Developing High Performance Space Networking Capabilities for the International Space Station and Beyond

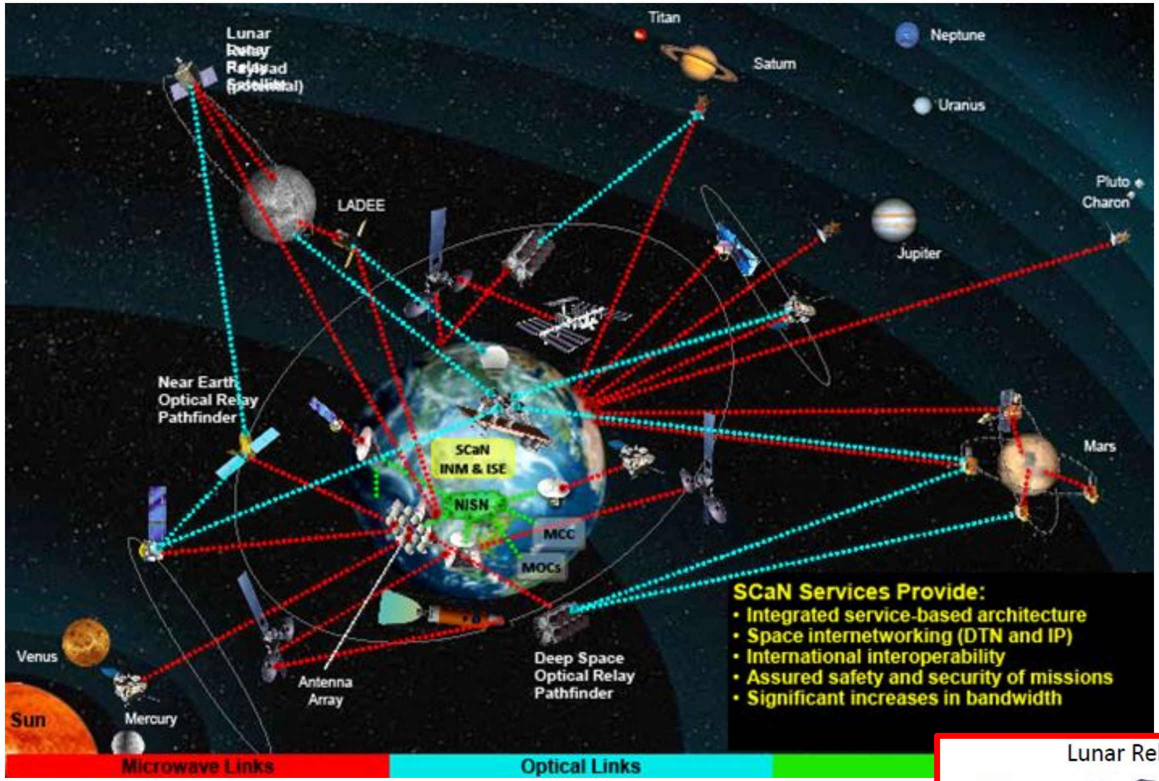
Daniel Raible[1], Rachel Dudukovich[1], Brian Tomko[1], Nadia Kortas[1], Blake LaFuente[1],

Dennis Iannicca[1], Tom Basciano[2], Bill Pohlchuck[3], Josh Deaton[4], Alan Hylton[5], John Nowakowski[1]

1 NASA Glenn Research Center, 2 NASA Johnson Space Center, 3 Boeing Aerospace Company, 4 HOSC Services Contract, 5 NASA Goddard Space Flight Center

High-rate Delay Tolerant Networking (HDTN)

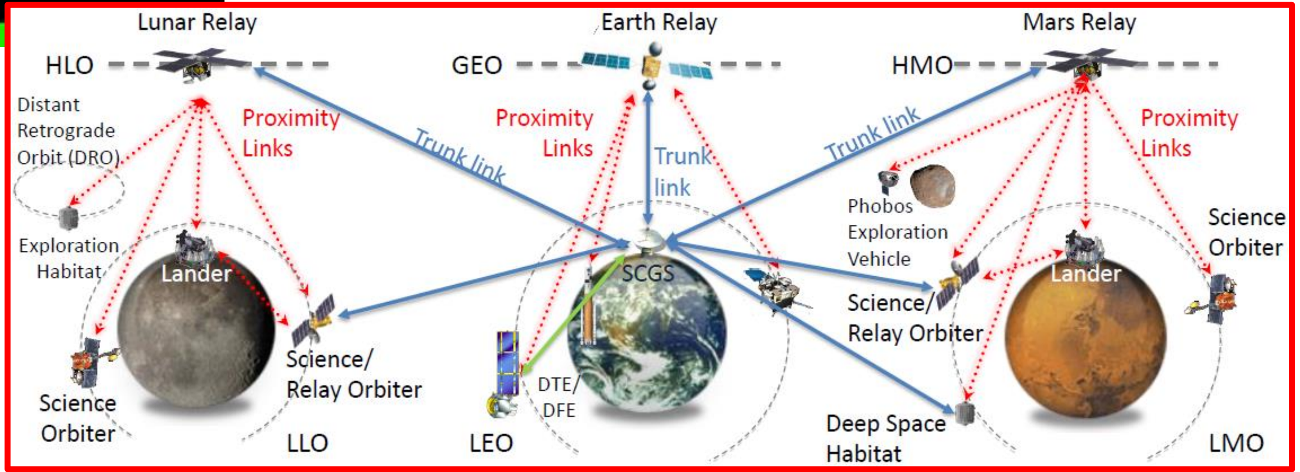
Investigating and developing mission solutions spanning the edge cases of space networking



Goals for Planetary Networks

- Reduced mission burden with short links for in-system communications, enables in-system telerobotics
- Common architecture reduces technology and development costs
- Reuse of hardware and software: family of products includes variants for different environments
- Reuse of spectrum

Network Topology Transformation



Store,
Carry,
Forward…

Delay/Disruption/Disconnection Tolerant Networking (DTN)

Buffers data until a transmit opportunity arises,
Where an end-to-end path may not be available

Space communication naturally exhibits several networking challenges

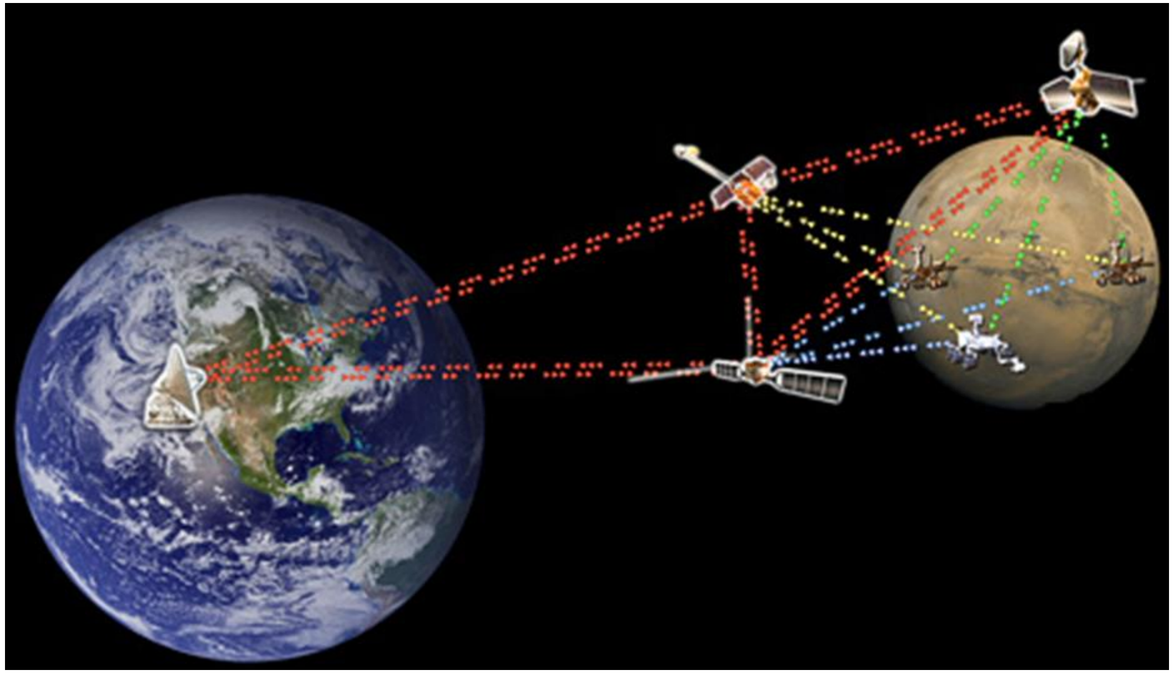
- Disconnection *interruptions in service, can be unscheduled*
- Buffering *levies storage requirements on s/c*
- Latency *negates ACK - based protocols*
- Security *imparts requirements on transceivers*

DTN approach organizes systems

- Multiple link (RF + optical) management, routing
- Establishes maximum transmission unit (MTU) bundle
- Ameliorates forward/return link & bus asymmetry
- Interoperability, pro & reactive bundles fragm

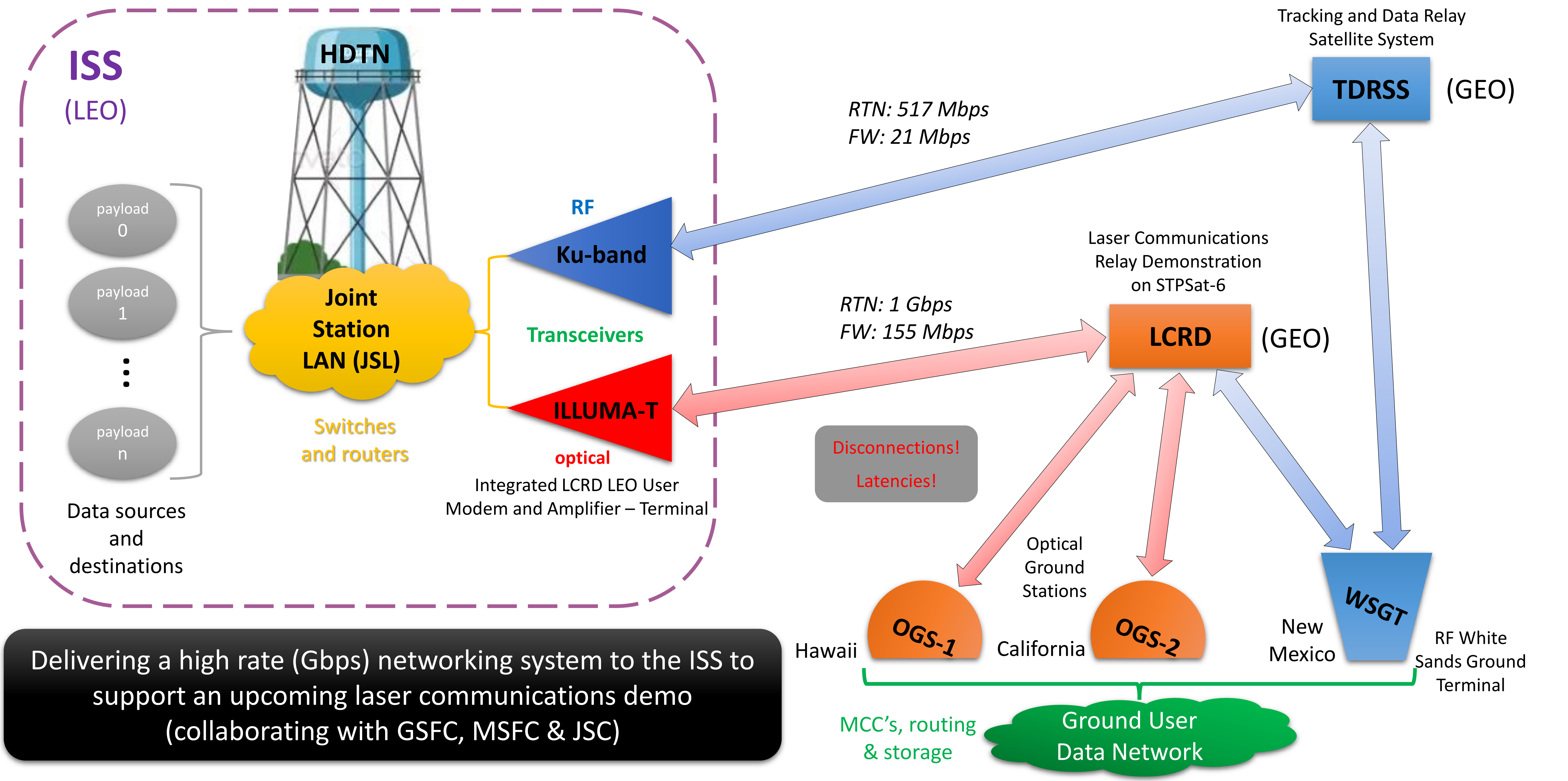
Development Goals

- Ensure DTN is not the communications bottleneck
- Open architecture: RFC 5050 bundle protocol (BP) specification Consultative Committee for Space Data Systems (CCSDS)
- Offload burden from CPU (separate control / data plane)
    - **1 Gb/s now, 100 Gb/s later**
- **Commercialization through open-sourcing of technology**



DTN unites the formerly disjoint collection of assets and links to form a cohesive system of systems

HDTN ISS Concept of Operations

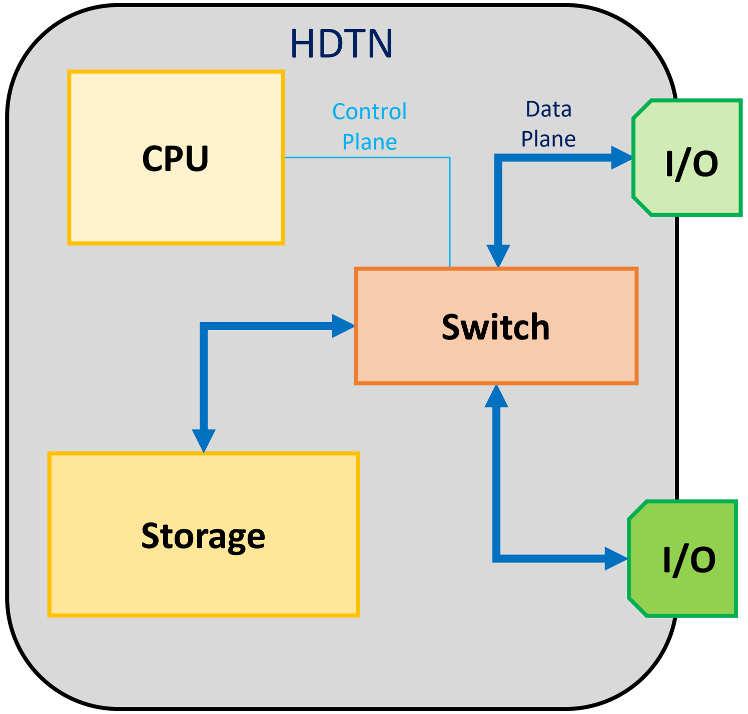


Delivering a high rate (Gbps) networking system to the ISS to support an upcoming laser communications demo (collaborating with GSFC, MSFC & JSC)

HDTN Architecture

Goal: Increase network data throughput to meet future user needs.
Enhance communications capability to increase mission science return.

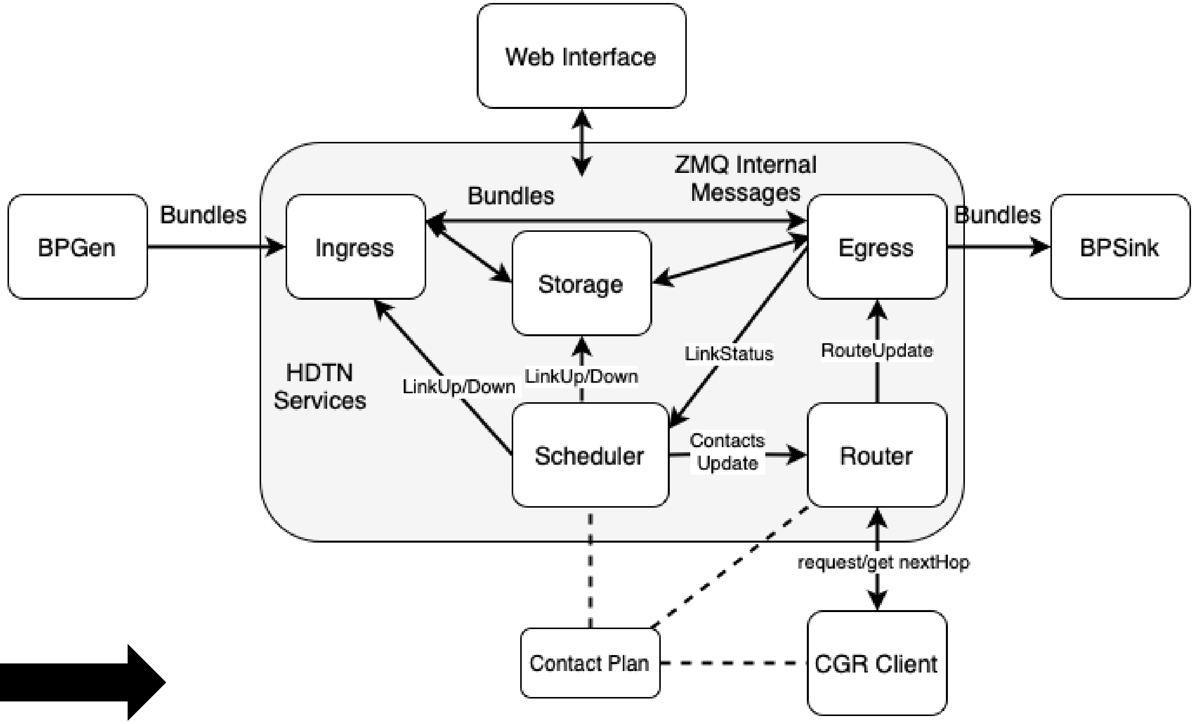
Initial Concept



System provides functionality at two levels

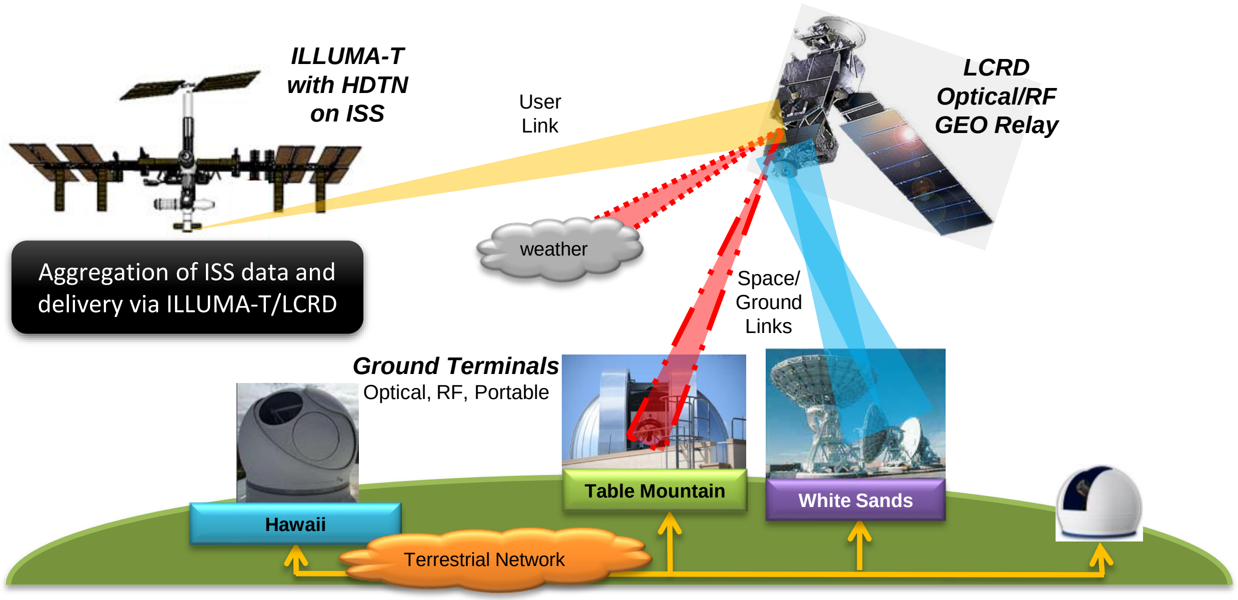
- Control Plane
- Data Plane

Current Architecture



Evaluating HDTN in the Laboratory

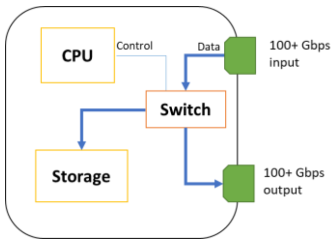
Traffic model



Development Approach:

- Constructed a modular and high speed software implementation of DTN
- Publicly releasing and updating products on GitHub repository, publishing support docs
- Leveraging COTS components for hardware acceleration reaching 100 Gbps
- Pursuing class-A software certification to support Artemis crewed missions

Prototype HDTN



Technology Maturity

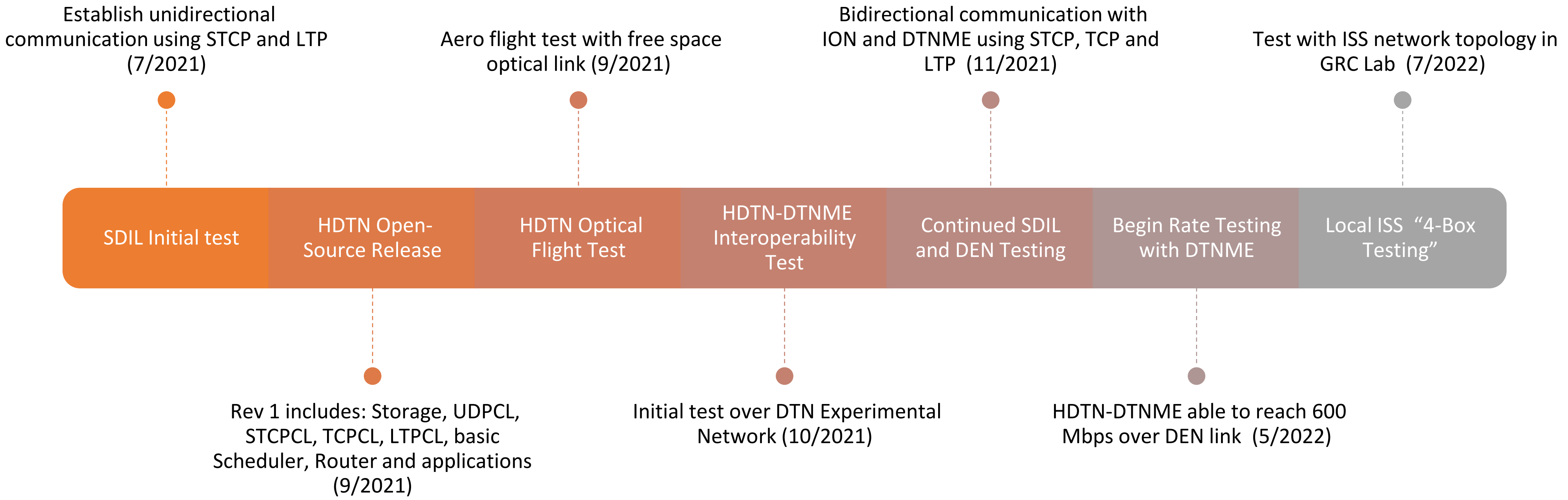
Flight Demonstrations

Commercial Solutions

Flight research aircraft
using laser links

Laboratory
emulation testbed

Testing Timeline

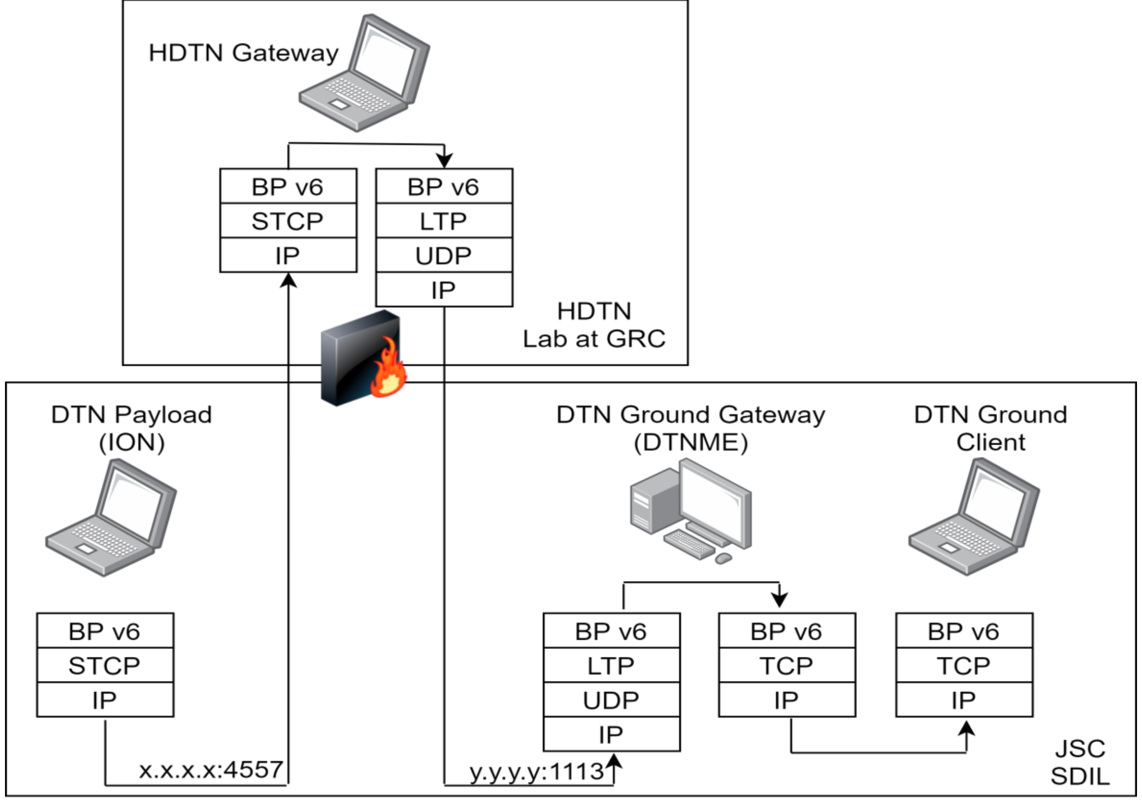


SDIL Interoperability Testing

- GRC tested HDTN with the Software Development and Integration Laboratory (SDIL) at the Lyndon B. Johnson Space Center (JSC)
- SDIL provides a realistic emulation of the network conditions on the ISS.
- Successful SDIL testing is a required step for any software to be deployed on the ISS.
- SDIL uses Wide Area Network (WAN) emulator to introduce losses on both the uplink and downlink as well as a fixed transmission delay (600 ms) to recreate network conditions on the ISS.

HDTN Laboratory at Glenn Research Center

SDIL Testing



- Bidirectional ping successful from ION node (Space) to DTNME node (Ground) via HDTN node (Gateway)
- Downlink (From Space to Ground): Large files transfer was tested
  - Successfully sent 1GB file with a data rate of up to 10 MBits/s.
- Uplink (From Ground to Space): Only a small file was sent.
- We believed low rates might be due to:
  - DTNME older version on the ground not including the capability to use LTP to connect to multiple remote clients.
  - VPN connection between GRC and JSC

Reset | Download History

Refresh Interval: 1s

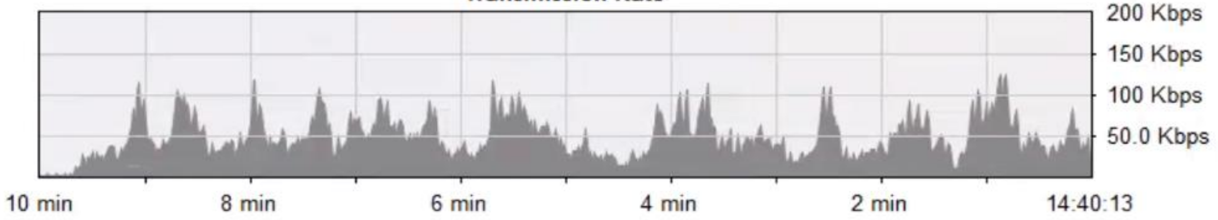
LAN A → LAN B

Rate: 1.000 Gbps **Delay:** 300 ms **Loss:** 1% **BER:** 0 **Traffic:** 0% **Queue:** 305 ms **Duplication:** 0% **Reordering:** 0%

	Transmission Count		Drops					Duplication		Reordering	
	Bytes	Frames	Loss	BER	Queue	Total	Percent	Frames	Percent	Frames	Percent
Forwarded	2,037,633,539	7,889,620	36	0	0	36	0.00045%	0	0.00%	0	0.00%
Background	0	0	0	0	0	0	0.00%	0	0.00%	0	0.00%

Queue		
Bytes	Frames	
0	0	Current
0	0	10 sec

Transmission Rate



Average Tx Rate

1 sec	21.28 Kbps
10 sec	42.99 Kbps
1 min	55.93 Kbps
10 min	48.10 Kbps

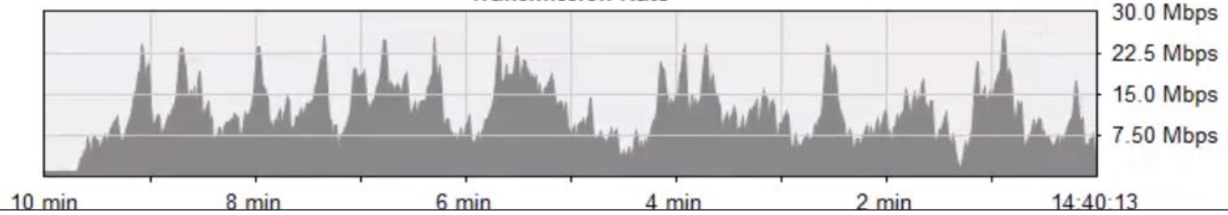
LAN B → LAN A

Rate: 1.000 Gbps **Delay:** 300 ms **Loss:** 1% **BER:** 0 **Traffic:** 0% **Queue:** 305 ms **Duplication:** 0% **Reordering:** 0%

	Transmission Count		Drops					Duplication		Reordering	
	Bytes	Frames	Loss	BER	Queue	Total	Percent	Frames	Percent	Frames	Percent
Forwarded	2,692,004,889,080	2,889,697,306	1,684	0	0	1,684	0.00005%	0	0.00%	0	0.00%
Background	0	0	0	0	0	0	0.00%	0	0.00%	0	0.00%

Queue		
Bytes	Frames	
0	0	Current
0	0	10 sec

Transmission Rate



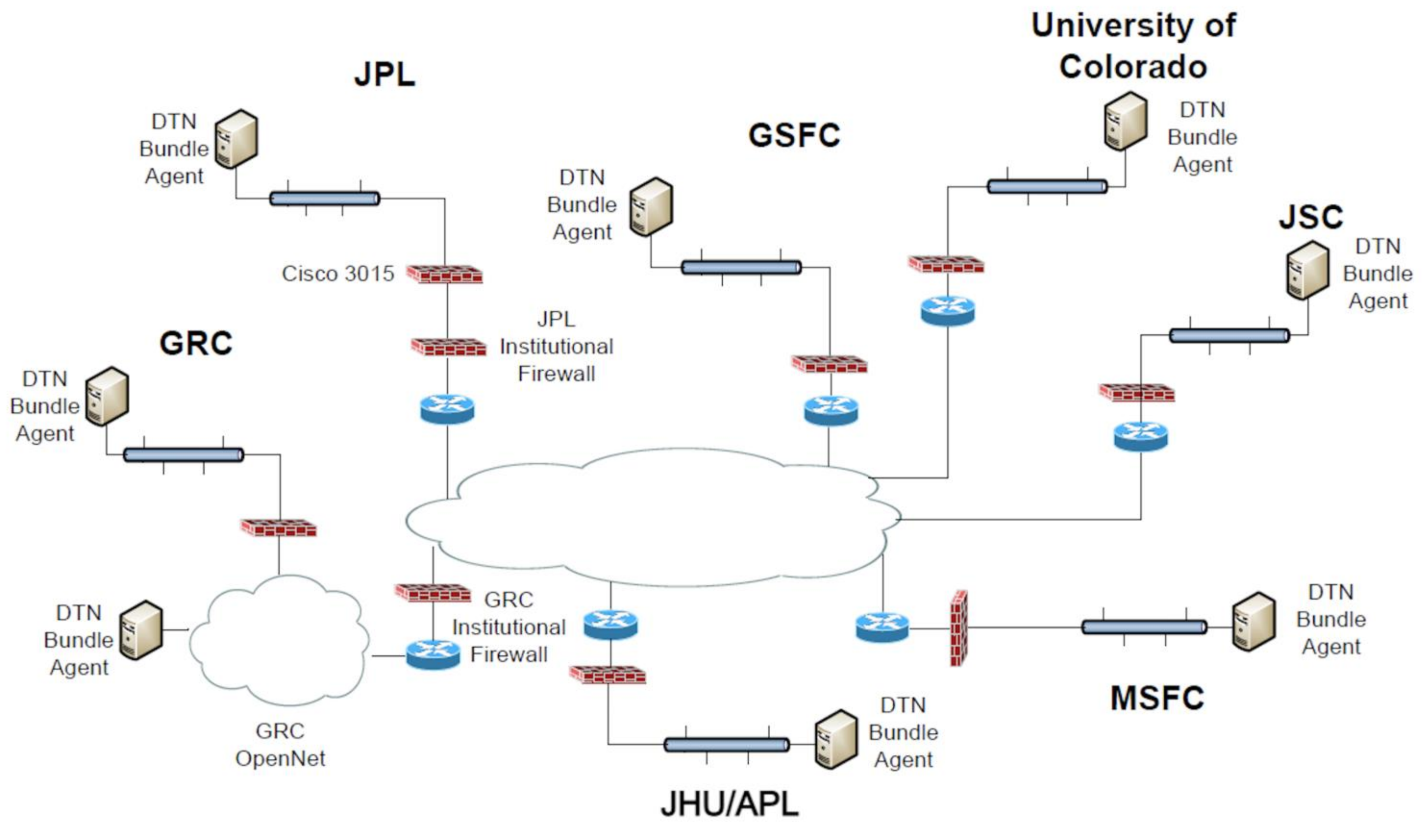
Average Tx Rate

1 sec	3.633 Mbps
10 sec	7.204 Mbps
1 min	10.78 Mbps
10 min	11.53 Mbps

DTNME Interoperability Testing

- Interoperability testing for both connectivity and data rates was completed between a Glenn Research Center (GRC) High-rate Delay Tolerant Networking (HDTN) node and a Marshall Space Flight Center (MSFC) Delay Tolerant Networking Marshall Enterprise (DTNME) node.
- Testing was conducted virtually between GRC and MSFC over the DTN Experimental Network (DEN).

Testing with DTNME



- ICMP Ping latency ~ 21ms without introducing any delay.
- Bidirectional Bundle Protocol Ping was tested for different configurations and convergence layers using both BPv6 and BPv7.
- Delays of 200ms and 700ms were added using the Linux kernel netimpair CLI to simulate bad network conditions and check how that impacts the latency for the different convergence layers.

Connectivity testing results BPv7

Convergence Layer	Emulated delay (ms)	Latency (ms) for DTNME to HDTN
LTP	0 200 700	24 to 33 223 to 231 724 to 732
TCPClv3	0 200 700	21 222 1277 to 1962
TCPCLv4	0 200 700	21 to 81 221 922 to 2124
TCPCLv4/TLS	0 200 700	21 to 81 222 896 to 2139
STCP	0 200 700	21 221 722
UDP	0 200 700	21 221 722

Connectivity testing results BPv6

Convergence Layer	Emulated delay (ms)	Latency (ms) for DTNME to HDTN
LTP	0 200 700	22 to 31 222 to 230 723 to 732
TCPClv3	0 200 700	21 222 1100 to 1800
STCP	0 200 700	21 222 722
UDP	0 200 700	21 222 722

- The connectivity testing results show that the recorded round-trip latency was consistent with the standard ping time with and without delay for all convergence layers
- The performance of BPv7 was comparable to BPv6 which shows that there was no delay introduced by version 7 update.

Data rates testing using LTP

Emulated delay (s)	Data rate (Mbps) with max 400 LTP sessions	Data rate (Mbps) with max 5000 LTP sessions
0	500 (+/- 100)	600
2	400 (+/- 100)	NA
4	400 (+/- 100)	NA

- Data rate tests were conducted by sending bundles at high rates from HDTN to DTNME using LTP with a Bundle Size of 100000 and LTP MTU 1360.
- 2 seconds and 4 seconds link delays were introduced (on the inbound and on the outbound) to mimic the delay on LCRD. We used HDTN UDP delay proxy to simulate the delay.
- We reached up to 600 Mbps data rates without delay and up to 500 Mbps with the delay.
- The DTNME server only has one core available, and that core is maxed up.

Config Tuning for high data rates


```
{
  "hdtnConfigName": "my hdtn config",
  "userInterfaceOn": "true",
  "mySchemeName": "unused_scheme_name",
  "myNodeId": 10,
  "myBpEchoServiceId": 2047,
  "myCustodialSsp": "unused_custodial_ssp",
  "myCustodialServiceId": 0,
  "isAcsAware": true,
  "acsMaxFillsPerAcsPacket": 100,
  "acsSendPeriodMilliseconds": 1000,
  "retransmitBundleAfterNoCustodySignalMilliseconds": 10000,
  "maxBundleSizeBytes": 10000000,
  "maxIngressBundleWaitOnEgressMilliseconds": 2000,
  "maxLtpReceiveUdpPacketSizeBytes": 65536,
  "zmqIngressAddress": "localhost",
  "zmqEgressAddress": "localhost",
  "zmqStorageAddress": "localhost",
  "zmqRegistrationServerAddress": "localhost",
  "zmqSchedulerAddress": "localhost",
  "zmqRouterAddress": "localhost",
  "zmqContactGraphRoutingAddress": "tcp://localhost:5556",
  "zmqBoundIngressToConnectingEgressPortPath": 10100,
  "zmqConnectingEgressToBoundIngressPortPath": 10160,
  "zmqConnectingEgressBundlesOnlyToBoundIngressPortPath": 10161,
  "zmqBoundIngressToConnectingStoragePortPath": 10110,
  "zmqConnectingStorageToBoundIngressPortPath": 10150,
  "zmqConnectingStorageToBoundEgressPortPath": 10120,
  "zmqBoundEgressToConnectingStoragePortPath": 10130,
  "zmqRegistrationServerPortPath": 10140,
  "zmqBoundSchedulerPubSubPortPath": 10200,
  "zmqBoundRouterPubSubPortPath": 10210,
  "zmqMaxMessagesPerPath": 5000,
  "zmqMaxMessageSizeBytes": 100000000,
  "inductsConfig": {
    "inductConfigName": "myconfig",
    "inductVector": [
      {
        "name": "i1 from DTNME",
        "convergenceLayer": "ltp_over_udp",
        "myEndpointId": "ipn:10.1",
        "boundPort": 4556,
        "numRxCircularBufferElements": 10000,
        "thisLtpEngineId": 10,
        "remoteLtpEngineId": 32001,
        "ltpReportSegmentMtu": 1360,
        "oneWayLightTimeMs": 200,
        "oneWayMarginTimeMs": 200,
        "clientServiceId": 1,
        "preallocatedRedDataBytes": 200000000,
        "ltpMaxRetriesPerSerialNumber": 50,
        "ltpRandomNumberSizeBits": 32,
        "ltpRemoteUdpHostname":[redacted]
        "ltpRemoteUdpPort": 4556,
        "ltpRxDataSegmentSessionNumberRecreationPreventerHistorySize": 0,
    "ltpMaxExpectedSimultaneousSessions": 500
      }
    ]
  },
```



```
  "outductsConfig": {
    "outductConfigName": "myconfig",
    "outductVector": [
      {
        "name": "for egress DTNME",
        "convergenceLayer": "ltp_over_udp",
        "nextHopEndpointId": "ipn:3.1",
        "remoteHostname": "GroundNode",
        "remotePort": 4556,
        "bundlePipelineLimit": 6000,
        "finalDestinationEidUris": [
          "ipn:[redacted]
          "ipn:[redacted]",
          "ipn:[redacted]
        ],
        "thisLtpEngineId": 10,
        "remoteLtpEngineId": 32001,
        "ltpDataSegmentMtu": 1360,
        "oneWayLightTimeMs": 2000,
        "oneWayMarginTimeMs": 200,
        "clientServiceId": 1,
        "numRxCircularBufferElements": 10000,
        "ltpMaxRetriesPerSerialNumber": 500,
        "ltpCheckpointEveryNthDataSegment": 0,
        "ltpRandomNumberSizeBits": 32,
        "ltpSenderBoundPort": 4556,
       "ltpMaxSendRateBitsPerSecOrZeroToDisable": 0
      },
      {
      "name": "bpsink",
      "convergenceLayer": "stcp",
      "nextHopEndpointId": "ipn:2.1",
      "remoteHostname": "localhost",
        "remotePort": 4557,
          "bundlePipelineLimit": 5,
          "finalDestinationEidUris": [
          "ipn:2.1"
        ],
        "keepAliveIntervalSeconds": 17
      }
    ]
  },
```

Long-Delay Testing

- Current estimates suggest ISS to ground may experience significant delays over ILLUMA-T/LCRD
  - One-way: 1-2 seconds
  - Round trip: 2-4 seconds
- This may be challenging even for specialized convergence layers such as LTP

UDP Delay Proxy

- HDTN has developed a UDP delay proxy to simulate seconds long delays at gigabit rates
- Testing with netem and Linux Traffic Control (TC) was useful for packet loss
- Conventional tools did not perform well for delays greater than 1 second
  - Possibly due to large amounts of data being queued
- HDTN implementation uses Boost.Asio C++ library
- CPU usage is low, but can require significant amounts of memory
- Circular buffer memory usage is approximately:
  - *data rate* x *one-way delay*

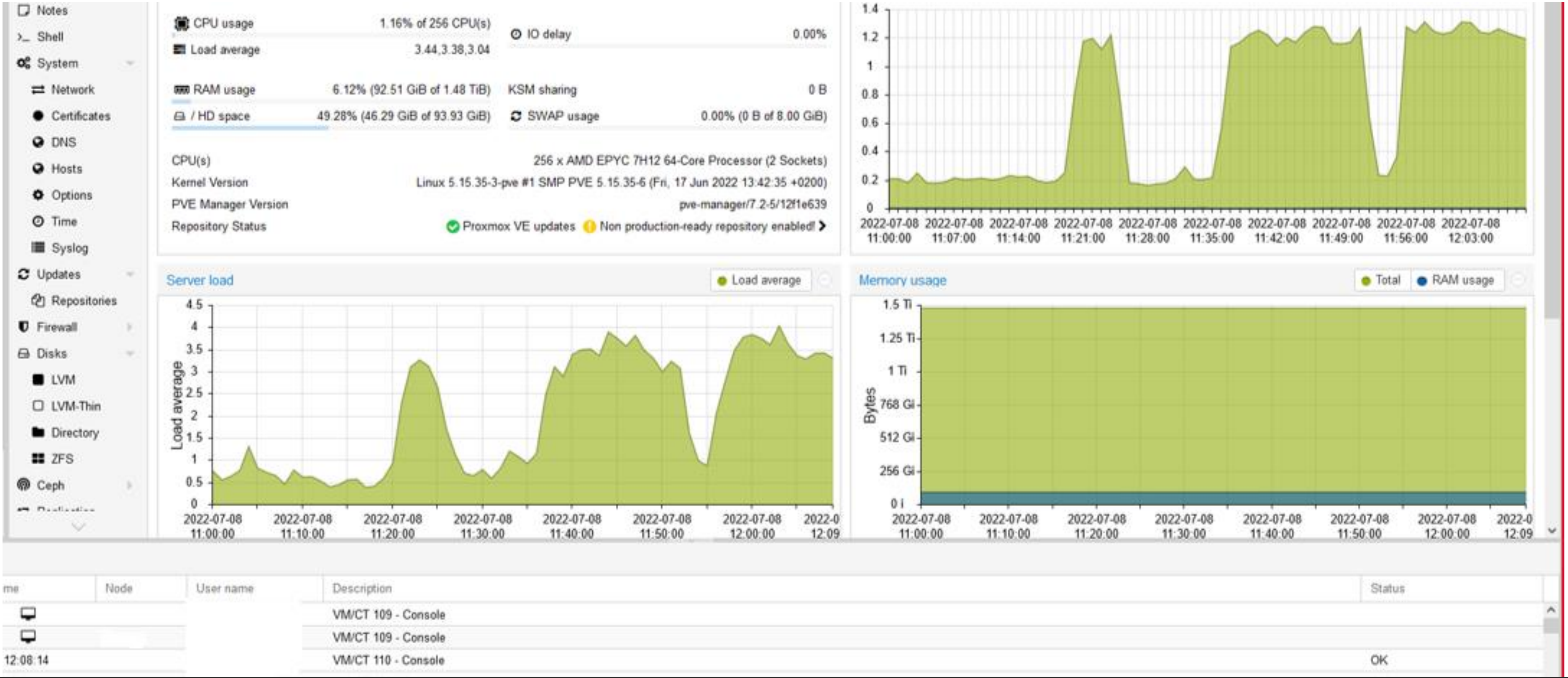
LTP Long Delay Testing

Bundle Size	Raw rate Mbit/s	RptSeg rate Mbit/s	Sender CPU	Receiver CPU	Loss DS	Bundles/ sec	DS UDP pack/sec	LTP simultaneous sessions
100KB	998	0.39	100%	70%	0%	1250	92000	5000
50KB	999	0.85	105%	92%	0.047%	2500	103000	10000
10KB	994	3.68	150%	140%	0.267%	12000	112000	48000
5KB	824	12	200%	200%	70%	21000	143000	84000

- The HDTN LTP implementation was recently optimized for significant round trip times
- Testing was conducted on lab servers with UDP delay proxy simulating a 4 second round trip time
- Testing with HDTN sender and receiver show HDTN can reach up to 1 Gbps rates even with 4 second round trip time
- The key to achieving high rates was to allow tens of thousands of simultaneous sessions

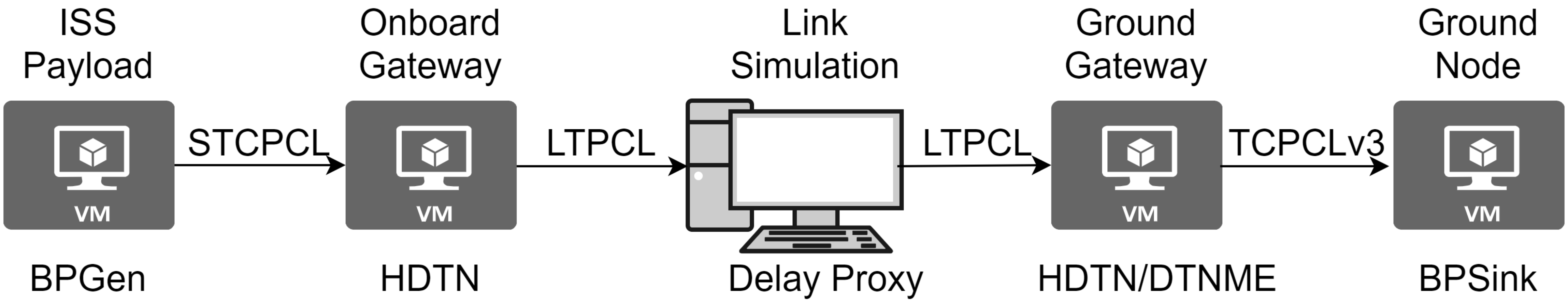
Virtualization

- The ISS hosts several DTN payloads as well as the DTN onboard gateway on a ZBook Laptop
- The payloads and gateway each run on their own Kernel-based Virtual Machine
- HDTN seeks to reach rates of 1 Gbps in this environment



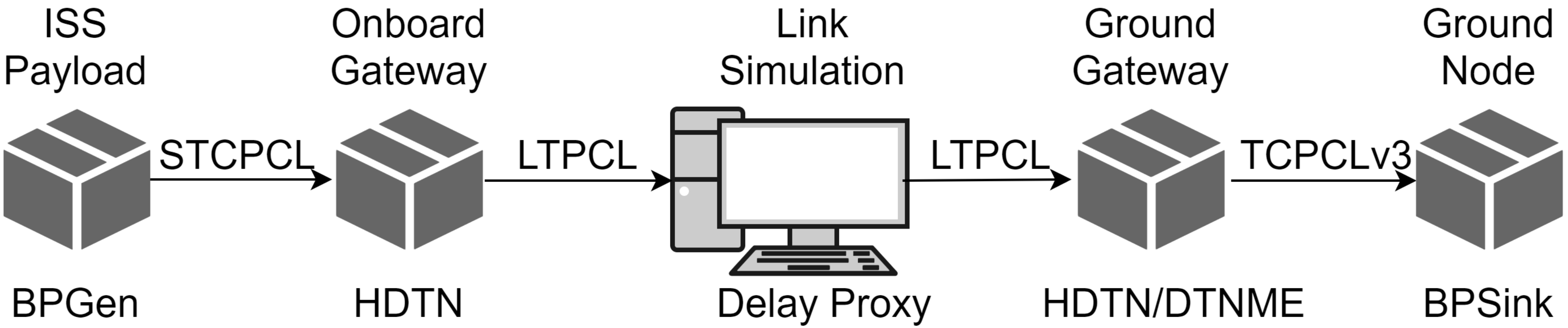
ProxMox Virtual Environment

GRC Local Testing – Virtual Machines



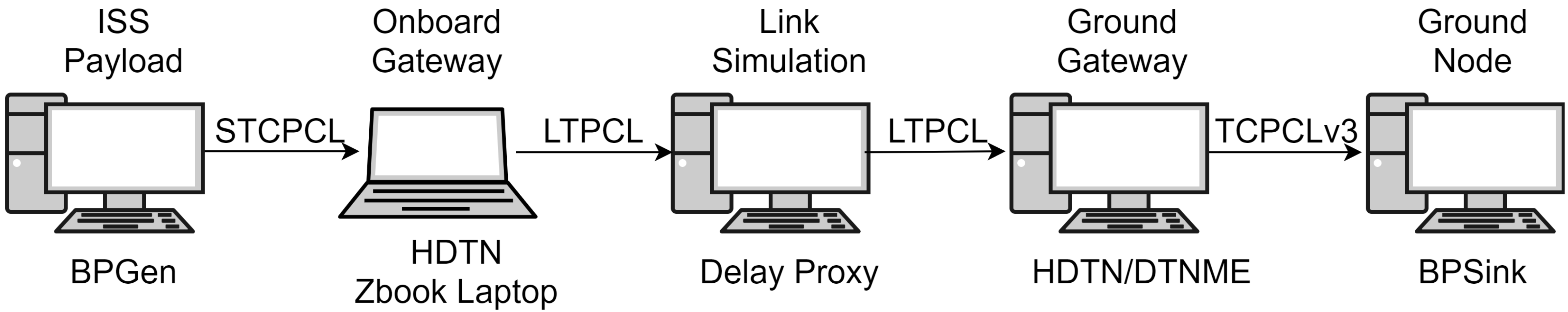
- Initial testing conducted on GRC's virtualization server
  - 256 x AMD EPYC 7H12 64-Core Processor (2 Sockets)
  - 1.48 TiB RAM
- Each node is a Kernel-based Virtual Machine (KVM)
  - 16 cores based on host architecture
  - 32 GB RAM
  - Network device is virtio network bridge
  - ISS payload and ground node are Oracle Linux 8 to support Telescience Resource Kit (TReK)
  - Onboard gateway and ground gateway are Ubuntu 20.04.4

GRC Local Testing – Linux Containers

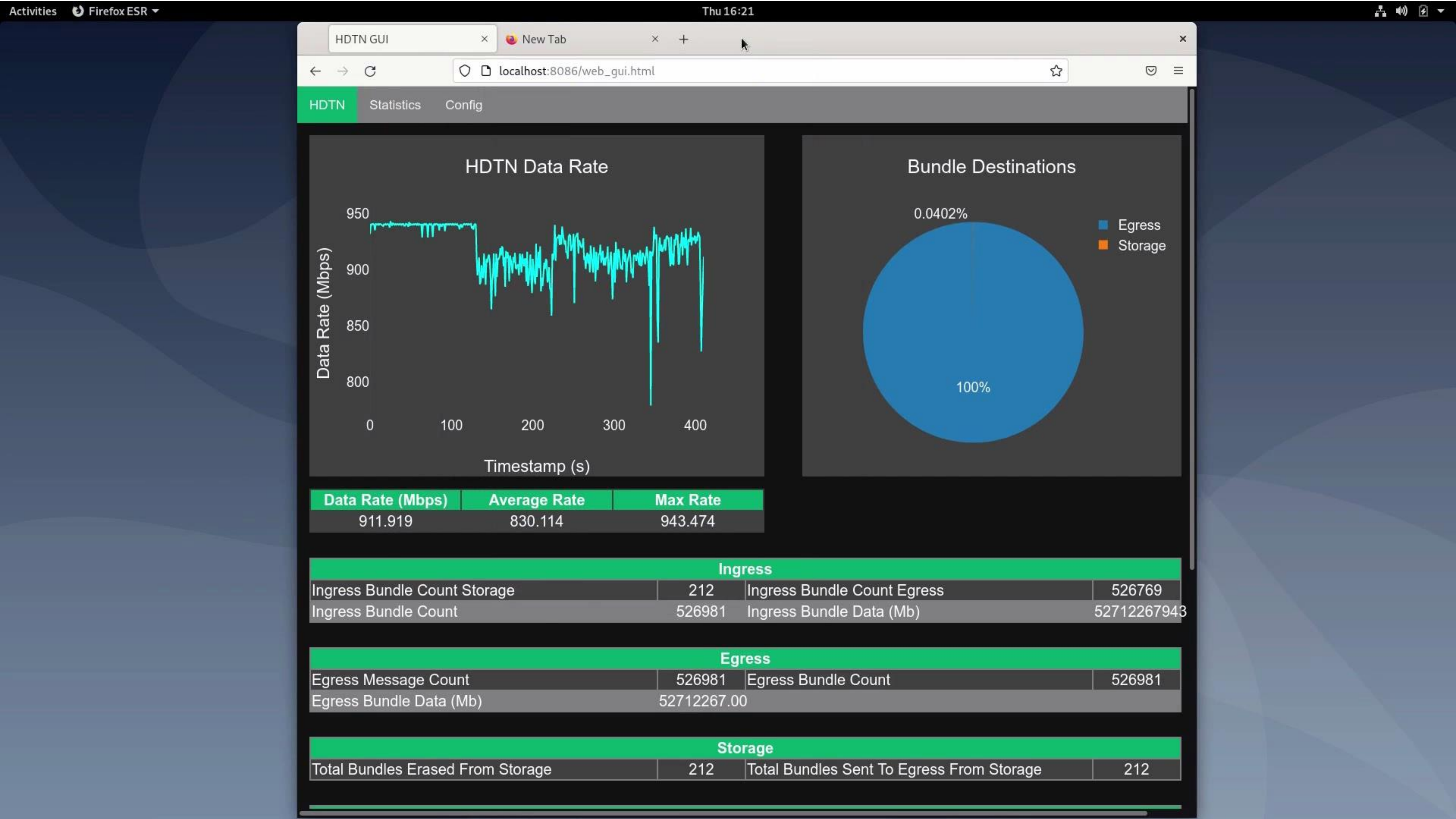


- LXC testing also conducted on GRC’s virtualization server
  - 256 x AMD EPYC 7H12 64-Core Processor (2 Sockets)
  - 1.48 TiB RAM
- Each node is a Linux Container (LXC)
  - 8 cores based on host architecture
  - 32 GB of RAM
  - Debian GNU/Linux 11 (bullseye)

GRC Local Testing – Lab Servers



- The ISS payload, ground gateway, and ground node were running on:
  - Debian 10
  - 32 GB RAM
  - 8 x Intel Core i7-7700K CPU @ 4.2 GHZ
- The onboard gateway is running on an HP ZBook laptop identical to what is used on the ISS
  - Debian Linux 10
  - 32 GB RAM
  - 8 x Intel Core i7-7700HQ CPU @ 2.8 GHZ


Activities
Firefox ESR
Thu 16:21
HDTN GUI
New Tab
localhost:8086/web_gui.html
HDTN
Statistics
Config
HDTN Data Rate
950
900
850
800
Data Rate (Mbps)
0
100
200
300
400
Timestamp (s)
Bundle Destinations
0.0402%
Egress
Storage
100%
Data Rate (Mbps)
Average Rate
Max Rate
911.919
830.114
943.474
Ingress
Ingress Bundle Count Storage
212
Ingress Bundle Count Egress
526769
Ingress Bundle Count
526981
Ingress Bundle Data (Mb)
52712267943
Egress
Egress Message Count
526981
Egress Bundle Count
526981
Egress Bundle Data (Mb)
52712267.00
Storage
Total Bundles Erased From Storage
212
Total Bundles Sent To Egress From Storage
212

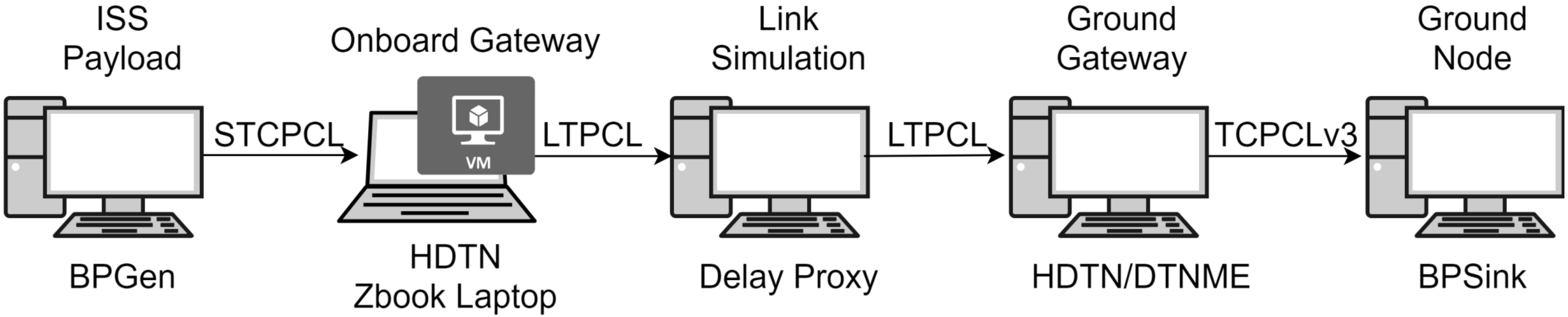
HDTN/DTNME Ground Gateway Initial Test

Average Data Rates	KVM	LXC	Lab Servers
Iperf3 UDP	1.5 Gbps	1.5 Gbps	945 Mbps
Iperf3 TCP	16.4 Gbps	18.4 Gbps	661 Mbps
HDTN	260 Mbps	~900 Mbps	930 Mbps
DTNME **	~240 Mbps	~430 Mbps	930 Mbps

- Initial testing was done with HDTN as the onboard gateway as well as ground gateway
- Delay proxy was set to 0 ms (no delay)
- While Iperf3 shows the KVM nodes are capable of 1.5 Gbps over UDP, performance is poor running HDTN
- The onboard gateway node processing and memory remains low
- On LXC and servers, HDTN performance was as expected (~900 Mbps)
- The same test was performed
  - With and without hardware acceleration enabled
  - With and without web interface enabled
- Tests were repeated with DTNME as the ground gateway
- DTNME ** results are preliminary as we are still investing proper configuration

Resource Usage for Onboard Gateway (KVM)

GRC Local Testing – ISS Onboard Gateway VM



- In this test, the setup is the same as the lab servers test but the ISS DTN Gateway runs on the ISS DTN Gateway image that is used for ISS testing
- The default settings are:
  - 2 Cores
  - 1600 MB RAM

ISS DTN Gateway VM

Average Data Rates	ISS DTN Gateway VM	Lab Servers
Iperf3 UDP	515 Mbps	945 Mbps
Iperf3 TCP	935 Mbps	661 Mbps
BPSink Receiver	~ 25 Mbps	930 Mbps

- Initial config = 2 CPUS, 1600 MB RAM
- Next tests were increasing the RAM and CPU allocation for the VM

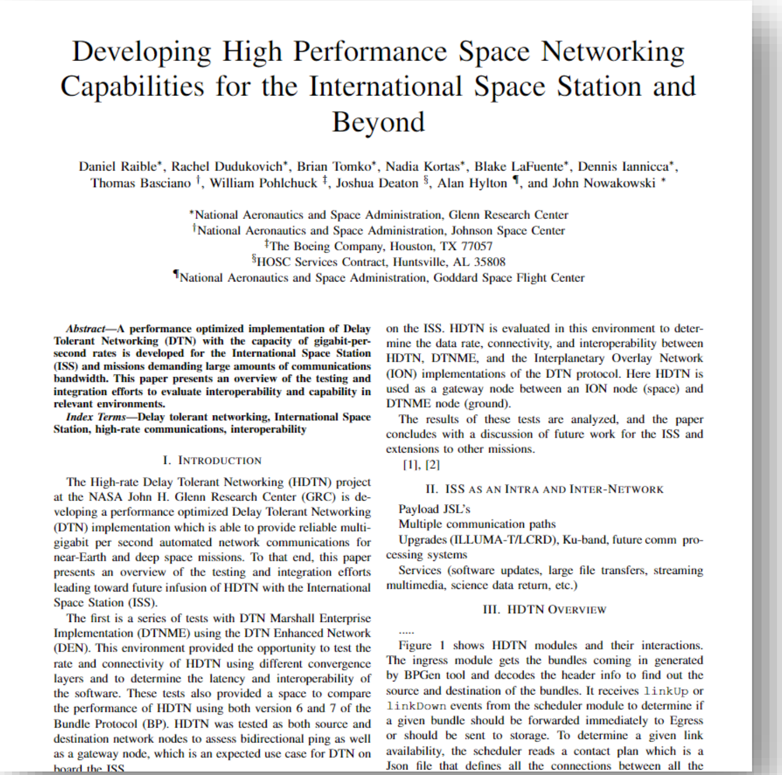
CPUs	RAM	BPSink Rate
2	1600	25 Mbps
2	8192	25 Mbps
4	8192	26 Mbps
6	16384	26 Mbps

Conclusions

- We have shown interoperability with DTNME
  - Test with custody transfer
  - TCPCL versus UDP
- Gigabit rates are possible in 4-hop ISS configuration with HDTN and DTNME
- Our local testing indicates the low rates experienced during the SDIL remote test is likely due to KVM
- We will explore optimization and tuning of the KVM virtualization
- The next step of interoperability testing will be with the Telescience Resource Kit
  - CFDP and Ionizer tools
  - Determine baseline rates for CFDP/ION versus HDTN BP file transfer tools
- Deliver final config files to ISS DTN team for testing

Further Information on High-rate Delay Tolerant Networking (HDTN)

Forthcoming corresponding publication:



Developing High Performance Space Networking Capabilities for the International Space Station and Beyond

Daniel Raible*, Rachel Dudukovich*, Brian Tomko*, Nadia Kortas*, Blake LaFuente*, Dennis Iannicca*, Thomas Basciano †, William Pohlchuck ‡, Joshua Deaton §, Alan Hylton ¶, and John Nowakowski *

*National Aeronautics and Space Administration, Glenn Research Center
†National Aeronautics and Space Administration, Johnson Space Center
‡The Boeing Company, Houston, TX 77057
§HOSC Services Contract, Huntsville, AL 35808
¶National Aeronautics and Space Administration, Goddard Space Flight Center

***Abstract*—A performance optimized implementation of Delay Tolerant Networking (DTN) with the capacity of gigabit-per-second rates is developed for the International Space Station (ISS) and missions demanding large amounts of communications bandwidth. This paper presents an overview of the testing and integration efforts to evaluate interoperability and capability in relevant environments.**

***Index Terms*—Delay tolerant networking, International Space Station, high-rate communications, interoperability**

I. Introduction

The High-rate Delay Tolerant Networking (HDTN) project at the NASA John H. Glenn Research Center (GRC) is developing a performance optimized Delay Tolerant Networking (DTN) implementation which is able to provide reliable multi-gigabit per second automated network communications for near-Earth and deep space missions. To that end, this paper presents an overview of the testing and integration efforts leading toward future infusion of HDTN with the International Space Station (ISS).

The first is a series of tests with DTN Marshall Enterprise Implementation (DTNME) using the DTN Enhanced Network (DEN). This environment provided the opportunity to test the rate and connectivity of HDTN using different convergence layers and to determine the latency and interoperability of the software. These tests also provided a space to compare the performance of HDTN using both version 6 and 7 of the Bundle Protocol (BP). HDTN was tested as both source and destination network nodes to assess bidirectional ping as well as a gateway node, which is an expected use case for DTN on board the ISS. HDTN is evaluated in this environment to determine the data rate, connectivity, and interoperability between HDTN, DTNME, and the Interplanetary Overlay Network (ION) implementations of the DTN protocol. Here HDTN is used as a gateway node between an ION node (space) and DTNME node (ground).

The results of these tests are analyzed, and the paper concludes with a discussion of future work for the ISS and extensions to other missions.

[1], [2]

II. ISS as an Intra and Inter-Network

Payload JSL's

Multiple communication paths

Upgrades (ILLUMA-T/LCRD), Ku-band, future comm processing systems

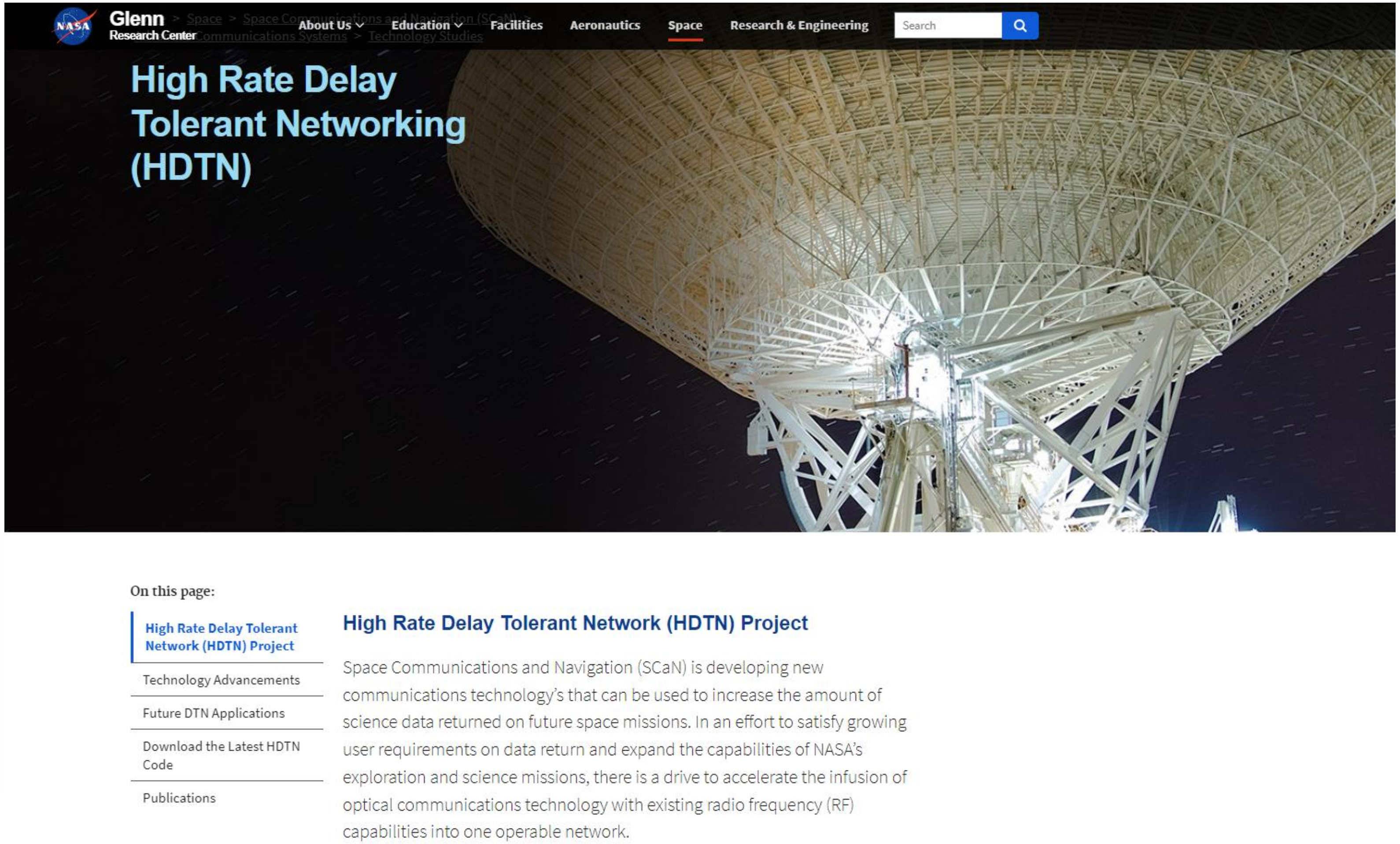
Services (software updates, large file transfers, streaming multimedia, science data return, etc.)

III. HDTN Overview

.....

Figure 1 shows HDTN modules and their interactions. The ingress module gets the bundles coming in generated by BPGen tool and decodes the header info to find out the source and destination of the bundles. It receives `linkUp` or `linkDown` events from the scheduler module to determine if a given bundle should be forwarded immediately to Egress or should be sent to storage. To determine a given link availability, the scheduler reads a contact plan which is a Json file that defines all the connections between all the

Website with code links and publications:
https://www1.grc.nasa.gov/space/scan/acs/tech-studies/dtn/



Thank you

- Code is publicly available at:

https://github.com/nasa/HDTN