A Delay Tolerant Networking-based Approach to a High Data Rate Architecture for Spacecraft

Alan Hylton
NASA Glenn Research Center
21000 Brookpark Rd.
Cleveland, OH 44140
alan.g.hylton@nasa.gov

Daniel Raible
NASA Glenn Research Center
21000 Brookpark Rd.
Cleveland, OH 44140
daniel.e.raible@nasa.gov

Gilbert Clark
NASA Glenn Research Center
21000 Brookpark Rd.
Cleveland, OH 44140
gilbert.j.clark@nasa.gov

***Abstract*—Historically, it has been the case that SWaP placed such severe constraints on radios that the links between spacecraft and the ground were relatively slow. This meant that the radio link was normally a significant bottleneck in returning scientific data. Over recent years, however, a combination of more efficient radio design, intelligent waveforms, and highly directed, high-frequency RF / optical systems have led to a rapid increase in the amount of data that can be pushed through radio and optical links. This has led to some cases where the radio links are capable of moving data much more quickly than the spacecraft and instruments are capable of actually generating it! In some instances, scientific data can therefore be lost not because the downlink is too slow to support the data rate, but instead because the spacecraft was not designed in a way that would let it fully utilize both the radio and the networking services available to it.**

The High Data Rate Architecture (HiDRA) project describes a packet-based approach to building modern, distributed spacecraft systems. It presents a means for spacecraft and other assets to participate in both present and future Delay Tolerant Networks (DTN), while simultaneously ensuring that the asset is able to fully utilize the new, high-speed links that have been seeing more widespread development and deployment in recent years. With this in mind, this paper begins with a discussion regarding HiDRA's evolution. Next, it discusses the capabilities and limitations of NASA's present DTN-enabled networks. Of particular note is the way in which principles of network design at the terrestrial level (e.g. use of programmable networks / software-defined networks, separation between data and control plane, infusion of COTS Ethernet switch chips, etc.) can all be translated into the space environment as well. After this, the paper discusses the design and implementation of a present prototype reference implementation of High-Rate DTN (HDTN), which is intended to demonstrate future high-rate networking concepts as part of a coherent demonstration on the International Space Station (ISS). The goal, of both the research and of this implementation, is to help develop a ready-made toolbox of ideas, approaches, and examples from which mission designers can draw when putting together new missions. Assuming all goes as planned, this should not only work to reduce the cost of individual mission design, but also improve the rate at which science data can be returned for mission participants to review.

TABLE OF CONTENTS

1. INTRODUCTION

Historically, the volume of scientific data returned from space missions has been largely constrained by the capabilities of the mission's communication system. Such communication systems are generally constrained by three things: size, weight, and power (SWaP). As technology has improved, however, there have been incremental (albeit marked) improvements to each of the individual S, W, and P areas of radio performance. For example, amplifier and front-end design has advanced to allow one to run radios at higher power without increasing the associated size, weight, and power (SWaP) of the system. Indeed, over time radio frequency (RF) based systems have evolved to higher frequency carrier waves allowing increased bandwidth, travelling wave tube amplifiers (TWTA) have advanced to greater powers and efficiencies, and antenna technology has been pushed to larger and deployable apertures to offer an increase in signal gain. Improved antenna design and efficiency has led to a significant increase in signal gain. Improvements in computational architecture have led to the development of superior waveforms while simultaneously offering remarkable new platforms and approaches to radio design (e.g. software-defined radio, or SDR). At the link level, SDRs are now being employed to continually poll and reconfigure link parameters to operate at a point of maximum optimality.

Unfortunately, no incremental improvement to a radio system can eliminate the fundamental requirement that, in order to improve the performance of a link, one must eventually give it more bandwidth. While it is possible to replace a waveform on a spacecraft with something that can pack more data into an equivalent number of symbols, the trade-off this process makes is that it makes the radio link more fragile: the larger signal to noise ratio (SNR) requirement means (significantly) more power is required to close the link. Additionally, since signals propagate well through space, high-power RF systems can blanket incredibly large areas with the signals they produce: as such, careful coordination is necessary to ensure that one craft's spectrum profile (center frequency, bandwidth, power, etc) either does not put it in conflict with that of another craft. Alternatively, in the event that crafts do share a spectrum profile, care must be taken to ensure that all the craft involved can treat the spectrum as a common physical medium (e.g. through common multiple access schemes). Given the limitations both on bandwidth and on directionality with lower-frequency RF systems, higher frequency alternatives have been the subject of significant research and development over many years.

One example of such a system is the free-space optical system. Free-space optics are at a reasonably advanced stage of development, and have seen a number of successful deployments. One example of such a deployment was the Lunar Laser Communications Demo (LLCD) [1]. LLCD was successful in

demonstrating laser communications between the moon and the Earth at a rate of 622 Mbps, but its deployment revealed a fundamental issue with such high-frequency systems: the large rates involved meant that the radio was no longer the bottleneck in data return! In the case of LLCD, while the laser payload could transmit data at 622Mbps, the connection to the host spacecraft, the Lunar Atmosphere and Dust Environment Explorer (LADEE), was constrained to 40Mbps. Thus, while this demonstration illustrated that free-space optical could be a viable technology, it also demonstrated a fundamental issue with the deployment of such high-frequency systems in the general case: it is likely that spacecraft would lack the bus and processing capabilities necessary to source data at such extremely high transmission rates. Thus, rather than the radio link acting as the bottleneck for scientific data return, the bottleneck had become the spacecraft itself.

Another case study to consider is with the International Space Station (ISS), whose infrastructure has been and will continue to be a constantly evolving process. From a data source perspective the performance of imaging and remote sensing technologies have also advanced, and in fact in certain scenarios have outpaced standard practice communication systems [2][3]. The ISS has enjoyed several RF systems upgrades, and is currently adding laser communication capability. These state of the art technologies will help to realize higher data rates, but must also interface with the existing and aging data infrastructure, which introduces considerable rate mismatches between the existing and emerging capabilities. The ISS additionally presents an interesting challenge in that it hosts a large network of experiments and sensors (data sources), which must be routed to a variety of communication systems. In this way it is a large multiple-in-multiple-out (MIMO) system whose traffic must be carefully managed to successfully route data to the correct destination at a certain quality of service (QoS).

The next generation of space communication technologies is currently being prepared for deployment, such as with the upcoming NASA Laser Communications Relay Demonstration (LCRD). LCRD will place a laser relay terminal in geostationary orbit (GEO) over the continental United States [4]. The system will support several data rates up to the Gbps regime, and feature the capability to optically link a user satellite in Low-Earth Orbit (LEO) with a terminal on the ground. An important aspect of the demonstration is the Integrated LCRD LEO User Modem and Amplifier Terminal (ILLUMA-T), which will be installed on the ISS in 2020 where it will serve as the user terminal for the overall demonstration, and also provide an alternative high speed path for moving data to and from the ISS [5][6]. This capability will add to the MIMO architecture of the ISS, which will require additional network storage and management to utilize to its full capacity. However, the system-wide issue discussed with LLCD, the ISS, and LCRD remain.

To address this issue, the NASA John H. Glenn Research Center is conducting the High Data-Rate Architecture (HiDRA) project. HiDRA was commissioned to examine the current bottlenecks and rate asymmetries in space data and communications systems. The goal was to allow mission designers to, if they should choose, integrate elements of the high data rate architecture into their own missions, allowing them to more effectively utilize the high-rate, high-frequency links of the future. Attaining this goal has involved a careful blend of evangelism, infusion planning, lab testing, and commercial development that represents an extremely active body of work. However, the work HiDRA has taken on is not limited to the fundamental rate mismatch between legacy spacecraft bus systems and the capacity of future high-rate links.

Although the hardware and software particulars of the HiDRA implementation will be application dependent, the relationship between delay tolerant network (DTN) as a protocol (introduced in Section 2), and the architecture across the CPU, FPGA, memory and I/O elements will remain constant. The essence of HiDRA is to create a networked buffering solution which is general enough to provide mission scalability, with the particular reconfigurability to enable required performance in any particular scenario. Performance goals include target rates of 200Gbps in support of future optical communications payloads. The current status of the HIDRA effort is presented here as a reference architecture under development, with current work leading towards a technology demonstration aboard the ISS.

Packetized Data and On-Board Switching

A fundamental design principle of HiDRA is that the architecture must include support for both present and future networking technologies in order to be successful in its realization of NASA's vision of a solar system internet. Along these lines, one core assumption that the High Data Rate Architecture makes is that spacecraft work with data that has been demodulated and packetized. This is somewhat different than the approach to signal processing commonly adopted by existing spacecraft, where signals are considered (and routed) signals solely in terms of their RF components. For example, in a (notional) present-day relay, an RF signal would come in a receiving antenna, pass through a series of RF switches, filters, amplifiers, etc, and pass out a transmitting antenna. A relay, as illustrate in Figure 1, would neither demodulate nor actively process the data.

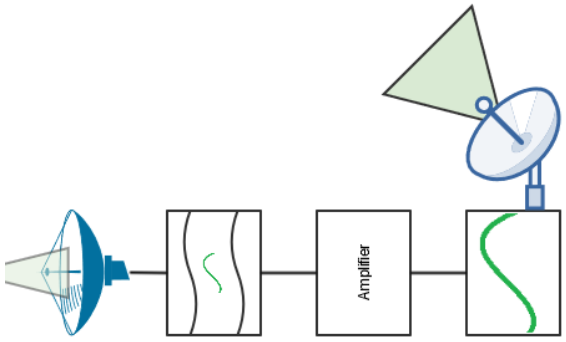

Figure 1. Basic traditional relay architecture

HiDRA, on the other hand, operates on packets as its atom. Data enters a notional HiDRA system through a high-speed modem, where the data is demodulated and transformed into a packet stream. The packet stream is expected to pass through a programmable, line-rate ASIC, which switches the traffic in much the same way that a modern-day, terrestrial switch or router would. In that vein, some HiDRA prototypes to date have been built entirely on the use of COTS switching and routing chipsets: one present area of active work involves integrating COTS switch platforms (e.g. Broadcom's Tomahawk, Barefoot's Tofino, etc.) into coherent, high-rate platforms that might be capable of operating in space [7].

While such an effort would both reduce the cost and improve the capability of modern space systems, there are a few challenges associated with pursuing this path. One such challenge involves taking the existing architectures and adapting them to operate in space environments. This practice tends to be somewhat arcane, and also tends to incur significant expense due to the substantial NRE involved. While this challenge is a substantial one in the longer-term (and will need to be addressed), in the shorter-term we hope that such solutions could be infused into elements that might be less sensitive to radiation (e.g. assets in low-earth orbit, or LEO).

Another challenge that is presented when trying to adapt COTS devices to spacecraft buses involves the idea that the protocols used by space assets are often rather different than the protocols used by terrestrial devices. While modern space-centric protocol standards do include provisions to operate over, for example, Ethernet and the Internet Protocol (IP), there is some layer of glue required to make the systems talk to one another. While in many cases the glue can be relatively direct (e.g. running a Consultative Committee for Space Data Systems (CCSDS) framing layer directly over IP), significant issues can arise in cases where assumptions for space-centric networks differ from the assumptions made in Earth-centric networks.

2. Making Delay / Disruption Tolerant Networks

What is a DTN?

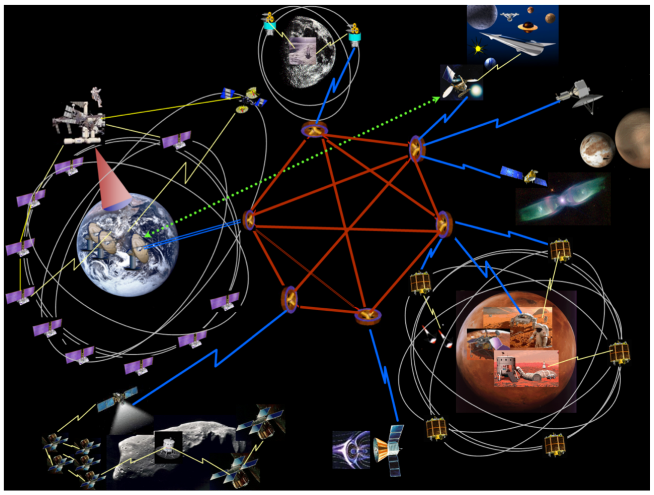

Figure 2. Typical DTN depiction

One such assumption is the idea that an end-to-end path is expected to exist between any given source and any given destination at every point in time. From a mathematical perspective, terrestrial connectivity is generally represented as a connected graph, where routing between different sources and destinations often occurs by applying path-finding algorithms to a mathematical model of the network through which data must flow.

Space systems operate differently. For example, assume that there are three assets in space: a source, a relay, and a sink. Assume that the relay is in orbit around a body, and (due to the position of other interfering elements) the relay cannot talk to the source and the sink at the same time. This means that, in order for the source to communicate with the sink, it must pass data to the relay, which in turn must store it and later forward it to the sink. From a mathematical perspective, the major difference is that the network model involves a temporal component than is not usually required when modeling a terrestrial network.

The assumptions of connectivity and latency are illustrated in Figure 2. In particular, we see the variance within the single network. There are GEO assets whose connections to the Earth are essentially static, whereas their connections to LEO assets (say, the ISS), change several times per hour. We also see latencies that admit closed-loop protocols, such as Earth-LEO, and interplanetary communications that will have one-way light times in the minutes.

Another assumption upon which terrestrial systems are based is that, even in the presence of extreme queuing / processing delay and the like, latency is relatively low. A signal can reasonably circumnavigate the planet in a few hundred milliseconds. In most cases, one can therefore expect some degree of real-time interaction between two peers who might happen to be exchanging data (e.g. a client and a server). For space systems, this may or may not be the case.

While many communication links in space systems are short enough that real-time communication (and therefore the use of e.g. the Transmission Control Protocol, or TCP) is an option, there are other links where such real-time interaction is not supported. As one example, communication with Mars has a one-way propagation delay that is a few minutes in the best case. Alternative protocols for these situations have been developed and are in use (e.g. Licklider Transport Protocol [8]).

To function given these constraints, a NASA Delay- and Disruption-Tolerant Network will store, carry, and forward data as appropriate. In other words, a DTN functions as a type of abstract architecture: it describes a pattern of implementing networks that address challenges inherent to operating in such environments [9].

For the purposes of this paper, we assert that the following assumptions apply to any implementation of a NASA-centric DTN:

- Links that exist between successive hops along the data's path (e.g. between its source and its sink) are expected to be constrained
- No end-to-end path need exist in order for applications and / or the network to function as expected
- DTN should gracefully scale to support a variety of networks, any of which may be fundamentally different in terms of both capacity and scope
  - Different links may involve extremely different properties, so no single, common transport technique exists which may apply to all links

These design objectives come with fundamental trade-offs which affect the ease with which DTN-compatible protocols can be processed at high rates. Notably, this is not intended as a criticism of the architecture: instead, it simply describes a part of the nature of the decisions that were made.

Rather than focusing exclusively on the abstract architecture to illustrate the effect that specific decisions have had, it is often easier to instead emphasize the way the architecture has been realized in a representative example. For the purposes of

this document, the representative example of a DTN we will be focusing on is the protocol suite described by the Bundle Protocol [10] and related documents that may be found below.

Scaling vs. Computational Complexity

One core tenet of DTN is that of scalability: it suggests this idea having seen the difficulties with scaling that IPv4 has encountered over the course of a number of years. With this in mind, BP mandates that every field in its primary header must be encoded in a variable-length form [11]. Such fields include the length of the bundle, the source and destination of the bundle, and even flags that relate to the way the bundle should be processed. Further, bundles are constructed of numerous sub-blocks, each of which may describe a specific aspect of the bundle (e.g. encryption, additional source information, etc). This ensures that BP may be almost infinitely extended: no matter how many nodes may come, no matter how many flags may be needed, and no matter what new capabilities may be needed, the protocol can support them without mandating that future revisions be deployed.

This approach presents a significant challenge when building a system that can process data at high-rates: one cannot access a field within a bundle without first decoding every other field that has come before. Compare the design of the Bundle Protocol to the design of IPv4 for a moment: in IPv4, the fields which are used to make routing decisions are always contained at fixed offsets within the header. In the case of BP, the primary header must be decoded in its entirety in order to know where in the header routing information may be found. Furthermore, once one knows where routing information is located, there is additional decoding overhead that comes into play. In BP, one must loop across multiple bytes and extract a value 7 bits at a time, which is not the most efficient operation to perform on modern processors though it can still be optimized to an extent, given that:

- one is able to bound the maximum length of an SDNV to the register size of the machine (64-bit in most cases), and
- an operation equivalent to SSE's `lzcnt` exists on the platform in question.

On Convergence Layers as Elephant Flows

As another example of its general design, BP assumes that no single type of transport can be utilized in all cases. As such, BP relies on the utilization of so-called Convergence Layers (or CLs) to push bundles to their eventual destinations. These CLs essentially act as tunnels, forwarding bundles from one hop to the next in the manner that best fits the link being utilized. This abstraction offers great benefits: for example, in the case of space links where propagation delay may be measured in hours, one may swap a closed-loop protocol like TCP for an open-loop protocol like LTP [8]. Unfortunately, this decision also carries consequences when considered in terms of the way modern networking hardware operates.

To understand these consequences, we offer some background in the way modern network equipment tracks network data. In most cases, network traffic can be considered to consist of a number of logical flows, or coherent pieces of data that consist of multiple packets. To easily track which pieces of data are coherent, hardware computes a kind of unique ID for that packet: in most cases for IP-based hardware, this is a 5-tuple that consists of the protocol type, the source IP, the destination IP, the source port, and the destination port. Packets that share a 5-tuple are assumed to be part of the same flow. Since flows are considered to be coherent data streams, there is a fundamental assumption that the packets associated with each flow should be delivered (and processed) in order. In cases where a single, extremely large piece of coherent data must be delivered in such a fashion, a large number of packets will share a single 5-tuple and will need to be delivered in order. Such a flow is commonly referred to as an elephant flow.

Scaling network processing capability relies on the ability to split packets across various paths. In the network case, one is able to incrementally add more capacity by splitting some percentage of flows off to new hardware. In the case of a host, the idea is similar: modern network interface cards support multiple queues, or essentially places that packets can be delivered. Different threads of execution can be added and assigned to specific hardware queues, allowing the application to achieve efficient horizontal scaling. Elephant flows, however, prevent such approaches to scaling by forcing large numbers of packets to take a specific path through the network and / or to be delivered to a single hardware queue on the host.

Unfortunately, the way convergence layers function makes them appear as individual elephant flows when routing DTN traffic through IP networks. For example, if we assume that:

- We have a convergence layer that is built on top of either UDP or TCP,
- We multiplex delivery of any number of bundles through that convergence layer, and
- The convergence layer is realized as a long-lived connection between a single DTN source node and a single DTN sink node, then

given that BP operates at a network layer higher than that of TCP/UDP and IP, the network hardware will not be able to differentiate one bundle from another. Instead, from the perspective of the network hardware, a convergence layer acts as a single, long-lived flow that moves data between the source node and the sink node. As we move more or fewer bundles through the CL, we adjust the rate of that single flow but no matter how many bundles we push through it, a CL always remains a single flow nonetheless. With this in mind, the flow that represents a CL will always trend toward an elephant flow as the bundle volume increasing through that CL increases.

Additional Notes / Experimental Results

It is worthwhile to note that the above represents a synthesis of lessons learned from various experiments to date. Most instances of past experiments have involved the use of testbeds capable of emulating varying degrees of both disruption and delay.

Experiments conducted on such configurations have helped HiDRA gain valuable operational experience with the existing implementations. During initial testing, for example, such experiments illustrated the need for significant, locally-managed memory to accommodate bundle storage during outage periods.

Additionally, such experiments have documented varying effects related to, for example, the lack of reliability checks within bundles, varying support for fragmentation, lack of interoperability between implementations, limitations on scalability that arise due to the use of a flat address space, and issues related to the lack of standardized discovery mechanisms in the system [12] [13].

While these issues vary in terms of their individual significance, it is important to note there is a difference between issues that exist at an architectural level versus issues that exist at the implementation level. This document focuses only on a subset of elements with a clear origin in the way the architecture was constructed, rather than focusing on issues with the various implementations themselves.

Mapping Back to Architecture

Since both of the elements identified in this case (generalized support and convergence layers) are described at the architecture level instead of the BP level, we can identify two fundamental scaling challenges that we expect to be present in the DTN architecture as a whole:

- The overhead to decode and evaluate a DTN frame is higher than the overhead to decode and route an IP packet
  - Effect: vertical scaling is less effective than it would be otherwise
- DTN traffic flowing through a single convergence layer cannot rely on hardware to load-balance across multiple hardware queues on the NIC for parallel processing
  - Effect: unable to determine where individual bundles stop and start, so cannot distribute across multiple threads / cores / etc.

Thus, the standard approaches to making networking scale are rather difficult without first addressing the challenges referenced above.

3. HDTN Approach

Why Another Implementation?

Generally speaking, Interplanetary Overlay Network (ION) is not the only existing DTN implementation - several implementation presently exist and are either free or open-source. Generally, each implementation is optimized for different situations. ION, for example, has been optimized to run on small, embedded systems. Another implementation of DTN, called DTN2, was primarily designed to offer an experimental and easily extensible platform for working with DTN routing approaches. A third implementation called Postellation was designed to be extremely simple for new users to set up and work with, and included a discovery mechanism and quite a bit of configuration automation. Other implementations exist, each of which has different goals in mind.

Unfortunately, there is no such thing as a (truly) free lunch[14]. In this instance, different implementations of DTN come with different benefits and drawbacks: for example, [15] demonstrates that DTN2's emphasis on extensibility comes at the expense of performance. In the experiment described therein, a DTN implementation called IBR-DTN was able to read and write bundles to disk in the same amount of time it took for DTN2 to work with the bundle in memory. Given the overhead commonly associated with storage media, the different design goals are illustrated as substantial differences in the way the code behaves and performs in given situations.

There have also been approaches to building faster / high-performance DTNs in e.g. FPGAs. Generally speaking, such attempts have acted as direct ports of existing DTN implementations (e.g. the IONAC-Lite project [16]). Unfortunately, the standard for BP is relatively large, involves a number of subsystems (e.g. storage, network processing, convergence layers, etc), and is therefore extremely complex to represent at the level of an ASIC / FPGA. This approach also adds great complexity to e.g. ensuring that software implementations of DTN stay synchronized with the implementations that have been ported to the hardware. To work around these issues with complexity / monolithic design, HDTN is split into two discrete components: one that handles data-plane operations, and another that handles control-plane operations.

The data-plane component of HDTN consists of a five-stage pipeline: an ingress stage, an unpack stage, a switch stage, a repack stage, and an egress stage. This pipeline is illustrated in Figure 3. There is also an optional storage stage of the pipeline, as well as an interface that allows interaction between the external control-plane interface and the internal switching controller.

It is important to note that no state is actively shared between any two states of the processing pipeline. In the event that data must be shared between different stages of the pipeline, it is shared as a separate message that passes through the same pipeline as the bundle data. This avoids a number of common issues with livelock and deadlock, but comes at the cost of complexity and memory overhead: state must be replicated at each stage of the pipeline, and it can take time for state to migrate between successive stages.

HDTN Data Plane Processing Pipeline

The first stage of the HDTN pipeline is the ingress stage. In this stage, the bundle enters the system through a supported ingress port. One (or more) threads continually poll one (or more) ingress ports (e.g. convergence layers, hardware queues on a NIC, etc.) to watch for new data. Immediately upon receiving a new bundle, the bundle is tagged with a 64-bit hash that corresponds to the appropriate source and destination. The bundle is then forwarded to the unpacking stage of the pipeline.

The unpacking stage transforms the bundle into what is called an intermediate format. The intermediate format is a format that preserves all information associated with the bundle, but is also not strictly compliant with relevant BP / other specifications. This step is done to reduce computational overhead associated with bundle processing in future stages of the pipeline. Once this transformation has been completed, the bundle's meta-data and payload data is passed to the switching stage of the pipeline.

The switch stage of the pipeline is essentially a small table. This table matches the bundle source and destination to a table of known sources and destinations. It also applies any QoS rules, prioritization, and other flow control that must be applied in order for bundle processing to proceed as required. The switch relies on notifications from the control-plane interface to tell it which bundle sources / destinations are active, as well as to provide it with information with regard to how different bundle traffic should be prioritized across various egress paths. If a table entry exists, the switch tags the bundle with the appropriate egress information and passes it to the fourth stage of the pipeline: the packing stage.

In the packing stage of the pipeline, the metadata and payload data are rebuilt into a standards-compliant frame. The packing stage then forwards the bundle to an appropriate egress stage as directed by the switch. This fifth egress stage of the pipeline, writes the bundle to a convergence layer, hardware queue, etc. as directed by the switch. At this point, the bundle leaves the system. There are, however, a few special cases that can alter the flow of this basic operation.

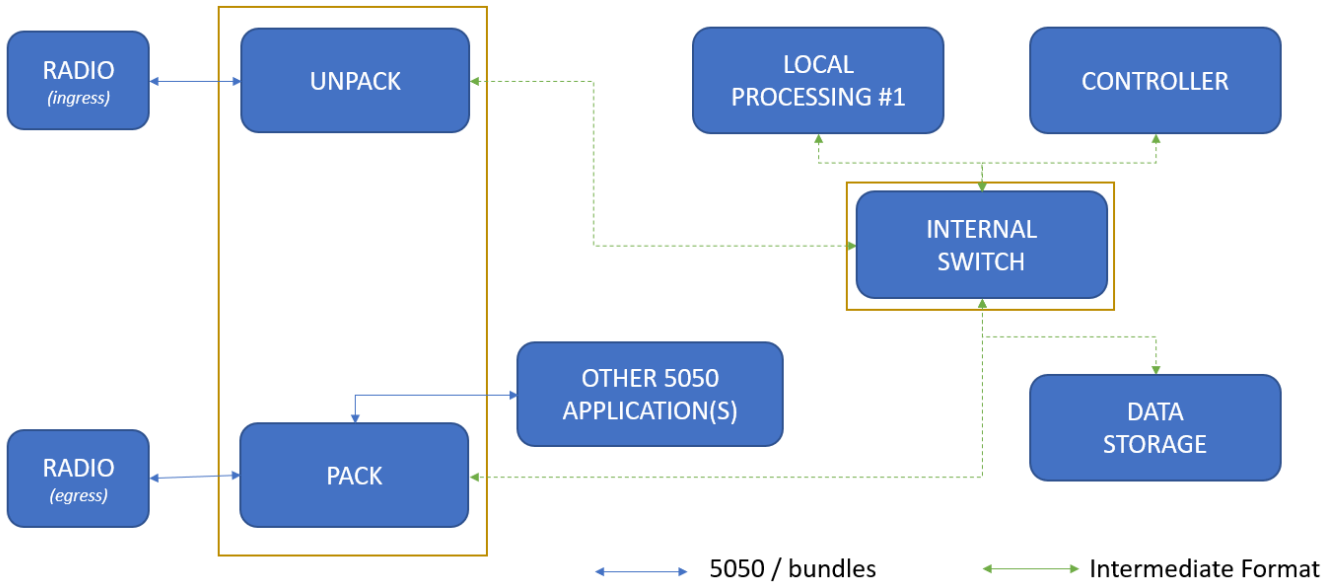

Figure 3. HDTN processing pipeline

Bundle Storage—In the event that the switch receives a bundle and has no information about it, the bundle passes to a special stage of the pipeline called storage. The storage stage of the pipeline accepts the bundle payload data and meta-data that the switch was unable to forward. The storage stage then adds it to a persistent storage device, and records its existence (as well as its source / destination tag) in an index.

Every so often, an external control interface will provide a contact schedule, QoS information on how existing data should be dispatched (e.g. if stored data should only be sent after all new data has been sent, etc), and data rates for each anticipated contact. Storage will monitor both this schedule and the current time as recorded by the system. When the storage believes that the time has come to begin releasing stored data, it will add an entry to the switch's table to indicate that an egress is available, and will then start passing indicated data to the switch stage of the pipeline.

The switch stage, now having an associated entry in its table, will forward data to the packing stage, which will in turn transform it into a bundle and pass it along to the egress stage.

Custody Transfer—While custody transfer is presently a work in progress, it does represent a special case. The present approach will therefore be discussed here, with the note that, should issues arise, it may be subject to change. Generally, however, both the switching and the storage stages of the pipeline are required to handle custody.

When a bundle is received at the switching stage of the pipeline and that bundle requires custody transfer, the switch first examines its table. If the switch has an entry for that bundle and the bundle can therefore be forwarded, then the switch forwards that bundle and immediately generates a custody response. The switch injects the custody response into itself, and proceeds to process the custody notification as it would any other bundle.

In the event that the switch does not have an entry for that bundle, the bundle is forwarded to storage. When the bundle has been successfully written to storage does the storage stage of the pipeline generate a positive custody notification and forward it to the switch.

Scaling / Acceleration

While substantially more complex than a monolithic design, one of the benefits of a pipeline approach is that it lends itself well to both incremental acceleration and to horizontal scaling techniques. In the case that all pipeline stages are implemented in software, for example, each stage can be executed as one or more threads with multiple-producer, multiple-consumer queues that are shared between them. Packets / bundles can also be load-balanced across multiple instances of e.g. ingress, unpacking, switch, or storage stages.

Mitigating Elephant Flows—This approach also allows one to e.g. avoid the elephant flow problem by building egress adapters that can build parallel convergence layers, and internally load-balance incoming bundles in round-robin fashion across each of them. This allows hardware to balance incoming / outgoing packets across multiple convergence layers, splitting the single elephant flow into multiple, much smaller flows. The egress adapters can also determine when and how to split the number of parallel convergence layers it has active at any given time. While this solution is admittedly sub-optimal (especially in the case that adding additional convergence layer flows can have side effects), it is a mitigation that allows for some degree of load-balancing to occur in COTS hardware.

Additionally, such an architecture need not have every component co-located on a common piece of hardware. Instead, the design allows for stages to be spread across one or many different discrete pieces of hardware - as long as the bus connecting those pieces of hardware is capable of supporting the aggregate ingress and egress rates expected by the system.

Facilitating Vertical Scaling—Such an approach also mitigates challenges associated with vertical scaling by allowing for the incremental construction of hardware-based fast paths

through various stages of the pipeline. Rather than forcing the entirety of a DTN implementation to reside in a single logical piece of hardware, it becomes possible to replace individual stages of the pipeline with dedicated hardware accelerators (e.g. unpacking stage, switching stage, storage stage, etc), and to wire those accelerators together in whatever way might make the most sense given the anticipated load on the system.

It also becomes possible to integrate COTS products into such a system: one could, for example, implement a DTN storage solution as a plug-in to an existing commercial Network-Attached Storage / Storage Area Network offering. The scope of work involved with doing so would be far better defined than might be the case should one attempt to integrate the same kind of solution into a more monolithic implementation of DTN.

Disadvantages

One of the primary disadvantages of such an approach is its complexity. Rather than utilizing a single moving part with a rather well-defined interface, the system can rapidly scale to a point beyond the capability of a human to easily understand and / or manage it.

Additionally, scaling in such situations must be handled extremely carefully: for example, in the case where there are more stages of the pipeline than there are available cores on a machine, the way the various stages of the pipeline are laid out are going to have a heavy influence on the operation of the system as a whole. On a similar note, each stage of the pipeline will come with different processing / scaling requirements. With this in mind, horizontal scaling normally only shows relatively linear results when it is handled in a way that actually distributes the load in a meaningful way.

Getting such a configuration correct takes time and tweaking, and can be difficult without a firm grasp of how the system works. On the plus side, however, such a system does present an opportunity for self-optimization, assuming that the system has been appropriately instrumented: load triggers can spawn new processing tasks automatically, within user-defined constraints that are related to the total resources available to the machine. Again, however, such schemes do involve additional complexity, and a mis-configuration of such a management scheme can lead to edge-cases where the performance of the system degrades in a way that is anything but graceful.

4. PLATFORMS

At the moment, HiDRA has been evaluating the pipelined, data-plane implementation of DTN on a number of different platforms.

Here, we take advantage of several benefits to a technology demonstration on the ISS. Among these are the ability to use more COTS equipment, within reason. This drives down development costs and time, and also affords alternatives to esoteric hardware (and hardware architectures). The current software is being prototyped on two readily available platforms:

- Double Shot MACCHIATObin
  - CPU: Armada 8040 (ARMv8 Cortex-A72, 4 cores)
  - RAM: 16GB DDR4
  - NIC: Dual 10GigE, 2.5GbE, GigE
  - PCIe: PCIe x4 (ver 3.0)
  - Power: 36W (System)
- Xeon-D
  - CPU: Xeon D-1541 (x86-64, 8 cores)
  - RAM: 32GB DDR4
  - NIC: Dual 10GigE, Dual
  - PCIe: PCIe x16 (ver 3.0)
  - Power: 45W (TDP)

Both of these options use the Mini-ITX form factor (170mm x 170mm), and are physically compatible with our potential ISS location. Moreover, we have also demonstrated As such, we anticipate that one of these platforms will be very representative of our actual payload.

Each option features one PCIe expansion slot. Presently, trade studies are being conducted to determine if the FPGA approach will be followed for the ISS application. Essentially, there are three options:

1. PCIe SSD, no FPGA
2. FPGA, SATA SSDs
3. FPGA and PCIe SSD

Options 1, at first glance, is the most straightforward. There are 8TB SSD's with read/write rates at ≈3000MBps which are commercially available at the time of this writing, which could satisfy the requirements for the ISS demonstration. Many passes worth of the data could be buffered, and by separating the bulk storage from system storage, it is reasonable to attempt a pure software solution for several 1Gbps links. The issues would, however, show up past certain inflection points. As a certain boundary, such an SSD cannot go beyond 24Gbps. It is most likely that more restrictive bottlenecks would surface before it got that far.

For Option 2, let's assume that we use a standard GigE link, even if the modem has a 1.2Gbps capability. Then in one 22 minute pass, we accumulate 165GB of data. The FPGAs presently used have 256GB of onboard DDR4 memory. Thus, all data from a single pass will fit on the dedicated RAM prior to being stored on the non-volatile SSD. As SATA III supports rates up to 6Gbps, it is reasonable to benchmark such a system. The biggest difference between this and Option 1 is extensibility - support for links beyond the onboard 10GigE interface become possible.

Finally, the FPGAs in question have ports to also be a PCIe host [17], and could be connected to, for example, an NVMe compatible SSD directly. This increases complexity and cost, but provides fast, bulk storage that is not on the system bus. This type of solution will be more applicable to future iterations where we encroach on our greater speed goals in the 200+Gbps range. In previous HiDRA papers (e.g., [18]), the FPGA approach is explained in greater detail. The relevant information includes that a functioning store, carry, and forward prototype was created, but it was not DTN-compliant, nor was Ethernet in hardware. The demarcation of the software and hardware implementation sides of the HiDRA is important to optimize the system for performance and scalability. Certainly functions, such as routing and traffic monitoring is best realized in software. Direct handling of the data flow, and especially when considering the 200+Gbps rates, is best realized in hardware, although for platforms utilizing slower physical links the full implementation could be realized though software. FPGAs are a natural choice for the hardware acceleration as they offer significant throughput performance through concurrency, and may be reconfigured to scale the architecture across a range of space mission classes. High rate FPGA development platforms utilizing local memory storage

and standard PCIe and Quad Small Form-factor Pluggable (QSFP) interfaces offer the performance and I/O required to address MIMO spacecraft systems utilizing emerging laser communication technologies [17]. Such a system, when paired with a CPU running the DTN protocol, would serve as an onboard DTN node to provide bundle storage and services at rates concomitant with the space network goals over the next decade.

5. INTERNATIONAL SPACE STATION

The ISS features an Ethernet LAN, the Joint Station Lan (JSL) [19]. The JSL connects to such Ethernet data sources, such as the External High Definition Camera (EHDC). The EHDC, in turn, has an internal switch connectings radios (WiFi), power supplies, the encoder, and the camera itself. As the EHDC provides MPEG2 movies, DTN could be used to store, carry, and forward its data - thereby becoming an automatic outage recorder. DTN has found a place on the ISS, and became a service in 2016 [20]. DTN has also been integrated into the Telescience Resource Toolkit (TReK) as a means to remotely provide command and control to payloads aboard the ISS [21]. Not only are these projects markers of great progress, but they also demonstrate the current scale of DTN on ISS. In [21], we see the ISS DTN topology suggests a three-hop network. We note that previous links are not removed with the inclusion of optical terminals, so rather more paths become available for data to get to the ground. Moreover, the optical ground stations might not be geographically co-located with the RF ground stations. Therefore, the terrestrial network (and infrastructure) will become more complex as well.

ILLUMA-T, the upcoming optical terminal for the ISS, will add a new data path. ILLUMA-T will use the LCRD system to relay data; the performance data are shown in Table 1:

	Data Rate	Data Return per Week
Return Link	1.2Gbps	18.76Tb
Forward Link	51Mbps	0.75Tb

Table 1. ILLUMA-T performance characteristics from [22]

ILLUMA-T expects to enjoy roughly 22 minutes of contact per orbit, which lasts 92 minutes. Per week, the expected contact time is about 33.5 hours. Hence the best utilization of the optical link will require buffering during times of outage. The data rates shown in TReK are on the order of 15Mbps, which is not entirely fair as TReK is end-to-end. However IBR's published performance is roughly 310MBps, and we note that desired data rates on the ISS can go beyond those of ILLUMA-T's, for example, if bundles are also directed over the Tracking and Data Relay Satellites (TDRS). Therefore, while DTN provides a useful data and network management solution, the system falls short of the capabilities of the links available. Despite the relatively lower data rates, more and more experiments are using DTN onboard the ISS, though typically for individual data rates below 30Mbps. We note that these numbers are all preliminary - actual rates may change based on e.g. operational constraints.

HiDRA's HDTN is a proposed payload for the ISS, designed to accomodate the emerging technologies and data needs. As a step towards the higher-rate capabilities desired, HiDRA for the ISS will, at a minimum:

- provide a persistent data storage capacity of 2TB, and
- saturate at least two gigabit Ethernet links.

We have the goal of supporting and saturating up to 10GigE. The ISS solution is currently undergoing trade studies to determine if the HDTN implementation should be pure software (with commercial off the shelf (COTS) solid state drives (SSD's), or if it should use a hybrid software and FPGA approach.

6. CONCLUSION

With the increase of commercial activities in space, alone with the additional international organizations ramping up their respective space programs, there is a growing clash between new and old technologies, high and low communication data rates, optical and RF systems creating an ever-expanding heterogeneous space system. Previous research has investigated network management of dissimilar capabilities[23][24], and has taken advantage of the short round-trip-times (RTT) characteristic of the near-Earth domain to present workable solutions. Lunar distances and especially deep space present a great challenge for real-time feedback control from Earth, so these techniques and parameter tunings are not extensible to such domains. A multi-hop multi-path test bed was constructed to emulate the dynamic and delay-centric space environment to stress both protocol and hardware instantiations utilizing the ION implementation of DTN [13]. The experiments conducted revealed several deficiencies in the effectiveness of DTN to mitigate architectural challenges, and revealed the need for significant locally managed memory to accommodate bundle storage during outage periods. A high speed read and write requirement is levied upon the memory storage devices to maintain the burst capability with the physical layer optical links. Additional deficiencies were notes including the lack of reliability checks within the DTN bundle, varying support for fragmentation, lack of definition for convergence layers, difficulty in scalability and routing due to flat address spaces, and no standardized discovery mechanisms [12]. The identification of these challenges created a targeted research plan to develop solutions to evolve the DTN protocol into a workable architecture to operate with the emerging space communication systems. The HiDRA project is working towards a general yet practical toolkit and knowledge base to help usher in not the era of these new technologies, but of the system that contains them.

HiDRA is working towards the successful migration of DTN from a protocol to an operational implementation hinges on the ability to transmit data in the 10-200Gbps regime to meet the needs of current and future high speed space networks, such as those realized by laser communications. An early and successful experiment of the ION implementation of DTN running over a free-space laser communications link was conducted at the Jet Propulsion Laboratory (JPL) [25]. The many papers inspered from this experiment, as well as IONAC-Lite [16] have lead to not only the problem statement, but towards a solution. A potential solution is to consider a form of hardware acceleration for DTN through field programmable gate arrays (FPGAs) to serve as a direct memory access (DMA) controller, thus offloading the burden of the higher-rate data plane traffic from the CPU control plane [26]. The hybrid software/hardware implementation streamlines functionality between the respective sequential and concurrent processing elements to handle the systems tasks with which they are best suited. Early trials of this methodology have been examined the implications of custody transfer on the distribution of transfers and the inclusion of Contact Graph Routing (CGR)

for effective link establishment [27] [28].

As a simplistic overview, the FPGA design essentially allows software to manage memory and interfaces, loosely like a direct memory access (DMA). The software architecture laid out here is being designed to operate both with system storage, and with the FPGA's dedicated memory, which is off of the system bus [18]. Current strides have resulted in pieces of the HDTN software being created, and benchmarking has begun. Preliminary tests have shown bundle generation, processing, transmission, receiving, processing, and payload delivery working at 10Gbps rates in software. To make HiDRA practical, a web interface is being developed that both explores the system view, but also the (local) network view.

REFERENCES

[1] D. M. Boroson, B. S. Robinson, D. V. Murphy, D. A. Burianek, F. Khatri, J. M. Kovalik, Z. Sodnik, and D. M. Cornwell, "Overview and results of the lunar laser communication demonstration," *Proc.SPIE*, vol. 8971, pp. 8971 – 8971 – 11, 2014. [Online]. Available: https://doi.org/10.1117/12.2045508

[2] "Science - jpl's science division: People: Simon hook," *NASA*. [Online]. Available: https://science.jpl.nasa.gov/people/Hook/

[3] J. A. Robinson and W. L. Stefanov, "Earth science research on the international space station," *Committee on Earth Science and Applications from Space (CESAS) Space Studies Board*, 2016.

[4] B. Edwards, D. Israel, K. Wilson, J. Moores, and A. Fletcher, "Overview of the laser communications relay demonstration project," *SpaceOps 2012 Conference*, 2012.

[5] "Goddard tapped to build nasa's first integrated-photonics modem," *NASA*. [Online]. Available: https://gsfctechnology.gsfc.nasa.gov/Photonics.html

[6] "Nasa engineers tapped to build first integrated-photonics modem," *NASA*. [Online]. Available: https://www.nasa.gov/feature/goddard/2016/nasa-engineers-tapped-to-build-_rst-integrated-photonics-modem

[7] A. Hylton and D. E. Raible, "High data rate architecture (hidra)," *International Communications Satellite Systems Conferences (ICSSC)*, Oct 2016. [Online]. Available: https://doi.org/10.2514/6.2016-5756

[8] M. Ramadas, S. Burleigh, and S. Farrell, "RFC 5326, Licklider Transmission Protocol - Specification," *IETF Network Working Group*, 2008. [Online]. Available: https://tools.ietf.org/html/rfc5326

[9] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "RFC 4838, Delay-Tolerant Networking Architecture," *IETF Network Working Group*, 2007. [Online]. Available: https://tools.ietf.org/html/rfc4838

[10] K. Scott and S. Burleigh, "RFC 5050, Bundle Protocol Specification," *IETF Network Working Group*, 2007. [Online]. Available: https://tools.ietf.org/html/rfc5050

[11] W. Eddy and E. Davies, "RFC 6256, Using Self-Delimiting Numeric Values in Protocols," *Internet Research Task Force (IRTF)*, 2011. [Online]. Available: https://tools.ietf.org/html/rfc6256

[12] A. Hylton and D. E. Raible, "Networked operations of hybrid radio optical communications satellites," *AIAA SPACE Forum*, Aug 2014. [Online]. Available: https://doi.org/10.2514/6.2014-4440

[13] D. Raible and A. Hylton, "Integrated rf/optical interplanetary networking preliminary explorations and empirical results," *International Communications Satellite Systems Conferences (ICSSC)*, Sep 2012, 0. [Online]. Available: https://doi.org/10.2514/6.2012-15126

[14] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 1, no. 1, pp. 67–82, 1997.

[15] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf, "Ibr-dtn: A lightweight, modular and highly portable bundle protocol implementation," *Electronic Communications of the EASST*, vol. 37, 2011. [Online]. Available: https://journal.ub.tu-berlin.de/eceasst/article/view/512/544

[16] L. Torgerson, "Ionac-lite - a combination of energy and performance optimization is attained for high-speed delay tolerant networking," 2011. [Online]. Available: https://www.techbriefs.com/component/content/article/tb/techbriefs/electronics-and-computers/10723

[17] "Bittware.com/xilinx — xusp3r." [Online]. Available: http://www.bittware.com/xilinx/product/xusp3r/

[18] A. Hylton, D. E. Raible, and G. Clark, "On the development and application of high data rate architecture (hidra) in future space networks," *International Communications Satellite Systems Conferences (ICSSC)*, Oct 2017. [Online]. Available: https://doi.org/10.2514/6.2017-5415

[19] V. Studer, "International space station (iss) external high definition camera assembly (ehdca)," 2014. [Online]. Available: https://www.nasa.gov/sites/default/files/files/V_Studer-External_High_Definition_Camera.pdf

[20] "Disruption tolerant networking - reliable solar system internet connection," *NASA*, 2018. [Online]. Available: https://www.nasa.gov/content/dtn

[21] A. Schlesinger, B. M. Willman, L. Pitts, S. R. Davidson, and W. A. Pohlchuck, "Delay/disruption tolerant networking for the international space station (iss)," pp. 1–14, March 2017.

[22] A. Seas, Z. Gonnsen, and T. Yarnall, "Illuma-t (integrated lcrd leo user modem and amplifier terminal) payload," 2018. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20180002846.pdf

[23] V. W. S. Chan, "Optical satellite networks," *Journal of Lightwave Technology*, vol. 21, no. 11, p. 2811, Nov 2003. [Online]. Available: http://jlt.osa.org/abstract.cfm?URI=jlt-21-11-2811

[24] P. Clark and A. Sengers, "Wireless optical networking challenges and solutions," *Military Communications Conference*.

[25] J. Schoolcraft and K. Wilson, "Experimental characterization of space optical communications with disruption-tolerant network protocols," *2011 International Conference on Space Optical Systems and Applications (ICSOS)*, pp. 248–252, May 2011.

[26] M. N. Ellanti, S. S. Gorshe, L. G. Raman, and W. D. Grover, *Next Generation Transport Networks: Data, Management, and Control Planes*. Berlin, Heidelberg: Springer-Verlag, 2005.

[27] A. Hylton, D. Raible, J. Juergens, and D. Iannicca, "On applications of disruption tolerant networking to optical networking in space," *International Communications Satellite Systems Conferences (ICSSC)*, Sep 2012. [Online]. Available: https://doi.org/10.2514/6.2012-15228

[28] G. Araniti, N. Bezirgiannidis, E. Birrane, I. Bisio, S. Burleigh, C. Caini, M. Feldmann, M. Marchese, J. Segui, and K. Suzuki, "Contact graph routing in dtn space networks: overview, enhancements and performance," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 38–46, March 2015.