17th International Conference on Space Operations, Dubai, United Arab Emirates, 6 - 10 March 2023.
Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."

SpaceOps-2023, ID # 101 (390)

An acceleration study of a DTN implementation using a System-on-a-chip

Yu Morinaga[a]*, Kiyohisa Suzuki[b] and Hiroyuki Ito[c]

[a] *Japan Aerospace Exploration Agency, Tsukuba, Ibaraki, 305-8505, Japan,* morinaga.yu@jaxa.jp
[b] *Japan Aerospace Exploration Agency, Tsukuba, Ibaraki, 305-8505, Japan,* suzuki.kiyohisa@jaxa.jp
[c] *Japan Aerospace Exploration Agency, Tsukuba, Ibaraki, 305-8505, Japan,* hiroyuki.ito@jaxa.jp

* Corresponding Author

Abstract

In this paper, we introduce the status of an acceleration study of a Delay/Disruption Tolerant Networking (DTN) implementation. DTN is a new type of Internetworking suite that allows end-to-end data delivery in the extreme communication environments such as long delay and lack of end-to-end network connectivity. Therefore, DTN is regarded as a key technology of Interplanetary Networking for future Lunar and Mars missions. Japan Aerospace eXploration Agency (JAXA) has been participating in and contributing to the DTN standardization activity and has prototyped DTN protocol suite running on POSIX operating system, such as LINUX. In case of higher-performance internetworking-based communication scenarios using Ka-band/optical systems, the performance of DTN implementations also need to be high. However, it is difficult to improve high-throughput performance such as over Gbps by using only pure DTN software with power-saving electronics (e.g., lower TDP and frequency of CPU as used in spacecraft components or challenged networks), and thus it is important to consider an acceleration approach that is offloading functions of DTN software to hardware. Considering above situation, we have offloaded, for example, encoding and decoding of Bundle Protocol and Licklider Transmission Protocol to an FPGA and have implemented a hybrid software/hardware implementation of DTN protocol suite on a System-on-a-Chip (SoC). In addition, we have implemented real-time OS (RTOS) in a Processing System, which is part of SoC, instead of using POSIX operating system, because characteristics of RTOS, for the purpose of real time applications such as an embedded system, are often required for on-bord systems of spacecrafts, in generally. Up to the present, we have confirmed to achieve average throughput performance of the DTN implementation, 5.7Gbps on 10Gigabit-Ethernet at lab level. From those experimental facts, we studied the approach using SoC for DTN implementation had a certain feasibility for higher rate use. To adopt DTN protocol suite to spacecrafts with higher-rate links such as Ka-band/optical in future, we aim to develop DTN implementations that are more efficient and higher throughput performances, and we think it is necessary to increase the Technology Readiness Level (TRL) for future flight assets.

Keywords: (DTN, System-on-a-chip, FPGA, acceleration, RTOS)

Nomenclature

Gbps = gigabits per second
kkps = kilobits per second
MB = megabyte
sec = second

1. Introduction

The internet protocol suite known as TCP/IP provides internetworking service in the terrestrial communication environment, but in the extreme environment relative to generic terrestrial communication such as outer space where communication link is intermittently disrupted and/or round-trip time (RTT) is significantly long, the internet protocol suite might not work properly or could not work at all. On the other hand, a Delay/Disruption Tolerant Networking (DTN) enables to ensure internetworking in such environment. Therefore, DTN has attracted attention over years as a key network technology of Interplanetary Networking for future Lunar and Mars missions and other missions such as Low Earth Orbit (LEO) or Geosynchronous Earth Orbit (GEO) satellite systems using internetworking-based communications. Considering the importance of future DTN usage, Japan Aerospace eXploration Agency (JAXA) has studied DTN protocol suites including contact graph routing (CGR) that is the DTN routing procedure. [1]

Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."

The Consultative Committee for Space Data Systems (CCSDS), affiliating with the ISO TC20/SC13, is multi-national forum for the purpose of developing of communications and data systems standards. JAXA has been participating in CCCSD DTN Working Group and contributing to develop CCSDS recommended standards such as Bundle Protocol (BP) [2], Licklider Transmission Protocol (LTP) [3] and Schedule-Aware Bundle Routing (SABR). [4] Based on these CCSDS standards, JAXA has successfully produce own DTN protocol suite implementation domestically. [5] JAXA also has contributed feasibility study for DTN utilization. JAXA and the National Aeronautics and Space Administration (NASA) have conducted interoperability demonstration for the Interplanetary Overlay Network (ION) [6] using actual space link. [7] In addition, JAXA and Sony Computer Science Laboratories, Inc. have successfully demonstrated complete data file transfer in an experimental environment with the bit error rate of free-space optical communication. [8]

Recently, in order to establish a long-term human presence on the Moon such as Artemis program, NASA, the European Space Agency (ESA) and other space agencies are planning to serve the interoperable networking communication and navigation for typical use cases such as lunar orbiters, rovers, astronauts and other users on and around the Moon (e.g., LunaNet and Moonlight). [9] JAXA also started a feasibility study for a communication system, connected between lunar system and earth. [10][11] In these scenarios, links between some lunar and Earth nodes may be Ka-band/optical to implement high-performance internetworking-based communication systems. It is expected that the bandwidth and its data rates for each link exceed Gbps in order to meet the use cases for transferring massive amounts of scientific and engineering data. It means that the performance of an implementation architecture for DTN nodes needs to be high. However, it is difficult to improve high-throughput performance such as over Gbps by using only pure DTN software (In other words, all implemented in software program) with power-saving electronics (e.g., lower TDP and frequency of CPU as used in spacecraft components or challenged networks). Thus, it is important to consider offloading functions of DTN software to a Field-Programmable Gate Array (FPGA), which is the modern method of High-Performance Computing (HPC). Offloading functions of communication protocol is often used as an acceleration approach in terrestrial communication or communication satellite related field.

In this paper, we describe how we implement a hybrid software/hardware implementation of DTN protocol suite on a System-on-a-Chip (SoC) and how much it improves the performance. This paper is organized as follows. In section 2, we explain our DTN protocol suite implementation running on Real-time OS (RTOS). In section 3, we explain the acceleration approach of the DTN implementation. In section 4, we show our experimental description and its results. In section 5, we discuss the results. Finally, the conclusions are drawn in section 6.

2. Development of the DTN implementation running on RTOS

Up to the present, JAXA has successfully produce own DTN protocol suite implementation as a software running on POSIX operating system. [5] However, this software can run on only LINUX, not RTOS. Considering a DTN implementation with an onboard system for a spacecraft, the DTN implementation on RTOS could be needed. Since operation of spacecraft today is the "earth centric", so understanding and transitioning the status of spacecraft is by the way of telemetry and telecommand to/from the earth. In that case, reproducible and deterministic behaviour is preferred in view of operation on earth. In other words, behaviour of platforms such as operation system are as well. Therefore, as first step of feasibility study of DTN implementation towards components of spacecraft, we port programs of the DTN software running on LINUX to that of the software running on RTOS. The following components in the DTN software running on RTOS are used in acceleration study described in section 3 and the performance test of the DTN implementation in section 4.

- **Bundle Protocol**: BP is implemented based on CCSDS 634.2-B-1 in 2015 [2], so-called BP version 6 and is necessary to support the DTN in space applications. In following section, when we implement the BP, we utilize LTP Convergence Layer Adapter (LTPCLA), which is the communication interface for Bundle Protocol Agent (BPA) to interact with the protocols underlying BP.
- **Licklider Transmission Protocol**: LTP is implemented based on CCSDS 734.1-B-1 in 2015 [3] and provides reliable data transmission between LTP entities. In following section, LTP provides the service to send only red-part data segments, which is optional reliability mechanisms, to a destination LTP client in our DTN implementation.
- **Schedule-Aware Bundle Routing**: SABR is implemented based on CCSDS 734.3-B-1 in 2019 [4] and which enables to forward bundles in space environments by computing dynamic routing based on called "contact plan" that the visibilities, data rates, distance for each link of neighbour nodes are included.

Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."

To evaluate the performance of the DTN software running on RTOS, we conducted the simple performance test. Figure 1 shows the test configuration and protocol stack of end nodes. In taking the first step, we considered the economics and agile-like development style, DTN software is being implemented on FreeRTOS, which is distributed freely under the MIT open-source license. In this test case, 10MB data is transported from a source node (Raspberry Pi 1) to a destination node (DELL Optiplex 7010) on 100BASE-T Ethernet without delay and frame loss. Figure 2 shows the result of the DTN software, and the throughput is approximately 700Kbps.

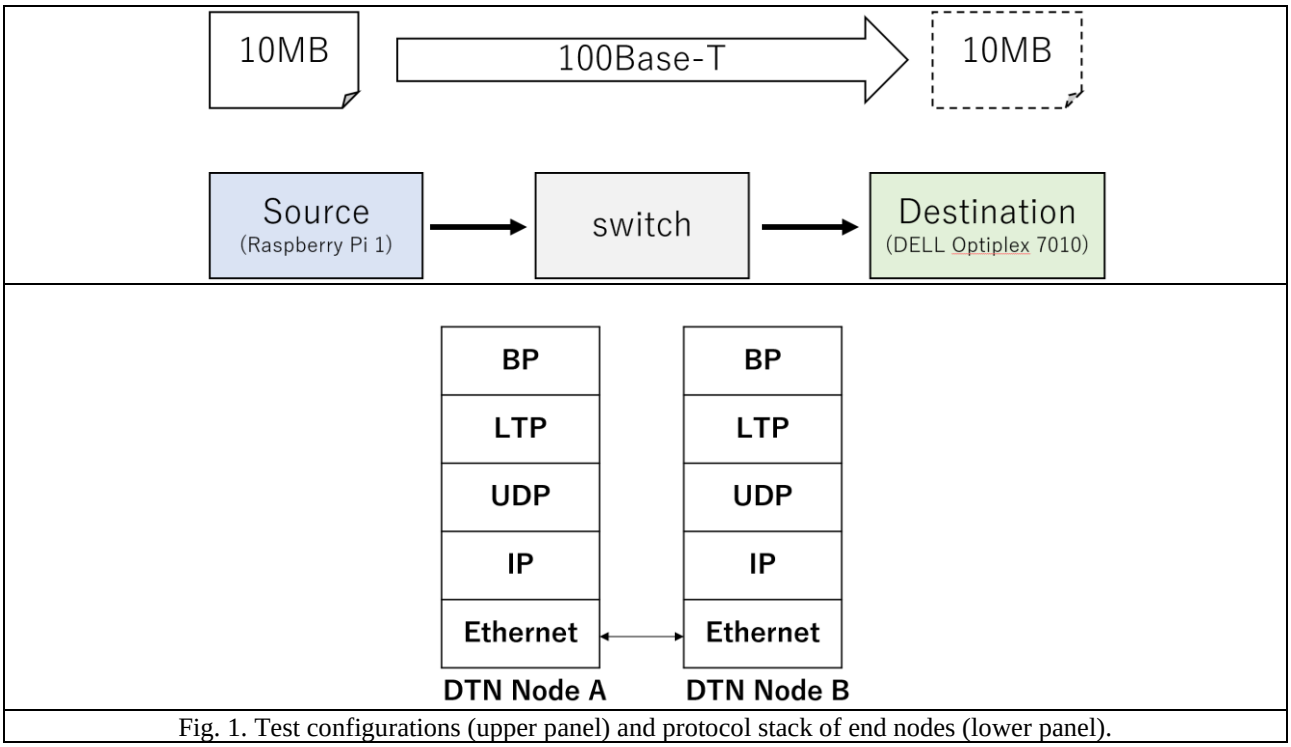


Fig. 1. Test configurations (upper panel) and protocol stack of end nodes (lower panel).

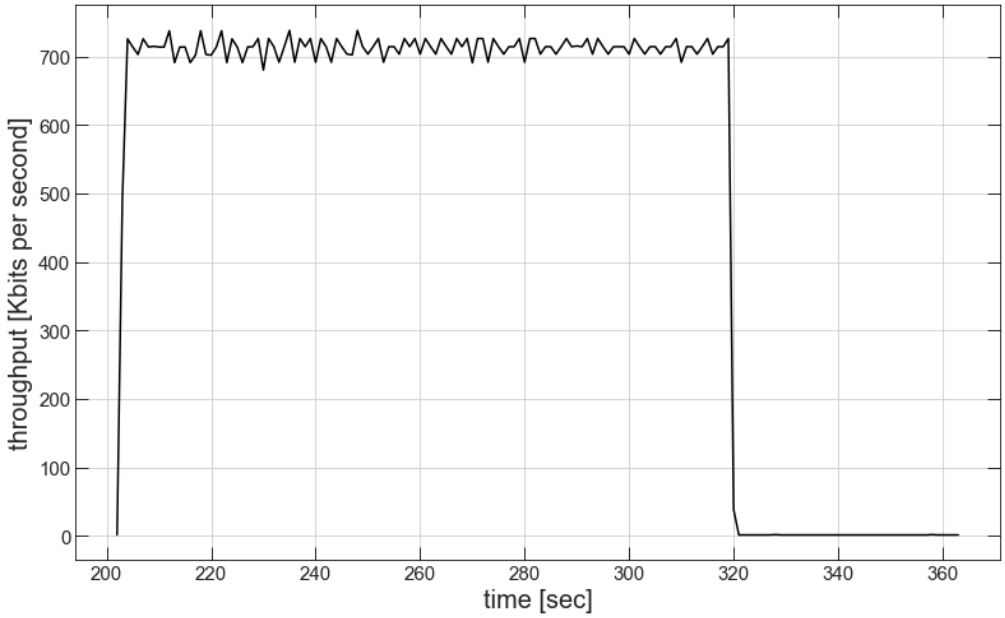


Fig. 2. Throughput of the DTN software running on RTOS

Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."

We confirmed the throughput is lower than expected from potential hardware capability. We presume that the main reason of this low performance is not caused by the DTN software itself (userland) but the kernel land including network device driver. Figure 3 shows the result of the UDP burst performance test without userland application such as DTN software to confirm our hypothesis, and confirms the maximum throughput is approximately 2.35Mbps. After results are obtained, we continue trying over 2.35Mbps throughput, however test is failed because device driver is crashed. This result supports our hypothesis that the main reason of unexpected low performance is not caused by the DTN software itself but the kernel land.
As the result, the DTN software can perform a certain level with Commercial-Off-The-Shelf (COTS) power-saving electronics and RTOS, and we must pay attention to the constraints in case of COTS utilization.

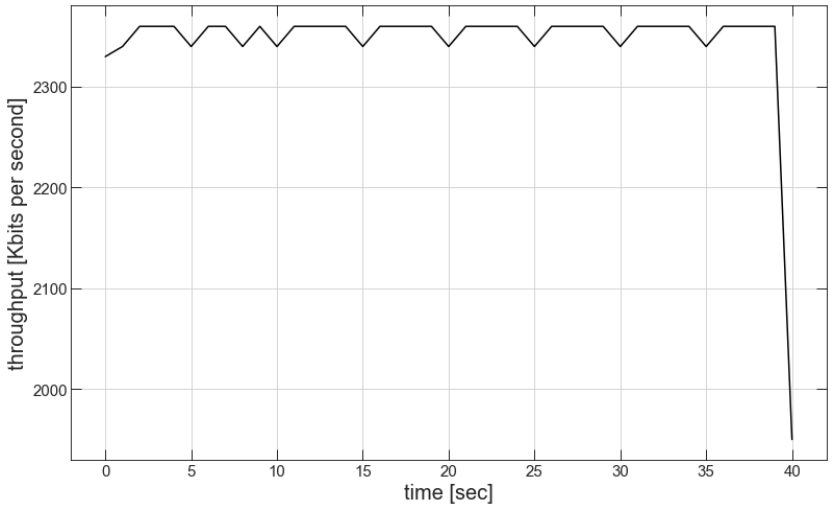


Fig. 2. Throughput of UDP burst performance test on FreeRTOS

We conclude that we succeed in porting programs of the DTN software running on LINUX to a kind of RTOS as a first step, however we find to identify bottlenecks including software and other elements to improve throughput.

3. acceleration of the DTN implementation

3.1 bottleneck analysis of the DTN implementation

At first, before accelerating the performance of the DTN implementation, we investigate and reveal the bottlenecks of the current DTN software (POSIX version) and other aspects. From the viewpoint of clarifying the boundary between DTN functions and other functions, we categorize the bottlenecks into two types that are caused by the DTN software or the rest of aspect (e.g., RTOS and device drivers). Additionally, we also consider the improving approach to these bottlenecks. We summarise the bottlenecks and their improving approach in Table 1.

Table 1. Bottleneck and improving approach

Type	No.	Bottleneck	Improving approach
DTN	1	Self-Delimiting Numeric Values (SDNV) encoding and decoding require high CPU utilization.	Offloading SDNV encoding and decoding of BP and LTP to hardware.
	2	Latency of shared memory allocation, reading and writing is high.	Reduce the number of accesses of memory (e.g., the direct memory access instead of using the shared memory.)
	3	Loop processes to check the status of the shared memory so called polling account for approximately ~90% of the total processes, and thus the efficiency tend to be down.	Improving the process of software (Trade-off between latency improvement of processes and readability of program codes) offloading the functions to hardware

Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."

			as possible.
	4	In case of a process is large, the latency of context switching tends to be high.	Optimize the program, for example, minimizing the amount of code and avoiding unnecessary computations to reduce memory usage and the processing time.
Other	1	Latency of memory for utilizing transferring and/or receiving data is high (e.g., shared memory, socket buffer and socket que).	As possible as not use the network device drivers that are produced by the RTOS and offload these functions to hardware.
	2	the performance of network device driver in particular the Ethernet driver is extremely lower than that of the CPU. Therefore, the DTN software cannot sufficiently utilize the processing resources of the computer	

3.2 Concept of the DTN node architecture for SoC

Considering the bottlenecks shown in Table 1, we implement a hybrid software/hardware implementation of DTN protocol suite on a SoC. In generally, a SoC is composed of two component parts on a single chip. One is software controlled by microprocessor, and another is hardware executed by a graphics processing unit (GPU) or FPGA. In case of utilizing GPUs, they tend to require high power consumption to enable HPC [12] and it is difficult to use GPUs in embedded systems for spacecrafts because of limited available power. Therefore, as hardware and software acceleration approach with power-saving electronics, it is suitable to use FPGA-based SoC for a DTN implementation.

In Figure 3, we represent the concept of our reference architecture to implement DTN node based on FPGA-based SoC system. As shown in Figure 3, the DTN node consists of three entities; First is a DTN software entity that DTN process is using CPU/RTOS, second is DTN hardware entity that is DTN process is using FPGA, last is underlying protocol entity that a protocol performs under DTN's convergence layer (In case of assuming high throughput, this entity might have received buffer). In addition, the architecture has storage accessible from both DTN software entity and DTN hardware entity. The storage connected to a DTN processing unit store transmission bundles and received LTP segments.

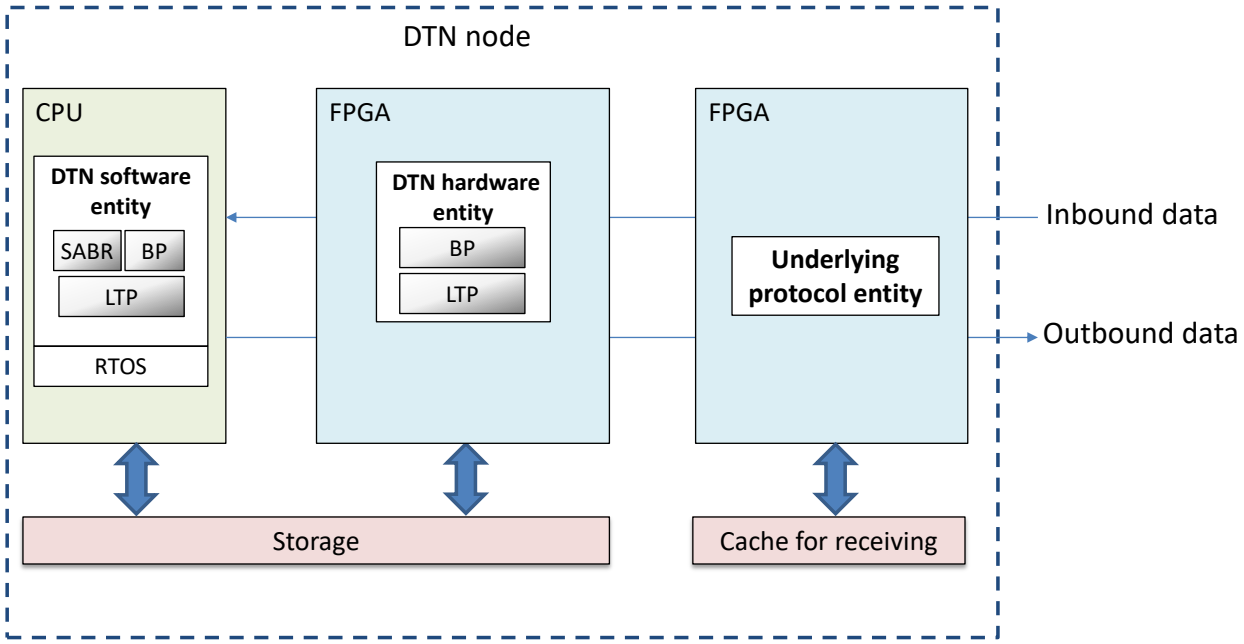


Fig. 3. Typical description of a DTN node architecture concept on SoC

Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."

In following sections, we describe the details of the respective entities as the DTN node architecture.

3.2.1 DTN software entity

As DTN software entity, we assign software functions to manage the state of the node at a particular time, such as bundle state information, node state information, convergence layer information and so on. In addition, processes which require a lot of conditional computations such SABR are implemented as CPU functions not FPGA because FPGA may result in poor performance to process such complicated procedures.

3.2.2 DTN hardware entity

DTN hardware entity has the offloaded functions which is separated from software program to logic circuit etc, those are listed in Table 2.

Table 2. Offloaded functions of LTP and DTN to an FPGA

DTN protocol	No.	Offloaded functions of LTP and BP
LTP	1	SDNV encoding and decoding
	2	management of underlying layer
	3	LTP segment generation
	4	LTP segment header parsing
BP	1	SDNV encoding and decoding
	2	BP header generation
	3	BP header parsing

3.2.3 Underlying protocol entity

As described in section 2, we confirmed that the performance of network device drivers on RTOS is extremely low and thus the throughput of the DTN software is low, too. To overcome this bottleneck, we offload the functions of network device drivers to an FPGA as the underlying layer entity.

4. performance test of the DTN implementation

4.1 implementation of the DTN node architecture

To implement the DTN node architecture described in section 3.2, we use Xilinx Zynq-7000 SoC ZC706 and Xilinx ZCU102. We implement the DTN software and hardware entities on ZC706 and underlying layer entities on ZCU102.

4.2 test scenario and configuration

In this section, we analyse the performance of the DTN implementation and show the results of them. To quantify the performance of the DTN implementation, we use the average throughput, which is the total transmission data size divided by the period in the transmission session. In this analysis, we conduct the performance tests in the simple network topology comprised of two DTN nodes. Additionally, the performance tests are executed in the various network environments with several scenario parameters, which are the delay, the Ethernet frame loss ratio and the LTP segment size. The scenario parameters are summarized in Table 3. We simulate two communication scenarios; one is normal case (N-1) without interruption and delay, and another is delay/disruption case (D-1) with interruption and delay. In all test cases, the link bandwidth between the source node and the destination node is 10G-base Ethernet. In addition, LTP segment sizes vary from 1430 Bytes to 8900 Bytes. The bundle size is 50MB and the transmission data size is also 50MB, which means that the transmission date is comprised of one bundle.

In Figure 4, the configuration of the performance test and the protocol stacks of the end nodes are illustrated. As shown in Figure 4, the traffic control emulator enables to embed the delay and/or Ethernet frame random loss in the link between the source node and the destination node. The traffic control emulator is only used in case D-1. On the other hand, the source node directly connects the network switch in case N-1.

In this performance test, the network delay corresponding to RTT is 500ms and 1000ms, and the frame loss ratio is 1%, 5% and 10%. Based on the standards required for interoperability, [13] frame loss ratio is less than or equal to 0.01% at frame size = 2048 octets. In our performance test, although the frame size is different, the frame loss ratio is higher than that of the standards. In this study, we assume the challenged network in the environments with higher error ratio and confirmed the feasibility of disruption tolerance of our DTN implementation in such environments.

Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."

First, we perform the test scenario corresponding to N-1 and evaluate the pure performance of the DTN implementation in the environments without embedded delay and disruption. Then we conduct the test scenario corresponding to D-1 and investigate the impact of delay and disruption on the performance of the DTN implementation.

Table 3. Scenario parameters in the performance test

case	Scenario parameters
normal case (N-1)	LTP segment size: 1430-8900Bytes delay: 0ms frame loss ratio: 0%
delay/disruption case (D-1)	LTP segment size: 8900Bytes delay: 0ms, 500ms, 1000ms frame loss ratio: 0%, 1%, 5%, 10% *The combination of delay and frame loss ratio is below. (loss,delay)=(0,500), (0,1000), (0,1000), (1,0), (5,0), (10,0), (5,500)

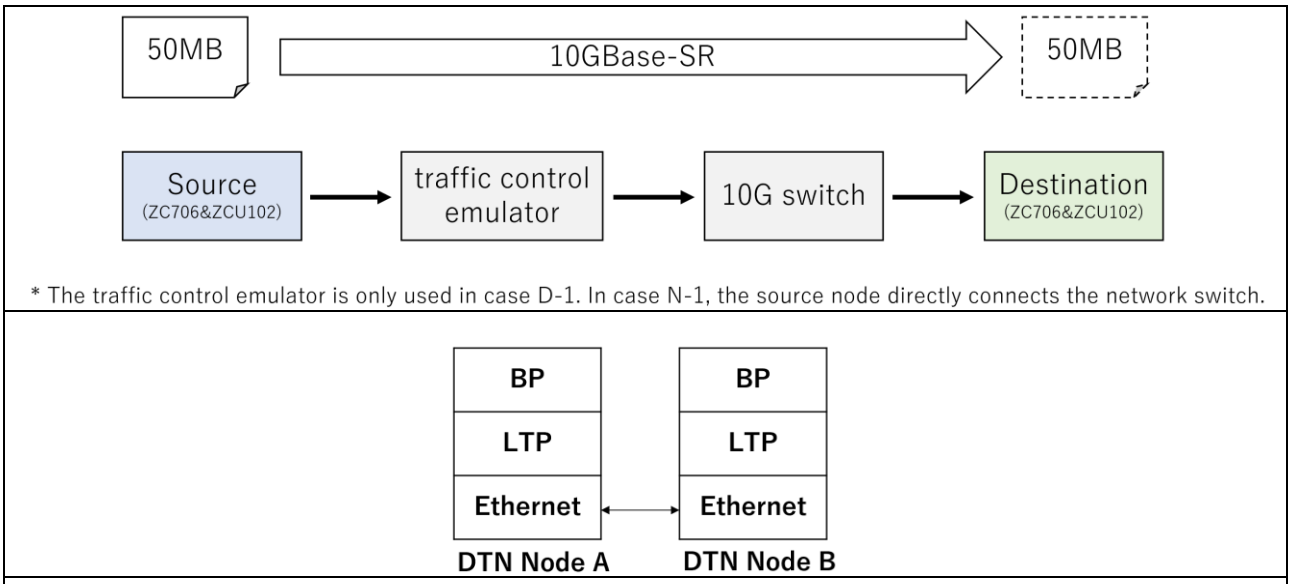


Fig 4. Test configuration (upper panel) and protocol stack of end node (lower panel)

4.3 Results

In Figure 5, we plot the distribution which represents the average throughput of the DTN implementation versus the LTP segment size in case N-1. The average throughput tends to be high with increasing the LTP segment size because in case of large LTP segment, overhead of the LTP processing is reducing. We found that the throughput achieves up to approximately 5.7Gbps at LTP segment size = 8900Bytes.

Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."

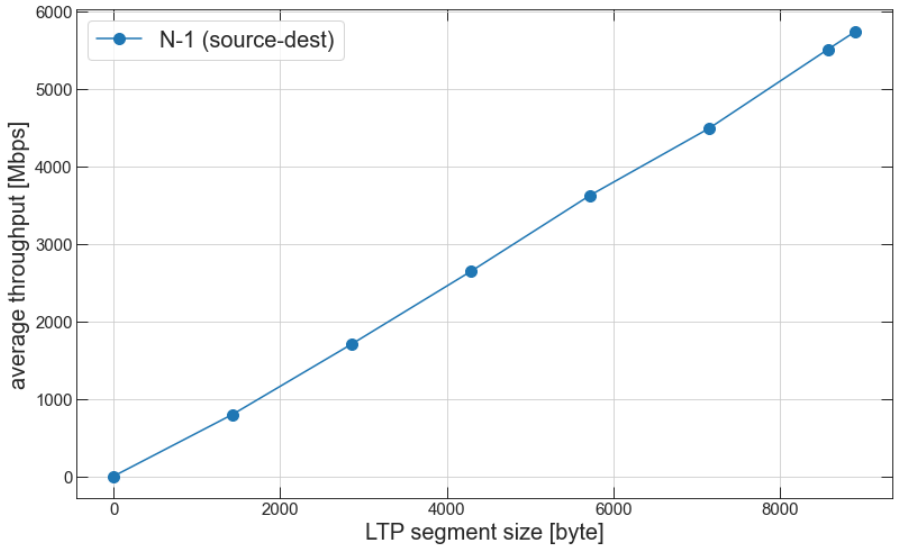


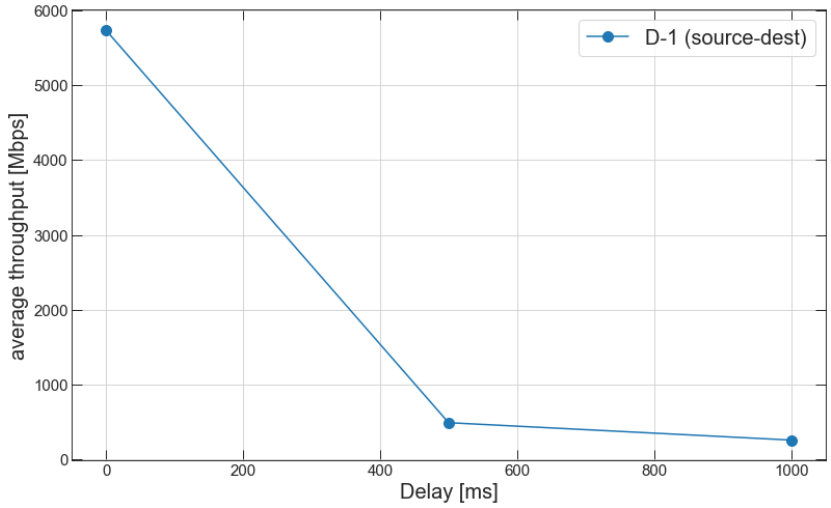
Fig. 5. Average throughput versus LTP segment size in N-1

Figure 6 shows the distributions of the throughputs in the environments where delay and/or frame loss are embedded. In the upper panel of Figure 6, we plot the distribution of the throughput in the link with 500ms and 1000ms delay. As shown in this figure, the throughput in case D-1 is highly affected by embedded delay. The average throughput is approximately 487Mbps at 500ms delay and 254Mbps at 1000ms delay, respectively. This is because the definition of the average throughput is total transmission data size divided by the period in the transmission session, and thus the throughput is simply decrease by the value of the delay. In order to complete a transmission session, all bytes of the indicated date block have been transmitted and the receiver has received the red-part of the block. [3] To receive the red-part of the block, it is necessary to exchange the three messages, which are Check Point (CP) of end of red-part, Report Segment (RS) and Report Acknowledge segment (RA), between the LTP entities. For example, in case that delay (RTT) is equal to 500ms, additional 1500ms is required to achieve this process of communications. Therefore, removing this additional time from the period in the transmission session, we confirmed the average throughput is comparable to that of case N-1 which is based on the scenario without embedded delay and frame loss.

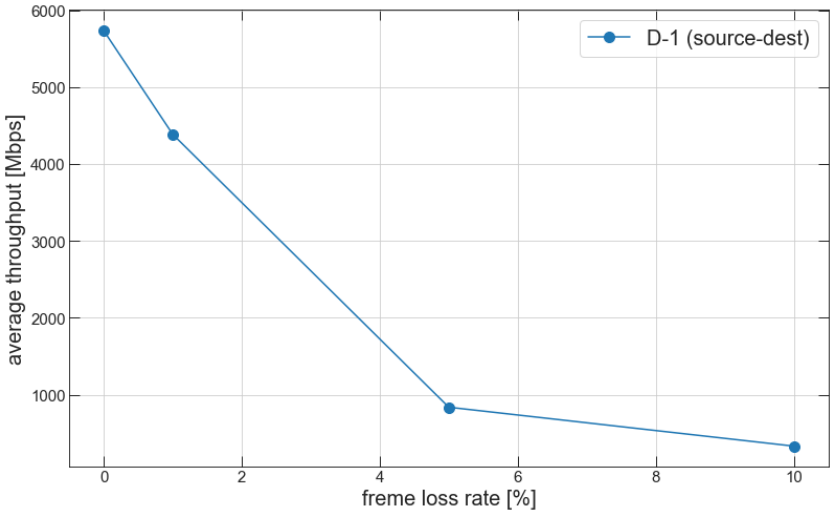
We plot the distribution of the average throughput in the link with 1%, 5% and 10% Ethernet frame loss in the middle panel of Figure 6. The throughput tends to decrease as the frame loss ratio increases. In the case of 1% Ethernet frame loss, the average throughput is approximately 4.3Gbps and the DTN implementation maintains high performance. The throughput decreases about 837Mbps and 330Mbps at 5% and 10% Ethernet frame loss, respectively.

The lower panel of Figure 6 represents the distribution of the average throughput in the link with 500ms delay and 5% Ethernet frame loss. In such environment, the throughput is approximately 211Mbps.

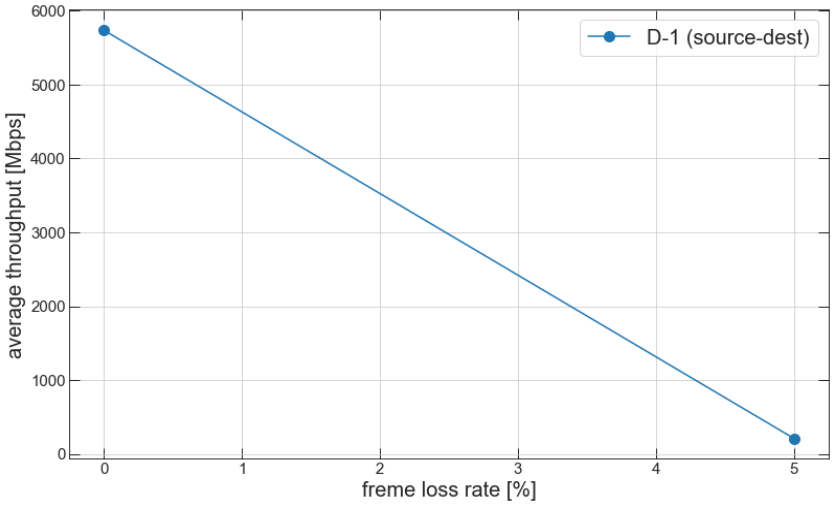
Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."



(a) Average throughput under delay (500ms-1000ms) environments



(b) Average throughput under frame loss (1-10%) environments



(c) Average throughput under 500ms delay and 5% frame loss environments

Fig. 6. Average throughput in the case D-1 and D-2

Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."

5. Discussion

To evaluate the effect of our acceleration approaches, we compare the throughputs of accelerated DTN implementation on the FPGA-based SoC (throughput = 5.7Gbps in case N-1) with that of the DTN implementation on the PCs (Raspberry Pi 1 and DELL Optiplex 7010) whose result is shown in Figure 2 (throughput = 700Kbps). Although the bandwidth is different, we confirmed the throughput of the DTN implementation on the SoC is significantly improved. In addition, focusing on the transfer efficiency for bandwidth which is the transmission data size divided by the bandwidth, the DTN implementation on the PC is only about 0.7% (throughput:700Kbps, bandwidth:100MBase), on the other hand, the transfer efficiency for bandwidth of the DTN implementation on the SoC is about 57% (throughput:5.7Gbps, bandwidth:10GBase). Therefore, results of the throughput and the transfer efficiency for bandwidth, we have successfully improved the performance of the DTN implementation.

In some of the acceleration approaches represented in Table 1 and Table 2, we confirmed the offloading approaches especially improve the performance such as throughput and transfer efficiency. Offloading the functions of BP and LTP to the FPGA resulted in reducing the CPU usage and consumption of time resource in RTOS. In addition, offloading the function of Ethernet as underlying layer protocol to an FPGA also contributes to reduce the CPU usage and consumption of time resource in RTOS.

The performance of the DTN implementation significantly improved, but there are still some problems. In our accelerated DTN implementation on the SoC, the CPU usage is approximately 100% to achieve over Gbps throughput. In order to make further improvement of the performance, it is assumed that the processing resource is insufficient. Therefore, as a future work, it is necessary to reduce the CPU usage by implementing a dual-core-compliant DTN software or offloading further functions of the DTN software. Considering the future DTN usage in various use cases, we need to confirm the further feasibility of the DTN implementation in various scenario cases, for example transferring multiple bundles, complex network topologies and various type of network environments with delay and disruption, in future studies. At present, JAXA started to develop a new BP implementation running on LINUX compatible with BP version 7 based on RFC 9171. [14] In near future, we will address to perform the acceleration of this BPv7 implementation based on the same approach presented in this paper.

6. Conclusion

In this paper, we have introduced results of an acceleration study of a DTN implementation based on a concept of software/hardware hybrid architecture. We have confirmed a feasibility of porting our implementation as a software program from on Linux to on RTOS. Then, we analysed the bottleneck of DTN software on the Linux and RTOS and reached some improving approach. Using the FPGA-based SoC, we have offloaded various functions, for example encoding and decoding of BP and LTP, and other functions including the network device driver on the RTOS to the FPGA. As the results, we have successfully improved the performance and confirmed the maximum average throughput achieves 5.7Gbps on 10Gigabit-Ethernet (57% as transfer efficiency for bandwidth). From those experimental facts, we have confirmed the acceleration approaches using the FPGA-based SoC for DTN implementation had a certain feasibility for higher rate use.

Acknowledgements

We thank Mr. Tadakatsu Abe and Mr. Susumu Tezuka for valuable comments and fruitful discussions.

References

[1] G.Araniti, et al., "Contact Graph Routing in DTN Space Networks: Overview, Enhancements and Performance", IEEE Communications Magazine, pp.38-46, March 2015.
[2] "CCSDS Bundle Protocol Specification", Recommendation for Space Data Systems Standards, CCSDS 734.2-B-1. Blue Book, September 2015.
[3] "Licklider Transmission Protocol(LTP) for CCSDS", Recommendation for Space Data Systems Standards, CCSDS 734.1-B-1. Blue Book. May 2015.
[4] "Scheduled-Aware Bundle Routing", Recommendation for Space Data Systems Standards, CCSDS 734.3-B-1. Blue Book. July 2019.
[5] S. Ogawa et al., Overview of standardization of Delay/Disruption Tolerant Networking (DTN) protocol suite and JAXA's DTN domestic production, SAT2017-51, 2017
[6] S. Burleigh et al., "Interplanetary Overlay Network: An Implementation of the DTN Bundle Protocol", IEEE International Conference on Communications, ICC 2011, Kyoto, Japan, 5-9 June, 2011

Please input the preferred copyright option as mentioned in the attached "*SpaceOps-2023 Copyright Policy for Manuscripts and Presentations*"
e.g. "Copyright ©2023 by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps. All rights reserved."

[7] K. Suzuki et al., JAXA-NASA Interoperability Demonstration for Application of DTN under Simulated Rain Attenuation, AIAA 2014-1920
[8] https://global.jaxa.jp/press/2022/01/20220127-1_e.html
[9] "_LunaNet Interoperability Specification Document version 4" September 12, 2022
[10] Tomohiro Araki, A trade-off study of lunar-earth optical communication links, Proc. SPIE 11852, International Conference on Space Optics, ICSO 2020
[11] T. Araki et al., "Recent R&D activities of the Lunar – the Earth Optical Communication Systems in Japan", IEEE ICSOS, pp. 32-35, March 2022
[12] S. Collange et al., "Power consumption of gpus from a software perspective," in International Conference on Computational Science, pp. 914–923, September, 2009
[13] "International Communication System Interoperability Standards (ICSIS)" March, 2019
[14] S. Burleigh et al., "Bundle Protocol Version 7", RFC 9171 January, 2022