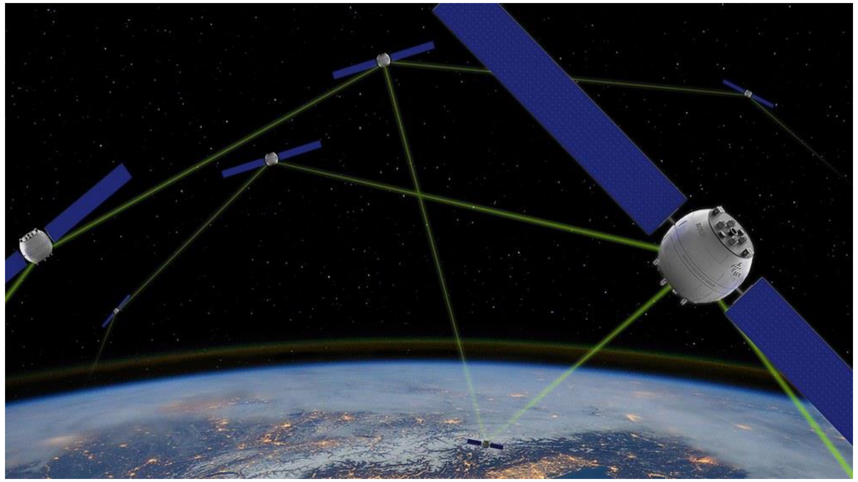
PRELIMINARY DESIGN OF THE STELLAR APPS SOFTWARE PLATFORM FOR DEVELOPING AND EXECUTING ON-BOARD APPLICATIONS

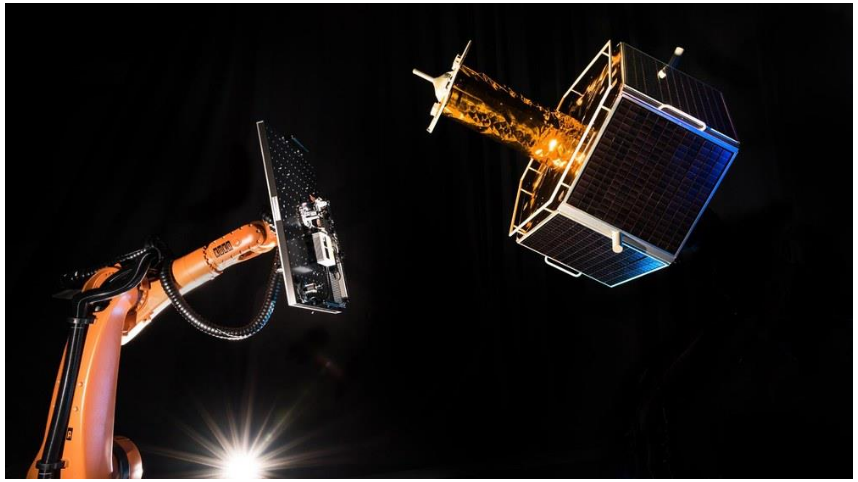
Hendrik Otte, Armin Purle-Kopacz, Kai Bleeke, Arnau Prat, Zain Haj Hammadeh, Andreas Lund, Jan-Gerd Meß, Michael Felderer, Daniel Lüdtke

German Aerospace Center (DLR)

Use Cases for Edge Computing and an Application Execution Platform

- AI-based autonomy and health management for constellations
- Sensor data preprocessing
- On-board scientific processing
- On-board digital twins
- Real-time processing of earth observation data and alarming (e.g. wild fires)
- Optical navigation for on-orbit servicing or rover navigation


Constellation Kepler - Credit: DLR (rendering), ESA/NASA (image of the Earth)

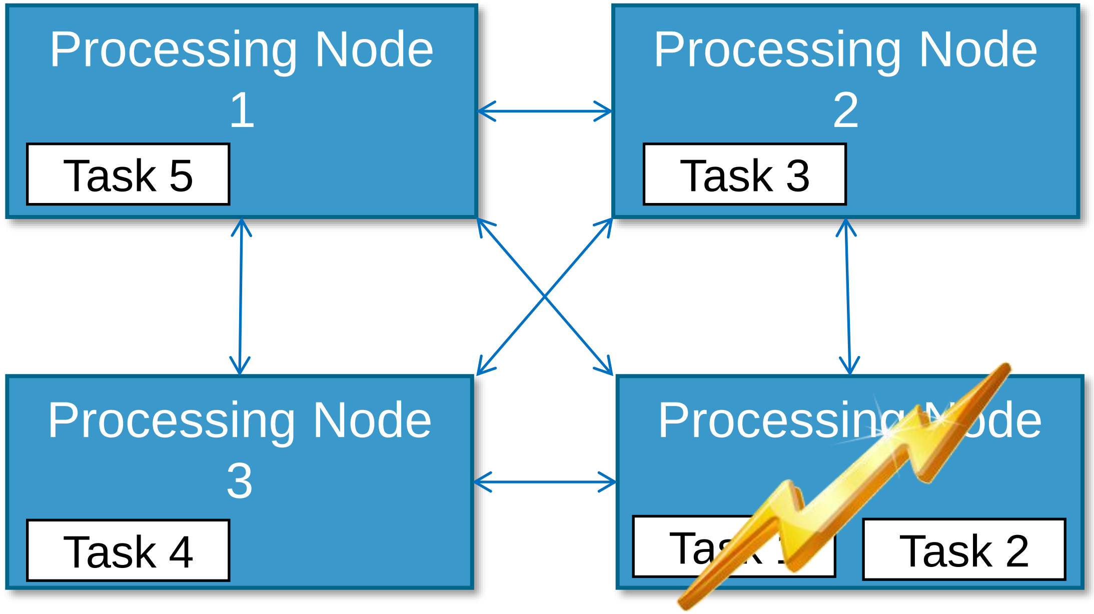

Robot and simulation setup at the European Proximity Operations Simulator

SCOSA

Scalable On-board computing for Space Avionics (ScOSA)

Combine space-qualified hardware with COTS (commercial-off-the-shelf) components

Reliable Computing Nodes (RCNs)	• Mission-critical tasks • Fallback
High-Performance Nodes (HPNs)	• High-performance tasks
In case of node-failures	• Reordering of the task mapping
Resilience	• Autonomous reintegration of reactivated nodes



*Lüdtke, Daniel und Firchau, Thomas und Gonzalez Cortes, Carlos Eduardo und Lund, Andreas und Nepal, Ayush Mani und Elbarrawy, Mahmoud Mostafa Hussein Hassan und Haj Hammadeh, Zain Alabedin und Meß, Jan-Gerd und Kenny, Patrick und Brömer, Fiona und Mirzaagha, Michael und Saleip, George und Kirstein, Hannah und Kirchhefer, Christoph und Gerndt, Andreas (2023) *ScOSA on the Way to Orbit: Reconfigurable High-Performance Computing for Spacecraft.* In: Proceedings - 2023 IEEE Space Computing Conference, SCC 2023. 2023 IEEE Space Computing Conference (SCC), 2023-07-18 - 2023-07-21, Pasadena, CA, USA. doi: 10.1109/SCC57168.2023.00015. ISBN 979-8-3503-4143-0.

ScOSA Features

Scalability

- Heterogeneous architecture (CPU, FPGA)
- Combination of COTS and radiation tolerant processors
- Reconfiguration for new mission phases

Data-driven programming model

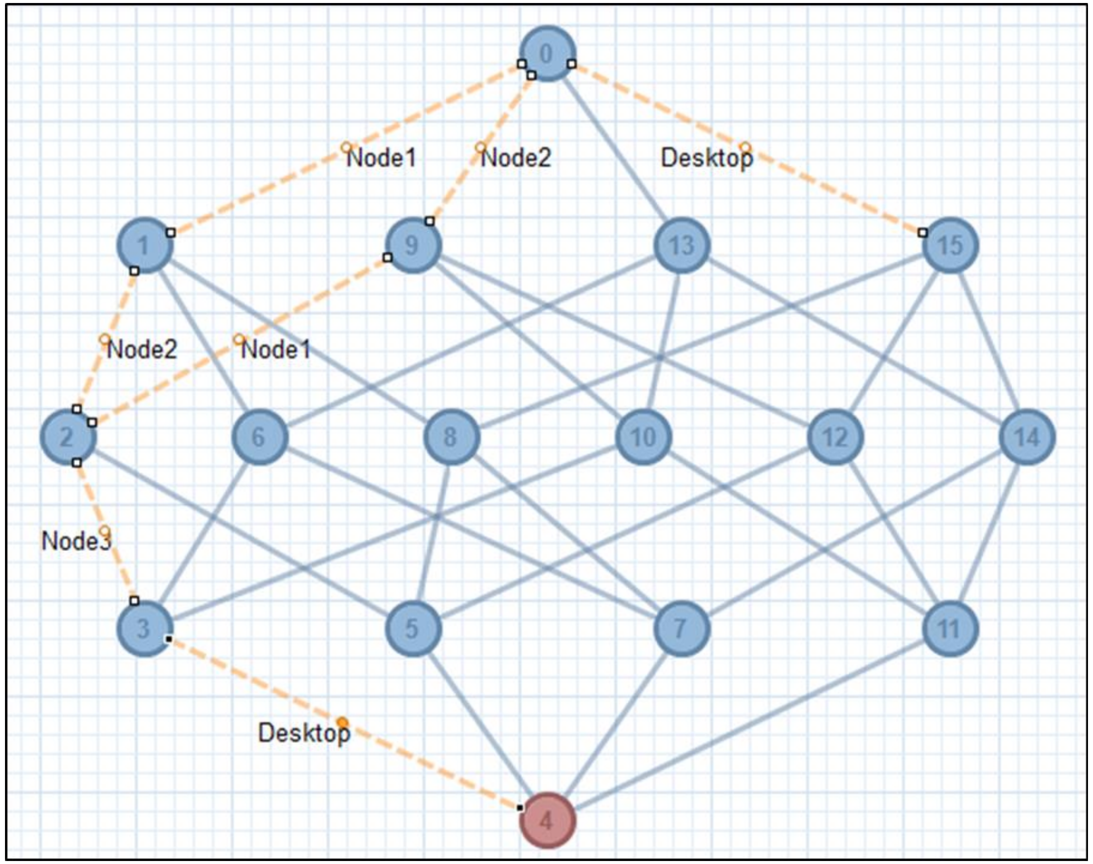
- Distributed middleware
- Event-based task activation
- Supports Linux and RTEMS

Network protocol (SpaceWire-IPC)

- Reliable messaging
- IPC between tasks on different nodes

Fault tolerance

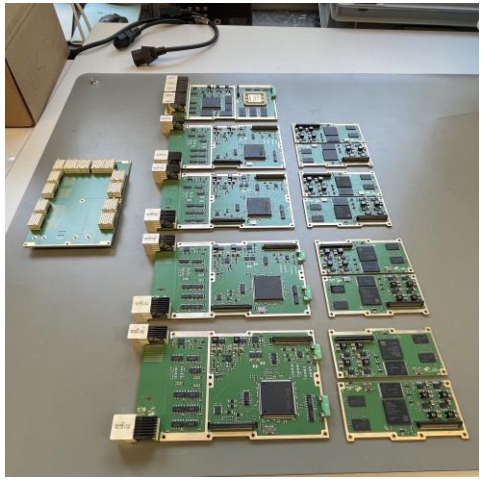
- Distributed FDIR subsystem at data, task, node, and system levels
- Fault isolation of unreliable components
- Checkpointing



ScOSA reconfiguration graph

ScOSA Flight Experiment on CAPTn-1

- Flight of a 9-node computer (8 Xilinx Zynq SoCs + 1 Leon3 GR 712 → 18 cores) on a 12U CubeSat in 2026
- Demonstrate system reliability and resilience in an operational space environment
- Demonstrate at least 5 relevant applications (e.g. rendezvous navigation, earth observation, …)
- Expected mission lifetime: 1 year + 1 additional year

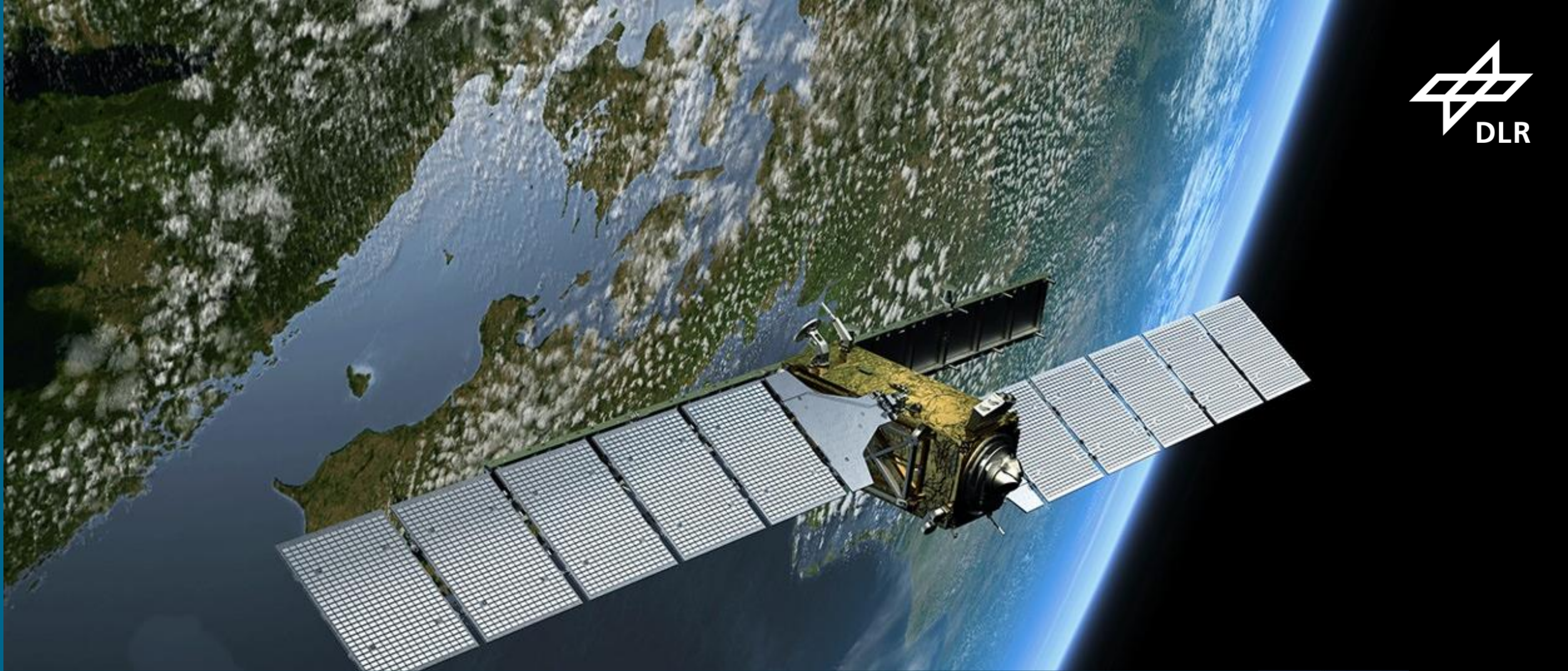


ScOSA boards

ScOSA payload integration

Lessons Learned from ScOSA

- Application developers want a more flexible development environment with custom libraries/runtimes
- Software integrators want strong separation and rights management for applications
- Update procedures should be easier for iterative testing in orbit
- Need for AI application environment



STELLAR APPS:
ON-BOARD APPLICATION FRAMEWORK FOR SPACE MISSIONS

What is Stellar Apps?

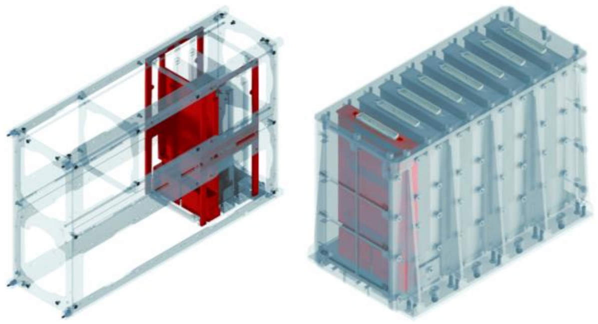
The Stellar Apps software platform enables the secure, efficient, and flexible use of the spacecraft on-board computing capabilities by a variety of third-party applications. The main features include:

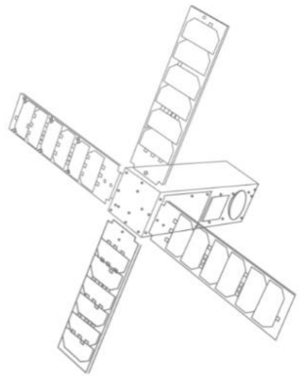
- Run applications in an isolated environment with configurable access to spacecraft and instrument data, ensuring the safety of the mission
- Enables incremental updates and replacement of applications in-orbit to support multi-mission spacecraft
- Provides a software development environment similar to the real execution environment, reducing application time-to-space
- Enables AI applications by providing AI libraries and accelerators

→ „App Store“ for Spacecraft (without the „Store“ part)

Stellar Apps Project

- Design and Development of the **Stellar Apps Software Platform**
- Evolution of the ScOSA OBC platform as EM for Stellar Apps
- Development of at least 7 DLR applications to increase the TRL including the possibility of flight testing
- Use of the 2nd mission year of CAPTn-1 for in-orbit demonstration of Stellar Apps and at least 6 applications



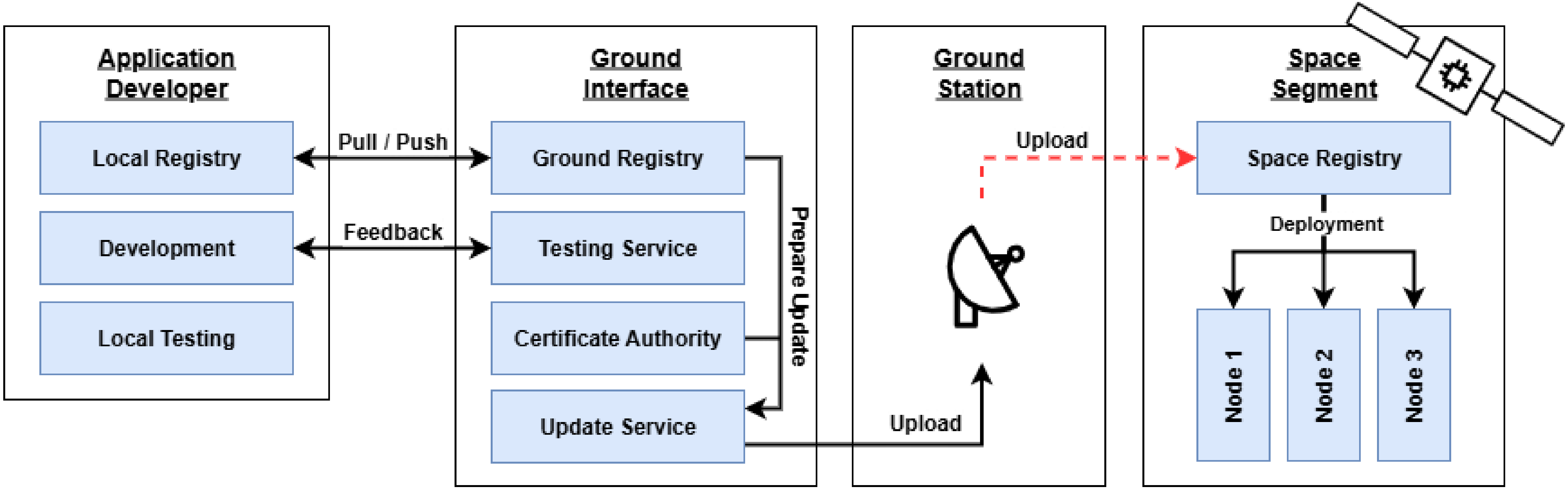


Key Requirements for the Software Platform

1. Isolation and separation of individual applications
2. Regular updates of applications
3. Strong cybersecurity measurements during the entire lifecycle of an application
4. Multi-tenant operations
5. Support for real-time applications
6. API-based communication between applications and the underlying hardware
7. Access to AI accelerators and commonly used runtime libraries
8. Distributed execution of applications on multiple computing nodes

Stellar Apps Software Platform

Main Components

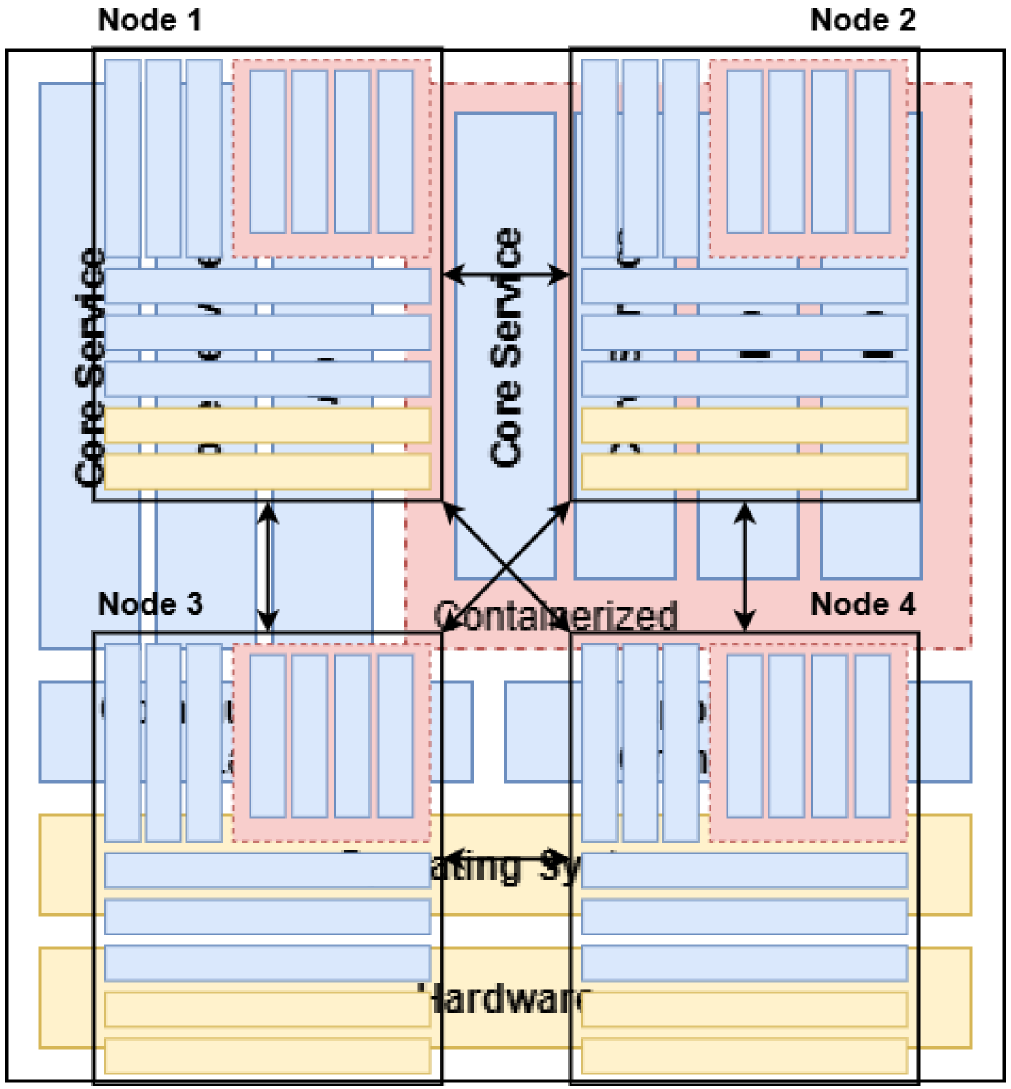


Stellar Apps Layered Software Stack

- Applications communicate with the
  - spacecraft components
  - other apps
  - ground

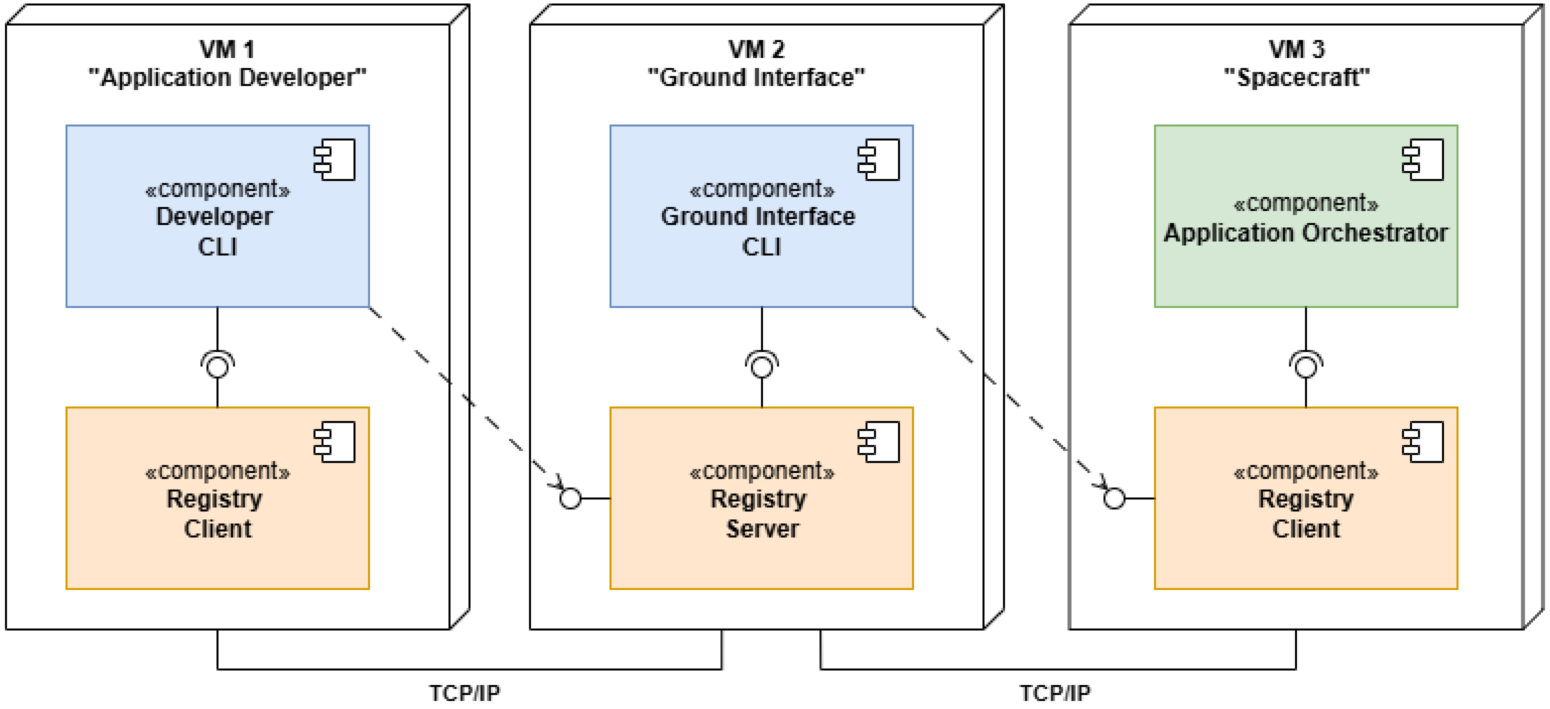
  by an web-service like API
  → abstraction of PUS, CSP, CAN, SpW…

- AI support by at least PyTorch and TensorFlow
- System supports distributed computing



Proof-of-Concept Stellar Apps System

- Applications developed in Rust
- API support for C/C++ and Python (future)
- Apps are containers following the Open Container Initiave (OCI)

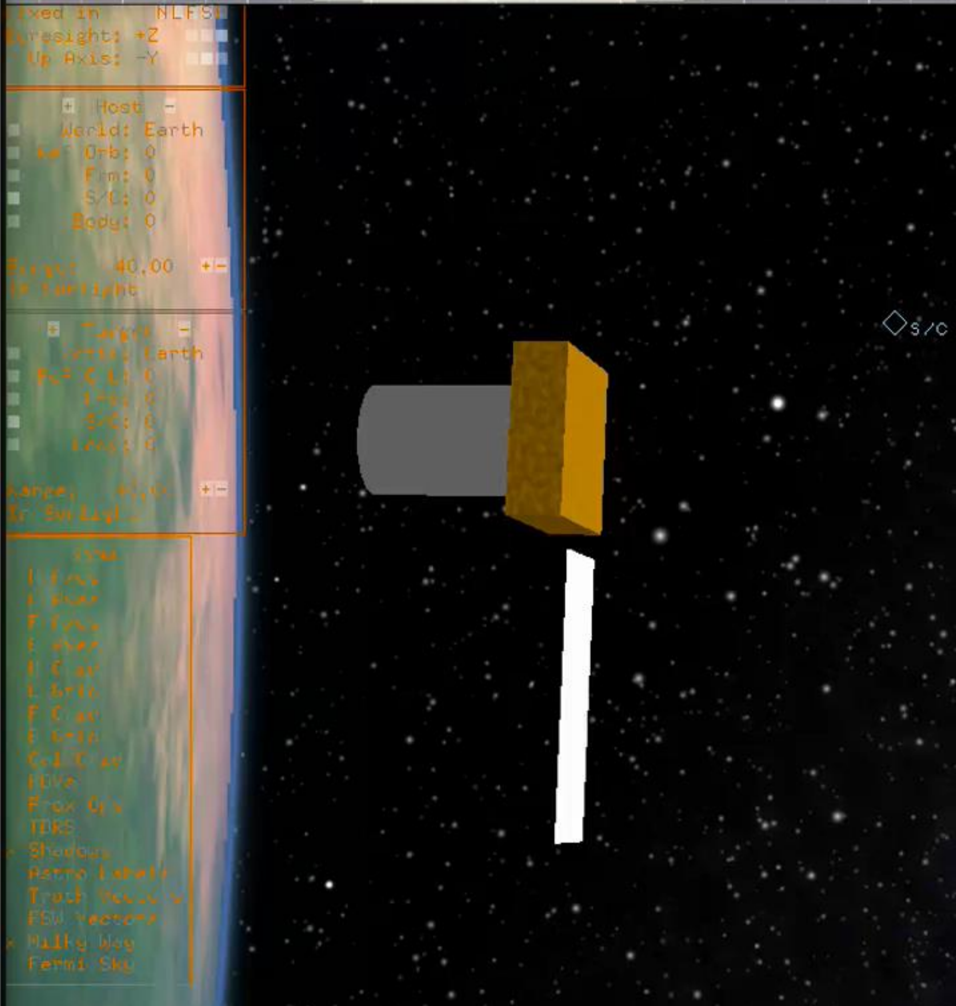


[09:56] satellite | app_orchestrator 0

[09:56] ground | ./reg 30001 1

[09:56] ground |

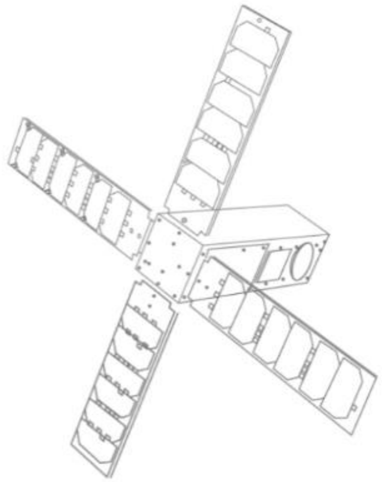
[09:56] developer |



Conclusions & Outlook

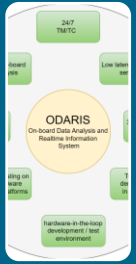
- Stellar Apps platform enables edge computing for future space missions
- **Stellar Apps Features:**

  isolated applications • real-time execution • AI support and acceleration • safety and cybersecurity features • fast to-orbit deployment • API-based communication



- We are interested to build an European “space app” ecosystem by defining a lightweight app API
- If you are interested to discuss: daniel.luedtke@dlr.de

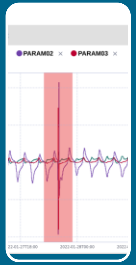
Stellar Apps Applications



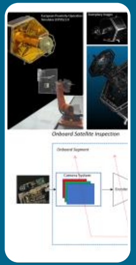
ODARIS: On-board Data Analysis and Real-time Information System

Satellite Attitude Estimation and Tracking

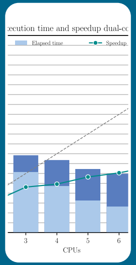
On-board Digital Twin



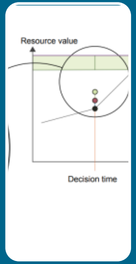
ATHMoS: Automated Telemetry Health Monitoring Service



Computational Sensing for Onboard Inference and Compression



Benchmarks



OBETTE: On-Board Event Triggered Timeline Extension

SEE Detection and Mitigation

3rd Party Apps

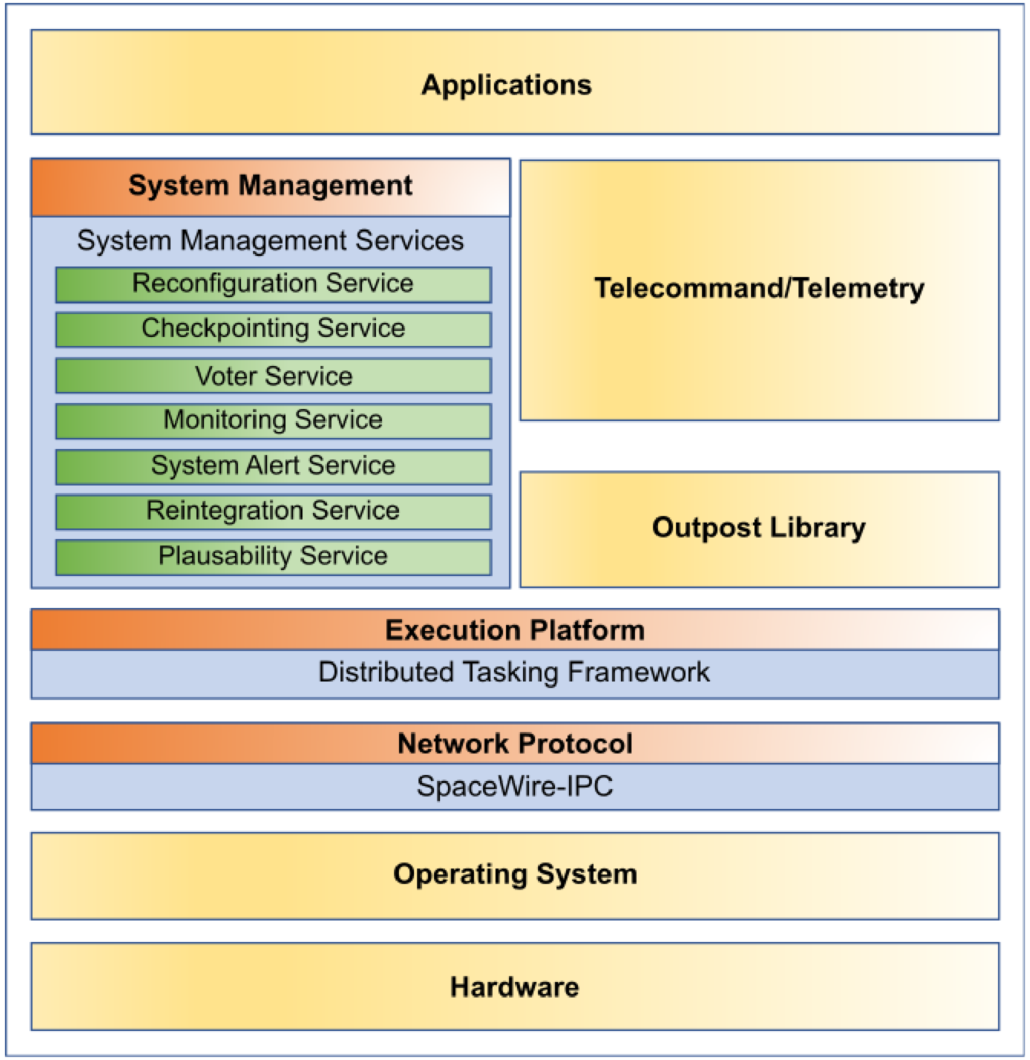
ScOSA System Software
Software Stack

Operating systems

- GNU/Linux
- RTEMS (real-time operating system)

SpaceWire-IPC

- SpaceWire (and Ethernet) protocol to support
  - Inter-process communication (IPC)
  - Reliable / unreliable messages
  - Large message transfer
  - Reconfiguration



ScOSA's layered software stack

Peng, Ting und Weps, Benjamin und Höflinger, Kilian und Borchers, Kai und Lüdtke, Daniel und Gerndt, Andreas (2016) *A New SpaceWire Protocol for Reconfigurable Distributed On-Board Computers.* In: Proceedings of the 2016 7th International SpaceWire Conference, SpaceWire 2016, Seiten 175-182. International SpaceWire Conference 2016, 25.-27. Okt. 2016, Yokohama, Japan. doi: 10.1109/SpaceWire.2016.7771624. ISBN 978-0-9557-1968-4.

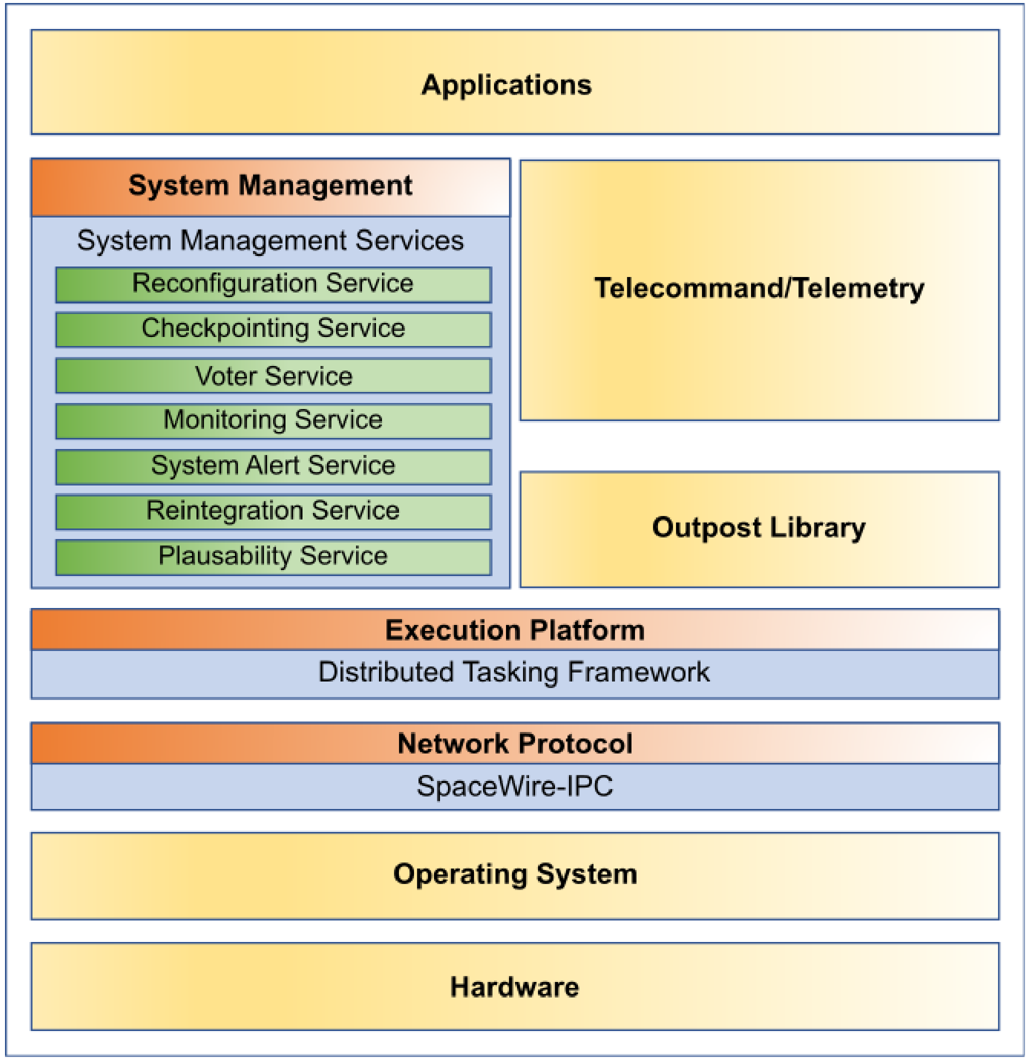
ScOSA System Software
Middleware

Distributed Tasking Framework

- Event-driven, multithreaded middleware
- Distribute applications across multiple nodes
- https://github.com/DLR-SC/tasking-framework

OUTPOST

- Library for low-level space system functions
- Abstraction layer for different hardware and operating systems
- Standardized communication protocols such as CCSDS and PUS
- https://github.com/DLR-RY/outpost-core



ScOSA's layered software stack

Imprint

Topic:	Preliminary Design of the Stellar Apps Software Platform for Developing and Executing On-board Applications
Date:	2025-10-17
Author:	Hendrik Otte, Armin Purle-Kopacz, Kai Bleeke, Arnau Prat, Zain Haj Hammadeh, Andreas Lund, Jan-Gerd Meß, Michael Felderer, Daniel Lüdtke
Institute:	DLR Institute of Software Technology
Image credits:	All images “DLR (CC BY-NC-ND 3.0)” unless otherwise stated