be detrimental. Instead, a space-efficient binary representation is chosen as the preferred space packet standard [1]. This is where the necessity of an MCS becomes apparent—the MCS server uses a Mission Database (MDB) to translate the space packet representation into human-readable values in a process called calibration. Alongside this functionality, MCS servers also archive all TM data received in calibrated and raw (uncalibrated) form, generate TCs, keep a command history, generate alarms if defined limits are violated (e.g., excessively high temperature), and expose interfaces to allow external systems to be connected for data evaluation and science support. Still, the operators normally do not use the MCS server directly but interface with it through a tool called the MCS client. The MCS client allows for the generation of synoptic displays, which are customized interfaces that show the calibrated TM data in an organized fashion and allow personnel to react quickly using prepared TCs.

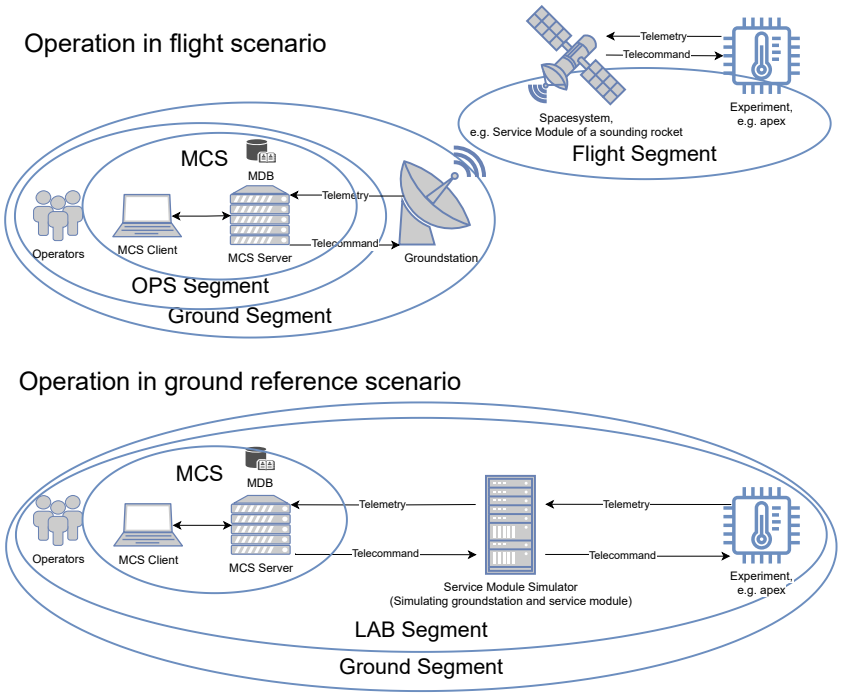

Figure 1. Basic ground segments for flight and ground scenarios.

Clearly, an MCS is an important backbone of space operations and, fortunately, is backed by standards.

And while these standards are built and utilized to perfection, the instrumentation of these in certain areas—e.g., ground segments, testing facilities, and development centers—is implemented in a myriad of ways by every single actor, especially when it comes to MCS. Be it the National Aeronautics and Space Administration (NASA) with the Telescience Resource Kit (TReK), the International Space Station (ISS) Columbus with the Columbus Distributed Mission Control System (CD-MCS), or the Microgravity User Support Center (MUSC) with SpaceMaster—all of which process CCSDS-compliant TM/TC data—all are vendor-locked-in, proprietary, protected, and closed-source solutions. In some cases, users of these solutions have no access to the underlying source code and are unable to verify the validity of the algorithms and scientific output except through black-box testing. Consequently, these users also lose the ability to improve their own products, and any new projects or changes in mission parameters warrant additional expensive contracts to make the needed modifications, in addition to the efforts required to keep the software working on modern or changing operating systems, to apply security patches to used libraries, and to manage the risk of a vendor declaring bankruptcy. The high cost of ownership, upkeep, and the complexity of such solutions also disqualify their use for small applications, e.g., sounding rockets, experimental rovers, or CubeSat missions. These drawbacks sometimes drive development engineers of smaller missions to ditch standards, robustness, and bandwidth efficiency altogether in exchange for a quick data-interface implementation with tools like National Instruments LabVIEW.

For these reasons and many more, the European Space Agency (ESA) encourages the use of open-source software whenever possible to circumvent vendor lock-in, empowering all member states with powerful tools and enabling faster integration of newer space standards [3].

Alongside this open-source policy, the ESA has started to support the implementation and rollout of the open-source MCS Yamcs [4], developed by Space Applications Services (SAS) for use in the ISS project [5,6]. NASA has also taken notice of the software and mentioned it in its 2025 "State-of-the-Art Small Spacecraft Technology" report [7] as having a Technology Readiness Level (TRL) of 9. NASA also employed it as the MCS for its Commercial Lunar Payload Services (CLPS) program, e.g., with Astrobotic's Peregrine moon lander [8] in 2024, and has planned its use with the NASA VIPER rover [9].

Some proprietary mission control systems (e.g., CD-MCS) are closed source and tied to institutional infrastructure. In contrast, our blueprint offers open-source licensing, modular deployment, and portable mission control components. This reduces vendor lock-in and lowers barriers for small missions.

Following the example of NASA and the ESA and emphasizing the importance of open-source software, the use of standards, and transparency, Yamcs was chosen as the main MCS for this study. While a direct comparison of Yamcs and the other MCSs mentioned above would be interesting, it was not included, as it would not only exceed the scope of this document but also prove difficult, as public documentation and citation of those tools pose their own legal challenges. Furthermore, the drawbacks of closed-source software have already been discussed, demonstrating that the use of these tools is unlikely to be future-proof.

This study describes the implementation of a secure, open-source mission control center and ground segment. It focuses on the ground segment for the second-generation advanced processors, encryption, and security experiment (apex) Mk.2/Mk.3 payloads, flown aboard the Material Physics Experiments Under Microgravity (MAPHEUS)-10 sounding

rocket [10] on 12 December 2021 from the Swedish Space Corporation (SSC) Esrange facility in Kiruna, Sweden.

The blueprint showcases the first open-source ground segment validated in a sounding rocket mission, including CCSDS-compliant TM and TC pipelines, containerized deployment of all subsystems, secure multi-site operation, and portable mission control on commodity hardware. This study will also discuss how this system can be retrofitted for use in other experiment types and mission profiles.

Remark: This study is part two of a three-part series describing the apex Mk.2/Mk.3 experiments [11], open-source ground segment, and service module simulator [12].

2. Materials and Methods

2.1. Overview

Prior to discussing implementation details, we provide a generalized overview of the concepts and objectives. The most important requirement for the new ground segment was to increase flexibility without compromising security. This is especially important due to the necessity of including remote commanding capabilities as a core component and not an afterthought. The system is designed with a wide range of flexibility, from easy integration of new capabilities to offloading all components onto a single laptop functioning as a mobile control center. The key reason for allowing such flexibility is to reduce the rigid communication structures between the different components in exchange for a broker-based message-queue approach. This system serves as a communication pipeline between all subsystems, allowing the MCS to receive data even from remote locations. A message queue can be thought of as a bundle of message channels. Each channel has its own name (topic) and can contain one piece of information (message) at a time. Systems can listen (subscribe) to one or multiple topics and receive any new message entered into a named channel by other systems called publishers. For example, the temperature sensor of a weather station could publish its value to the topic *"sensors/roof/temperature"*, and a remote-control unit could subscribe to this topic and receive the value for display. The last piece within the message-queue setup is the broker: the server provides the overall message-queue service to all the different clients as a connection endpoint, encrypts communications, authenticates users, and enforces that subscription or publishing is only allowed to the correct clients and with their associated topics. With this information in mind, a generalized setup would foresee TM data being fed into an incoming telemetry queue, which different services subscribe to; e.g., packet decoders would receive the latest TM data, process it, and publish it to another queue for additional preprocessing. Then, the last subscriber would be the MCS, which receives, calibrates, and displays the data. As topics can be generated on the fly, adding or removing new services can also be done during operations without downtime. It also allows changing routing/data flow by changing topics for subscription and publishing within the services. Each client also communicates with the broker via encrypted channels and can choose to use a username/password or a more secure public/private-key scheme for authentication.

While this improves information flow and security, it still lacks the vision of how to span the single system to a multi-site ground segment (breaking up the rigid communication structures). While it is possible to host the broker at the leading site and have all other clients/services connect to it, this also means that those remote services will not be able to communicate with each other should the connection to the broker drop. It would also necessitate new firewall rules for each new service added, which would slow down the implementation of new features while also increasing the attack surface. This is where bridges come into play: multiple brokers, each installed at their site, can choose to bridge parts of the topics to other brokers. As bridging works transparently, the overall

message-queue system can be set up for testing on one machine, including all TM ingests and TC receivers, preprocessing services, and MCS endpoints. This approach can then later move the first services to their own physical broker at the launch site, which then bridges the data flow to the home control center via a single broker-to-broker communication link—no matter the number of added services. All communications between both brokers are encrypted, and the systems can be configured to cache or temporarily store all incoming data if it cannot be forwarded to the bridge partner. With that, an automatic retransmission is carried out as soon as connectivity is restored, preventing any data loss. Also, if connectivity to the remote site is lost, e.g., due to an internet outage, locally running services connected to the local broker will still retain their functionality. In this instance, Mosquitto (Eclipse Foundation, Brussels, Belgium) is used, an open-source implementation of the Message-Queueing Telemetry Transport (MQTT) standard supported by the Eclipse Foundation. It was chosen due to its security, performance, minimal resource requirements, and included features, as benchmarked against other message queues in a prior master's thesis [13]. For the apex experiments in the MAPHEUS-10 campaign, brokers were installed at the launch site for both the flight (OPS) and the laboratory (LAB) setups. These brokers transmitted and received TM and TC messages from the ground antenna to the broker at the ground segment in the MUSC, where they were processed at the MCS. In addition to OPS and LAB TM and TC, control messages and data for the service module simulator (Test Support Equipment (TSE) stream) within the LAB, along with the countdown clock, were transferred using MQTT. The countdown clock implementation came in two varieties. Within the OPS environment, the existing SSC countdown clock was captured and sent to the MUSC to be included in the MCS as a time reference. For the LAB environment, a synthetic countdown clock was simulated according to the SSC standard to allow for more complex flight simulations and reference experiments executed by the service module simulator. For a better overview of the segments and systems used, see Figure 2.

In addition to the data transferred via MQTT, legacy Internet Protocol (IP)-based data communication also took place. This included, foremost, video data from a self-built weather camera as well as OPS and LAB videos from the apex Mk.2 microscope. Additionally, scientists, as well as operators, needed to be connected to the MCS at the MUSC to allow for remote commanding within OPS and LAB contexts, unless they were using the mentioned mobile control center.

Regarding the MQTT data services, most were written in Python 3 to strike a balance between performance, ease of development, and flexibility in terms of the hardware platform used. These services were then deployed in the form of Docker (Docker Inc., Palo Alto, CA, USA) containers and implemented with automatic service restoration in case of issues.

All communication channels (MQTT, video, MCS) were provided through the use of a two-factor-authentication (2FA)-secured virtual private network (VPN) service as an additional layer of security.

This VPN service, hosted at the MUSC at the German Aerospace Center (DLR) in Cologne, also served as the central communication endpoint.

All systems mentioned later in this study (the MAPHEUS service module simulator in the LAB segment and the apex-node, apex-video, and remote consoles in the OPS segment) at the SSC directly connect with their own VPN clients through the Ethernet-connected, provided hardware firewall and internet connection to the VPN endpoint at the MUSC at DLR. All other defined connections, e.g., MQTT broker communication, video, or direct MCS access, are transported via this virtual, secure network. This secure endpoint also allows remote OPS connections via the internet to connect to the mission control center (MCC) and establish secure remote commanding capabilities. This can be done either with

an OpenVPN-enabled (OpenVPN Inc., Pleasanton, CA, USA) and provisioned client or by using the mobile control center.

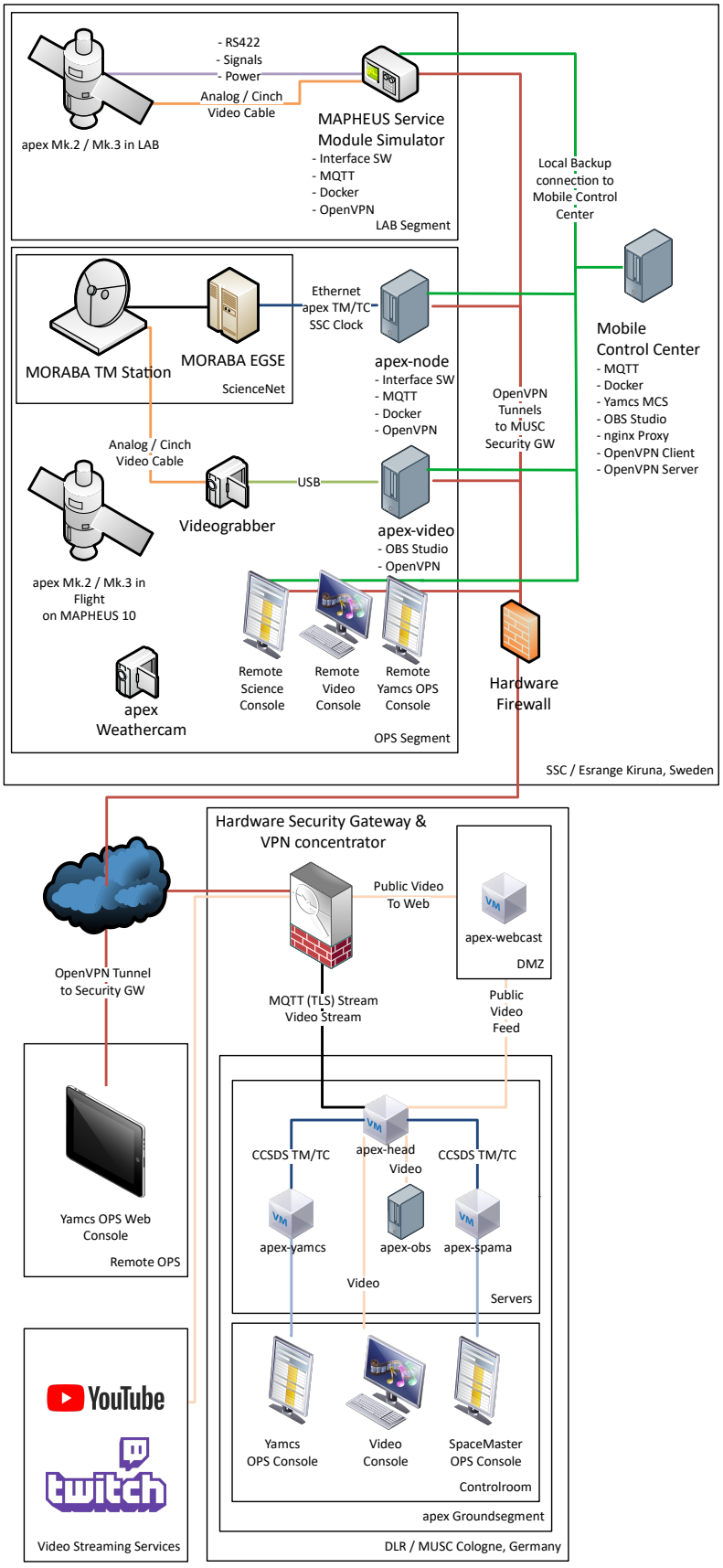

Figure 2. Ground-segment architecture of apex.

In case of a complete internet outage or issues at the MCC, the mobile control center can also replace the MCC with a local backup to allow all operations to take place at the SSC without the need for an internet connection. During these outages, the mobile control center can also provide a VPN server interface to allow external scientists to connect to the apex payloads, making the MCC redundant.

With the key communication guidelines outlined, we now examine the specific systems in the operations and laboratory contexts and, later on, the mission control center, where all systems are terminated, data is computed, and remote access capabilities are provided.

2.2. OPS Segment

Within the operations segment, data is received and transmitted to the payload in flight. This is done via multiple antennas as well as specialized ground equipment to control the rocket, its flight parameters, and the downlink and upload of parameters needed to support science operations. As such, a scientific data network (ScienceNet) exists to provide all the services needed by scientists to conduct experiments. ScienceNet provides an interface between rocket and payload operations in the form of two User Datagram Protocol (UDP) ports, exposing the TM and TC links to the individual experiments onboard the rocket. An additional stream is the network broadcast of the countdown clock, allowing ground stations to synchronize their operations to the mission time and view the current status and holds. In addition to these network links, an analog video link via a Bayonet Neill–Concelman (BNC) cable carries the analog Phase-Alternating Line (PAL) video format of the experiment downlink, if needed by the experiment. The control station can be seen in Figure 3, with more detailed communication services shown in Figure 4.

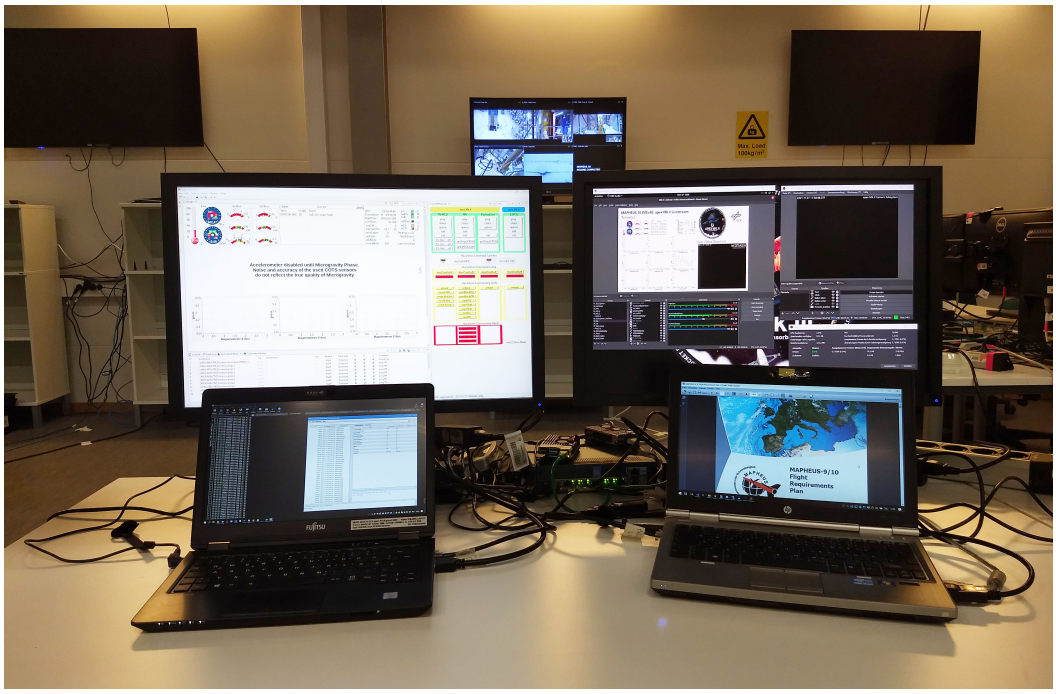

Figure 3. OPS segment at the SSC Esrange in Kiruna: (**left**) laptop and monitor used with the MCS; (**right**) laptop and monitor used for live video streaming, social media, and general notes. Between the monitors are the firewall, apex-node, and apex-video. The remote science console is not shown.

To allow remote control of these links, the data exposed here needs to be forwarded to the MCC. This is done via two Raspberry Pi (RPi) 4 (Raspberry Pi Foundation, Cambridge, UK) single-board computers: one has two network interfaces and is connected to ScienceNet and the MCC via a VPN and an additional firewall, and the other provides a video grabber

that converts the analog video signal into digital data and streams it to the MCC via a VPN and an additional firewall.

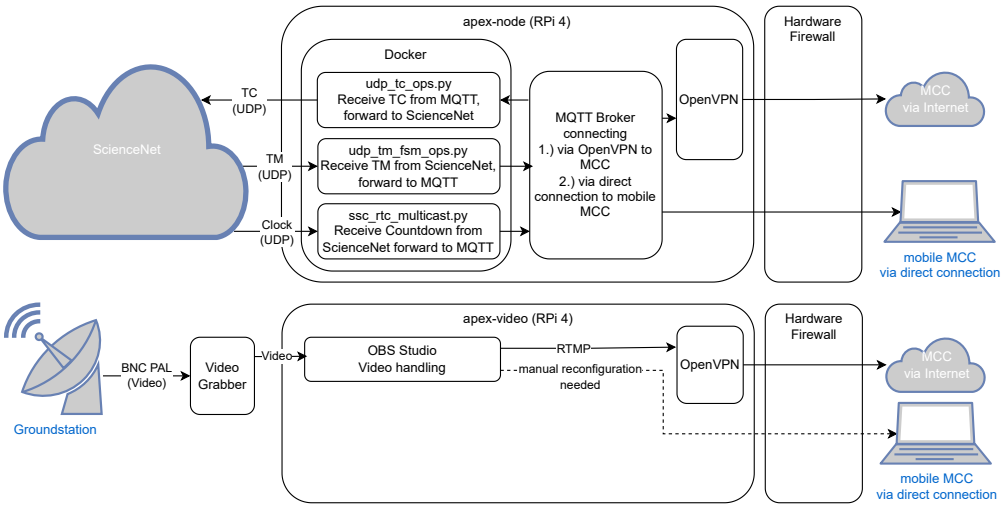

Figure 4. Detailed overview of selected data communications in the OPS segment. The mobile MCC is detailed in Section 2.4.

Looking at the system for data handling, referred to as the apex-node, this system runs Debian and a local Mosquitto broker that connects to the main broker within the MCC. There are multiple services running on this node, all programmed in Python and run as Docker containers to allow for quick restarts in case of issues and provide greater flexibility.

The first set of services considers the TM/TC handling. Incoming raw TM packets are received by the CCSDS space packet Finite State Machine (FSM). Because all payload TM data is forwarded by the MAPHEUS service module to the ground via a protocol-agnostic serial multiplexer, it is common for TM packets to be fragmented and arrive as data chunks on the ground. The FSM receives these chunks, reassembles them into correct CCSDS space packets, and validates their integrity via a Cyclic Redundancy Check (CRC). Then, the reconstructed packets are transmitted to the local Mosquitto broker, which is bridged via a firewall and a VPN to the main broker at the MCC.

In parallel to the TM handling, there is a second service that receives TCs from the MCC via the reverse connection, transforms them into UDP packets, and transmits them to the correct receiver in ScienceNet for uploading to the rocket in flight.

The last data service is the countdown clock receiver, which, in this case, receives the countdown clock provided by the SSC, parses its data, and transmits it in the same way as the TM data to the MCC for use in the MCS.

As the bridging between the brokers works transparently, it is also possible to connect a local copy of the MCC to the apex-node to maintain operations during lights-out scenarios where connectivity to the real MCC at the MUSC is lost. This approach enhances operational resiliency.

The analog video is converted via the second RPi and is referred to as apex-video. This system runs Debian and uses *Open Broadcaster Software (OBS) Studio* to convert the analog video to a digital video stream that is forwarded via a firewall and a VPN to the MCC as a directed data stream using the Real-Time Messaging Protocol (RTMP).

2.3. LAB Segment

The laboratory segment (see Figure 5, with more detailed communication services shown in Figure 6) allows the payload to work without the service module and can be used for checkout, testing procedures, and ground experiments. This is done via the MAPHEUS

service module simulator (MSMS), which simulates power, signals, and TM/TC capabilities for experiments. The MSMS consists of a hardware simulator, an RPi 4, and a video grabber. The RPi 4 runs Debian, uses Docker for its services, and connects via its own Mosquitto broker, firewall, and VPN to the MCC. All data transferred is designated to its own MQTT topics to allow the MCS to distinguish data from flight and laboratory segments. This applies to both the payload TM/TC and the MSMS control messages (TSE stream), which allow remote control of the simulator through the MCS, as well as the reading of system telemetry, such as payload power consumption or potential errors.

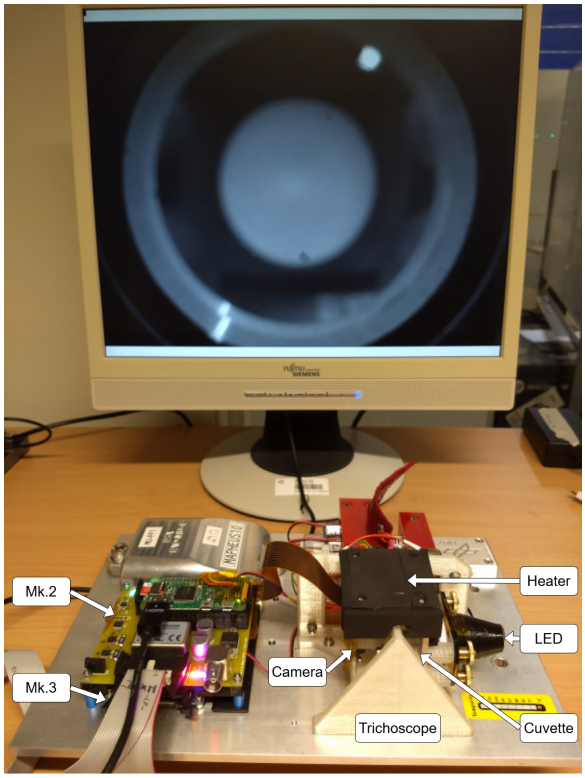

Figure 5. LAB segment at the SSC Esrange in Kiruna: The payload and monitor are in the center. The MAPHEUS service module simulator and remote Yamcs console are located to the left but are not shown.

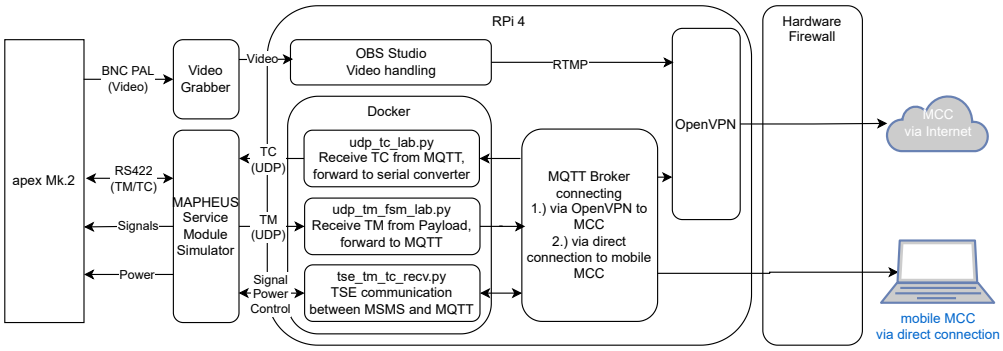

Figure 6. Detailed overview of selected data communications in the LAB segment.

The system also includes a countdown clock simulator and an additional network interface that mimics ScienceNet for a specialized checkout of the apex-node and MCC. Within this checkout mode, the MSMS can simulate the complete signal chain between the payload and endpoint in the ScienceNet/apex-node and becomes indistinguishable from the real setup. To create a flight-like simulation, TM/TC data can also be synthetically

malformed or fragmented in the same way the service module does during flight, along with the use of a video grabber to simulate the video downlink.

Since the MSMS is described in a dedicated study [12], it is not described here. Please note that the MSMS employed here was part of the second-generation apex family. The MSMS has since been improved as part of its third-generation upgrades and is now known as the multiple service module simulator, emphasizing its ability to simulate not only MAPHEUS but also the Rocket Experiments for University Students (REXUS), Astrobotic Peregrine, and SSC Orbital Express service modules. The MSMSv3 has also been integrated into the Multi-Role Mission Support System (MRMSS), which allows it to function as a mobile control center.

2.4. Mission Control Center

The mission control center at DLR in Cologne, MUSC, hosts the different systems needed to calibrate, visualize, and control payload data (see Figure 7, with more detailed communication services shown in Figure 8). Although hosted in this center, it can also be offloaded onto a single laptop for independent use in the field. Scientists can also use the MCC via a VPN to control experiments remotely if local attendance is not possible or if additional experts need to be given command capabilities in contingency operations.

Connectivity to internal services is provided by a firewall system via a 2FA-secured VPN. After this connection has been successfully established, different virtual machines (VMs) become available.

The apex-head system is the main server VM of the network, running Debian, Docker, the main Mosquitto broker, and an RTMP proxy-enabled nginx instance. All systems from the OPS and LAB segments deliver their data to the apex-head either via the broker or, in the case of video, via an RTMP stream to the nginx instance.

Depending on the type of data delivered to the broker, different services are used to forward it to their target systems. TM/TC data is subjected to a TM and TC flight recorder for central archiving. The data is then forwarded (TM) or received (TC) to/from MCSs via UDP network sockets. Countdown clock data is calibrated and directly injected into the Yamcs MCS via its Python API in the same way that the TSE stream to and from the MSMS is handled.

Video data arriving at the nginx instance can be individually recorded for archiving. The nginx instance also routes video to different systems and other VMs, e.g., apex-obs.

Regarding the MCSs used, apex employed two different solutions. As part of the open-source ground segment, Yamcs by SAS was used, which was extended for testing purposes by the closed-source solution SpaceMaster by SEA Datentechnik Troisdorf. Both systems were implemented independently of each other, shared the same MDB, received the same TM data, and could command via the apex-head.

The SpaceMaster server in this scenario was installed on the apex-spama VM, running Debian. As already described, it received data and sent commands through its own network sockets on the apex-head system. The client components were located in the MUSC control room in the form of a Windows computer running the SpaceMaster client. As SpaceMaster is closed-source software, it is not part of the described open-source ground segment; however, it is mentioned here because it was used during operations to validate SpaceMaster TC components and capabilities for small-scale missions.

The primary MCS, Yamcs, was installed on the Debian-backed apex-yamcs VM. TM/TC connectivity was accomplished via the apex-head using the same method as with SpaceMaster. Specialized subsystem data, like the countdown clock or the MSMS, were introduced via the apex-head and then ingested using the Yamcs Python API to allow for seamless integration. Yamcs supports the use of different server instances, allowing the

separation of data from different environments, such as OPS and LAB data. Due to the direct integration of the countdown clock and automated commanding within Yamcs in reaction to observed parameters, it enabled fully automated tests and flight simulations within the LAB environment. Real-time command and control during flight were achieved using the Yamcs Studio desktop client installed on the laptop within the SSC Kiruna control room/OPS segment, which was connected via a firewall and a VPN.

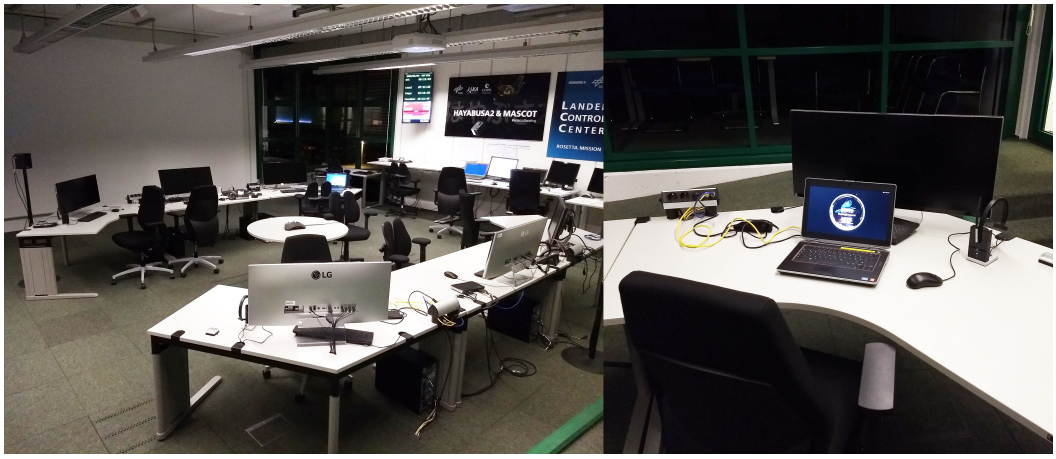

Figure 7. MUSC mission control center at DLR in Cologne. The local laptop serves as a redundant system to allow direct control of the payload from the MCC; apex is located at the center front console.

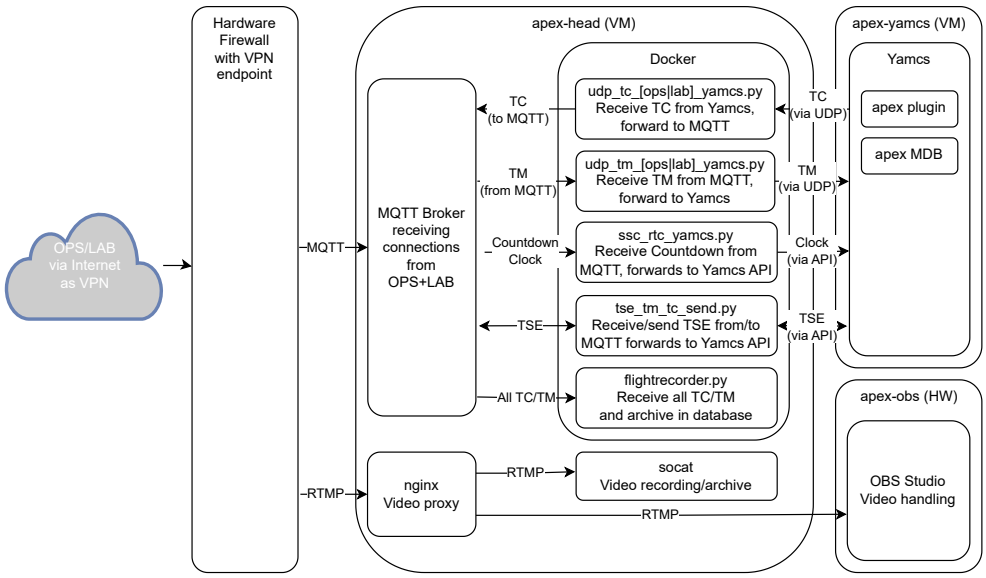

Figure 8. Detailed overview of selected data communications in the MCC.

As Yamcs itself is protocol-agnostic, a small plugin was written to allow the server to correctly identify timestamps and compute the CRC checksum of the employed CCSDS space packets [1] for the TM data, as well as include TC sequence counters and compute CRC checksums for the outgoing TC space packets to Mk.2/Mk.3.

The implementation of the apex MDB was done in XML Telemetric and Command Exchange (XTCE) format, since it is another recommended CCSDS standard [14] and is natively supported by Yamcs.

An additional method for visualizing TM data was implemented by the apex-yamcs-grafana, a Debian VM that provided the visualization framework Grafana. This framework was directly connected to apex-yamcs to allow graphing of all required data during OPS and

LAB checkouts using the SAS-provided Grafana (Grafana Labs, New York, NY, USA) Yamcs plugin. These visualizations also proved extremely useful for the Go/NoGo decisions during the challenging MAPHEUS-10 campaign, as discussed in the first study [11].

With the routing of TM/TC data covered, we now examine video data distribution. In addition to the live video provided from the apex-video system in the OPS environment, there was also the apex-yamcs-display VM. This Ubuntu VM ran the Yamcs Studio desktop client and ffmpeg to produce a video stream of the visualized payload TM and synoptic displays. The video data was sent to the apex-head to be received and retransmitted by another system, the apex-obs.

The apex-obs, also operating on Ubuntu, was a physical machine with a dedicated graphics card located in the MUSC and running OBS Studio (OBS Project, open-source software) . With this software, it was possible to generate a complex video-mixer studio setup that enabled processing multiple sources, including pregenerated presentation slides, real-time information, short-text updates, different video channels (e.g., live video and Yamcs Studio displays), and music into a single video stream, which was then forwarded via the apex-video-web proxy service VM located in a Demilitarized Zone (DMZ) to the streaming services Twitch and YouTube for real-time viewing around the world.

OBS Studio was controlled by the laptop at Kiruna via a VPN and remote desktop. A local setup was available as part of the mobile control center, tested and ready for use in contingency operations.

2.5. Flight Operations

As already mentioned, flight operations took place locally at the SSC Esrange in Kiruna, with all data being received in ScienceNet, sent to the MCC for processing, and finally controlled via three VPN-connected laptops at the local OPS segment in Esrange.

The main data handling and TM/TC operations were achieved via the remote Yamcs OPS console. This system had access to the synoptic displays via Yamcs Studio, data export and archive capabilities via Yamcs Web, real-time packet analysis at the hex level via Yamcs Packet Viewer, and additional visualization via Grafana and its Yamcs plugin. Also, debug data from the apex-node showed the real-time operations of the CCSDS space packet FSM and data ingestion.

The second console was the remote science console. The apex Mk.2 Science Camera Platform (SCP) served as an instrumented high-definition science camera to capture the first microgravity flight and behavior of *Trichoplax adhaerens*—the simplest organized metazoan [11]. The behavioral patterns were studied through observation, and the scientists involved agreed to support the public relations campaign on Twitch and YouTube by supplying still frames generated from the analog video downlink sent by Mk.2 during flight. The remote science console accessed the Mk.2 live stream, which was captured by apex-video and sent to the apex-head via a VPN using a Video LAN Client (VLC). This allowed the scientist to monitor the organism in real time. If an opportune point in the stream was found, the scientist could save this point as a still frame, which was then automatically uploaded by a local Python script to the apex-obs and inserted into the running Twitch and YouTube streams. If, for any reason, the test subject showed abnormal behavior, this display could be blocked and publication stopped.

The third and last console was the remote video console, which allowed for control of the apex-obs computer running at the MCC and also controlled the live stream to Twitch and YouTube.

The remote Yamcs OPS console could also be used in the LAB segment during ground tests, launch preparation, and after successful recovery of the payload to extract all captured internal high-quality camera data and environmental data from both Mk.2 and Mk.3.

2.6. Security Assessment

While the presented blueprint allows for additional capabilities not available in traditional ground segments, e.g., remote access to TM/TC capabilities and multi-site communication, it is important to highlight the impact on security and assess the steps taken to allow for secure use of the newly developed capabilities.

Starting with the baseline of using a non-MCS system or a directly attached LabVIEW instance in ScienceNet to control an experiment, the user receives and sends data according to the experiment's needs via UDP to and from a specific IP and port, which represents the experiment's gateway. Access to this network is highly controlled, allowing only local, well-known users to connect in order to avoid posing a risk to the mission. The most important security requirements are as follows: (1) No unauthorized user is allowed access to this network, and the newly added functionalities are not allowed to impact security. (2) Experimental data (TM and video) are treated as confidential and must not be leaked to unauthorized third parties. (3) TCs are only allowed to be sent from authorized sources to their own experiment.

To reach these goals, multiple precautions are in place. Starting within ScienceNet, the only component directly connected to this secure network is the apex-node, which is responsible for receiving TM and countdown clock data and forwarding TC to the gateway for transmission to the payload. The data entering this system is directly unpacked from its UDP structure by a Python script and transferred to an internal Mosquitto broker. While the connection between the script and broker is internal and not exposed to a network, it still uses Access Control Lists (ACLs) and username/password authentication on an individual script basis to restrict the script to reading/writing the required topics. The apex-node is then attached via a second Ethernet interface to a firewall that only allows the apex-node to reach the VPN gateway in the MUSC at DLR and, at the same time, blocks any incoming traffic not associated with this VPN connection. This is an additional layer of security, as the SSC firewall already implements Network Address Translation (NAT) and does not allow a system from the internet to directly attack the internal systems, e.g., the firewall used by the apex-node. With this communication path established, the apex-node can connect to the MUSC VPN server via its own certificate pair and username/password, which then allows the internal Mosquitto broker to connect to the apex-head server in the MUSC using its own certificate. With this, we have the following security profile: (1) Data is securely ingested into the apex-node, which forwards it internally to its own local Mosquitto broker. (2) The apex-node connects via a 2FA-secured VPN to the MUSC. (3) The Mosquitto broker opens another secure tunnel within the secure VPN tunnel, which does not terminate at the MUSC VPN or firewall appliance but within the isolated apex network at the MUSC, which means data is encrypted twice over the internet and once from endpoint to endpoint.

Within the isolated network, data (TM and countdown clock data) is decrypted on the apex-head and sent to the Yamcs MCS as if it were directly connected within ScienceNet using UDP.

TCs are sent the same way back from Yamcs via UDP to the appropriate ingestion script on apex-head, sent securely via this script to the internal apex-head Mosquitto broker, and then via the existing, broker-to-broker bridge and VPN connection back to the apex-node in Kiruna, describing the complete round-trip of data from ScienceNet to the isolated apex network at the MUSC.

As video data are captured via an analog video capture card, this interface is less sensitive and is decoupled from ScienceNet. Nevertheless, to allow for confidential video transfer, the apex-video system is also connected to a protective firewall that only allows the RPi to connect to the MUSC VPN gateway. Once this connection is established, the video stream is transferred using unencrypted RTMP to the appropriate video proxy on

the apex-head VM for consumption and use on the apex-obs or apex video consoles, or forwarded publicly using the apex-webcast in a DMZ.

The same connectivity described for the apex-node is also used by the MAPHEUS service module simulator. As it is directly attached to the payload—and not ScienceNet—it does not require a second network interface and can directly interface with its local firewall to connect to the MUSC VPN server to provide its TM/TC interfaces via the apex-head to the Yamcs MCS, as well as its video via the apex-head to the apex-obs system.

The last three missing operations concern access to the internal video stream, a scientist to capture still images for the live stream, and the connection of an operator to the Yamcs MCS during operations.

The internal video stream can be accessed via a dedicated VPN profile and a user-specific account that can only access the video proxy on the apex-head system. As this proxy uses multiple ports for the different video streams available, the user can only access the video data intended for him/her. The same applies to PI use for capturing still images: their ruleset is only extended by allowing them to upload images to the apex-obs via Secure Shell (SSH), authenticated with a certificate on the apex-obs. The final access to Yamcs is given via another VPN profile, which only allows a connection to the Yamcs interface (using HTTPS), which is also protected by a username and password and sophisticated Role-Based Access Control (RBAC).

To summarize, access to ScienceNet is severely limited and controlled, and all data is either single- or double-encrypted using independent systems based on TLS 1.2 and appropriate key lengths (4096 bits). Direct access to the TM/TC interfaces in ScienceNet is denied by design for all systems except the Yamcs MCS. The only access to the user-facing facilities is given to the experiment's operator and scientist.

We acknowledge that an additional proxy outside of the secured apex network to shield the Yamcs MCS would be appropriate to increase security in a future implementation, as it would block direct malicious attacks against the underlying MCS by a malicious user; however, having access to an appropriate 2FA-secured VPN account with access rights to this MCS would have similar implications to having a malicious user on site at Kiruna. The only redeeming factor would be that a successfully attacked or compromised network at the MUSC could only be used to disrupt the services for apex or spam commands to the payload. Other payloads could not be attacked, as the connections allowed on the apex-node within ScienceNet are hardcoded.

We declare that a formal internal security assessment was conducted in cooperation with the Mobile Rocket Base (MORABA) and SSC to allow for installation and use of the described system before the MAPHEUS-10 mission. This assessment found the precautions taken to be appropriate and authorized the system's use.

3. Results

The showcased ground segment was successfully used during the MAPHEUS-10 campaign to support all operations for the apex Mk.2 SCP and apex Mk.3 Student Experiment Sensorboard (SES) experiments. Among these operations were the pre-tests during development, the bench test, the environmental test, and the flight campaign. As the bench test started at the beginning of the COVID-19 pandemic, the remote-control capabilities proved to be invaluable. The environmental test was completed without physical attendance, as all checkouts could be observed remotely due to the apex-node and apex-video used within this scenario. During the bench test, additional personnel remotely monitored the payload from the MUSC MCC to support the engineers at the test site in Munich. During both tests, all MAPHEUS-10 payloads were combined into their final configuration and connected to the service module. Due to this, the OPS segment tools were used on site to support the

flight-like evaluation. While the purpose of these tests was to validate the MAPHEUS-10 payload stack, they also served as an informal check of the apex MCC. During these early tests, the necessity for a CCSDS space packet FSM became apparent, which was introduced to increase the robustness of TM calibration in the MCS.

The validation of these changes was performed during the on-site preparation for the flight campaign at the SSC Esrange. The focus of this preparation was on confirming that all hardware arrived at the launch site in flight-worthy condition. Additionally, countdown rehearsals and comparison experiments with *Trichoplax adhaerens* were undertaken. The first two preparations were conducted with a flight-like configuration using the OPS segment while the hardware was still on the test benches or already in the launch tower on top of the sounding rocket. The last one was achieved using the LAB segment, since the mounting plate holding the apex experiments could be removed from the payload container.

Comparison experiments within the LAB instance concentrated on gathering scientific data on the organism in question using the apex Mk.2 SCP microscope (Trichoscope) under resting/Earth gravity conditions of 1 g (see Figure 5). This data was later compared with the Trichoscope recordings during flight to analyze the changes in behavior of the organism under microgravity. The LAB experiments followed a specific procedure, which started with the biologists of our interdisciplinary workgroup cultivating specimens in a laboratory, selecting them for diversity and use in the experiments, and inserting them into a stainless steel cuvette. After this, the cuvette was installed in the Trichoscope, and the apex Mk.2 was connected to the MSMS. The apex engineering team then powered up the experiment via the MCS using the remote Yamcs console in the LAB instance and started recording Trichoscope videos via TC. During the experiment, video recordings of the organism, as well as environmental and performance telemetry from the apex Mk.2, were monitored by biological and engineering personnel. After the conclusion of the experiment, recording was stopped using a TC, and the systems were powered down using the MSMS. The LAB segment was also used for extracting recorded data, deleting old data from the internal memory, and adjusting camera focus, relying on the MSMS and apex MCC setup.

The flight-like tests consisted of either full-length test countdowns or shortened versions and focused on training personnel and checking systems to prepare for the hot countdown and subsequent flight. As mentioned, these were done in the OPS segment and included the preparation of the organisms in the LAB instance, insertion into the Trichoscope, installation of the mounting plate within the payload container, and subsequent payload checkouts. During this time, the experiments were powered up by the MAPHEUS service module, and the subsequent boot-up TM was supervised by apex engineering using the remote Yamcs console in the OPS instance. The countdown was then rehearsed as planned, checking that the payload reacted correctly to signals given by the service module using the received TM data. Telecommands were also sent from both the OPS segment and the MUSC MCC to validate correct functioning. Additionally, the countdown clock and video-streaming capabilities were tested, and personnel were trained on their use to ensure a smooth hot countdown. This also included testing contingency operations in scenarios such as the payload needing to record for longer durations, there were issues with the service module's signaling, or other unexpected problems. During this time, the distributed ground segment and its systems performed according to plan, especially in terms of the hot countdown.

The flight campaign was the strongest proof of the newly acquired capabilities, culminating in a successful flight, a public relations event in the form of the live video streams [15–17], and an accompanying Twitter campaign.

While the first apex [18] was flown without live TM/TC during the mild MAPHEUS-8 summer campaign, the flight carrying living *Trichoplax* during the unfavorable weather con-

ditions of MAPHEUS-10 (down to −38 °C in the launch area) and the necessary Go/NoGo decisions would not have been possible without live TM. Due to these capabilities, temperature issues could already be identified during the initial test countdowns, and mitigations could be planned in a timely manner. This resulted in a successful exposure of well-tempered organisms to microgravity conditions and the recording of their behavior in that environment.

During the flight campaign, no service or connection outages were registered, and the mission was flown as described via the MCC at the MUSC. As part of initial checkout, the mobile MCC was verified to demonstrate that the required flexibility was successfully achieved and that the engineered solution could be used without any external dependencies in a local setup.

Additional experiments included the measurement of round-trip times for the complete signal chain. This was conducted during a static test countdown (TCD) on the launch pad (2021-12-01 16:11:31 UTC–2021-12-01 16:16:55 UTC) and during the suborbital flight (HCD, 2021-12-06 08:08:41 UTC–2021-12-06 08:13:33 UTC). For both flights, three phases were defined: pre-apogee (preA/~150 km height), apogee (A/259 km height), and post-apogee (postA/~150 km height). Within each of these phases, a sequence of 10 measurements was performed, recording the time from remotely sending a TC until the corresponding TM was received, in milliseconds. For each phase within a flight, each flight, and both flights combined, the mean values and standard deviations were calculated, as shown in Figure 9.

Comparing the mean values of both flights showed only slight deviations, while the individual ones within the flights were quite high. This needs to be explained in the context of the overall data system and its bottlenecks. While the operation of the TM/TC microcontroller unit on apex Mk.2 could explain the variances in the round-trip time, environmental data acquisition was stopped during the benchmarks so as not to interfere with the measurements. The apex MCC systems were also sized accordingly so as not to be overwhelmed by the apex Mk.2 science TM (4 Hz, 82 bytes per packet). There were, however, two bottlenecks that could have contributed to these results, considering that each variation added up. First, the internet is an unreliable medium, especially in terms of consistent latency. The measurements shown include two data transfers between Sweden and Cologne: (1) Cologne sends the TC for uplink to Sweden. (2) Sweden sends the downlinked TM to Cologne. The latencies observed can fluctuate, as the apex MCC uses the default internet connection at the SSC Esrange. With that, sudden downloads by other guests or streaming of high-resolution video may interfere with the local connection, as with Cologne at the MUSC at DLR. Additionally, sudden spikes in bandwidth demand along the complete path may also interfere with the measured standard deviation. The second bottleneck is the available bandwidth between the rocket and ground, as well as the data handling within the service module. Space-to-ground bandwidth is a scarce resource, and the service module and its data handling systems need to send and receive data according to available buffers for the most efficient use of this bandwidth. This means that the defined round-trip times cannot be guaranteed in all scenarios. This becomes apparent when thinking about the fact that the service module needs its own share of bandwidth for housekeeping reasons, as well as the other six experiments with different needs in addition to apex [10].

The combined values show a mean round-trip time of 557.42 ms and a standard deviation of 119.41. These values represent complete end-to-end testing times and could be improved by using a direct fiber connection between both centers or a virtual one with guaranteed response times (e.g., Multiprotocol Label Switching). Another option for future operation would be to initially use the mobile MCC as the primary system and allow

external parties to join the operations using an internal VPN server and firewall system. This would decrease the latency measured during flight for on-site operators and eliminate the varying latency induced by using an internet connection.

Finally, the results need to be compared with the requirements set by the experiments to determine whether performance is appropriate. While biological samples can endure high latencies, experiments in material sciences, e.g., molten metal, require stricter regulation, as in the case of the Electromagnetic Levitator (EML). The EML requires a "turnaround time of 5 s between telemetry and video reception [...] and a command reception by the EML" [19]. This means that only the TC needs to be transferred within 5 s, representing only half a round trip. Evaluating apex MCC performance at 557.42 ms shows that it is well within acceptable limits, even for material sciences experiments.

ID	Flight	Phase	Seq	ms	ID	Flight	Phase	Seq	ms	ID	Flight	Phase	Seq	ms
1	TCD	preA	1	495	11	TCD	A	1	520	21	TCD	postA	1	697
2	TCD	preA	2	429	12	TCD	A	2	769	22	TCD	postA	2	663
3	TCD	preA	3	496	13	TCD	A	3	539	23	TCD	postA	3	722
4	TCD	preA	4	614	14	TCD	A	4	657	24	TCD	postA	4	477
5	TCD	preA	5	688	15	TCD	A	5	510	25	TCD	postA	5	633
6	TCD	preA	6	570	16	TCD	A	6	540	26	TCD	postA	6	533
7	TCD	preA	7	486	17	TCD	A	7	523	27	TCD	postA	7	521
8	TCD	preA	8	645	18	TCD	A	8	773	28	TCD	postA	8	439
9	TCD	preA	9	535	19	TCD	A	9	410	29	TCD	postA	9	602
10	TCD	preA	10	554	20	TCD	A	10	446	30	TCD	postA	10	463
x̄ Phase in ms				**551.2**					**568.7**					**575**
σ Phase				**79.91**					**124.5**					**102.01**
x̄ Flight in ms														**564.97**
σ Flight														**100.63**

ID	Flight	Phase	Seq	ms	ID	Flight	Phase	Seq	ms	ID	Flight	Phase	Seq	ms
31	HCD	preA	1	521	41	HCD	A	1	379	51	HCD	postA	1	542
32	HCD	preA	2	630	42	HCD	A	2	434	52	HCD	postA	2	599
33	HCD	preA	3	520	43	HCD	A	3	581	53	HCD	postA	3	480
34	HCD	preA	4	1010	44	HCD	A	4	463	54	HCD	postA	4	289
35	HCD	preA	5	683	45	HCD	A	5	524	55	HCD	postA	5	545
36	HCD	preA	6	547	46	HCD	A	6	460	56	HCD	postA	6	622
37	HCD	preA	7	485	47	HCD	A	7	564	57	HCD	postA	7	367
38	HCD	preA	8	523	48	HCD	A	8	815	58	HCD	postA	8	455
39	HCD	preA	9	471	49	HCD	A	9	723	59	HCD	postA	9	530
40	HCD	preA	10	594	50	HCD	A	10	659	60	HCD	postA	10	481
x̄ Phase in ms				**598.4**					**560.2**					**491**
σ Phase				**158.9**					**138**					**101.96**
x̄ Flight in ms														**549.87**
σ Flight														**137.86**
x̄ combined in ms														**557.42**
σ combined														**119.91**

Figure 9. End-to-end latency measurements from the remote Yamcs console to apex Mk.2 and back during a static test countdown (TCD) and a suborbital flight (HCD), showing the mean value and standard deviation of both.

While the setup worked very well, an internet outage at short notice would have caused issues, as reconnecting the remote consoles to the mobile MCC would have taken several seconds for the MCS and longer for the video streams. This would not have impacted the science conducted by apex but would have disrupted the public relations event and the scientists observing their specimens. The reason for these different impacts is that, due to the Mosquitto broker setup, both MCSs (at Cologne and local) could be active at the same time, and the only change would be the reconnection of the console from the remote MCS to the local one. As the video was transferred as direct RTMP transmission, a restart of the processes on the apex-video with a different target to the mobile MCC and the start of streaming from Sweden would have been necessary. A better solution,

as described above, would be to initially use the mobile MCC as the primary system and include external scientists to expand its capabilities.

Regarding system performance and scalability, apex Mk.2 transmitted TM data at 4 Hz, 82 bytes per packet, with 40 TM parameters per packet. The resulting system load was so negligible that even the Yamcs MCS would not have necessitated a virtualized server but could have run on an RPi. Regardless, this was not done in favor of the backup and high-availability capabilities of the MUSC VMware vSphere environment, which added operational security. Due to the flexibility of the Mosquitto broker, it would also be possible to connect multiple payloads at the same time to different Yamcs instances running on the same—or multiple—servers, allowing the showcased MCC to operate as a multi-client-capable system and managed service. For further missions, extended benchmarking should be considered to provide a more comprehensive overview of each system's performance.

To test adaptability to other payloads, a functional replica of the DLR M-42 radiation experiment, flown on Astrobotic's Peregrine moon lander [20] in 2024, was built and tested with the setup. As Astrobotic uses a different approach than MAPHEUS to connect TM/TC (RS-422 via Serial-Line Internet Protocol (SLIP) in UDP [21]), it was necessary to adapt the MAPHEUS service module simulator to accept this new interface. As the M-42 functional replica used CCSDS-space-packet-compliant TM/TC data, it was only necessary to change the Yamcs MDB to correctly calibrate the new TM and TC parameters.

Beyond the MAPHEUS-10 use case, the architecture can be generalized to other mission types. Elements that are mission-independent include the containerized Yamcs MCS, the broker-bridged communication backbone, the VPN security layer, and the portable MCC. Mission-specific adaptations are limited to the MDB reflecting payload parameters, the interfacing scripts at the link layer (e.g., UDP vs. RS-422/SLIP), and operational procedures. This separation makes the blueprint readily transferable to other sounding rockets, CubeSats, or small landers with only modest adaptation effort.

Discussing the limitations of the adaptability in more detail, as long as the interface remains the same and the system is CCSDS-space-packet-compliant, only the MDB at the Yamcs MCS needs to be changed. If, however, the interface changes, the incoming (TM) and outgoing (TC) scripts on the apex-node need to be adapted. This could be the case if, e.g., data is no longer transferred via UDP. Also, the interface takes into consideration the fragmented nature of TM data transmitted by the MAPHEUS service module by employing a CCSDS space packet FSM. This system would also need to be changed to reflect the changed Application Process Identifier (APID) of the packets. Still, these are changes that would be necessary regardless of whether Yamcs or another MCS were used without the proposed MCC.

Although the proposed setup has not been used with CubeSat hardware, we successfully tested these systems directly connected to Yamcs. The reason for not deploying the new MCC in this case was the limited use by a single developer in a directly connected scenario, which did not necessitate advanced networking capabilities.

4. Conclusions

The overall ground segment was tested successfully during the campaign and, since then, has been constantly evolving as part of the MRMSS—the support system for the MSMSv3 and new ground segments.

The use of open-source tools allowed the development of a secure, flexible, and maintainable MCC, which has since been expanded to additional experiments, gained new features, and been deployed to newer operating systems (OSs) without considerable effort.

Additionally, the remote video-streaming system developed was retrofitted in early 2020 to allow for the continuation of the EML experiments onboard the ISS without the

physical attendance of scientists at the MUSC. The system allows for remote viewing of video and TM data via a 2FA-secured VPN. Due to its ease of use and popularity among the science community, the system is still in use today as part of routine operations.

Another system that was retrofitted for the EML use was the TSE system, which transferred MSMS data during the apex mission and was redesigned to bind the High-Rate Data Processor (HRDP)/science video system into Yamcs to make the aforementioned tools directly controllable via the MCS.

While the presented blueprint was successfully validated during the MAPHEUS-10 campaign, several limitations must be acknowledged. First, the benchmarking focused exclusively on end-to-end latency measurements; extended evaluation of throughput, scalability under high load, and systematic fault-injection tests remain for future work. Second, no external penetration testing of the security architecture has been conducted, although an internal review by the MUSC, MORABA, and SSC has been completed. Third, interoperability with proprietary mission control systems could not be systematically assessed due to confidentiality constraints. Addressing these aspects in future campaigns will further strengthen the generality and robustness of the proposed open-source ground segment.

Author Contributions: Conceptualization, N.M. and S.F.; methodology, N.M.; software, N.M.; validation, N.M. and S.F.; formal analysis, N.M.; investigation, N.M.; resources, J.-P.d.V.; data curation, N.M.; writing—original draft preparation, N.M.; writing—review and editing, N.M., S.F. and J.-P.d.V.; visualization, N.M.; supervision, N.M.; project administration, N.M.; funding acquisition, J.-P.d.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: We want to thank the Institute for Frontier Materials on Earth and in Space at DLR Cologne for the vacuum testing and for allowing us to fly on MAPHEUS-10. We also want to express our gratitude to MORABA for flight certification of the hardware, campaign, and on-site support with the Swedish Space Corporation (SSC) at Esrange. We also want to thank the following MUSC colleagues: Oliver Küchemann, for providing the initial apex test MDB and Yamcs plugin; Daniel Embacher and Daniel May for the initial SpaceMaster server installation; Sven Jansen and Hans-Herbert Fischer for supporting the SpaceMaster configuration and operation; and Stephan Sous for supporting apex as the exploration control room responsible at MUSC. This work was accomplished under basic funding by the DLR Microgravity User Support Center (MUSC) and the DLR Institute of Aerospace Medicine, both in Cologne, Germany.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

2FA	Two-Factor Authentication
ACL	Access Control List
apex	advanced processors, encryption, and security experiment
APID	Application Process Identifier
ASCII	American Standard Code for Information Interchange
BNC	Bayonet Neill–Concelman
CCSDS	Consultative Committee for Space Data Systems

CD-MCS	Columbus Distributed Mission Control System
CLPS	Commercial Lunar Payload Services
CRC	Cyclic Redundancy Check
DLR	German Aerospace Center
DMZ	Demilitarized Zone
EML	Electromagnetic Levitator
ESA	European Space Agency
FSM	Finite State Machine
HRDP	High-Rate Data Processor
IP	Internet Protocol
ISS	International Space Station
MAPHEUS	Material Physics Experiments Under Microgravity
MCC	Mission Control Center
MCS	Mission Control System
MDB	Mission Database
MORABA	Mobile Rocket Base
MRMSS	Multi-Role Mission Support System
MSMS	Multiple Service Module Simulator
MUSC	Microgravity User Support Center
MQTT	Message-Queueing Telemetry Transport
NASA	National Aeronautics and Space Administration
NAT	Network Address Translation
OBS	Open Broadcaster Software
OS	Operating System
PAL	Phase Alternating Line
RBAC	Role-Based Access Control
REXUS	Rocket Experiments for University Students
RPi	Raspberry Pi
RTMP	Real-Time Messaging Protocol
SAS	Space Applications Services
SCP	Science Camera Platform
SES	Student Experiment Sensorboard
SLIP	Serial-Line Internet Protocol
SSC	Swedish Space Corporation
SSH	Secure Shell
TC	Telecommand
TReK	Telescience Resource Kit
TRL	Technology Readiness Level
TM	Telemetry
TSE	Test Support Equipment
UDP	User Datagram Protocol
VLC	Video LAN Client
VM	Virtual Machine
VPN	Virtual Private Network
XTCE	XML Telemetric and Command Exchange

References

1. CCSDS. CCSDS 133.0-B-2 Space Packet Protocol. 2020. Available online: https://ccsds.org/Pubs/133x0b2e2.pdf (accessed on 4 August 2025).
2. Sellmaier, F.; Uhlig, T.; Schmidhuber, M. *Spacecraft Operations*; Springer: Berlin/Heidelberg, Germany, 2022.
3. ESA. ESA Open Source Policy. European Space Software Repository (ESSR). 2025. Available online: https://essr.esa.int/esa-open-source-policy (accessed on 4 August 2025).
4. Sela, A. Yamcs—A Lightweight Open-Source Mission Control System. In Proceedings of the SpaceOps 2012 Conference, Stockholm, Sweden, 11–15 June 2012. [CrossRef]

5. Schmitt, M.; Diet, F.; Mihalache, N. Yamcs for lean Commercial Control Centres: The ICE Cubes Control Centre. In Proceedings of the 2018 SpaceOps Conference, Marseille, France, 28 May–1 June 2018; p. 2682.
6. Mariën, G.; Jacobs, C.; Klai, S.; Karl, A.; Van Hoof, D.; Pieters, L.; Michel, A. SOLAR, 9 years of operations as external payload on the ISS: The technical challenges overcome. In Proceedings of the 2018 SpaceOps Conference, Marseille, France, 28 May–1 June 2018; p. 2715. [CrossRef]
7. Weston, S.V.; Burkhard, C.D.; Stupl, J.M.; Ticknor, R.L.; Yost, B.D.; Austin, R.A.; Galchenko, P.; Newman, L.K.; Soto, L.S. State-of-The-Art Small Spacecraft Technology. 2025. Available online: https://ntrs.nasa.gov/citations/20250000142 (accessed on 4 August 2025).
8. SAS. How Yamcs Sets the Pulse of Mission Control. 2023. Available online: https://www.spaceapplications.com/news/yamcs-mission-control (accessed on 4 August 2025).
9. Mihalache, N.; Trimble, J. Open MCT and Yamcs. In Proceedings of the 8th European Mission Operations Data System Architecture Workshop (ESAW), Darmstadt, Germany, 2–3 November 2021. Available online: https://airdrive.eventsair.com/eventsairwesteuprod/production-atpi-public/5237a2af87db47898e1d5427e6a4baa2 (accessed on 4 August 2025).
10. Nuessle, D. DLR MAPHEUS 10 Sounding Rocket Campaign. DLR Weblog. 2021. Available online: https://www.dlr.de/en/latest/news/2021/04/20211206_journey-into-microgravity-for-seven-experiments (accessed on 4 August 2025).
11. Maas, N.; de Vera, J.-P.; Schmidt, M.J.; Reimann, P.; Randall, J.G.; Feles, S.; Hemmersbach, R.; Schierwater, B.; Hauslage, J. apex Mk.2/Mk.3: Secure live transmission of the first flight of *Trichoplax adhaerens* in space based on components off-the-shelf. *Eng* **2025**, *6*, 241. [CrossRef]
12. Maas, N.; Feles, S.; de Vera, J.P. apex MRMSS: A multi-role mission support system and service module simulator for payloads of sounding rockets and other space applications. *Eng* **2025**, *6*, 241.
13. Maas, N. Evaluating Message Queues for use in Distributed Filesystems. Master's Thesis, University Trier, Trier, Germany, 2017. Available online: https://nico-maas.de/Master_Thesis_Nico_Maas.pdf (accessed on 4 August 2025).
14. CCSDS. CCSDS 660.0-B-2 XML Telemetric and Command Exchange Version 1.2. 2020. Available online: https://ccsds.org/Pubs/660x0b2.pdf (accessed on 4 August 2025).
15. Maas, N. Flight apex Mk.II in MExA on MAPHEUS 10. YouTube. 2021. Available online: https://www.youtube.com/watch?v=H4EP9TFVHm4 (accessed on 4 August 2025).
16. Maas, N. MAPHEUS 10 Launch (Radarhill). YouTube. 2021. Available online: https://www.youtube.com/watch?v=9N74bLjJrR0 (accessed on 4 August 2025).
17. Maas, N. MAPHEUS 10 Launch (Dome/apex Groundsegment Weathercam). YouTube. 2021. Available online: https://www.youtube.com/watch?v=h_K23H79Ijo (accessed on 4 August 2025).
18. Maas, N.; Willnecker, R.; Hemmersbach, R.; Hauslage, J. apex: A new commercial off-the-shelf on-board computer platform for sounding rockets. *Rev. Sci. Instrum.* **2019**, *90*, 105101. [CrossRef]
19. Glaubitz, B.; Kullack, K.; Dreier, W.; Seidel, A.; Soellner, W.; Diefenbach, A.; Schneider, S. The Electro-Magnetic Levitator (EML) on Board the ISS-an Overview and Outlook. In Proceedings of the Deutscher Luft- und Raumfahrtkongress 2015, Rostock, Germany, 22–24 September 2015. Available online: https://www.dglr.de/publikationen/2015/370222.pdf (accessed on 4 August 2025).
20. Astrobotic. Astrobotic Peregrine Payload Manifest. 2024. Available online: https://www.astrobotic.com/lunar-delivery/manifest/ (accessed on 4 August 2025).
21. Astrobotic. Astrobotic Lunar Landers Payload User's Guide v.5.0. 2021. Available online: https://www.astrobotic.com/wp-content/uploads/2022/01/PUGLanders_011222.pdf (accessed on 4 August 2025).