OPTIMISED CCSDS PROTOCOL STACK FOR HIGH DATA RATE (ESA HRLTP)

Final Presentation

Project Overview

Project Team

The HR-LTP Activity must push the state of the art in space-to-ground communication:

- Define communication protocol stacks for high-rate payload data downlink including configuration of the individual protocol layers.
- Identify shortcomings in existing protocols, and outline a open (CCSDS) standard for optical and RF communication links.
- Develop on-board (FPGA) and on-ground implementations proposed protocols.
- Implement the qualification model integrating the prototypes above.
- Validate the open ARQ scheme with the on-board and the ground prototypes with simulated link characteristics.
- Provide input for further standardization

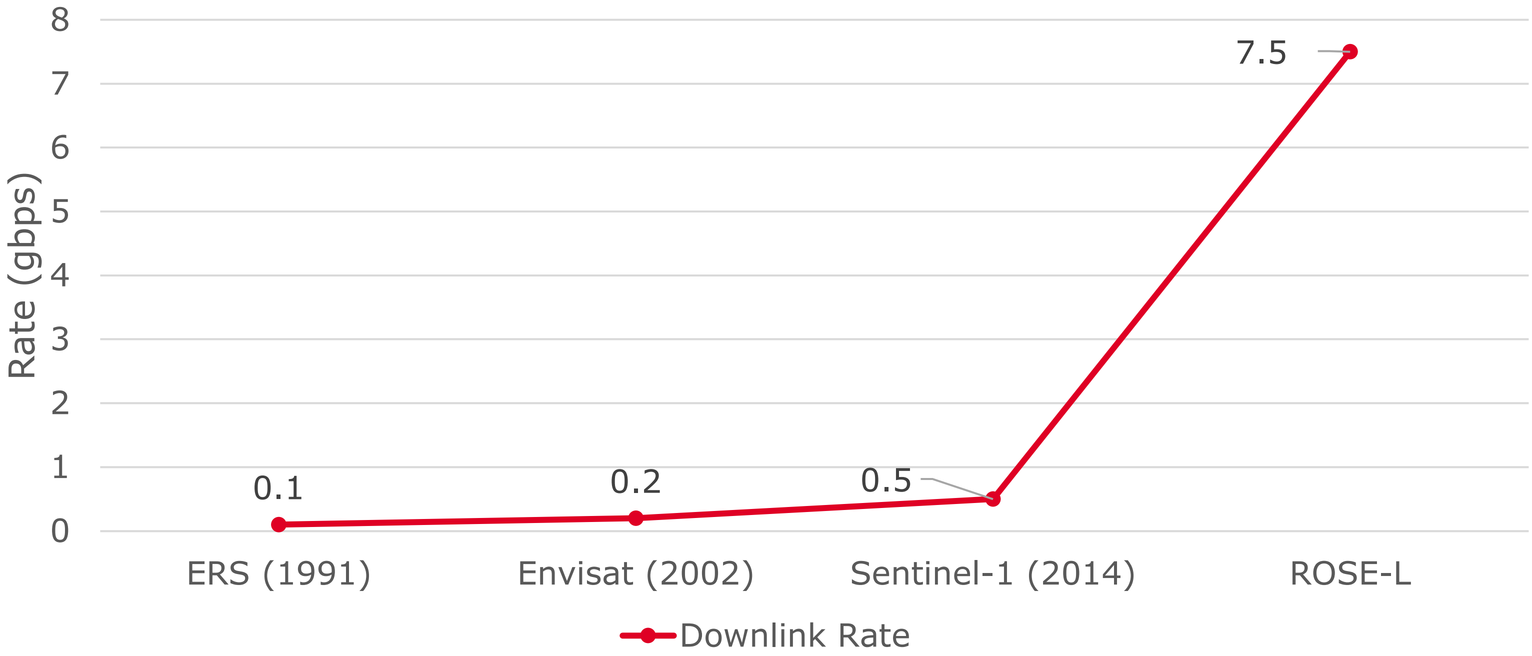
Project duration: 14 months

Consortium: GMV GmbH, TESAT, DLR

Background

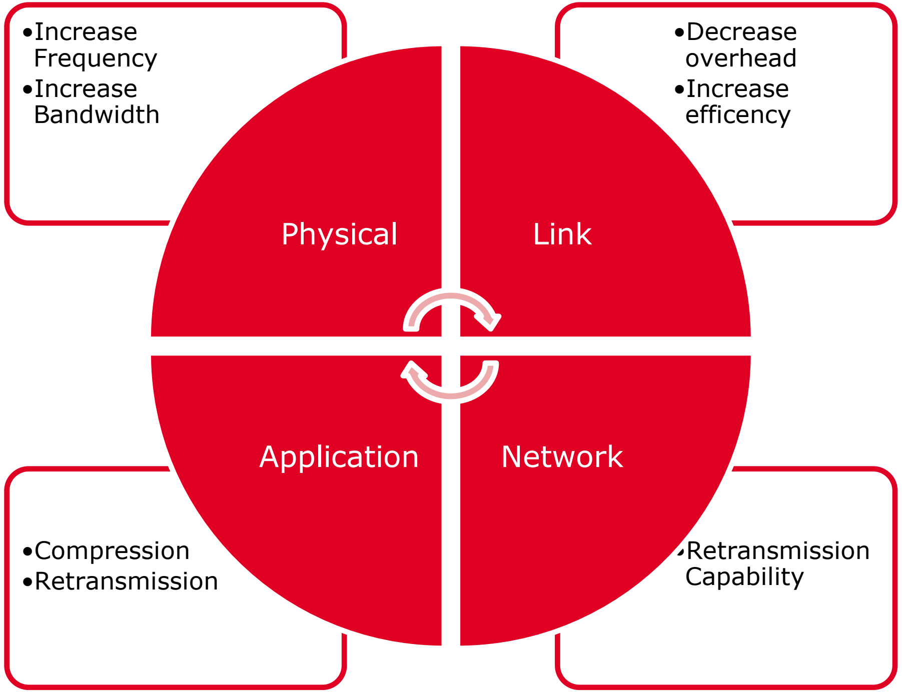
Background

- **Increased demand for scientific data is pushing downlink data rates.**



Increasing capacity?

How do you increase link performance?



Physical Layer

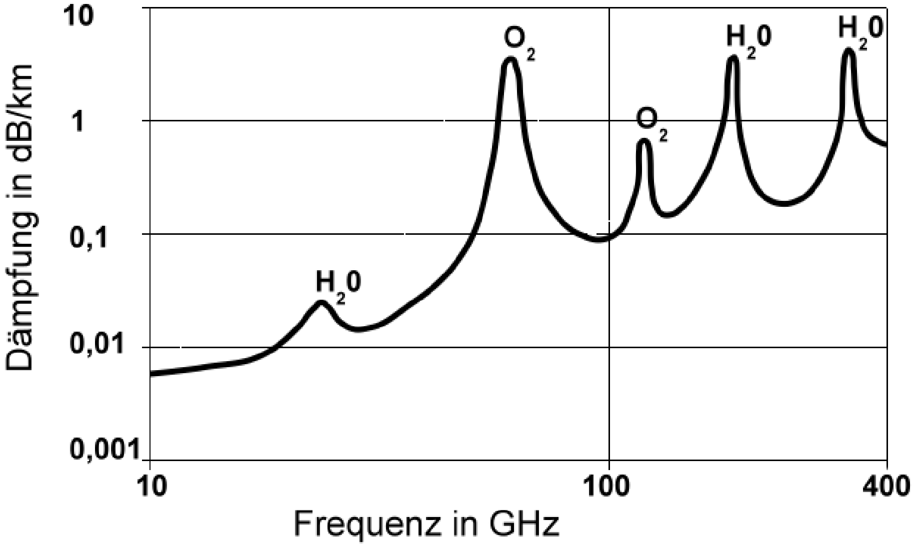
Higher bandwidths can be used (e.g. Ka-)

Pros:

- Increased channel bandwidth – higher capacity
- Increased gain with smaller antennas

Cons:

- Subseptable to fading by rain and atmospheric conditions



Physical Layer - Continued

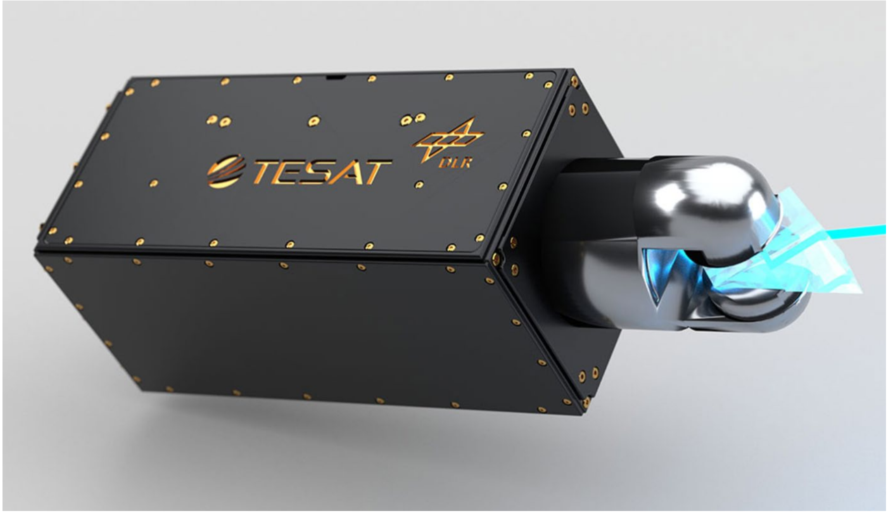
Alternately, optical may be used:

Pros:

- High throughput
- Precise pointing capability
- Commonality between Inter-Satellite Links and S/G

Cons:

- Risk of rain/cloud occlusion



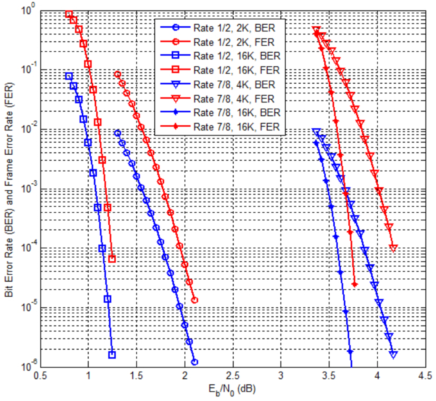
gmv

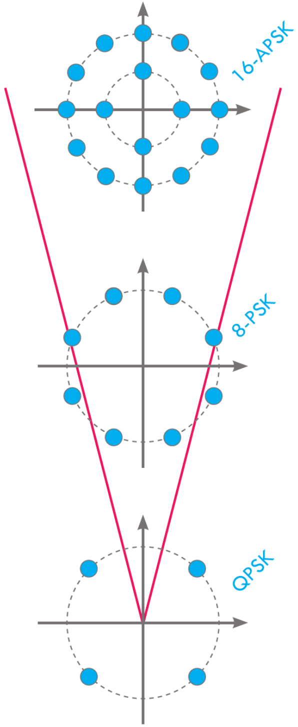
Link Layer

For a given channel, capacity may be increased within the "link" layer:

- **Modulation:** increases the number of symbols (bits) which may be sent in parallel.

- **Coding:** Overhead induced by error coding mechanisms: allows higher performance but reduces reliability
  - Increased error correction provides higher short-term reliability, but does not solve issues relating to longer-duration outages.

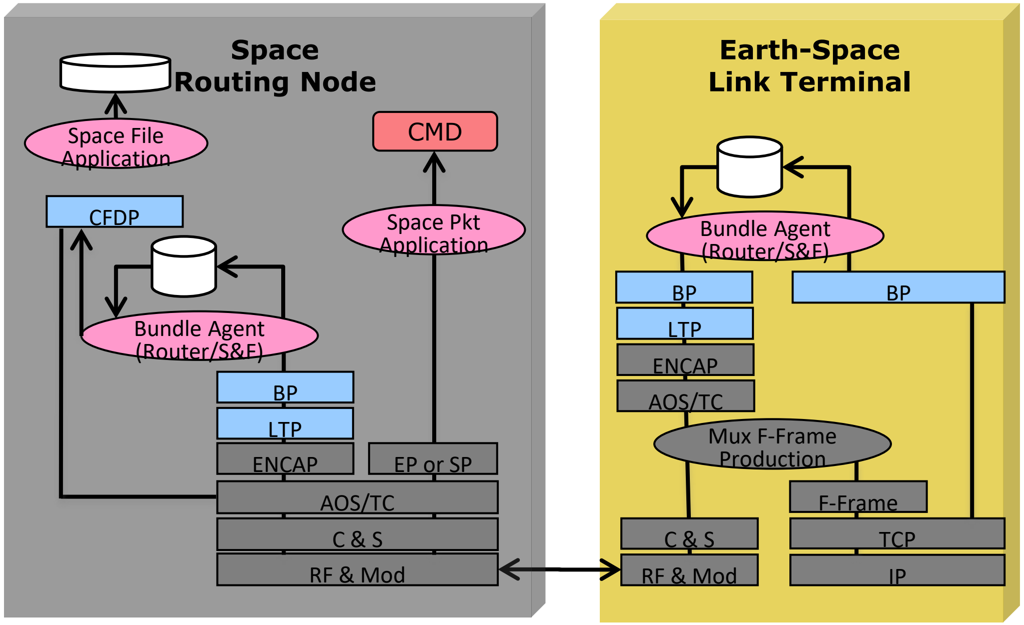




Upper Layers

We can always increase performance at higher levels…

- **Network:** Enable automated retransmission; *just retransmit whatever data was lost by lower levels*
  - Generically-applicable…
  - But requires bidirectional communication
- **Application:** Compress payload data, allowing more of it to be sent?
  - Application-specific
  - Higher CPU/resource utilization



Study Hypothesis

We can increase physical link capacity at the cost of reliability, or we can increase link "goodput" with the application of error correction, etc.

But: can we get the best of both worlds by using "smarter" link-lever protocols:

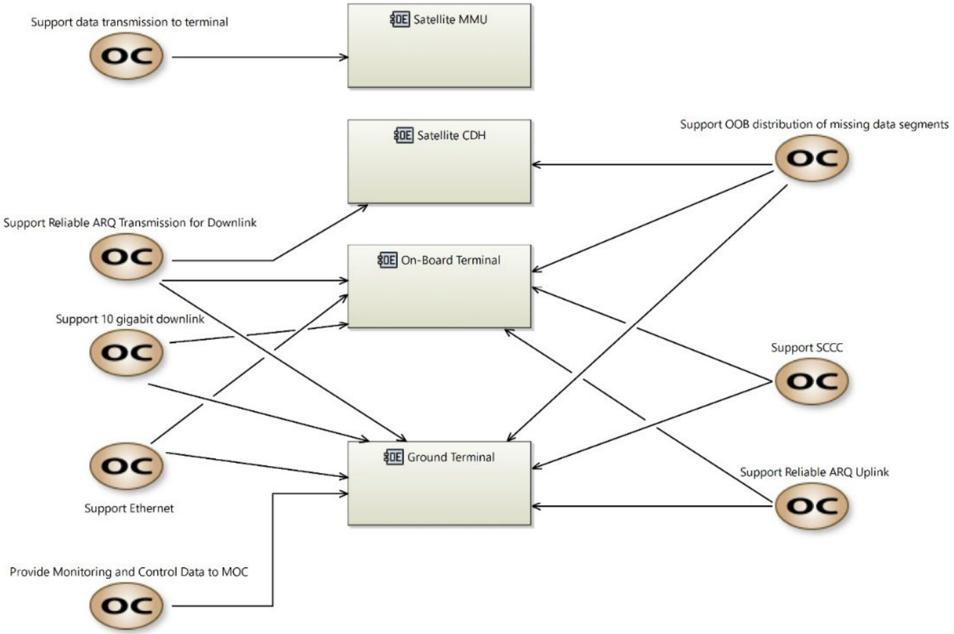
allow the network to manage it's own retransmission

Let's prove it…

System Engineering

System Engineering – Use-cases

- *System engineering was managed via the Arcadia method*
- *Use-cases were developed, encorpating requirements, consortium experience, etc.*



Scenario Definition

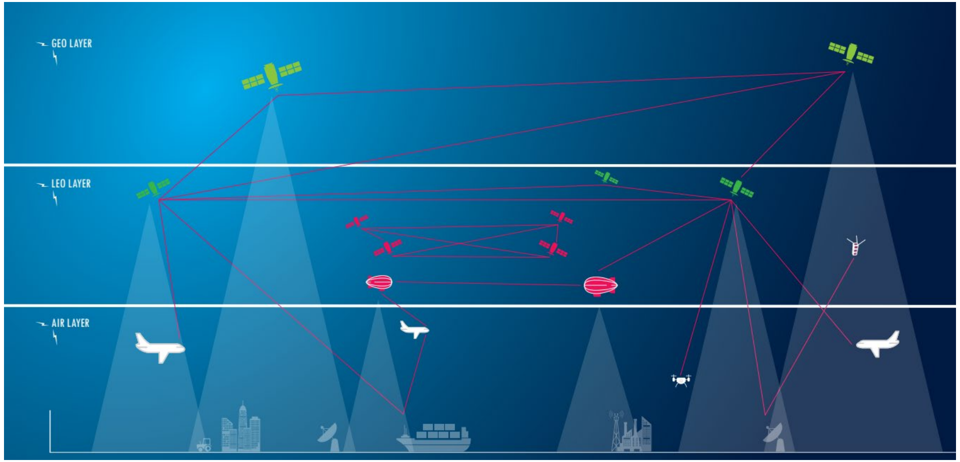
Multiple scenario types were considered:

- Optical ISL/DTE
- Ka- downlink

Scenarios were defined:

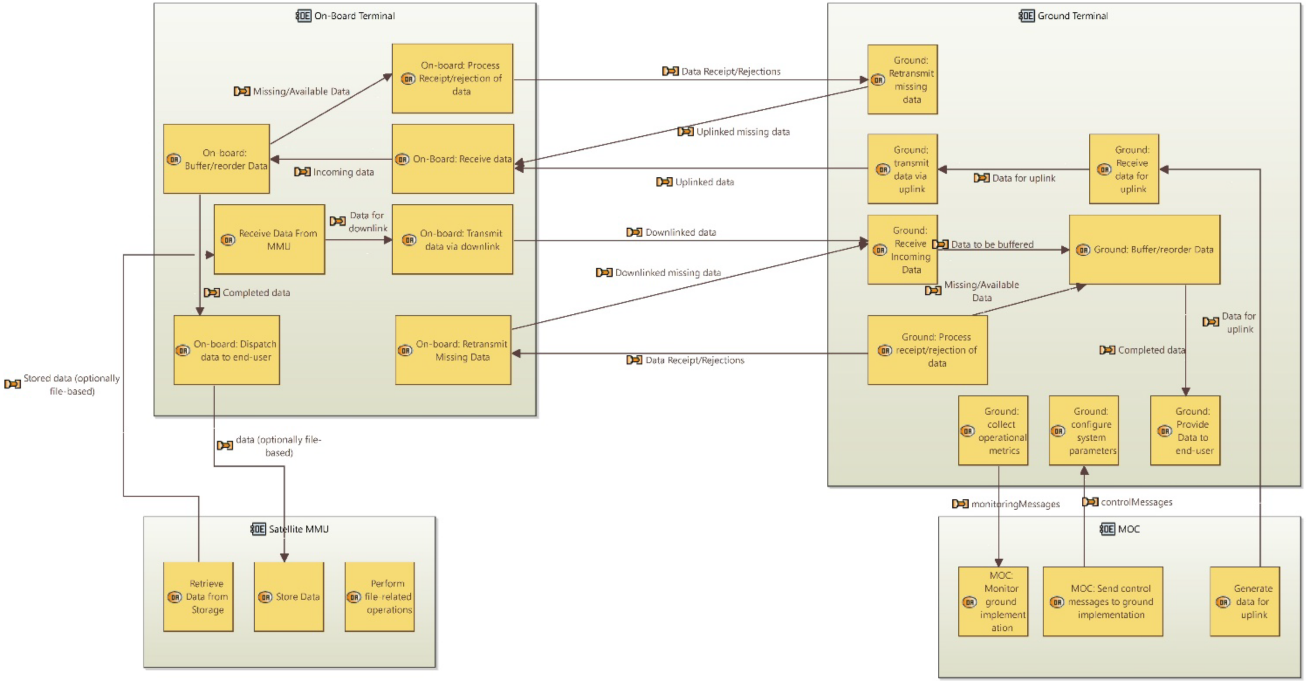
- DTE with simultaneous uplink availability
- DTE with deferred uplink
- GEO (regenerative) relayed DTE
- LEO (regenerative) relayed DTE
- DTE with simultaneous uplink availability

Erasure vectors were created for all, to be used in testing



System Engineering – Operational Activities

- Behaviour & Interaction between the on-board and ground prototypes were modelled as operational activities



Protocol Analysis

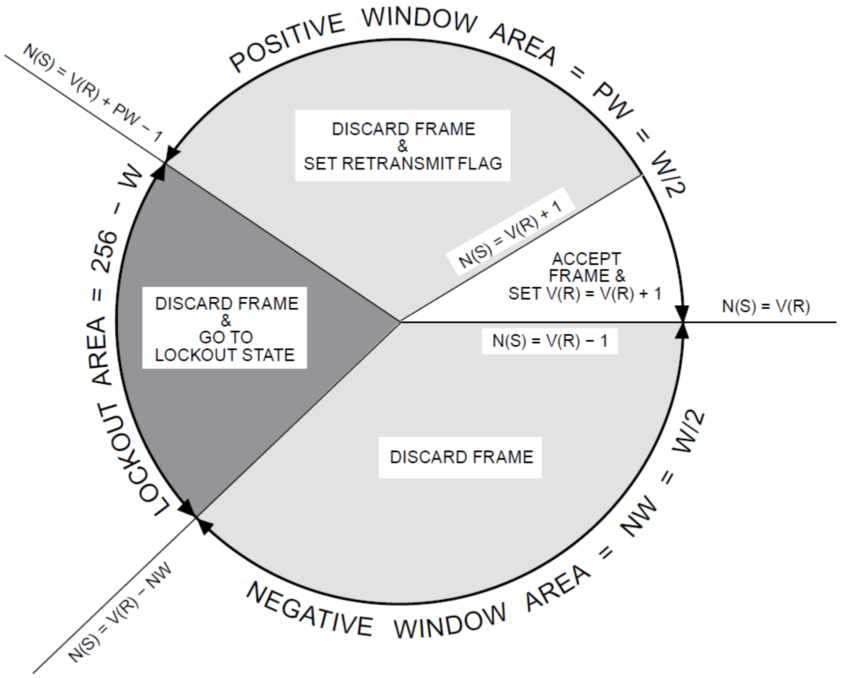
Protocol Requirements

- Provide automated retransmission
- Simplify Protocol Design
- Enable implementation on FPGA & ASIC
- Support > 10gbps data rates

What is ARQ?

Automatic Repeat ReQuest (ARQ) is a family of protocols/methods to enable reliability

- **Stop and Wait** – Send 1 PDU, wait until acknowledgement, send next
- **Go back N** – Send *N* PDUs before requiring acknowledgement, acknowledgement sent as current PDU – N
- **Selective Acknowledgment (SACK)** – Send N PDU's, allowing acknowledgement of any PDU's within that range



Protocol Evaluation

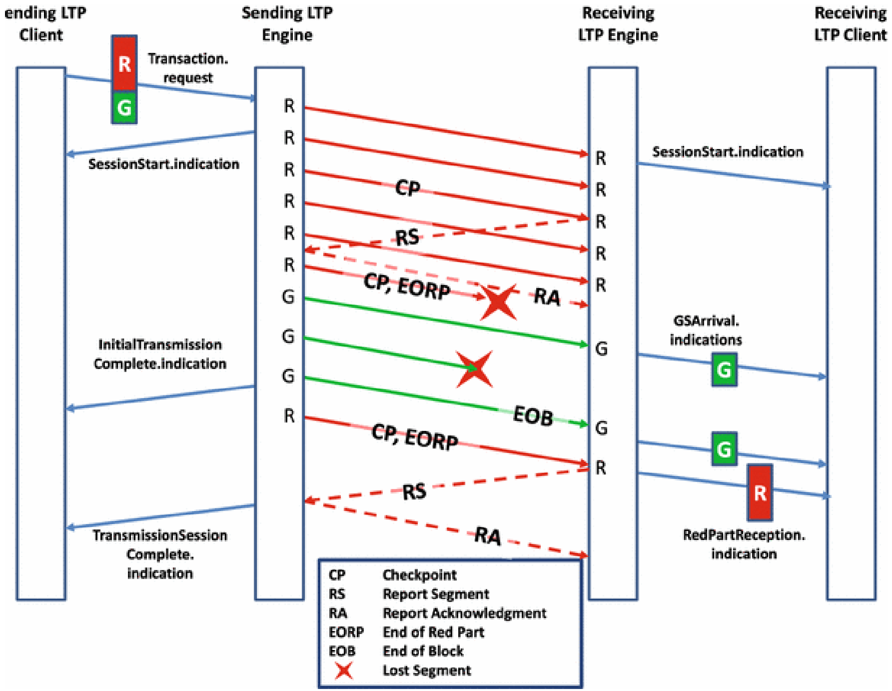
- Protocol design started with the analysis of multiple ARQ protocols, considering suitability for space-links

Protocol	Suitable for long delays	Suitable for high data-rate	Message-based	ACK/NAK based	Congestion Control independent	ARQ flavour	Stream-based	Synchronous connection setup
TCP		X		(S)ACK		GBN, SR	X	X
QUIC				(S)ACK		GBN, SR	X	X
SCPS-TP				SACK, SNACK		GBN, SR	X	X
SRT		X	X	ACK, NACK	X	SR	X	X
NORM		X	X	NACK	X	SR		
LTP	X		X	SACK	X	SR		
CFDP	X	X	X	NACK	X	SR		
COP-1/-P			X	ACK	X	GBN		
802.11 BA			X	ACK	X	SR		/
DLR Patent	X	X	X	X	X	SR		
AX.25	/		X	ACK	X	GBN	X	X

Protocol Selection

The Licklider Transmission Protocol (LTP) was selected as a baseline for the proposed protocol

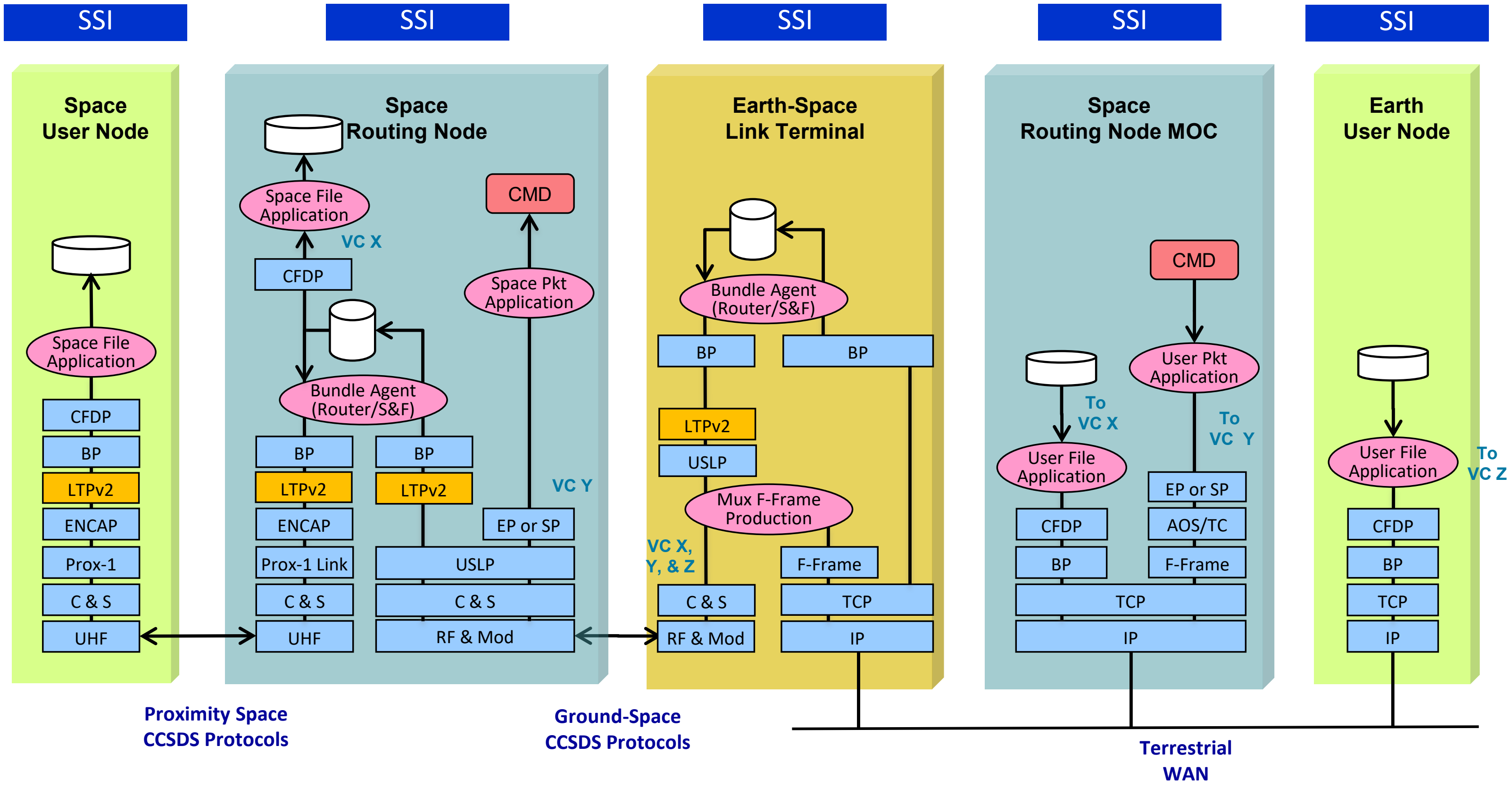
- Existing CCSDS standard…
- With consortium familiarity.
- Provides reliable and unreliable channels.



LTP is not without problems:

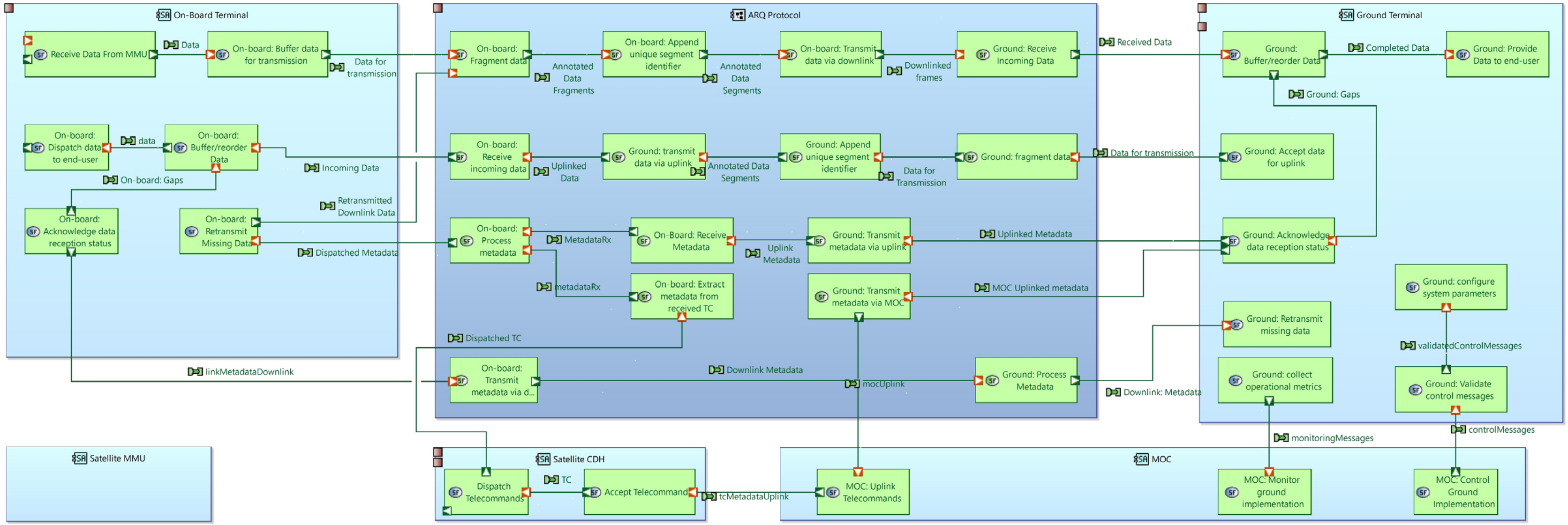
- Relatively complex internals required to support mixed (unreliable/reliable) sessions
- Extensive utilization of variable length fields
- Many different on-the-wire data representations

LTPv2 in SSI


SSI
SSI
SSI
SSI
SSI
Space
User Node
Space File
Application
CFDP
BP
LTPv2
ENCAP
Prox-1
C & S
UHF
Space
Routing Node
Space File
Application
VC X
CFDP
CMD
Space Pkt
Application
Bundle Agent
(Router/S&F)
BP
BP
LTPv2
LTPv2
VC Y
ENCAP
EP or SP
Prox-1 Link
USLP
C & S
C & S
UHF
RF & Mod
Earth-Space
Link Terminal
Bundle Agent
(Router/S&F)
BP
BP
LTPv2
USLP
Mux F-Frame
Production
VC X,
Y, & Z
F-Frame
C & S
TCP
RF & Mod
IP
Space
Routing Node MOC
CMD
User Pkt
Application
To
VC X
To
VC Y
User File
Application
EP or SP
CFDP
AOS/TC
BP
F-Frame
TCP
IP
Earth
User Node
User File
Application
To
VC Z
CFDP
BP
TCP
IP
Proximity Space
CCSDS Protocols
Ground-Space
CCSDS Protocols
Terrestrial
WAN

Prototype Design

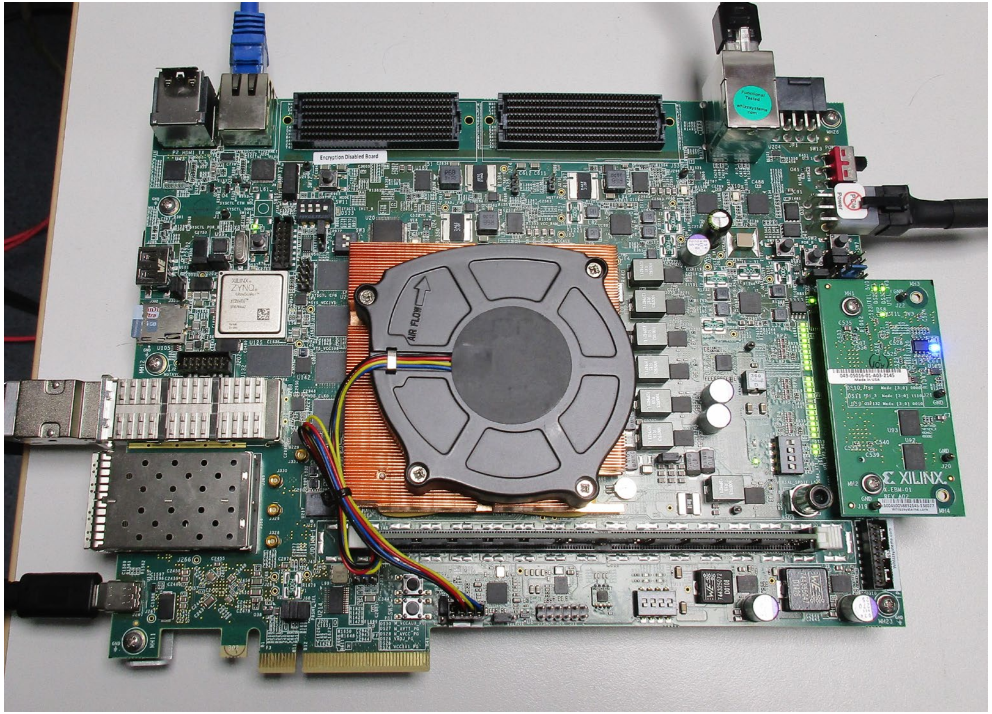
Overall System Design

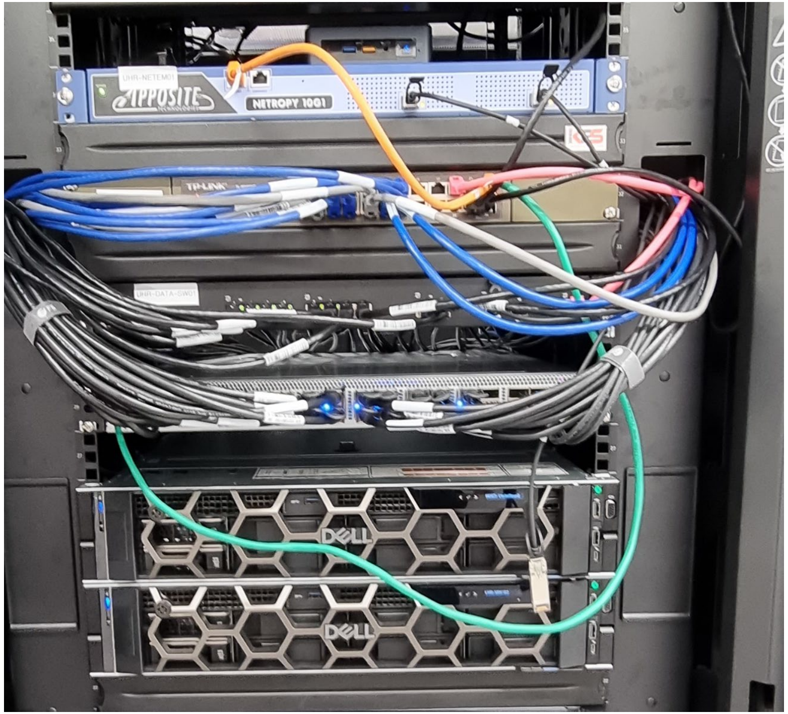


Two prototypes were built:

- **Xilinx Versal FPGA –** Emulating next-generation On-Board Computers
- **A PC** – Replicating a ground station

LTPv2 – Physical Implementations

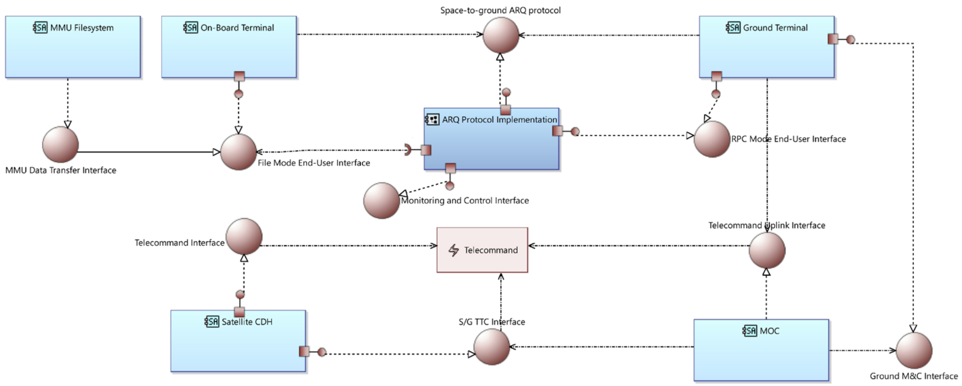


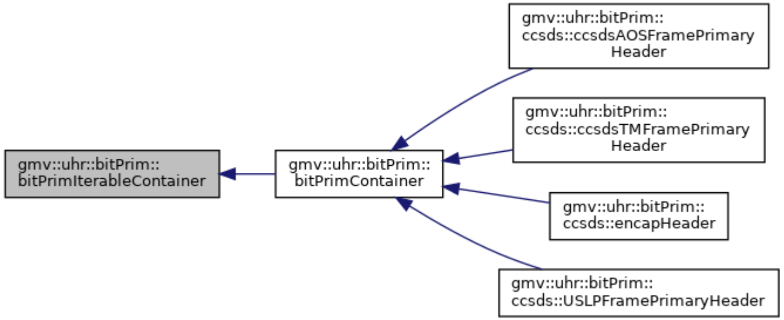


gmv

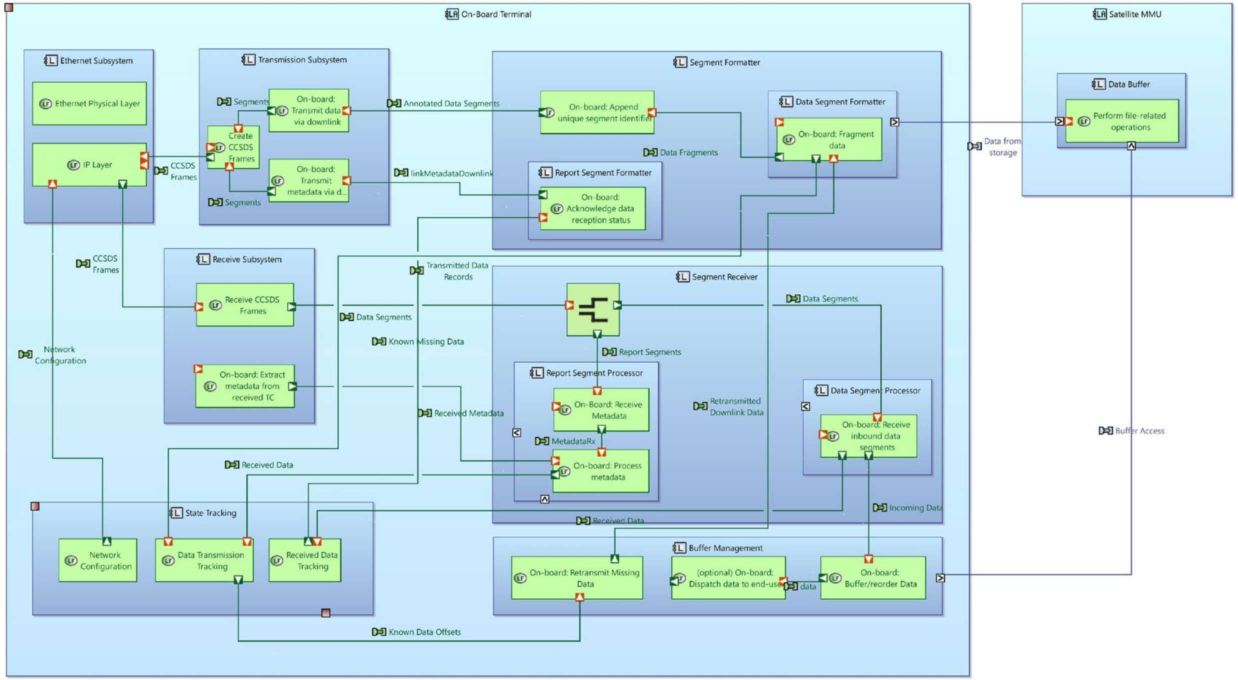
External Interfaces

- Space link provided via USLP, encapsulated in UDP packets
- Telemetry provided via OpenMetrics (via HTTP)
- Commanding provided via RPC system(s)
- External file management provided by an emulation of the SAVOIR file management service:
  - *Used by the FPGA prototype to manage data transfer.*



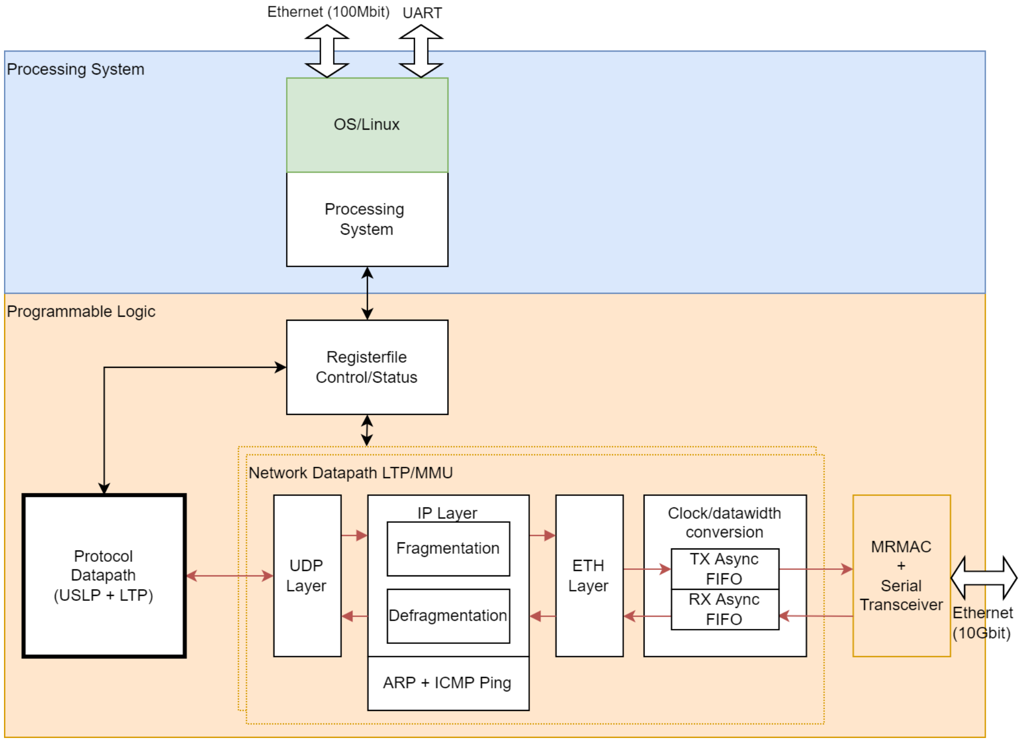


On-Board Prototype – Design

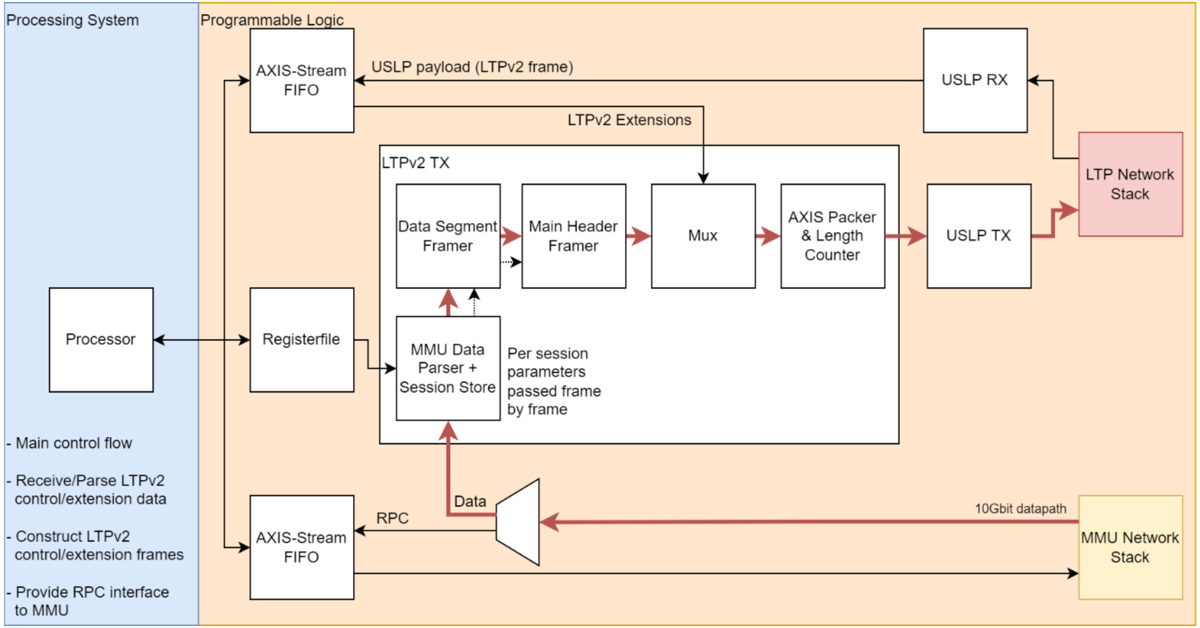


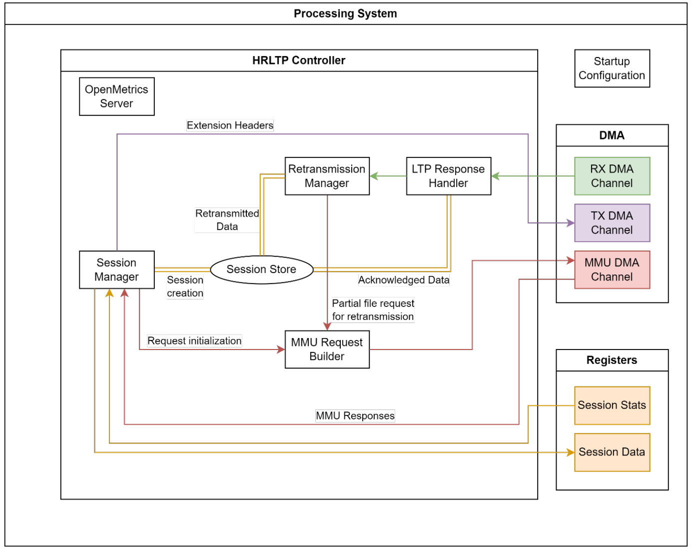
gmv

On-Board Prototype – Data Plane

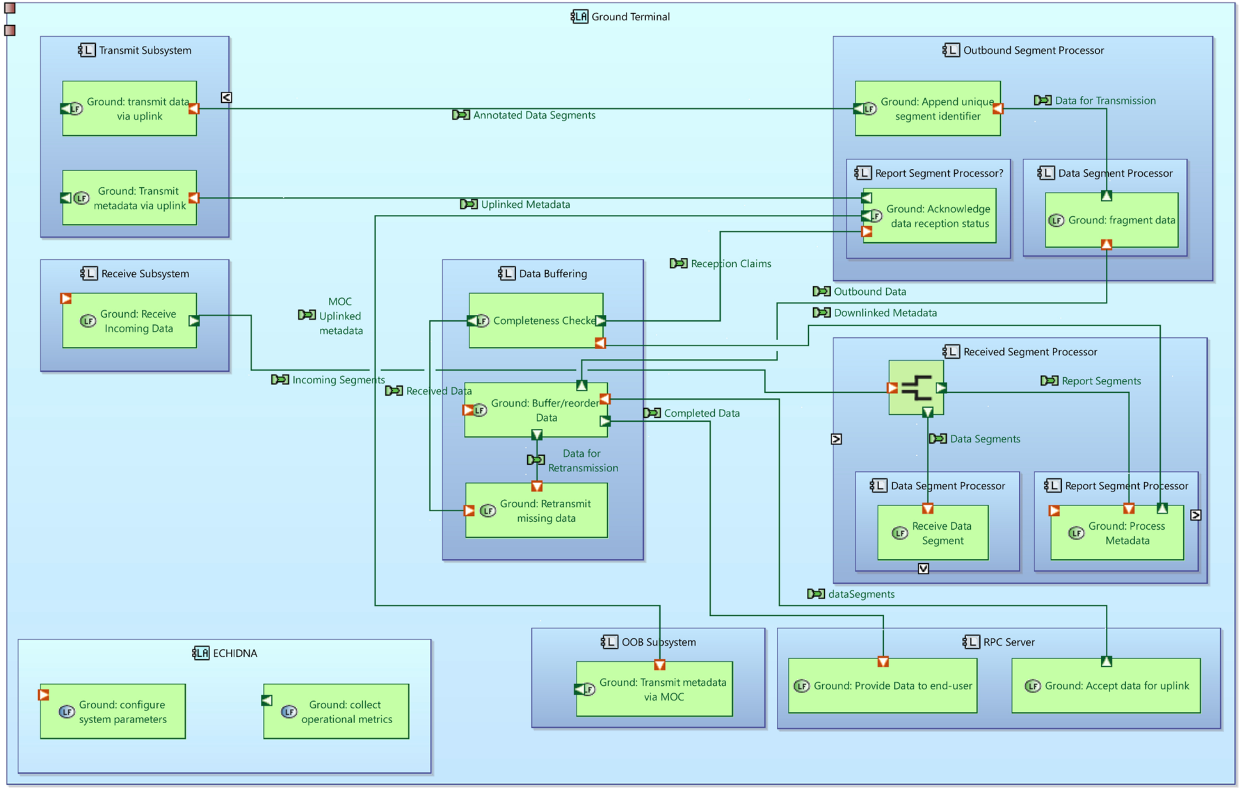


On-Board Prototype – LTPv2





Ground Prototype - Design



Ground Prototype - ECHIDNA

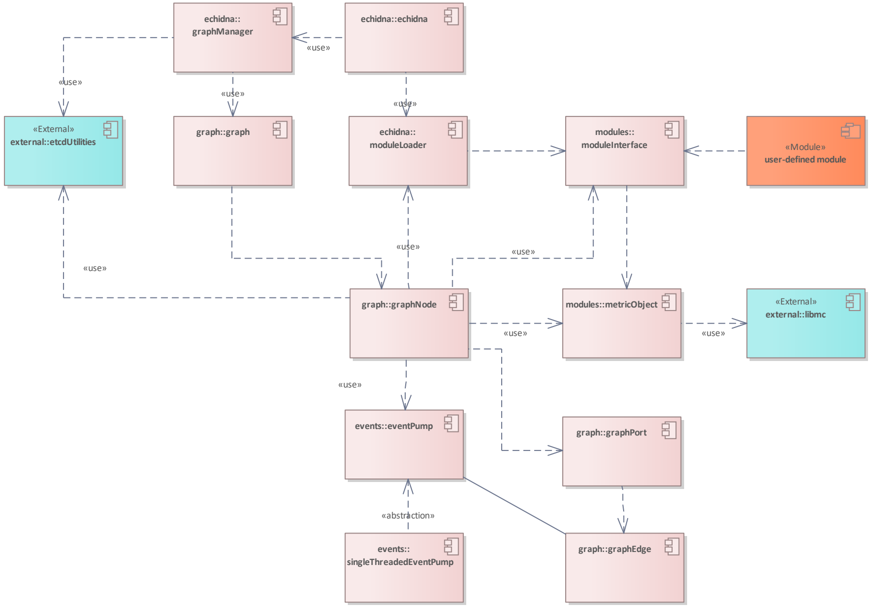
Ground Prototype was developed within the ECHIDNA framework:

- A modular framework for high-rate applications.

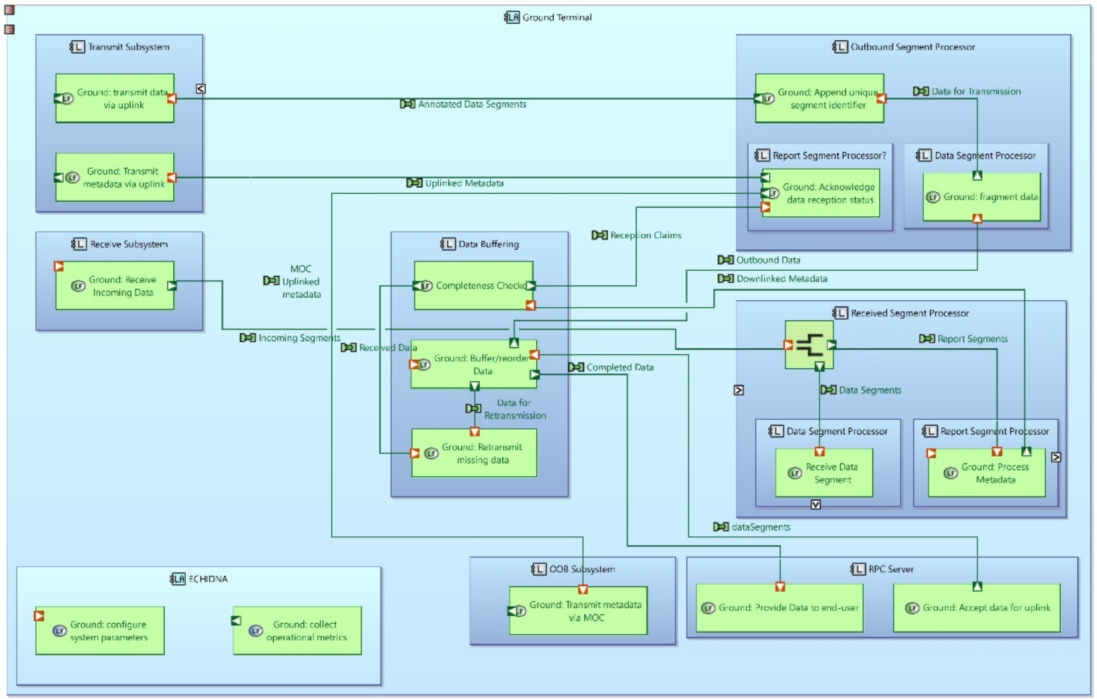
Open-source tooling used for other components:

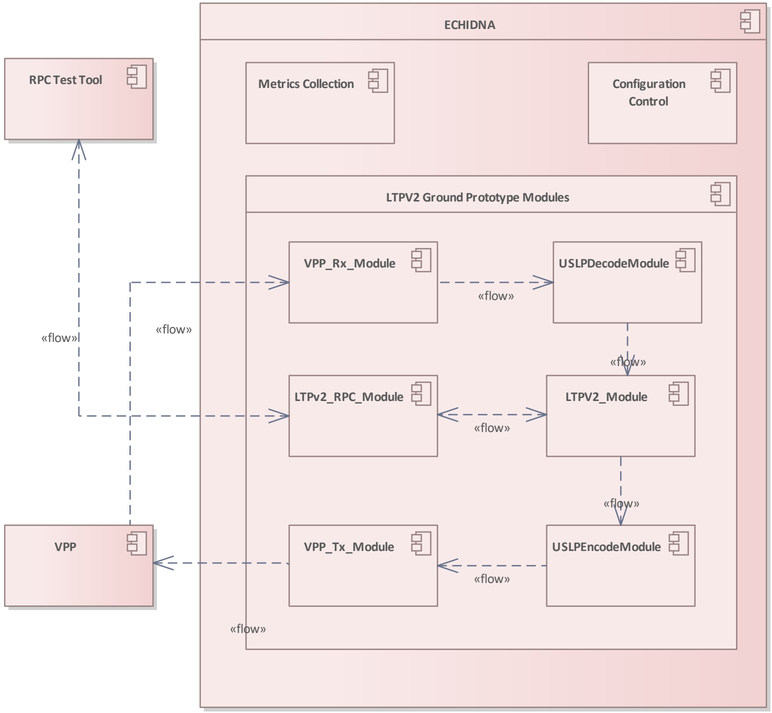
- Prometheus for monitoring
- VPP/DPDK for networking
- Etcd for configuration control

Developed on Ubuntu, deployed on bare-metal, VM's, and containers.



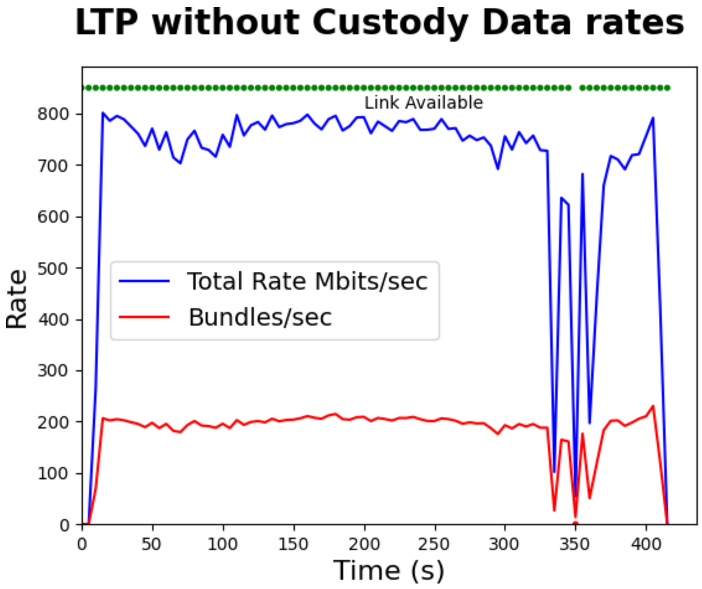
Software Prototype



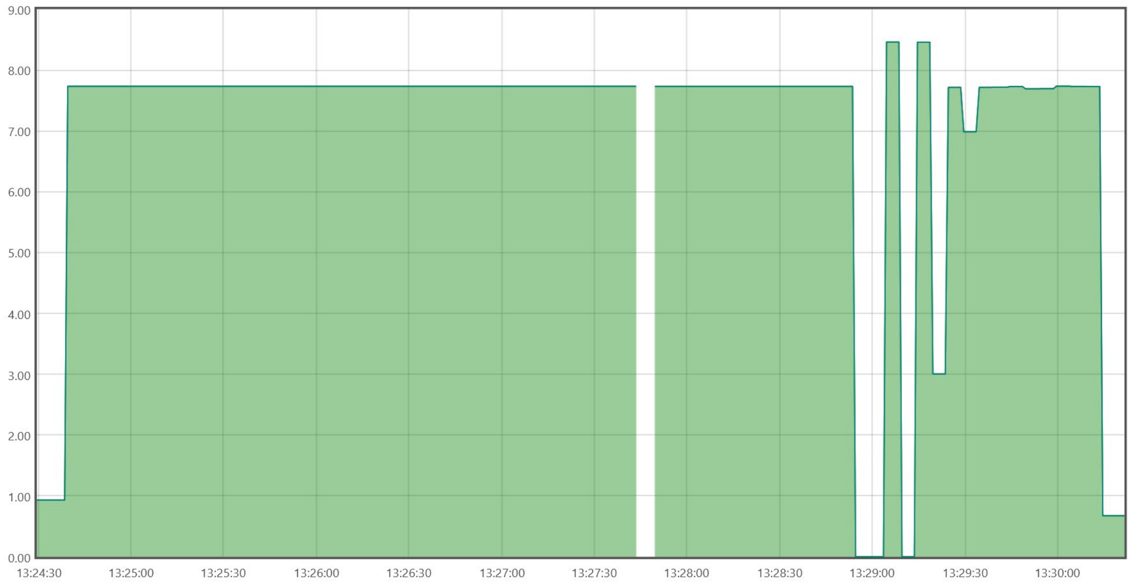


Project Results

LTPv2 –Performance



NASA HDTN



LTPv2 FPGA->CPU (reliable)

Demonstration

LTPv2 –Validation Process

- Validation performed via ESOC Jira – Zephyr Plugin
- All tests performed by GMV staff, and recorded

Test execution matrix by test cycle

Test Case	Latest	HRLTP-C1 - filePlayer RPC Testing	HRLTP-C10 - VPP Test Cycle	HRLTP-C11 - Populate etcd with configuration data	HRLTP-C12 - docker-compose - deployment	HRLTP-C13 - LTP-RPC - Sending and receiving file read data	HRLTP-C2 - Startup	HRLTP-C4 - Verify HRLTP-controller	HRLTP-C5 - Kubernetes Deployment	HRLTP-C6 - LTPv2 - build and run	HRLTP-C7 - USLP - build and run	HRLTP-C8 - LTP-RPC - Sending and receiving counting data.	HRLTP-C9 - Install and Run Echidna on SLES15
HRLTP-T1 Open existing file with filePlayer No environment	Result: Pass By: Jeremy Pierce Mayer Actual End Date: 19/04/2023	Result: Pass By: Jeremy Pierce Mayer Actual End Date: 19/04/2023											
HRLTP-T11 docker-compose - etcd configuration ubuntu20.04	Result: Pass By: Pablo Hinojsa Lopez Actual End Date: 24/04/2023				Result: Pass By: Pablo Hinojsa Lopez Actual End Date: 24/04/2023								
HRLTP-T12 kubernetes - echidna deployment ubuntu20.04	Result: Pass By: Pablo Hinojsa Lopez Actual End Date: 24/04/2023								Result: Pass By: Pablo Hinojsa Lopez Actual End Date: 24/04/2023				
HRLTP-T13 Install Echidna on Suse Linux Enterprise Server SLES 15 No environment	Result: Pass By: Matthias Urban Actual End Date: 21/04/2023												Result: Pass By: Matthias Urban Actual End Date: 21/04/2023
HRLTP-T16 FPGA Hard Reset No environment	Result: Pass By: Jeremy Pierce Mayer Actual End Date: 20/04/2023							Result: Pass By: Jeremy Pierce Mayer Actual End Date: 20/04/2023					
HRLTP-T17 kubernetes - etcd configuration ubuntu20.04	Result: Pass By: Pablo Hinojsa Lopez Actual End Date: 24/04/2023								Result: Pass By: Pablo Hinojsa Lopez Actual End Date: 24/04/2023				
HRLTP-T18 kubernetes - cluster and host setup ubuntu20.04	Result: Pass By: Pablo Hinojsa Lopez Actual End Date: 24/04/2023								Result: Pass By: Pablo Hinojsa Lopez Actual End Date: 24/04/2023				

High Level Project Results

- Implementing the LTPv2 protocol on a PC server as well as a modern FPGA
  - Data rates of 10gbps could be achieved between prototypes, using MMU emulation and reliable/unreliable transmission
- The project was managed via agile Model-Based System Engineering (MBSE)
- Both prototypes were developed in geographically separated locations, using the development processes of GMV and TESAT
  - Interoperability was ensured via a series of tests, conducted with representative data exchanged via file and packet captures
  - Tests showed interoperability between the two projects, as well as ensuring that the performance requirements outlined the ITT could be met
- Representative space-to-ground scenarios were analyzed and updated

Thank you

INNOVATING SOLUTIONS