



National Aeronautics and
Space Administration

HDTN User Guide for V2.0.0

Document No: HDTN-USER-036



Revision: 2.0.0
Effective Date: 7/18/2025

Space Operations Mission Directorate (SOMD)
Space Communications and Navigation (SCaN)
High-Rate Delay Tolerant Networking (HDTN) Project
John H. Glenn Research Center
CAGE Code No.: 1QFP5
21000 Brookpark Road, Cleveland, Ohio 44135

HDTN User Guide for V2.0.0



AUTHORIZED by CM when under FORMAL Configuration Control
Signature/Date

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 2 of 69

PREFACE

Space Communications and Navigation (SCaN) is developing new communications technologies to increase the amount of science data returned on future space missions. The High-Rate Delay Tolerant Networking (HDTN) project at NASA Glenn Research Center (GRC) will provide reliable internetworking as a high-speed path for moving data between spacecraft payloads, and across communication systems that operate on a range of different rates.

This document provides the user information regarding how to set up the configuration file(s) and contact plan(s) to initialize and/or use the HDTN architecture. It also provides the external communication lanes to connect HDTN's component(s) to customized or external applications.

The following government contributors aided in this document: Stephanie Booth, Rachel Dudukovich, Nadia Kortas, Ethan Schweinsberg, Brian Tomko, Blake LaFuente, Timothy Recker, Prash Choksi, and Shaun McKeehan.

The following non-government contributors aided in this document: Evan Danish and Tad Kollar.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 3 of 69

DOCUMENT HISTORY LOG

Status (Preliminary/ Baseline/ Revision/ Canceled)	Document Revision	Effective Date	Description
Baseline	1.0.0	01/18/2023	Initial Release of User Guide for v1.0.0
Revision	1.1.0	12/8/2023	<p>Fixed</p> <ul style="list-style-type: none"> - All “volatile” variables, especially “volatile bool”, have been replaced with “std::atomic” with proper memory-ordering semantics. - All convergence layer telemetry variables/counters use “std::atomic” with proper memory-ordering semantics. - Routing library now uses reloaded contact plans from the telemetry API. - Fixed bug where router link state information would become out of sync with actual link state. - Addressed minor custody transfer bug that was found during testing. <p>Added</p> <ul style="list-style-type: none"> - Bundle Protocol security (BPsec) support. “-bpsec-config-file” command-line argument was added to bpgen, bpsink, bpsendfile, bpreceivefile, and hdtm-one-process. This argument is used to specify BPsec config file location. - Added “slip_over_uart” convergence layer, allowing bidirectional communication and framing of bundles over a COM/Serial port. - Added “-cla-rate” command-line argument to bpgen. This argument, defaulting to zero for unlimited, can be used to set the rate for LTP and UDP connections. - Added config option “neighborDepletedStorageDelaySeconds” allowing for optional rerouting around neighboring nodes with depleted storage. Zero disables; otherwise, the value is interpreted as the amount of time to wait before forwarding to the neighbor after a depleted storage message is received. Requires that custody be enabled. - Added functionality to support API calls for retrieving configuration and statistics information of HDTN (refer to Section 8.0)

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 4 of 69

			<ul style="list-style-type: none"> - Added FPrime support by adding BpSendPacket and BpReceivePacket tools which can convert Fprime data to bundles and vice versa. <p>Changed</p> <ul style="list-style-type: none"> - All bundle data types use “padded_vector_uint8_t” instead of “std::vector<uint8_t>” to remain uniform across both inducts, outducts, and BundleView. This results in API changes for: <ul style="list-style-type: none"> ▪ “LtpClientServiceDataToSend” ▪ the outduct “Forward” calls ▪ the internal buffers of “BundleViewV6” and “BundleViewV7” - Added support for “BundleViewV6” and “BundleViewV7” to recycle their canonical block header objects whenever the bundle view object is reused in the creation or loading of bundles. - Combined router and scheduler into one module. - Updated routing logic. Minor bug fixes and improved handling of interrupted/failed contacts. Upon a failed contact, HDTN will attempt to calculate a new route avoiding the failed node. - Changed “-bundle-rate” argument to floating point type to allow for rates slower than one bundle-per- second. - Enabling BP version 6 style priority is optional in BP version 7. <p>Removed</p> <ul style="list-style-type: none"> - Removed “finalDestinationEidUris”, “udpRateBps”, and “ltpMaxSendRateBitsPerSecOrZeroToDisable” fields from the HDTN configuration. These values now come from the contact plan or command- line arguments.
Canceled	1.2.0		<p>Fixed</p> <ul style="list-style-type: none"> - BpSourcePattern can now receive non-custody-signal bundles for outduct convergence layers SlipOverUart and BpOverEncap. - Fixed BundleViews v6 and v7 bug when using a mixture of both payload admin records and payload non-admin records. - Building HDTN as shared libraries now works on all platforms.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 5 of 69

			<ul style="list-style-type: none"> - HDTN now cleanly exits when an LTP outduct or induct has a bad hostname. Stcp, TcpclV3, and TcpclV4 inducts would cause a “use after free” event when deleting a closed connection. - UdpBatchSender now works on Apple using a syscall to sendmsg x (the sendmmsg equivalent). - Windows now builds with warning level 4 (previously level 3) and treats all compiler warnings as errors; fixed all level 4 warnings. - Linux now builds with all warnings (-Wall, -Wextra, -Wpedantic); fixed all warnings. Only the CI/CD pipeline enables “Treat warnings as Errors” (-Werror option). - Fix LTP RedPartReceptionCallback where the isEndOfBlock parameter now reports if the EOB came from the red data (correct behavior) instead of whether the last segment received from any part of the red data was the EOB (previous incorrect behavior). - Fix LTP GreenPartSegmentArrivalCallback where the offsetStartOfBlock parameter incorrectly had the length of the block added to it. - Fix BPsec decoding mishandles larger parameter and result sizes (caused by the wrong type cast). <p>Added</p> <ul style="list-style-type: none"> - New supported native platforms (g++ or Clang). <ul style="list-style-type: none"> ▪ macOS Apple Silicon M2 on Ventura (using ARM CPU NEON instructions) ▪ macOS Intel x64 on Ventura ▪ FreeBSD Intel x64 ▪ OpenBSD Intel x64 ▪ Linux ARM64 (aarch64) (using ARM CPU NEON instructions) - HDTN now prints the Git commit hash (from which it was built) to the logs at initialization (in addition to the HDTN version). - Added the option to build the web interface with CivetWeb (statically linked without SSL support) instead of Boost Beast (default) in order to reduce HDTN binary file size. See the README for instructions. - Added experimental “bp_over_encap_local_stream” and “ltp_over_encap_local_stream” convergence layers, allowing HDTN to generate CCSDS
--	--	--	--

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 6 of 69

			<p>encap packets over a cross-platform local stream. On Windows, this is accomplished using a full-duplex named pipe. On Linux/POSIX, this is accomplished using a local AF_UNIX duplex socket.</p> <ul style="list-style-type: none"> - Added EncapRepeater.cpp demo application to serve as an example for writing code to intercept/extract CCSDS Encap packets from HDTN. - Add enforceBundlePriority config option; setting this enforces strict priority ordering of forwarded bundles. When true, HDTN will forward bundles by priority. This will have a performance impact, as it requires passing all bundles through storage. <p>Changed</p> <ul style="list-style-type: none"> - HDTN config now has mandatory boolean field enforceBundlePriority (default to false). - Egress now disables LTP ping during times when the contact plan DOES NOT allow transmission. Likewise, Egress will reenable LTP ping (back to its config file value) during times when the contact plan allows transmission. - UdpBatchSender no longer has its own thread and io_service due to now being completely asynchronous on all platforms; user provides an io_service to its constructor. The LtpEngine io_service is what runs the UdpBatchSender when using LTP over UDP when ltpMaxUdpPacketsToSendPerSystemCall config variable is greater than 1. - Windows now builds with warning level 4 (previously level 3) and treats all compiler warnings and linker warnings as errors. - Linux now builds with all warnings (-Wall, -Wextra, -Wpedantic).
Canceled	1.2.1		<p>Fixed</p> <ul style="list-style-type: none"> - Fixed a race condition in egress that caused dropped bundles. Handle the error where no outduct exists for a bundle in egress. Egress now returns that bundle to its sender. This condition is careful not to mark the link down in storage or ingress. - Fixed missing function calls in the router to recompute routes when using the link up and link down API calls. <p>Changed</p>

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 7 of 69

			<ul style="list-style-type: none"> - Write fatal level errors to a fatal level log file when CMake cache variable LOG_TO_ERROR_FILE=ON
Revision	1.3.0	05/23/2024	<p>HDTN Version 1.3.0 is a minor release with several features promoted out of the HDTN development branch to the master branch. The most notable additions are a new application to support multimedia streaming over Real-time Transport Protocol and a new configuration file editor has been added to the HDTN web interface. The HDTN user guide for version 1.3.0 includes updates for version 1.2.0. A user guide specific to version 1.2.0 was not released due to the short time span between the release of HDTN version 1.1.0, 1.2.0, and 1.3.0. At the time of the HDTN v1.3.0 minor release, the software and engineering processes are undergoing improvements to reach NASA Procedural Requirements 7150.2D Class B compliance. HDTN is currently Class D. Full Class B compliance is planned for HDTN v2.0.0 in the second half of 2024. New features added to the master branch are considered stable, however additional testing will be conducted prior to the release of HDTN v2.0.0.</p> <p>Fixed</p> <ul style="list-style-type: none"> - Fixed a re-route issue related to the API command for updating a link up/down and link state. <p>Added</p> <ul style="list-style-type: none"> - Added support for multimedia streaming over HDTN.
Canceled	1.3.1	11/27/2024	<p>Changed</p> <ul style="list-style-type: none"> - Ported LaTeX User Guide to Word Format. - Changed to CM templated for HDTN. - Changed sentence “At the same time, HDTN defines a new internal messaging format better suited to higher-rate operation.” In section 1.0 to “At the same time, HDTN defines a new data format better suited to higher-rate operation.”. - Summarized supported convergence layers under Convergence Layers and Routing Protocols - Section BP verbiage in Section 20.2. - Figure numbering. - Moved all authors to the TM title page and preface of document. - Added to section 5.4.2. - Chapter 13 introduction section was clarified.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 8 of 69

			<ul style="list-style-type: none"> - Changed GUI picture to reflect the revision number. <p>Removed</p> <ul style="list-style-type: none"> - Sections 3.1-3.6 - Section 10 subsections and were summarized in one section - Section 15.1 - Sections 20.2.1-20.2.2 <p>Added</p> <ul style="list-style-type: none"> - Added MacOS installation and dependencies
Revision	2.0.0	07/18/2025	<p>Changed</p> <ul style="list-style-type: none"> - Rewrote Section 1.0. - Added in Windows 11 support - Made section 5.1 more readable - Updated description of the “max-bundle-size-bytes” parameter for the BpSendFile (Section 12.10) and BpSendPacket (Section 12.12) - Updated description of the “max-rx-bundle-size-bytes” parameter for the BpReceiveFile (Section 12.3) and BpReceivePacket (Section 12.13)

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 9 of 69

SIGNATURE PAGE

Prepared By:

Robert Jones Digitally signed by Robert Jones
Date: 2025.07.16 10:49:41 -04'00'

Signed on behalf of
Stephanie L. Booth
HDTN Software Developer
NASA John H. Glenn Research Center

Concurred By:

Jose Lombay-gonzalez Digitally signed by Jose Lombay-gonzalez
Date: 2025.07.16 13:29:39 -04'00'

José Lombay-González
HDTN Software Lead
NASA John H. Glenn Research Center

Rachel Dudukovich Digitally signed by Rachel Dudukovich
Date: 2025.07.17 08:55:00 -04'00'

Rachel M. Dudukovich
HDTN Software Engineering Lead
NASA John H. Glenn Research Center

Daniel Raible Digitally signed by Daniel Raible
Date: 2025.07.17 10:20:45 -04'00'

Daniel Raible
HDTN Principal Investigator
NASA John H. Glenn Research Center

Approved By:

John Nowakowski Digitally signed by John Nowakowski
Date: 2025.07.17 11:44:00 -04'00'

John J. Nowakowski
HDTN Project Manager
NASA John H. Glenn Research Center

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 10 of 69

TABLE OF CONTENTS

1.0	HIGH-RATE DELAY TOLERANT NETWORKING OVERVIEW	13
2.0	DOCUMENTS	14
2.1	Applicable Documents.....	14
2.2	Reference Documents	14
3.0	ARCHITECTURE.....	15
4.0	BUILD ENVIRONMENT REQUIREMENTS	16
4.1	Tested Platforms	16
4.2	Dependencies	16
4.2.1	Linux Dependencies	16
4.2.2	MacOS Dependencies.....	17
4.2.3	Windows Dependencies.....	17
4.3	Limitations	17
5.0	HDTN BUILD PROCEDURES.....	18
5.1	Notes on HDTN CMake	18
5.2	Build HDTN on Linux	18
5.2.1	Optional x86 Hardware Acceleration.....	19
5.2.2	Storage Capacity Compilation Parameters	19
5.3	Build HDTN on Mac	20
5.4	Build HDTN on Windows with its Dependencies	20
5.4.1	HDTN Developers	21
5.4.2	Setup Instructions for Developers Using Installed HDTN Libraries within their own Projects.....	23
5.5	Build HDTN on ARM	24
5.5.1	Debugging Errors/Problems	24
5.6	Building for ARM on x86.....	24
5.6.1	Setting up ARM Chroot on x86 Desktop.....	24
5.6.2	Setting up HDTN Dependencies in the Chroot Environment.....	24
5.6.3	Compiling HDTN	25
5.6.4	Useful Commands.....	25
6.0	RUNNING HDTN.....	25
6.1	Directory Structure.....	26
6.2	Unit Tests.....	26
6.3	Integrated Tests.....	26
7.0	GRAPHICAL USER INTERFACE	26
7.1	Running the Web Interface	26
7.2	Statistics Page	26
7.3	System View Page	27
7.4	Config Page.....	28
7.5	Statistics Logging.....	28
8.0	GETTING STARTED WITH THE API	29
8.1	API Calls.....	29
8.1.1	HDTN Version	29
8.1.2	HDTN Configuration.....	29
8.1.3	Storage	30
8.1.4	Expiring Storage	30
8.1.5	Inducts.....	30
8.1.6	Outducts	31
8.1.7	Maximum Send Rate for an Outduct	31
8.1.8	BP Security Configuration.....	32

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 11 of 69

8.1.9	Upload Contact Plan	32
8.1.10	Ping.....	33
8.1.11	Take Link Down	33
8.1.12	Bring Link Up.....	34
9.0	COMMAND LINE INTERFACE.....	34
9.1	Usage.....	34
9.2	Examples.....	34
10.0	SIMULATIONS	35
11.0	HDTN APPLICATIONS.....	35
12.0	RUNSCRIPT	36
12.1	Path Variables	36
12.2	BpSink.....	36
12.3	BpReceiveFile.....	37
12.4	Egress.....	38
12.5	Router.....	38
12.6	Ingress.....	39
12.7	Storage	39
12.8	HDTN One Process.....	40
12.9	BpGen	40
12.10	BpSendFile.....	41
12.11	Bping.....	42
12.12	BpSendPacket	43
12.13	BpReceivePacket	44
12.14	CleanUp	44
13.0	CONFIG FILES.....	45
13.1	hdtm_config.....	45
13.2	sink_config.....	48
13.3	gen_config.....	48
13.4	bpsec_config	49
13.5	distributed_config.....	52
14.0	CONTACT PLANS.....	54
14.1	JSON Fields	54
15.0	CONVERGENCE LAYERS AND ROUTING PROTOCOLS	55
15.1	TCPCLv3	55
15.2	TCPCLv4.....	55
15.3	UDPCL	56
15.4	LTP	56
15.5	STCP	58
16.0	TEST CONFIGURATIONS AND INSTRUCTIONS	58
16.1	TCP Loopback Test	58
16.2	Two Node LTP Test.....	59
16.3	Four Nodes STCP Test	59
16.4	File Transfer Test.....	60
16.5	Integrated Tests.....	61
17.0	GETTING STARTED WITH STREAMING VIA HDTN	61
17.1	Configuration Parameters for BPSendStream.....	61
17.1.1	Example JSON Configuration File for BPSendStream	61
17.2	Configuration Parameters for BPrecevStream	62
17.2.1	Example JSON Configuration File for BPrecevStream	62
17.3	Runscripts for Streaming Scenarios.....	63

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 12 of 69

17.3.1	File Streaming.....	63
17.3.2	Live Streaming.....	63
17.4	More Details.....	63
18.0	CONTAINERIZATION.....	63
18.1	Docker Instructions.....	63
18.2	Docker Compose Instructions.....	64
18.3	Kubernetes Instructions.....	64
19.0	TROUBLESHOOTING.....	65
19.1	Logging.....	65
19.2	LTP Tuning Recommendations.....	65
20.0	NOTES.....	66
20.1	TLS Support for TCPCL Version 4.....	66
20.2	BP Version 6 and Version 7.....	66
	APPENDIX A Acronyms.....	67
	APPENDIX B TBD, TBR, and TBW Table.....	69

TABLE OF TABLES

Table 2-1:	Applicable Documents.....	14
Table 4-1:	Commercial Off-the-Shelf Utilities and Supporting Files.....	16
Table 12-1:	BpSink Instantiation Parameters.....	37
Table 12-2:	BpReceiveFile Instantiation Parameters.....	37
Table 12-3:	Egress Instantiation Parameters.....	38
Table 12-4:	Router Instantiation Parameters.....	39
Table 12-5:	Ingress Instantiation Parameters.....	39
Table 12-6:	Storage Instantiation Parameters.....	40
Table 12-7:	HDTN One Process Instantiation Parameters.....	40
Table 12-8:	BPGen Instantiation Parameters.....	41
Table 12-9:	BpSendFile Instantiation Parameters.....	42
Table 12-10:	Bping Instantiation Parameters.....	43
Table 12-11:	BpSendPacket Instantiation Parameters.....	43
Table 12-12:	BpReceivePacket Instantiation Parameters.....	44

TABLE OF FIGURES

Figure 3-1:	HDTN Architecture.....	15
Figure 5-1:	LTP CMake Example.....	23
Figure 7-1:	Statistics Page of the Web Interface.....	27
Figure 7-2:	System View Page of the Web Interface.....	28
Figure 7-3:	Pop-up Graphic Displaying Data from the Storage Module.....	28
Figure 11-1:	HDTN Application Architecture.....	36
Figure 16-1:	HDTN Loopback Test.....	59
Figure 16-2:	HDTN Two Nodes Test.....	59
Figure 16-3:	HDTN Four Nodes STCP Routing Test.....	60
Figure 16-4:	HDTN File Transfer with Confidentiality Test.....	61

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 13 of 69

1.0 HIGH-RATE DELAY TOLERANT NETWORKING OVERVIEW

The High-Rate Delay Tolerant Networking (HDTN) project stems from Delay Tolerant Networking (DTN) but takes advantage of modern hardware platforms to substantially reduce latency and improve throughput. This enables the core actions of DTN (i.e. store, carry, and forward) on disparate links by utilizing the DTN's Bundle Protocol (BP). HDTN is able to achieve an efficient DTN structure by its massively parallel pipelined and message-oriented architecture, allowing the system to scale gracefully as its resources increase. For questions and comments on this project, feel free to reach out to the contributors found on the Github page at <https://github.com/nasa/HDTN>.

2.0 DOCUMENTS

Documents listed in this section are the standards, specifications, handbooks, and other publications that guide this document.

2.1 Applicable Documents

Applicable documents are either cited in the body of this document or contain provisions directly related to, and necessary for, the performance of the activities described herein.

Table 2-1: Applicable Documents

Document Number	Revision	Document Title	Effective Date	Last Updated
HDTN-REQ-008	A	HDTN Software Requirements Specification (SRS)	2024-09-11	2024-09-11
RFC 4838	1	Delay-Tolerant Networking Architecture	2007-04-01	2007-04-01
RFC 5050	1	Bundle Protocol Specification	2007-11-01	2007-11-01
RFC 7116	2	Licklider Transmission Protocol (LTP), Compressed Bundle Header Encoding (CBHE), and Bundle Protocol IANA Registries	2014-02-01	2022-06-22
RFC 7242	1	Delay-Tolerant Networking TCP Convergence-Layer Protocol	2014-06-13	2014-06-13
RFC 9171	2	Bundle Protocol Version 7	2022-01-01	2022-12-14
RFC 9174	1	Delay-Tolerant Networking TCP Convergence-Layer Protocol Version 4	2022-01-31	2022-01-31

2.2 Reference Documents

Reference documents are those that **may** assist in understanding the intent of this document but are not required to execute the processes contained in this document.

NASA. (2024). *NASA/HDTN: High-rate delay Tolerant Network (HDTN) software*. High-rate Delay Tolerant Network (HDTN) Software. <https://github.com/nasa/HDTN>

OpenJS Foundation. (2024). *Node.js - Download node.js®*. Node.js. <https://nodejs.org/en/download/package-manager>

Fraire, J. (n.d.). *dtnsim* [Review of *dtnsim*]. Bitbucket. Retrieved 2024, from <https://bitbucket.org/lcd-unc-ar/dtnsim/src/master/>

NASA. (2022). *Installing and configuring HDTN simulator*. Installing and Configuring HDTN Simulator. https://docs.google.com/document/d/1KrKyO_pr-v9CeS5n_ectpfWtwkYL40PdLICGh-24zZY/edit?tab=t.0#heading=h.1fly6dvqx3j3

NASA/HDTN. (2023). *runscript_distributed.SH at master*. Runscript Distributed File. https://github.com/nasa/HDTN/blob/master/tests/test_scripts_linux/runscript_distributed.sh

NASA/HDTN. (2023). *HDTN_DISTRIBUTED_DEFAULTS.JSON*. HDTN Distributed Defaults File. https://github.com/nasa/HDTN/blob/master/config_files/hdt/hdt_distributed_defaults.json

NASA/HDTN. (2023). *Routing_Test*. Routing Test Files. https://github.com/nasa/HDTN/tree/master/tests/test_scripts_linux/Routing_Test

4.0 BUILD ENVIRONMENT REQUIREMENTS

In this section, the run environments, including tested platforms, architectures, and dependencies are detailed.

4.1 Tested Platforms

- OS
 - Linux
 - Ubuntu Desktop 18.04, 18.10, 20.04.2, 20.10
 - Ubuntu Server 20.04, 20.10, 22.04
 - Debian 10, 11, and 12
 - Raspbian
 - Red Hat Enterprise Linux (RHEL) 8
 - Oracle Enterprise Linux (OEL) 8
 - Windows
 - Windows 11 (64-bit)
 - Windows 10 (64-bit)
 - Windows Server 2022 (64-bit)
 - Windows Server 2019 (64-bit)
 - MacOS 14
 - FreeBSD 14
 - OpenBSD 7
- Chipset
 - ARM32
 - ARM64
 - x86_64

4.2 Dependencies

4.2.1 Linux Dependencies

See Table 4-1 for what the HDTN build environment requires.

Table 4-1: Commercial Off-the-Shelf Utilities and Supporting Files

Developer	Utility or Library	Version	Release Date	Source	Notes
CMake.org	CMake	3.16.3	2020 JAN 21	https://github.com/Kitware/CMake	
Boost.org	Boost	1.66.0	2017 DEC 18	https://www.boost.org	Included with Red Hat Linux 8
Boost.org	Boost	1.69.0	2018 DEC 12	https://www.boost.org	Required for TCPCLv4 TLS v 1.3
Boost.org	Boost	1.70.0	2019 APR 12	https://www.boost.org	Required for HTTPS/WSS support for Web User Interface
ZeroMQ.org	ZeroMQ	4.34	2021 JAN 17	https://github.com/zeromq/libzmq	

GNU.org	gcc	9.3.0	2020 MAR 12	https://gcc.gnu.org	Debian 8.3.0-6
OpenSSL.org	OpenSSL	1.1.1f	2020 MAR 31	https://www.openssl.org	Optional; needed to support BPSec
Adam Kennedy	Strawberry Perl	5.32.1.1	2021 JAN 24	https://strawberryperl.com	
GStreamer Team	GStreamer	1.20.0	2022 FEB 03	https://gstreamer.freedesktop.org	Required for streaming

4.2.2 MacOS Dependencies

Install Homebrew, which is a package manager for macOS

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh) "
```

****NOTE:** Homebrew is installed at "/opt" for Apple Silicon-based Macs, but it is installed at "/usr/local" for intel-based Macs. This user guide will reference as if Homebrew was installed at "/opt".

Add in your .zprofile

```
eval "$(/opt/homebrew/bin/brew shellenv)"
export CPATH="$HOMEBREW_PREFIX/include"
```

Install ZeroMQ, OpenSSL and Boost

- ZeroMQ: brew install zmq
- OpenSSL and gnutls: brew install openssl gnutls
- Boost version 1.85: brew install [boost@1.85](#)

4.2.3 Windows Dependencies

HDTN supports 9 permutations of the Visual Studio compilers on Windows:

- Versions: 2022, 2019, and 2017 (note: for 2017, only versions 15.7 and 15.9 have been tested)
- Editions: Enterprise, Professional, and Community

4.3 Limitations

- Ubuntu distributions have been known to install older non-compatible versions of -CMake.
- Some processors do not support hardware acceleration or the RDSEED instruction. (Note: In the CMake file, both are set to "ON" by default.) This support will be auto-detected by CMake if not cross-compiling.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 18 of 69

5.0 HDTN BUILD PROCEDURES

5.1 Notes on HDTN CMake

- All of HDTN’s modules/libraries contain their own CMakeLists.txt file.
- The root CMakeLists.txt file adds all modules/libraries to the HDTN project using the CMake `add_subdirectory` command.
- Package config information is exported to the installation directory (`install root/lib/cmake`) when a “make install” is performed.
 - An example of this can be found in `tests/unit_tests_import_installation/CMakeLists.txt`
- Package config information is for writing custom software projects that only use portions of the HDTN codebase such as a library of a particular convergence layer.
- HDTN CMake optimizes the build
 - Tests the compiler for more recent C++ standards
 - Tests the compiler and the CPU for specific x86 hardware instructions and utilizes those, if available
 - For operational usage, the CMake “Release mode” is recommended.
 - For troubleshooting, the CMake “Debug mode” is covered in Section 19.0.
- HDTN CMake supports building its libraries as either static or shared
 - Uses CMake’s `GENERATE_EXPORT_HEADER`
 - Required for building or using .dll files on Windows
 - Required for using GCC C++ visibility support

5.2 Build HDTN on Linux

The dependencies can be installed using the following command(s):

On Debian/Ubuntu

```
sudo apt-get install cmake build-essential libzmq3-dev
sudo apt-get install libboost-dev libboost-all-dev openssl libssl-dev
sudo apt install git-all
```

On RHEL/OEL

```
sudo dnf install epel-release
sudo yum install cmake boost-devel zeromq zeromq-devel
sudo dnf install git-all
```

To build HDTN in “Release mode”, perform the following steps. (Note: If the `-DCMAKE_BUILD_TYPE` is not specified, HDTN is built in “Release mode” by default).

- `git clone https://github.com/nasa/HDTN.git`
- `export HDTN_SOURCE_ROOT=/home/username/HDTN` (set to filepath containing HDTN)
- `cd $HDTN_SOURCE_ROOT`
- `mkdir build`
- `cd build`
- `cmake ..`
- open CMakeCache.txt
 - To build HDTN static
 - Set `BUILD_SHARED_LIBS:BOOL=OFF`
 - `CMAKE_CXX_FLAGS_RELEASE:STRING=-O3 -DNDEBUG`

Uncontrolled when printed. Verify that this is correct version before use.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 19 of 69

- To build HDTN shared libraries
 - Set `BUILD_SHARED_LIBS:BOOL=ON`
 - `CMAKE_CXX_FLAGS_RELEASE:STRING=-O3 -DNDEBUG -fPIC`
- `make`
 - Adding `-j8` (i.e. `make -j8`) to the `make` will speed up the processing time but requires a system with at least 8 cores.

5.2.1 Optional x86 Hardware Acceleration

HDTN build environment sets the following CMakeCache variables to “On” by default:

`USE_X86_HARDWARE_ACCELERATION` and `LTP_RNG_USE_RDSEED`.

Notes:

- If building natively (i.e. not cross-compiling), the HDTN CMake build environment will check the processor’s CPU instruction set and the compiler to determine which HDTN hardware accelerated functions will build and run on the native host. CMake automatically sets various compiler definitions to enable supported HDTN hardware accelerated features.
- If cross-compiling, the HDTN CMake build environment will check the compiler to determine if the HDTN hardware accelerated functions will build. It is up to the user to determine if the target processor can support/run those instructions. CMake will automatically set various compiler definitions to enable supported HDTN hardware accelerated features only if they compile.
- Hardware accelerated functions can be turned off by setting `USE_X86_HARDWARE_ACCELERATION` and/or `LTP_RNG_USE_RDSEED` to “Off” in the `CMakeCache.txt`.
- If building for ARM or any non X86-64 platform, `USE_X86_HARDWARE_ACCELERATION` and `LTP_RNG_USE_RDSEED` must be set to “Off”.

If CMake finds support for the CPU instructions and `USE_X86_HARDWARE_ACCELERATION` is turned “On” then some or all of the following features will be enabled:

- Fast SDNV encode/decode (BPv6, TCPCLv3, and LTP) requires SSE, SSE2, SSE3, SSSE3, SSE4.1, POPCNT, BMI1, and BMI2.
- Fast batch 32-byte SDNV decode (not yet implemented into HDTN but available in the common/util/Sdnv library) requires AVX, AVX2, and the above “Fast SDNV” support.
- Fast CBOR encode/decode (BPv7) requires SSE and SSE2.
- Some optimized loads and stores for TCPCLv4 requires SSE and SSE2.
- Fast CRC32C (BPv7 and a storage hash function) requires SSE4.2.
- The HDTN storage controller will use BITTEST if available. If BITTEST is unavailable, it will use ANDN if BMI1 is available.

If `LTP_RNG_USE_RDSEED` is turned “On”, this feature will be enabled if CMake finds support for this CPU instruction:

- An additional randomness source for LTP’s random number generator requires RDSEED. This feature can be disabled for potentially faster LTP performance.

5.2.2 Storage Capacity Compilation Parameters

HDTN build environment sets two CMake cache variables by default:

`STORAGE_SEGMENT_ID_SIZE_BITS` and `STORAGE_SEGMENT_SIZE_MULTIPLE_OF_4KB`.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 20 of 69

- The `STORAGE_SEGMENT_ID_SIZE_BITS` flag must be set to the recommended default, 32 or 64. It determines the size/type of the storage module's `segment_id_t`. Setting the flag to 32-bit significantly decreases memory usage because we don't need to use the RAM.
 - If this value is 32, the formula for the max segments (S) is given by

$$S = \min(\text{UINT32_MAX}, 64^6) \approx 4.3 \text{ billion}$$
 segments since `segment_id_t` is a `uint32_t`. A segment allocator using 4.3 Billion segments uses about 533 MByte RAM), and multiplying by the minimum 4KB block size gives 17TB bundle storage capacity. Make sure to appropriately set the `totalStorageCapacityBytes` variable in the HDTN JSON config so that only the required amount of memory is used for the segment allocator.
 - If this value is 64, the formula for the max segments (S) is given by

$$S = \min(\text{UINT64_MAX}, 64^6) \approx 68.7 \text{ billion}$$
 segments since `segment_id_t` is a `uint64_t`. Using a segment allocator with 68.7 Billion segments, when multiplying by the minimum 4KB block size gives ~281TB bundle storage capacity.
- The flag `STORAGE_SEGMENT_SIZE_MULTIPLE_OF_4KB` must be set to an integer of 1 or greater. It determines the minimum increment of bundle storage based on the standard block size of 4096 bytes. (Note: One is the default and recommended.) Example:
 - If `STORAGE_SEGMENT_SIZE_MULTIPLE_OF_4KB=1`, a `4KB*1=4KB` block size is used. A bundle size of 1KB would require 4KB of storage. A bundle size of 6KB would require 8KB of storage.
 - If `STORAGE_SEGMENT_SIZE_MULTIPLE_OF_4KB=2`, a `4KB*2=8KB` block size is used. A bundle size of 1KB would require 8KB of storage. A bundle size of 6KB would require 8KB of storage. A bundle size of 9KB would require 16KB of storage. If `STORAGE_SEGMENT_ID_SIZE_BITS=32` then bundle storage capacity could potentially double from ~17TB to ~34TB.

For information on how the Storage works, see `module/storage/doc/storage.pptx` in the HDTN main repository.

5.3 Build HDTN on Mac

Be sure to have installed the MacOS dependencies described in section 4.2.2.

1. `mkdir -p build`
2. `export LDFLAGS=-L$HOMEBREW_PREFIX/opt/boost@1.85/lib`
3. `export CPPFLAGS=-I$HOMEBREW_PREFIX/opt/boost@1.85/include`
4. `cmake -S . -B build -DCMAKE_CXX_FLAGS="-Werror" -DCMAKE_C_FLAGS="-Werror" -DUSE_X86_HARDWARE_ACCELERATION:BOOL=ON -DLTP_RNG_USE_RDSEED:BOOL=OFF -Dlibzmq_LIB:FILEPATH=$HOMEBREW_PREFIX/lib/libzmq.dylib -Dlibzmq_INCLUDE:PATH=$HOMEBREW_PREFIX/include`
5. `make -C build/ -j8`
6. `export HDTN_SOURCE_ROOT=$PWD`
7. `./build/tests/unit_tests/unit-tests`

5.4 Build HDTN on Windows with its Dependencies

HDTN build environment on Windows requires:

- One of the supported Visual Studio compilers. Visual Studio must be installed for C++ Development during setup.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 21 of 69

- PowerShell (recommended Visual Studio Code with the PowerShell extension installed)
- 7-Zip
- Perl (needed for building OpenSSL) with perl.exe in the Path environmental variable (Strawberry Perl for Windows has been tested)

To build HDTN and its dependencies in Release mode and as shared libraries (shared .dll files for both HDTN and its dependencies), simply run the PowerShell script in `building_on_windows\hdt_n_windows_cicd_unit_test.ps1` from any working directory. The working directory does not matter. Once finished, HDTN and its dependencies will be installed to `C:\hdt_n_build_x64_release_vs2022` (suffix will be 2019 or 2017 if that's the Visual Studio compiler installed). The script will also run HDTN's unit tests after the build. Once completed, you will see the following message:

"Remember, HDTN was built as a shared library, so you must prepend the following to your Path so that Windows can find the DLL's of HDTN and its dependencies:"

It will print four directory locations to be added to your Path environmental variable to facilitate use of HDTN outside this PowerShell script.

- From the Windows Start Menu, type `env`
- Open Edit environmental variables for your account
- Double click Path
- Add the four directories. (Omit the directory containing `hdt_n_install\lib` if modifying HDTN source code within Visual Studio. You will later build and install your HDTN binaries within Visual Studio.)
- If you are a user of HDTN and you are NOT going to modify HDTN source code within Visual Studio, also add this directory to your Path:
`C:\hdt_n_build_x64_release_vs2022\hdt_n_install\bin`
- Click OK
- Click New
- Add the following new variable: `HDTN_SOURCE_ROOT`
- Set the variable value to your source root (the folder that contains README.md).
Example `C:\path\to\hdt_n`
- Click OK
- Click OK

If you are a user of HDTN and you are NOT going to modify HDTN source code within Visual Studio, you can reference any of the .bat file example tests located in `HDTN_SOURCE_ROOT\tests\test_scripts_windows`. Note that these scripts were intended for developers, so you will have to modify the scripts, fixing any lines that reference `HDTN_BUILD_ROOT`, so, for example, if you see `%HDTN_BUILD_ROOT%\common\bpcodec\apps\bpgen-async.exe`, replace it with `bpgen-async.exe`. Also note that these .bat files reference config files located in `HDTN_SOURCE_ROOT\config_files`, so feel free to modify those .json configs to meet your needs.

5.4.1 HDTN Developers

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 22 of 69

If you are a developer and you are going to modify HDTN source code within Visual Studio, you may delete the directory `C:\hdt_n_build_x64_release_vs2022\hdt_n_install` and continue on with the next set of instructions.

Launch Visual Studio 2022 and open HDTN as a project with these steps:

- File >> open >> cmake
- Open HDTN root CMakeLists.txt
- Make sure drop down configuration at the top is set to x64-Release. You may need to go to Manage Configurations if not.

Then click Project >> view CMakeCache.txt Add these lines (change vs2022 directory suffix if different):

- BOOST_INCLUDEDIR:PATH=C:\hdt_n_build_x64_release_vs2022\boost_1_78_0_install
- BOOST_LIBRARYDIR:PATH=C:\hdt_n_build_x64_release_vs2022\boost_1_78_0_install\lib64
- BOOST_ROOT:PATH=C:\hdt_n_build_x64_release_vs2022\boost_1_78_0_install
- OPENSSL_INCLUDE_DIR:PATH=C:\hdt_n_build_x64_release_vs2022\openssl-1.1.1s\install\include
- OPENSSL_ROOT_DIR:PATH=C:\hdt_n_build_x64_release_vs2022\openssl-1.1.1s\install
- libzmq_INCLUDE:PATH=C:\hdt_n_build_x64_release_vs2022\libzmq_v4.3.4_install\include
- libzmq_LIB:FILEPATH=C:\hdt_n_build_x64_release_vs2022\libzmq_v4.3.4_install\lib\libzmq-v143-mt-4_3_4.lib (note: may be v141 or v142)
- BUILD_SHARED_LIBS:BOOL=ON

Then click Project >> configure cache

It is now time to set up additional environmental variables in order to be able to run the .bat file tests located in `HDTN_SOURCE_ROOT\tests\test_scripts_windows`:

- Right click on the open tab within Visual Studio titled CMakeCache.txt and then click "Open Containing Folder"
- Copy the path at the top of the Windows Explorer window
- From the Windows Start Menu, type "env", open "Edit environmental variables for your account"
- Click New
- Add the following new variable: HDTN_BUILD_ROOT. The variable value will look something like `C:\Users\username\CMakeBuilds\17e7ec0d-5e2f-4956-8a91-1b32467252b0\build\x64-Release`
- Click OK
- Click New
- Add the following new variable: HDTN_INSTALL_ROOT; the value will look similar to HDTN_BUILD_ROOT except change "build" to "install". something like `C:\Users\username\CMakeBuilds\17e7ec0d-5e2f-4956-8a91-1b32467252b0\install\x64-Release`

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 23 of 69

- Click OK
- Double click the Path variable, add the HDTN_INSTALL_ROOT\lib folder to your Path, something like C:\Users\username\CMakeBuilds\17e7ec0d-5e2f-4956-8a91-1b32467252b0\install\x64-Release\lib. This step is needed because HDTN is built as a shared library with multiple .dll files, so this step allows Windows to find those .dll files when running any HDTN binaries.

Relaunch Visual studio so that it gets loaded with the updated environmental variables. Now build HDTN:

- Build >> Build All
- Build >> Install HDTN
- Run unit_tests.bat located in HDTN_SOURCE_ROOT\tests\test_scripts_windows
- For a Web GUI example, run test_tcpcl_fast_cutthrough_oneprocess.bat and then navigate to <https://localhost:8086> (note: to exit cleanly, do a ctrl-c in each cmd window before closing)

NOTE: Since CMake is currently configured to build HDTN as a shared library (because the CMake cache variable BUILD_SHARED_LIBS is set to ON), any time you make a source code change to HDTN, for it to be reflected in the binaries, don't forget to Build >> Install HDTN after the Build >> Build All step.

5.4.2 Setup Instructions for Developers Using Installed HDTN Libraries within their own Projects

HDTN utilizes modern CMake. When HDTN is installed, it installs the appropriate CMake Packages that can be imported. For an example of this use case, see

HDTN_SOURCE_ROOT\tests\unit_tests_import_installation\CMakeLists.txt for a project that imports the libraries and headers from an HDTN installation and builds HDTN's unit tests from that installation.

All libraries can be imported as a shared or static library from a built and installed HDTN. Figure 5-1 shows the cmake script to import the LTP library.

```
cmake_minimum_required(VERSION 3.12)

#set find_package search
SET(CMAKE_PREFIX_PATH $ENV{HDTN_INSTALL_ROOT})
|  CACHE PATH "Path for find_package to import installed hdtm libraries"
| )

find_package(LtpLib REQUIRED)
add_executable(my-app
|   src/main.cpp
| )
target_link_libraries(my-app
|   HDTN::LtpLib
| )
```

Figure 5-1: LTP CMake Example

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 24 of 69

5.5 Build HDTN on ARM

HDTN is compiled with x86 hardware optimizations by defaults. To compile for ARM, these optimizations must be disabled. To build HDTN on ARM running Ubuntu (follow Ubuntu dependencies in Sections 4.2 and 5.2):

1. `cd HDTN`
2. `mkdir build && cd build`
3. `cmake .. -DCMAKE_SYSTEM_PROCESSOR=arm`
4. `make -j`
5. `sudo make install -j`

These commands should work for any ARM platform running Ubuntu such as Raspberry Pi 4 or Nvidia Jetson Nano. Compiling on a Raspberry Pi 4 takes anywhere from 30 minutes to an hour. It is recommended to compile with only one or two cores (`make -j2`) on memory constrained devices to ensure `gcc` does not consume all RAM.

5.5.1 Debugging Errors/Problems

- For errors reporting similar to: `cpuid.h is not found for file HDTN/common/util/src/CpuFlagDetection.cpp`
 - Double check `CMakeList.txt` edits
- For errors reporting similar to: recompile with `-fPIC`
 - Double check `CMakeCache.txt` edits
- For errors reporting `./runscript.sh not found`
 - Run: `export HDTN_SOURCE_ROOT=/home/user/HDTN`
- For errors reporting similar to: `no tcpdump`
 - Run: `sudo apt install tcpdump`
- For runtime errors:
 - Check the log files under `HDTN/logs`. These are not created by default but can be created following the instructions in Section 19.1.

5.6 Building for ARM on x86

5.6.1 Setting up ARM Chroot on x86 Desktop

Run the following commands:

- `sudo apt install qemu-user-static`
- `sudo apt install debootstrap`
- `sudo qemu-debootstrap--variant=buildd--archarm64focal/var/chroot/http://ports.ubuntu.com/`
 - `focal` in the above command is the name of the Ubuntu Release and may need to be changed.
 - This final command creates an `armhf` operating system located at `/var/chroot`. Users can move it elsewhere, but it is recommended to keep it out of the `/home/` and `user` directories.
- To get into chroot, use the command: `sudo chroot /var/chroot`

At this point, users are now in an ARM environment. Users should run the commands in the following sections AFTER they enter the ARM environment.

5.6.2 Setting up HDTN Dependencies in the Chroot Environment

Note: Sudo does not exist in chroot

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 25 of 69

- `apt install make cmake build-essential software-properties-common`
- `add-apt-repository universe`
- `apt update`
- `apt install libboost-dev libboost-all-dev libzmq3-dev openssl libssl-dev`

5.6.3 Compiling HDTN

- Download the latest HDTN from Github.
- Unzip the file in your home directory.
 - Note: Users cannot write directly to ARM emulator directories in Windows Subsystem for Linux.
- `sudo mv HDTN /var/chroot/home`
- `sudo chroot /var/chroot`
- `cd home/HDTN`
- `mkdir build`
- `cd build`
- `cmake .. -DCMAKE_SYSTEM_PROCESSOR=arm`
- `make -j 1`
 - Multiple threads will cause a race condition.

5.6.4 Useful Commands

- `readelf -h executable`
 - Read executable header.
- `apt install cmake-curses-gui`
 - Installs CMakeCache.txt editor install for static builds.
- `ccmake ..`
 - Runs the CMakeCache.txt editor.

6.0 RUNNING HDTN

Note: Ensure your config files (described in Section 13.0) are correct, e.g., check that the outduct `remotePort` is the same as the induct `boundPort`, a consistent `convergenceLayer`, and the outduct's `remoteHostname` is pointed to the correct IP adress. `tcpdump` can be used to test the HDTN ingress storage and egress. The generated `pcap` file can be read using `wireshark`: `sudo tcpdump -i lo -vv -s0 port 4558 -w hdtm-traffic.pcap`

In another terminal, run: `./runscript.sh`

NOTE: The Contact Plan, which lists future contacts for each node, is located under `module/router/-contact_plans/contactPlan.json` and includes the source and destination nodes, the start and end times, and the data rates. Based on the schedule in the Contact Plan the router sends events on link availability to Ingress and Storage. When the Ingress receives the `Link Available` event for a given destination, it sends the bundles directly to egress. When the Link is `Unavailable` it sends the bundles to storage. Upon receiving `Link Available` event, Storage releases the bundle(s) for the corresponding destination. When a `Link Down` event is received, Storage stops releasing the bundles.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 26 of 69

There are additional test scripts located under the directories `test_scripts_linux` and `test_scripts_windows` that can be used to test different scenarios for all convergence layers.

6.1 Directory Structure

<code>common/</code>	Common Libraries and Utils
<code>module/</code>	HDTN Core Modules
<code>egress</code>	CL adapter(s) that forwards bundle traffic
<code>ingress</code>	CL adapter(s) that accepts traffic in bundle format
<code>storage</code>	Stores bundles
<code>router</code>	Sends link state and routes to other modules
<code>hdtm_one_process</code>	Combines the main processes into one HDTN process
<code>udp_delay_sim</code>	Proxy that simulates long delays
<code>telem_cmd_interface</code>	Web interface for stats display and configuration
<code>Config_files/</code>	HDTN config files
<code>tests/</code>	Example Test cases and experiments

6.2 Unit Tests

After building HDTN (see Section 4.0), unit tests can be run using the following command within the build directory:

```
./tests/unit_tests/unit-tests
```

6.3 Integrated Tests

After building HDTN (see Section 4.0), integrated tests can be run using the following command within the build directory:

```
./tests/integrated_tests/integrated-tests
```

7.0 GRAPHICAL USER INTERFACE

7.1 Running the Web Interface

This repository comes equipped with code to launch a web-based user interface to display statistics for the HDTN engine. It relies on a dependency called Boost Beast which is packaged as a header-only library that comes with a standard Boost installation. The web interface requires OpenSSL since the web interface supports both http as well as https, and hence both WS (WebSocket) and WSS (WebSocket Secure). The web interface is compiled by default. Anytime that HDTNOneProcess runs, the web page will be accessible at <http://localhost:8086>

To prevent the web interface from running, follow the normal build instructions for Linux. The only difference will be in the cmake command will now be:

```
cmake -DUSE_WEB_INTERFACE:BOOL=OFF ..
```

7.2 Statistics Page

This page, displayed in Figure 7-1, displays real-time telemetry of HDTN. At the top are three boxes displaying the current Data Rate in Mega-Bits Per Second, the Average Data Rate, and the Maximum Data Rate reached. All of these are measured in the Ingress module.

Beneath are three graphs. The first two display the data rate of the Ingress and Egress Modules in Mega-bits Per Second. The third graph is a pie chart displaying the location of data bundles received by Ingress - either sent to the Storage module or directly to Egress. If a bundle is sent from Storage to Egress, it will be measured on the pie chart as having gone to Egress.

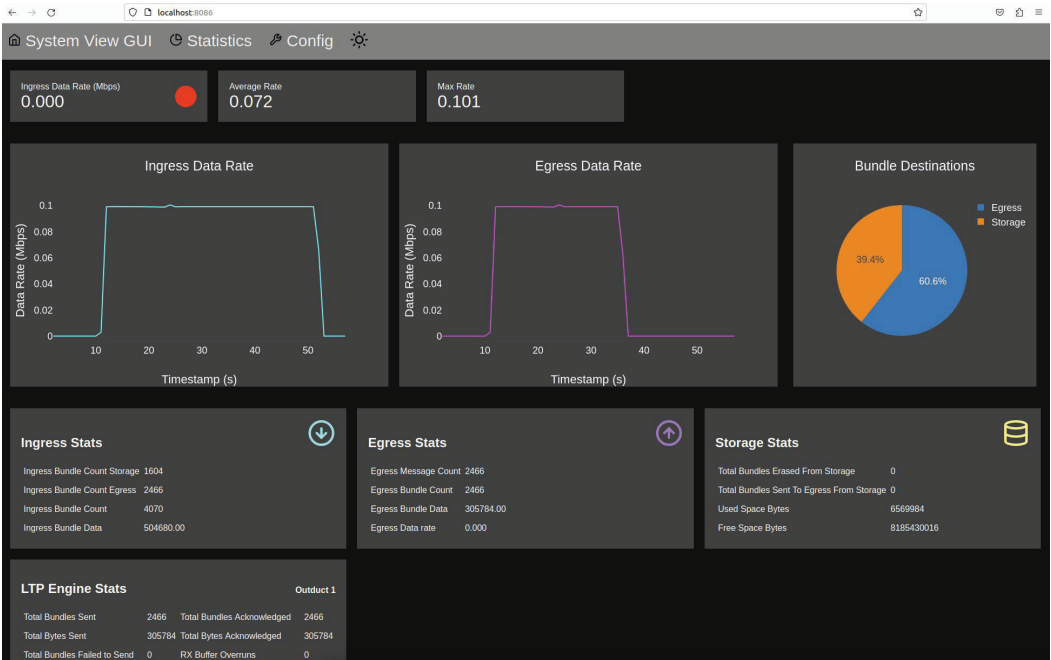


Figure 7-1: Statistics Page of the Web Interface

Beneath the graphs, cards display statistics for different parts of HDTN. At this time only Ingress and Egress are displayed.

7.3 System View Page

This page can be accessed by clicking System View GUI in the top left-hand corner of the Statistics page. As shown in Figure 7-2, this page displays a graphic of the different modules of HDTN as well as information on where the bundled data is coming from and where it is going. In the top row are adjustable settings for users to make the information more legible. On the left of the page are the IP addresses and IPN numbers from which data is being received. On the right is displayed the Nodes and IPN numbers to which data is being sent from this HDTN node. The graphic displays the data rate as data comes into Ingress, between the different modules of HDTN (Ingress, Storage, and Egress), and the rate as it leaves Egress. The Storage graphic displays the percentage and amount of storage space being used.

Users can also hover over each HDTN module and a pop-up graphic will appear displaying data for that module. An example is shown in Figure 7-3 which shows this information for the Storage module.

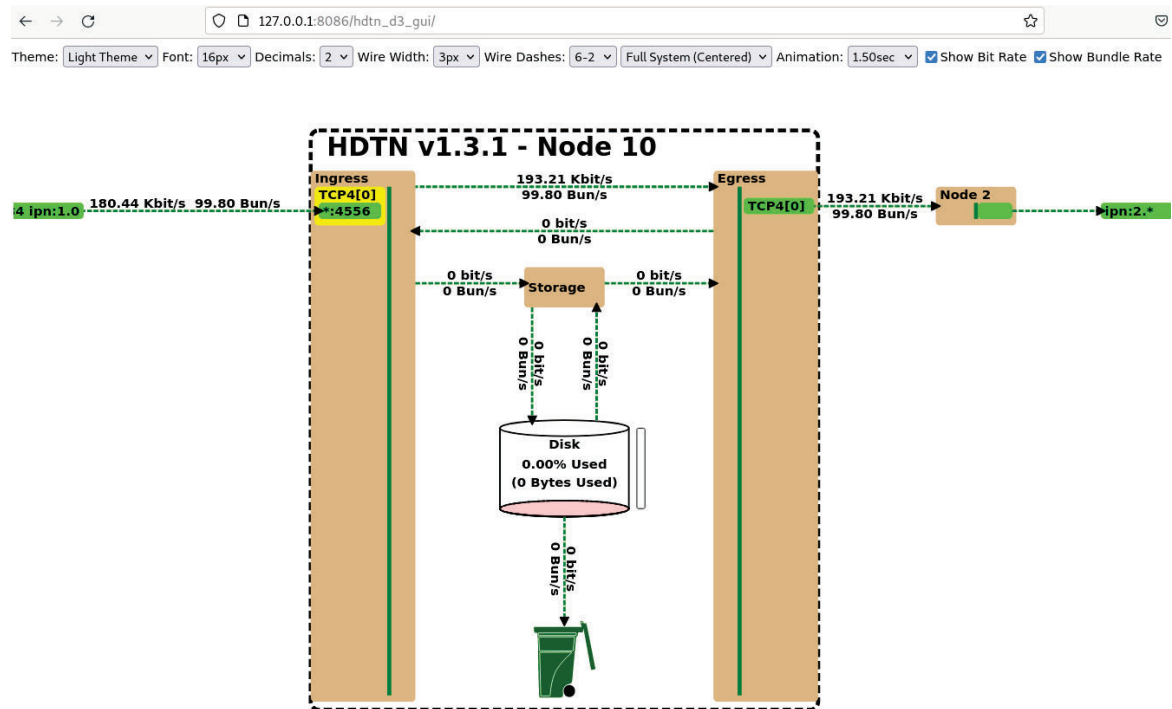


Figure 7-2: System View Page of the Web Interface

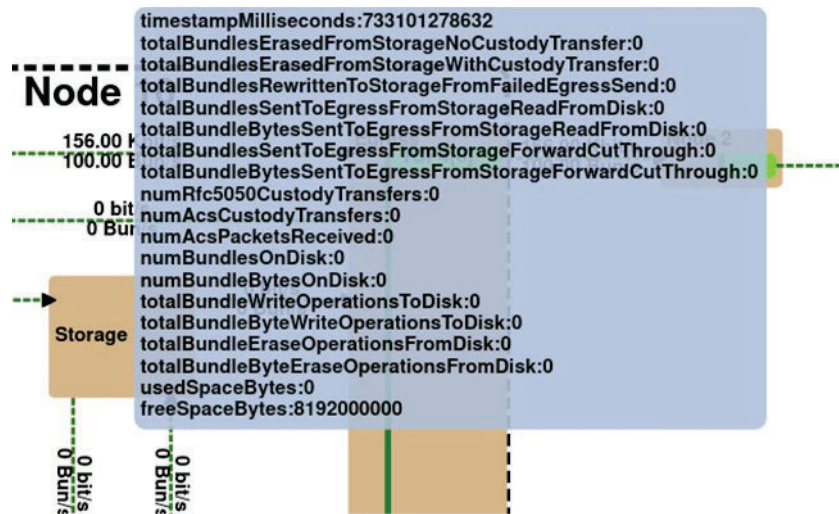


Figure 7-3: Pop-up Graphic Displaying Data from the Storage Module

7.4 Config Page

This page is not configured yet.

7.5 Statistics Logging

HDTN telemetry can be automatically logged to CSV files by compiling HDTN with the DO_STATS LOGGING CMake option. The command for enabling this is:

cmake -DDO_STATS_LOGGING:BOOL=ON. Files will be created in the /stats directory of the source code root. Statistics are logged on a 1 second interval. The following statistics are currently supported:

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 29 of 69

- `ingress_data_rate_mbps`
- `ingress_total_bytes_sent`
- `ingress_bytes_sent_egress`
- `ingress_bytes_sent_storage`
- `storage_used_space_bytes`
- `storage_free_space_bytes`
- `storage_bundle_bytes_on_disk`
- `storage_bundles_erased`
- `storage_bundles_rewritten_from_failed_egress_send`
- `storage_bytes_sent_to_egress_cutthrough`
- `storage_bytes_sent_to_egress_from_dis`
- `egress_data_rate_mbps`
- `egress_total_bytes_sent_success`
- `egress_total_bytes_attempted`

8.0 GETTING STARTED WITH THE API

The HDTN API is a ZeroMQ-Based Messaging API that follows a request-reply messaging pattern. The HDTN API leverages the ZeroMQ messaging library to facilitate efficient and flexible communication between HDTN and other software systems. It enables users to request information on the configuration and statistics of HDTN.

A simple python script named `run_api_commands.py` under `tests/tests_scripts_linux` has been provided as a reference. This script contains all the supported API calls. The API calls can be utilized in any programming language that support the ZeroMQ library. The HDTN API accepts and responds with JSON-encoded strings.

8.1 API Calls

8.1.1 HDTN Version

Call: `get_hdtm_version`

Description: This call is used to obtain the current HDTN version.

Syntax:

```
{
  "apiCall": "get_hdtm_version"
}
```

Parameter(s): None

Return Value: If the call succeeds, then the current HDTN version is returned as a JSON string.

Example:

```
JSON = {
  "apiCall": "get_hdtm_version"
}
```

8.1.2 HDTN Configuration

Call: `get_hdtm_config`

Description: This call is used to obtain the currently active configuration information for HDTN.

Syntax:

```
{
  "apiCall": "get_hdtm_config"
}
```

Uncontrolled when printed. Verify that this is correct version before use.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 30 of 69

Parameter(s): None

Return Value: If the call succeeds, then the currently active configuration information for HDTN is returned as a JSON string.

Example:

```
JSON = {
    "apiCall": "get_hdtm_config"
}
```

8.1.3 Storage

Call: get_storage

Description: This call is used to obtain storage statistics information of HDTN.

Syntax:

```
{
    "apiCall": "get_storage"
}
```

Parameter(s): None

Return Value: If the call succeeds, then the storage statistics information of HDTN is returned as a JSON string.

Example:

```
JSON = {
    "apiCall": "get_storage"
}
```

8.1.4 Expiring Storage

Call: get_expiring_storage

Description: This call is used to obtain expiring storage statistics information of HDTN.

Syntax:

```
{
    "apiCall": "get_expiring_storage",
    "priority": INTEGER,
    "thresholdSecondsFromNow": INTEGER
}
```

Parameter(s):

- **priority** specifies which bundles to look for based on priority when retrieving expired bundle.
 - Input: 0,1, or 2
- **thresholdSecondsFromNow** specifies how far in the future to look for expiring bundles in seconds.
 - Input: positive integer

Return Value: If the call succeeds, then the expiring storage statistics information of HDTN is returned as a JSON string.

Example:

```
JSON = {
    "apiCall": "get_expiring_storage",
    "priority": 1,
    "thresholdSecondsFromNow": 600
}
```

8.1.5 Inducts

Call: get_inducts

Description: This call is used to obtain all inducts statistics information of HDTN.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 31 of 69

Syntax:

```
{
  "apiCall": "get_inducts"
}
```

Parameter(s): None

Return Value: If the call succeeds, then all the inducts statistics information of HDTN is returned as a JSON string.

Example:

```
JSON = {
  "apiCall": "get_inducts"
}
```

8.1.6 Outducts

Call: get_outducts

Description: This call is used to obtain all outducts statistics information of HDTN.

Syntax:

```
{
  "apiCall": "get_outducts"
}
```

Parameter(s): None

Return Value: If the call succeeds, then all the outducts statistics information of HDTN is returned as a JSON string.

Example:

```
JSON = {
  "apiCall": "get_outducts"
}
```

8.1.7 Maximum Send Rate for an Outduct

Call: set_max_send_rate

Description: This call is used to set the maximum send rate in bit per second for a specific outduct of HDTN.

Syntax:

```
{
  "apiCall": "set_max_send_rate",
  "rateBitsPerSec": INTEGER,
  "outduct": INTEGER
}
```

Parameter(s):

- **rateBitsPerSec** specifies the maximum rate in bits per second. When the input is 0, there is no upper limit imposed on the maximum rate.
 - Input: 0 or positive integer
- **outduct** specifies the number of the outduct to which the maximum rate should be applied.
 - Input: positive integer

Return Value: If the call succeeds, then a success response is returned as a JSON string.

Example:

```
JSON = {
  "apiCall": "set_max_send_rate",
  "rateBitsPerSec": 64000,
  "outduct": 1
}
```

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 32 of 69

8.1.8 BP Security Configuration

Call: get_bpsec_config

Description: This call is used to obtain BP Security statistics information of HDTN.

Syntax:

```
{
  "apiCall": "get_bpsec_config"
}
```

Parameter(s): None

Return Value: If the call succeeds, then the BP Security statistics information of HDTN is returned as a JSON string.

Example:

```
JSON = {
  "apiCall": "get_bpsec_config"
}
```

8.1.9 Upload Contact Plan

Call: upload_contact_plan

Description: This call is used to upload a contact plan for HDTN.

Syntax:

```
{
  "apiCall": "upload_contact_plan",
  "contactPlanJson": JSON STRING
}
```

Parameter(s):

- **contactPlanJson** specifies the contact plan to apply.
 - Input: json string

Return Value: If the call succeeds, then a success response is returned as a JSON string.

Notes: The contact plan presented in the example illustrates a scenario with two nodes

Example:

```
JSON = {
  "apiCall": "upload_contact_plan",
  "contactPlanJson": {
    "contacts": [
      {
        "contact": 0,
        "source": 1,
        "dest": 10,
        "startTime": 0,
        "endTime": 2000,
        "rateBitsPerSec": 0,
        "owlt": 1
      },
      {
        "contact": 1,
        "source": 10,
        "dest": 20,
        "startTime": 0,
        "endTime": 2000,
        "rateBitsPerSec": 0,
        "owlt": 1
      }
    ]
  }
}
```

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 33 of 69

```

    },
    {
        "contact": 2,
        "source": 20,
        "dest": 2,
        "startTime": 0,
        "endTime": 2000,
        "rateBitsPerSec": 0,
        "owlt": 1
    }
]
}
}

```

8.1.10 Ping

Call: ping

Description: This call is used to ping a specific service of a HDTN node.

Syntax:

```

{
    "apiCall": "ping",
    "bpVersion": INTEGER,
    "nodeId": INTEGER,
    "serviceId": INTEGER
}

```

Parameter(s):

- **bpVersion** specifies which Bundle Protocol Version to use for the ping.
 - Input: 6 or 7
- **nodeId** specifies the node number.
 - Input: positive integer
- **serviceId** specifies which service to ping.
 - Input: positive integer

Return Value: If the call succeeds, then a ping reply is returned as a JSON string.

Example:

```

JSON = {
    "apiCall": "ping",
    "bpVersion": 7,
    "nodeId": 2,
    "serviceId": 1
}

```

8.1.11 Take Link Down

Call: set_link_down

Description: This call is used to take a link down from the outductVector.

Syntax:

```

{
    "apiCall": "set_link_down",
    "outductIndex": INTEGER
}

```

Parameter(s):

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 34 of 69

- **outductIndex** specifies which link to take down given the index of the outduct in the list of outducts.
 - Input: 0 or positive integer

Return Value: If the call succeeds, then a success reply is returned as a JSON string.

Example:

```
JSON = {
  "apiCall": "set_link_down",
  "outductIndex": 1
}
```

8.1.12 Bring Link Up

Call: set_link_up

Description: This call is used to bring a link up from the outductVector.

Syntax:

```
{
  "apiCall": "set_link_up",
  "outductIndex": INTEGER
}
```

Parameter(s):

- **outductIndex** specifies which link to bring up given the index of the outduct in the list of outducts.
 - Input: 0 or positive integer

Return Value: If the call succeeds, then a success reply is returned as a JSON string.

Example:

```
JSON = {
  "apiCall": "set_link_up",
  "outductIndex": 0
}
```

9.0 COMMAND LINE INTERFACE

A Command Line Interface (CLI) application is automatically built as part of the regular HDTN build process. It can be found under the `build/module/cli` folder after compiling HDTN. It is used to interact with the HDTN API for configuring and controlling the software.

9.1 Usage

The HDTN CLI is invoked by running the `hdt-cli` command. HDTN must be running for the CLI to work. Currently, it only supports a limited set of command options. To see the list of options, run `hdt-cli --help`. To get started:

1. Build HDTN (see instructions in the main README.md).
2. Install HDTN (see instructions in the main README.md).
3. Run an instance of HDTN (see instructions in the main README.md).
4. Run `hdt-cli --help` for a list of commands.
5. Run `hdt-cli <command1> <command2> ...` to run the desired CLI commands.

9.2 Examples

The following examples show how to use the CLI to configure and control HDTN.

Get a list of available options:

```
$ hdt-cli --help
Options:
```

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 35 of 69

```
--help                Produce help message
--hostname arg (=127.0.0.1) HDTN hostname
--port arg (=10305)    HDTN port
--contact-plan-file arg Local contact plan file
--contact-plan-json arg Contact plan JSON string
--outduct[0].rateBps arg Outduct rate limit (bits per second)
--outduct[1].rateBps arg Outduct rate limit (bits per second)
```

Upload a new contact plan file:

```
$ hdtm-cli --contact-plan-file=/home/user/contact_plan.json
```

Upload a new contact plan via JSON:

```
$ hdtm-cli --contact-plan-json= '{ "contacts": [ { "contact": 0, "source": 102, "dest": 10, "startTime
```

Set the rate limit of the first outduct to 1 Mbps:

```
$ hdtm-cli --outduct[0].rateBps =1000000
```

Set the rate limit of the first and second outducts to 1 Mbps:

```
$ hdtm-cli --outduct[0].rateBps=1000000 --outduct[1].rateBps =1000000
```

10.0 SIMULATIONS

HDTN can be simulated using DtnSim, a third party, open-source simulator for delay tolerant networks built on the OMNeT++¹ simulation framework. Use the “support-hdtm” branch of DtnSim which can be found in the [official DtnSim repository](#)². HDTN simulation with DtnSim has only been tested on Linux (Debian and Ubuntu). Follow the readme instructions for HDTN and DtnSim to install the software. More Details about the installation steps can be found [here](#)³.

11.0 HDTN APPLICATIONS

HDTN comes with many applications to use as isolated modules or in conjunction with HDTN. The current set of HDTN applications are:

- BPGen – Generates bundles (is intended to be used with its receiving tool BPSink).
- BPSink – Receives and validates bundles (is intended to be used with its generating tool BPGen).
- BPSendFile – Sends a single file or a directory of files (with recursion).
- BPreceiveFile – Receives the bundles in any order, reassembles the fragments, if applicable, and writes the file(s) to a user-specified directory.
- BPSendStream – Generates bundles from Real-time Transport Protocol (RTP) packets.
- BPreceiveStream – Receives bundles containing RTP packets and outputs the RTP packets.
- BPing – Generates ping bundles.
- BPSendPacket – Generates bundles from UDP or STCP payload data.
- BPreceivePacket – Receives bundles containing UDP or STCP payload data and outputs.

Figure 11-1 shows how these applications connect to HDTN’s modules.

¹ OMNeT++ license may be required. Please see <https://omnest.com/licensingfaq> for more information about this.

² <https://bitbucket.org/lcd-unc-ar/dtnsim/src/master/>

³ https://docs.google.com/document/d/1KrKyO_pr-v9CeS5n_ectpfWtwkYL40PdLICGh-24zZY/edit?tab=t.0#heading=h.1fly6dvqx3j3

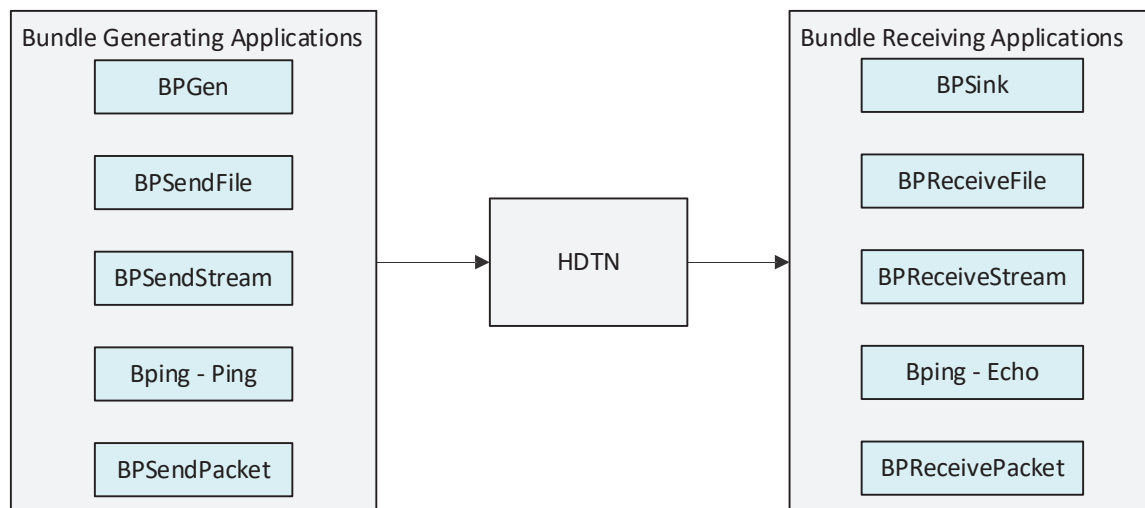


Figure 11-1: HDTN Application Architecture

12.0 RUNSCRIPT

One run script file (in JSON format) is required per each instance of HDTN. Each file starts with assigning path variables, starting up Egress (the outduct), and the bpsink (the method of where bundles will be received). Each file ends with starting up Ingress, assigning bpgen (the process by which bundles can be generated), and a cleanup procedure (optional). Within each file, two different setups can be found depending on if the setup is cut-through, i.e. by-passes Storage or utilizes Storage and, in turn, the Router. The cut-through option requires the link to be up and no BPv6 custody. Note: To avoid starting up Egress and Ingress separately, a method called hdtm-one-process combines the two with one command. An example runscript can be found at `HDTN/runscript.sh`.

12.1 Path Variables

The Path Variables section of each run script file points to the locations of all required configuration files. This way, one needs only to edit this section instead of each section if only changing a configuration file. The path variables contain locations of:

- config files
- hdtm config
- bpsec config
- sink config
- gen config

The section ends with a `cd $HDTN_SOURCE_ROOT` an already declared variable in your Linux path. This command is here if the run script is not in the HDTN source root directory (where HDTN runs).

12.2 BpSink

A typical instantiation of BpSink is:

```
./build/common/bpcodec/apps/bpsink-async --my-uri-eid=ipn:2.1 --
inducts-config-file=$sink_config --bpsec-config-file=$bpsec_config &
```

The command has three main parts: (1) the location of the bpsink code within HDTN, (2) the endpoint ID number, and (3) the inducts configuration file location.

The ipn:2.1 shows, in this particular case, the receiving node is the second endpoint. The \$sink_config response ties to the respectable Path Variable. For more information about the sink configuration file see section 13.2. bpsec-config-file is an optional parameter that is only required if BPsec is enabled and bpsink node is a security acceptor. This option specifies the location of the BPsec Config file which has the policy rules and security operation failure events handling. To end bpsink, it is recommended to insert a 3 second pause command to initialize before the configuration file starts the next section, i.e. sleep 3.

Table 12-1 shows BpSink's available parameters.

Table 12-1: BpSink Instantiation Parameters

Parameter	Description	Default Value
simulate-processing-lag-ms	Extra milliseconds to process bundle (testing purposes).	0
inducts-config-file	Inducts Configuration File.	""
my-uri-eid	BpSink Eid.	ipn:2.1
custody-transfer-outducts-config-file	Outducts Configuration File for custody transfer (use custody if present).	""
acs-aware-bundle-agent	Custody transfer should support Aggregate Custody Signals if valid CTEB present.	NA
bpsec-config-file	BpSec Configuration File.	""
max-rx-bundle-size-bytes	Max bundle size bytes to receive (default=10MB).	10000000

12.3 BpReceiveFile

A typical instantiation of BpReceiveFile on the receiving node is:

```
./build/common/bpcodec/app-s/bpreceivefile --save-directory=received -
-my-uri-eid=ipn:2.1 --inducts-config-file=$sink_config --bpsec-config-
file=$bpsec_config &
```

The command has 3 main parts: (1) the new directory where the received data is saved, (2) the receiving node endpoint id, and (3) the inducts configuration file location. The \$gen_config response ties to the respectable Path Variable. For more information about the gen configuration file see section 13.3.

bpsec-config-file is an optional parameter that is only required if BPsec is enabled and bpreceivefile node is a security acceptor. This option specifies the location of the BPsec Config file which has the policy rules and security operation failure events handling.

To end bpreceivefile, it is recommended to insert an 8 second pause command to initialize before the configuration file starts the next section, i.e. sleep 8.

Table 12-2 shows BpReceiveFile's available parameters.

Table 12-2: BpReceiveFile Instantiation Parameters

Parameter	Description	Default Value
inducts-config-file	Inducts Configuration File.	""
my-uri-eid	BpReceiveFile Eid.	ipn:2.1
custody-transfer-outducts-config-file	Outducts Configuration File for custody transfer (use custody if present).	""
acs-aware-bundle-agent	Custody transfer should support Aggregate Custody Signals if valid CTEB present.	NA

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 38 of 69

max-rx-bundle-size-bytes	Max size bytes to be received based on the Convergence Layer (default=10MB). Note: Does not have effect for the UDPCL.	10000000
outgoing-rtp-port	Destination port for the created RTP stream.	50560
outgoing-rtp-hostname	Remote IP to forward rtp packets to.	127.0.0.1
num-circular-buffer-vectors	Number of circular buffer vector elements for incoming bundles.	50
max-outgoing-rtp-packet-size-bytes	Max size in bytes of the outgoing rtp packets.	1400
shm-socket-path	Location of the socket for shared memory sink to gstreamer.	GST_HDTN_OUTDUCT_SOCKET_PATH
outduct-type	Outduct type to offboard RTP stream.	udp
gst-caps	Caps to apply to GStreamer elements before shared memory interface.	application/x-rtp, media=(string)video, clock-rate=(int)90000, encoding-name=(string)H264, payload=(int)96"
bpsec-config-file	BpSec Configuration File.	""

12.4 Egress

A typical instantiation of Egress is:

```
./build/module/egress/hdtn-egress-async --hdtn-config-file=$hdtn_config &
```

The command has two main parts: (1) the location of the egress code within HDTN and (2) the HDTN configuration file that the egress code requires. For more information about the HDTN configuration file see section 13.1. To end Egress, it is recommended to insert a 3 second pause command to initialize before the configuration file starts the next section, i.e. `sleep 3`.

Table 12-3 shows the Egress's available parameters.

Table 12-3: Egress Instantiation Parameters

Parameter	Description	Default Value
hdtn-config-file	HDTN Configuration File.	hdtn.json
hdtn-distributed-config-file	HDTN Distributed Mode Configuration File.	hdtn_distributed.json

12.5 Router

A typical instantiation of the Router in a runscript is:

```
./build/module/router/hdtn-router --contact-plan-file=contactPlan.json  
--hdtn-config-file=$hdtn_config &
```

The command has three main parts: (1) the location of the Router code within HDTN, (2) the contact plan and (3) the same HDTN configuration file that the Egress required. For more information about the HDTN configuration file see section 13.1. To end the Router, it is recommended to insert a 1 second pause command to initialize before the configuration file starts up the next section, i.e. `sleep 1`. An

example of a Routing scenario test with 4 HDTN nodes was added under
\$HDTN_SOURCE_ROOT/test/test_scripts_linux/Routing_Test.

Table 12-4 shows the Router's available parameters.

Table 12-4: Router Instantiation Parameters

Parameter	Description	Default Value
use-unix-timestamp	Use unix timestamp in contact plan.	NA
use-mgr	Use Multigraph Routing Algorithm.	NA
hdtm-config-file	HDTN Configuration File.	hdtm.json
hdtm-distributed-config-file	HDTN Distributed Mode Configuration File.	hdtm_distributed.json
contact-plan-file	Contact Plan file for link availability and routing.	DEFAULT_FILE

12.6 Ingress

A typical instantiation of Ingress is:

```
./build/module/ingress/hdtm-ingress --hdtm-config-file=$hdtm_config --
bpsec-config-file=$bpsec_config &
```

The command has two main parts: (1) the location of the Ingress code within HDTN and (2) the HDTN configuration file that the ingress code requires. For more information about the HDTN configuration file see section 13.1. bpsec-config-file is an optional parameter that is only required if BPsec is enabled and this HDTN node is a security source, verifier or acceptor. This option specifies the location of the BPsec Config file which has the policy rules and security operation failure events handling. To end Ingress, it is recommended to insert a 3 second pause command to initialize before the configuration file starts the next section, i.e. sleep 3.

Table 12-5 shows the Ingress's available parameters.

Table 12-5: Ingress Instantiation Parameters

Parameter	Description	Default Value
hdtm-config-file	HDTN Configuration File.	hdtm.json
bpsec-config-file	BpSec Configuration File.	""
hdtm-distributed-config-file	HDTN Distributed Mode Configuration File.	"hdtm_distributed.json"
masker	Which Masker implementation to use.	""

12.7 Storage

A typical instantiation of Storage is:

```
./build/module/storage/hdtm-storage --hdtm-config-file=$hdtm_config &
```

The command has two main parts: (1) the location of the storage code within HDTN and (2) the HDTN configuration file that the storage code requires. For more information about the HDTN configuration file see section 13.1. To end Storage, it is recommended to insert a 3 second pause command to initialize before the configuration file starts the next section, i.e. sleep 3.

Table 12-6 shows the Storage's available parameters.

Table 12-6: Storage Instantiation Parameters

Parameter	Description	Default Value
hdtm-config-file	HDTN Configuration File.	hdtm.json
hdtm-distributed-config-file	HDTN Distributed Mode Configuration File.	hdtm_distributed.json

12.8 HDTN One Process

A typical instantiation of hdtm-one-process is:

```
./build/module/hdtm one process/hdtm-one-process --hdtm-config-  
file=$hdtm_config --contact-plan-file=contactPlan.json --bpsec-config-  
file=$bpsec_config &
```

The command has three main parts: (1) the location of the hdtm-one-process code within HDTN, (2) the HDTN configuration file that the hdtm-one-process code requires and (3) the contact plan. For more information about the HDTN configuration file see section 13.1. bpsec-config-file is an optional parameter that is only required if BPsec is enabled and this HDTN node is a security source, verifier or acceptor. This option specifies the location of the BPsec Config file which has the policy rules and security operation failure events handling.

Table 12-7 shows the hdtm-one-process' available parameters.

Table 12-7: HDTN One Process Instantiation Parameters

Parameter	Description	Default Value
hdtm-config-file	HDTN Configuration File.	hdtm.json
bpsec-config-file	BpSec Configuration File.	""
contact-plan-file	Contact Plan file that router relies on for link availability.	DEFAULT_CONTACT_FILE
use-unix-timestamp	Use unix timestamp in contact plan.	NA
use-mgr	Use Multigraph Routing Algorithm.	NA
masker	Which Masker implementation to use.	""

12.9 BpGen

A typical instantiation of BpGen is:

```
./build/common/bpcodec/apps/bpgen-async --bundle-rate=100 --my-uri-  
eid=ipn:1.1 --dest-uri-eid=ipn:2.1 --duration=40 --outducts-config-  
file=$gen_config &
```

The command has six main parts: (1) the location of the bpgen application code within HDTN, (2) the bundle rate, (3) the endpoint ID, (4) the destination endpoint ID, (5) the duration, and (6) the gen configuration file that the bpgen code requires. For more information about the gen configuration file see section 13.3. In the example instantiation, the bundle rate was designated for 100 bundles per second with the bpgen's endpoint ID being the first node (ipn:1.1) and is sending to the second node (ipn:2.1), which matches the bpsink's endpoint ID. The duration value is in seconds, and in this case, 40 seconds. To end bpgen, it is recommended to insert a 8 pause command to initialize before the configuration file starts the next section, i.e. sleep 8.

If BPsec is enabled and BpGen node is a security source, it's required to add the option --bpsec-config-file=\$bpsec_config to specify the BPsec Config file which has the policy rules and security operation events handling.

Table 12-8 shows BpGen's available parameters.

Table 12-8: BpGen Instantiation Parameters

Parameter	Description	Default Value
bundle-size	Bundle size bytes.	100
bundle-rate	Bundle rate. (0=>as fast as possible)	1500
duration	Seconds to send bundles for (0=>infinity).	0
my-uri-eid	BpGen Source Node Id.	ipn:1.1
dest-uri-eid	BpGen sends to this final destination Eid.	ipn:2.1
my-custodian-service-id	Custodian service ID is always 0.	0
outducts-config-file	Outducts Configuration File.	""
custody-transfer-inducts-config-file	Inducts Configuration File for custody transfer (use custody if present).	""
bpsec-config-file	BpSec Configuration File.	""
custody-transfer-use-acs	Custody transfer should use Aggregate Custody Signals instead of RFC5050.	NA
force-disable-custody	Custody transfer turned off regardless of link bidirectionality.	NA
use-bp-version-7	Send bundles using bundle protocol version 7.	NA
bundle-send-timeout-seconds	Max time to send a bundle and get acknowledgement.	3
bundle-lifetime-milliseconds	Bundle lifetime in milliseconds.	1000000
bundle-priority	Bundle priority. 0 = Bulk 1 = Normal 2 = Expedited.	2
cla-rate	Convergence layer adapter send rate. 0 = unlimited.	0

12.10 BpSendFile

A typical instantiation of BpSendFile on the sending node is:

```
./build/common/bpcodec/apps/bpsend-file --max-bundle-size-bytes=500000
--file-or-folder-path=./flightdata --my-uri-eid=ipn:1.1 --dest-uri-
eid=ipn:2.1 --outducts-config-file=$gen config --bpsec-config-
file=$bpsec_config --upload-new-files &
```

The command has five main parts: (1) the maximum bundle size (2) the location of the folder or file to be sent, (3) the current node endpoint id, (4) the destination node endpoint id, (5) the outducts configuration file location. The \$gen config response ties to the respectable Path Variable. For more information about the gen configuration file see section 13.3.

bpsec-config-file is an optional parameter that is only required if BPsec is enabled and bpsendfile node is a security source. This option specifies the location of the BPsec Config file which has the policy rules and security operation failure events handling.

upload-new-files is an optional parameter that can be used to continue sending files when more files are added to the folder being sent.

To end BpSendFile, it is recommended to insert an 8 second pause command to initialize before the configuration file starts the next section, i.e. `sleep 8`.

Table 12-9 shows BpSendFile's available parameters.

Table 12-9: BpSendFile Instantiation Parameters

Parameter	Description	Default Value
max-bundle-size-bytes	Max size for file fragments (default 4MB).	4000000
file-or-folder-path	File or folder paths. Folders are recursive.	""
my-uri-eid	BpGen Source Node Id.	ipn:1.1
dest-uri-eid	BpGen sends to this final destination Eid.	ipn:2.1
my-custodian-service-id	Custodian service ID is always 0.	0
outducts-config-file	Outducts Configuration File.	""
custody-transfer-inducts-config-file	Inducts Configuration File for custody transfer (use custody if present).	""
bpsec-config-file	BpSec Configuration File.	""
skip-upload-existing-files	Do not upload existing files in the directory if and only if file-or-folder-path is a directory.	NA
upload-new-files	Upload new files copied or moved into the directory if and only if file-or-folder-path is a directory.	NA
recurse-directories-depth	Upload all files within max specified depth of subdirectories if file-or-folder-path is a directory (0->no recursion).	3
custody-transfer-use-acs	Custody transfer should use Aggregate Custody Signals instead of RFC5050.	NA
force-disable-custody	Custody transfer turned off regardless of link bidirectionality.	NA
use-bp-version-7	Send bundles using bundle protocol version 7.	NA
bundle-send-timeout-seconds	Max time to send a bundle and get acknowledgement.	3
bundle-lifetime-milliseconds	Bundle lifetime in milliseconds.	1000000
bundle-priority	Bundle priority. 0 = Bulk 1 = Normal 2 = Expedited.	2
cla-rate	Convergence layer adapter send rate. 0 = unlimited.	0

12.11 Bping

A typical instantiation of Bping is:

```
./build/common/bpcodec/apps/bping --my-uri-eid=ipn:1.1 --dest-uri-eid=ipn:2.2047 --outducts-config-file=$ping_config &
```

The command has six main parts: (1) the location of the bping application code within HDTN, (2) the endpoint ID, (3) the destination endpoint ID, and (4) the configuration file that the bping code requires is the same as BpGen config file. For more information about the gen configuration file see section 13.3. In the example instantiation, node is sending ping bundles to the node 2 (using the echo service number 2047).

Table 12-10 shows Bping's available parameters.

Table 12-10: Bping Instantiation Parameters

Parameter	Description	Default Value
bundle-rate	Bundle rate in Bundles per Second. (0=>as fast as possible)	1
duration	Seconds to send bundles for (0=>infinity).	5
my-uri-eid	BPing Source Node Id.	ipn:1.1
dest-uri-eid	BPing sends to this final destination Eid.	ipn:2.1
my-custodian-service-id	Custodian service ID is always 0.	0
outducts-config-file	Outducts Configuration File.	""
custody-transfer-inducts-config-file	Inducts Configuration File for custody transfer (use custody if present).	""
custody-transfer-use-acs	Custody transfer should use Aggregate Custody Signals instead of RFC5050.	NA
use-bp-version-7	Send bundles using bundle protocol version 7.	NA
bundle-send-timeout-seconds	Max time to send a bundle and get acknowledgement.	3
bundle-lifetime-milliseconds	Bundle lifetime in milliseconds.	1000000
bundle-priority	Bundle priority. 0 = Bulk 1 = Normal 2 = Expedited.	2

12.12 BpSendPacket

A typical instantiation of BpSendPacket is:

```
./build/common/bpcodec/apps/bpsendpacket --use-bp-version-7 --my-uri-eid=ipn:1.1 --dest-uri-eid=ipn:2.1 --outducts-config-file=$gen_config --packet-inducts-config-file=$send_config &
```

The arguments are repeated from the other applications in this section and have the same meaning, except for the packet-inducts-config-file argument, which is new. It refers to a config for an induct that will be used to receive a data payload from a local UDP or STCP client.

Table 12-11 shows BpSendPacket's available parameters.

Table 12-11: BpSendPacket Instantiation Parameters

Parameter	Description	Default Value
max-bundle-size-bytes	Max size for data in the payload block (default 4MB).	4000000
my-uri-eid	BpGen Source Node Id.	ipn:1.1
dest-uri-eid	BpGen sends to this final destination Eid.	ipn:2.1
my-custodian-service-id	Custodian service ID is always 0.	0
outducts-config-file	Outducts Configuration File.	""
bpsec-config-file	BpSec Configuration File.	""
custody-transfer-inducts-config-file	Inducts Configuration File for custody transfer (use custody if present).	""
packet-inducts-config-file	Inducts Configuration File for receiving packets.	""

custody-transfer-use-ac	Custody transfer should use Aggregate Custody Signals instead of RFC5050.	NA
force-disable-custody	Custody transfer turned off regardless of link bidirectionality.	NA
use-bp-version-7	Send bundles using bundle protocol version 7.	NA
bundle-send-timeout-seconds	Max time to send a bundle and get acknowledgement.	3
bundle-lifetime-milliseconds	Bundle lifetime in milliseconds.	1000000
bundle-priority	Bundle priority. 0 = Bulk 1 = Normal 2 = Expedited.	2

12.13 BpReceivePacket

A typical instantiation of BpReceivePacket is:

```
./build/common/bpcodec/apps/bpreceivepacket --my-uri-eid=ipn:2.1 --
inducts-config-file=$sink config --packet-outducts-config-
file=$receive config &
```

The packet-outducts-config-file argument is new and refers to config for an outduct that will be used to deliver a data payload to a local application listening on a UDP or STCP socket.

Table 12-12 shows BpReceivePacket's available parameters.

Table 12-12: BpReceivePacket Instantiation Parameters

Parameter	Description	Default Value
inducts-config-file	Inducts Configuration File.	""
my-uri-eid	BpReceivePacket Eid.	ipn:2.1
custody-transfer-outducts-config-file	Outducts Configuration File for custody transfer (use custody if present).	""
packet-outducts-config-file	Packet Outducts Configuration File.	""
acs-aware-bundle-agent	Custody transfer should support Aggregate Custody Signals if valid CTEB present.	NA
bpsec-config-file	BpSec Configuration File.	""
max-rx-bundle-size-bytes	Max size bytes to be received based on the Convergence Layer (default=10MB). Note: Does not have effect for the UDPCL.	10000000

12.14 CleanUp

If HDTN only needs to run for a certain amount of time and then end, add a line under all other sections (minus the path variables) after the instantiation command in the format of `<SectionName>PID=$!` For example, after bpngen's instantiation, the cleanup command will be: `bpngen_PID=$!`

Within the clean-up section, wait for HDTN to run. Then, from the bottom to the top of the configuration file sections, end them via format of `kill -2 $<PID name>`. A wait statement for at least 2 seconds between each kill command is included. Clean-up script example:

```
sleep 30
echo "\killing bpngen..." && kill -2 $bpngen_PID
sleep 2
echo "\killing HDTN storage..." && kill -2 $storage_PID
```

Uncontrolled when printed. Verify that this is correct version before use.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 45 of 69

```
sleep 2
echo "\killing HDTN ingress..." && kill -2 $ingress_PID
sleep 2
echo "\killing router..." && kill -9 $router_PID
sleep 2
echo "\killing egress..." && kill -2 $egress_PID
sleep 2
echo "\killing bpsink..." && kill -2 $bpsink_PID
This example will run HDTN for 30 seconds before closing HDTN.
```

The following options can be added: (1) `--bpsec-config-file=$bpsec_config`: if BPsec is enabled and this HDTN node is a security source, acceptor, or verifier it's required to add this option to specify the BPsec Config file which has the policy rules and security operation events handling. (2) `use-unix-timestamp` to use a contact plan with unix timestamp and (3) `use-mgr` to use Multigraph Routing Algorithm instead of the default CGR Dijkstra routing Algorithm.

To end `hdtm-one-process`, it is recommended that users insert a 10 second pause command to initialize before the configuration file starts the next section, i.e. `sleep 10`.

Note: When using the `hdtm-one-process`, the runscrip does not need to instantiate the Egress, Storage, Ingress, and Router separately.

13.0 CONFIG FILES

13.1 hdtm_config

The typical HDTN configuration file instantiation can be seen below. All values are default and changeable. More information on each line can be seen bulleted.

```
"hdtmConfigName": my hdtm config,
- User description of config file
"userInterfaceOn": true,
- When compiled determines if the interface is displayed
"mySchemeName": "unused scheme name",
- DTN scheme name
- Vestigial, still needs to be defined
"myNodeId": 10,
- Node running ID
- Must be an integer
"myBpEchoServiceId": 2047,
- Service number to ping if user wants to ping HDTN
- Must be an integer
"myCustodialSsp": "unused custodial ssp",
- Custodial scheme specific part
- Vestigial, still needs to be defined
"myCustodialServiceId": 0,
- Service ID where custody reports are sent to HDTN
- Must be an integer
-----
"isAcsAware": true,
- Aggregate Custody Signals (ACS)
- Specifies if HDTN is to use ACS
```

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 46 of 69

- Must be a Boolean

"acsMaxFillsPerAcsPacket": 100,

- How many custody signals to be packed into one ACS packet
- Must be an integer

"acsSendPeriodMilliseconds": 1000,

- Aggregation time in ms
- Must be an integer

"retransmitBundleAfterNoCustodySignalMilliseconds": 10000,

- Retransmits bundle that has custody after this timeout if a custody signal is not received by this time; in ms
- Must be an integer

"maxBundleSizeBytes": 10000000,

- The maximum size of the bundle HDTN can receive or send in Bytes
- NOTE: if the bundle is larger than this, the bundle will be dropped.

"bufferRxToStorageOnLinkUpSaturation": false,

- Must be a Boolean

"maxIngressBundleWaitOnEgressMilliseconds": 2000,

- During Cut-through, if egress cannot finish a bundle within this time it will give up cut-through and send to storage; in ms

"maxLtpReceiveUdpPacketSizeBytes": 65536,

- Maximum packet size in bytes that can be received by HDTN
- Set to the largest datagram the protocol will see on the network.
- 65536 is the typical max size of a local UDP packet will support
- This is unneeded if not using LTP as an irrelevant value

"neighborDepletedStorageDelaySeconds": 0,

- If non-zero, re-route around neighbors with depleted storage
- Attempt to send bundles to depleted neighbor after specified value in seconds
- Requires custody to detect depleted storage on neighboring nodes
- Setting to zero disables

"enforceBundlePriority": false,

- If set to true, enables strict priority ordering of forwarded bundles
- Lower priority bundles cannot be sent via cutthrough queues before higher priority bundles in storage
- Disables ingress-to-egress cutthrough queue, likely decreasing performance

"fragmentBundlesLargerThanBytes": 0,

- If non-zero, HDTN will attempt to fragment BPv6 bundles with payloads larger than value
- Does not fragment BPv7 bundles nor those with the NOFRAGMENT flag set

"zmqBoundRouterPubSubPortPath": 10200,

- ZMQ bound port of the router ZMQ pub-sub socket
- Must be an integer

"zmqBoundTelemApiPortPath": 10305,

- ZMQ bound port of the API socket
- Must be an integer

"inductsConfig": {
 "inductConfigName": "myconfig",
 "inductVector": [

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 47 of 69

```

    {
      "name": "stcp ingress",
      "convergenceLayer": "stcp",
      "boundPort": 4556,
      "numRxCircularBufferElements": 200,
    }
  ]
},
- inductConfigName and name are for user comment
- Can choose within the convergence layer: stcp, tcpcl v3, tcpcl v4, udp, ltp over udp
- boundPort and numRxCircularBufferElements must be integers
- NOTE: numRxCircularBufferElements differs for each convergence layer. STCP this represents
the number of bundles to buffer up; TCPCL this is the number of bundles or data fragments; LTP
this is the number of UDP packets; UDP this is the number of packets/bundles.

```

```

"outductsConfig": {
  "outductConfigName": "myconfig",
  "outductVector": [
    {
      "name": "stcp egress",
      "convergenceLayer": "stcp",
      "nextHopNodeId": 2,
      "remoteHostname": "localhost",
      "remotePort": 4558,
      "maxNumberOfBundlesInPipeline": 50,
      "maxSumOfBundleBytesInPipeline": 50000000,
    }
  ]
},
- outductConfigName and name are for user comment
- Can choose within the convergence layer: stcp, tcpcl v3, tcpcl v4, udp, ltp over udp
- remoteHostname is the IP or hostname that HDTN is sending bundles to. remotePort is the port
HDTN is sending bundles to.
- Final destination IDs can be multiple or one IPN URIs. IPN service number can be an * for a
service wildcard.
- nextHopNodeID, remotePort, maxNumberOfBundlesInPipeline, and
maxSumOfBundleBytesInPipeline must be integers.

```

```

"storageConfig": {
  "storageImplementation": "asio single threaded",
  "tryToRestoreFromDisk": false,
  "autoDeleteFilesOnExit": true,
  "totalStorageCapacityBytes": 8192000000,
  "storageDeletionPolicy": "never",
  "storageDiskConfigVector": [
    {
      "name": "d1",
      "storeFilePath": ".\\store1.bin"
    },
    {
      "name": "d2",

```

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 48 of 69

```

        "storeFilePath": ".\\store2.bin"
    },
]
},
- storageImplementation has 2 options: asio single threaded and stdio multi threaded. Default to
  asio_single_threaded.
- tryToRestoreFromDisk: if the bundle storage was left used, when HDTN reloads it can restore the
  state
- autoDeleteFilesOnExit: tells HDTN, when cleanly closed, to delete or save all storage bundles
- totalStorageCapacityBytes: storage module capacity or quota for bundles, in Bytes. NOTE: each
  storage item in the storageDiskConfigVector must be able to hold totalStorageCapacityBytes
  divided by the number items in the total storageDiskConfigVector
- storageDeletionPolicy: policy for deleting expired bundles from disk. Options are “never”: never
  delete expired bundles, “on expiration”: delete bundles as the expire, and “on storage full”: delete
  all expired bundles when storage reaches 90% capacity.
- storageDiskConfigVector is a striping scheme similar to RAID 0 but not using RAID itself.
  NOTE: can have unlimited storage vector size i.e. number of storeFilePath(s).

```

Miscellaneous notes for the HDTN configuration file:

- Depending on the convergence layer there may be additions to the “inductVector”, “outductVector”. See section 15.0 for details.
- Ingress, Egress, and/or storage are optional additions to the HDTN configuration file depending on the HDTN node need.

13.2 sink_config

A typical sink configuration file includes an “inductConfigName” and an “inductVector”. The “inductVector” requires the name, convergence layer, the bound port number, and the number of received circular buffer elements. For details about this section, see section 13.1 since the bp sink config file is a copy of the inductConfigName/inductVector of the hdtm config file. This is because the bp sink config file goes to the BPSink application detailed in section 11.0. Note: Depending on the convergence layer there may be additions to the “inductVector”. See section 15.0 for details. Structure example:

```

"inductsConfig": {
    "inductConfigName": "myconfig",
    "inductVector": [
        {
            "name": "stcp ingress",
            "convergenceLayer": "stcp",
            "boundPort": 4556,
            "numRxCircularBufferElements": 200,
        }
    ]
},

```

13.3 gen_config

A typical gen configuration file includes an “outductConfigName” and an “outductVector”. The “outductVector” requires: the name, convergence layer, next hop, remote hostname, remote port, bundle pipeline limit, and the final destination endpoint ID. For details about this section see section 13.1 since the gen config file is a copy of the outductConfigName/outductVector of the hdtm config file. This is due to the gen config file going to the BPGen application detailed in section 11.0 Note: depending on the convergence layer there may be additions to the “outductVector”. See section 15.0 for details. Structure

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 49 of 69

example:

```
{
  "outductConfigName": "myconfig",
  "outductVector": [
    {
      "name": "bpgen",
      "convergenceLayer": "tcpcl v3",
      "nextHopNodeId": 10,
      "remoteHostname": "localhost",
      "remotePort": 4556,
      "maxNumberOfBundlesInPipeline": 5,
      "maxSumOfBundleBytesInPipeline": 50000000,
    }
  ]
}
```

13.4 bpsec_config

A typical BPsec configuration file includes a “bpSecConfigName” which is user description of the config file, “policyRules” and “securityFailureEventSets” vectors which include the BPsec security policy rules and the security operation failure events handling. An overall structure example:

```
{
  "bpsecConfigName": "my BPsec Config",
  "policyRules": [
    {
      "description": "Bpsource confidentiality",
      "securityPolicyRuleId": 1,
      "securityRole": "source",
      "securitySource": "ipn:1.1",
      "bundleSource": [
        "ipn:1.1"
      ],
      "bundleFinalDestination": [
        "ipn:2.1"
      ],
      "securityTargetBlockTypes": [
        1
      ],
      "securityService": "confidentiality",
      "securityContext": "aesGcm",
      "securityFailureEventSetReference": "default_confidentiality",
      "securityContextParams": [
        {
          "paramName": "aesVariant",
          "value": 256
        },
        {
          "paramName": "ivSizeBytes",
          "value": 12
        },
        {
          "paramName": "keyFile",

```

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 50 of 69

```

        "value": "config_files/bpsec/ipn1.1_confidentiality.key"
    },
    {
        "paramName": "securityBlockCrc",
        "value": 0
    },
    {
        "paramName": "scopeFlags",
        "value": 7
    }
]
}
],
"securityFailureEventSets": [
{
    "name": "default_confidentiality",
    "description": "default bcb confidentiality security
operations event set",
    "securityOperationEvents": [
        {
            "eventId": "sopCorruptedAtAcceptor",
            "actions": [
                "removeSecurityOperationTargetBlock"
            ]
        }
    ]
}
]
}
]
}

```

More information on each line of the policy rules vector is detailed below:

```

"description": "Bpsource confidentiality",
    - User description of bpsec policy rule file
"securityPolicyRuleId": 1,
    - Policy rule Id
"securityRole": "source",
    - security role can be configured as "source", "verifier" or "acceptor"
"securitySource": "ipn:1.1",
    - security source Endpoint ID
"bundleSource": [
    "ipn:1.1"
],
    - a vector of bundle source endpoint IDs (wildcard can also be used).
"bundleFinalDestination": [
    "ipn:2.1"
],
    - a vector of bundle final destination endpoint IDs (wildcard can also be used).
"securityTargetBlockTypes": [
    1
],
    - a vector of all the security Target Block Types, in this example 1 corresponds to the payload
      block

```

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 51 of 69

- bundle Source Endpoint ID

```
"securityService": "confidentiality",
```

- Security service could be either “confidentiality” or “integrity”

```
"securityContext": "aesGcm",
```

- Security Context can be “aesGcm” for confidentiality or “hmacSha” for integrity

```
"securityContextParams": [
  {
    "paramName": "aesVariant",
    "value": 256
  },

```

- Security Context parameters is a vector of security context parameters and their values. The parameters and their possible values:
 - o “aesVariant” possible values are 128, 256
 - o “shaVariant” possible values are 256, 512 or 384
 - o “ivSizeBytes” possible values are 12 or 16
 - o “keyFile” specifies the location of the key
 - o “keyEncryptionKeyFile” specifies the location of the key-encryption key
 - o “securityBlockCrc” defines optional Cyclic Redundancy Check (CRC) to identify block corruption
 - o “scopeFlags” defines any additional data to be included along with the block-type-specific data

```
"securityFailureEventSetReference": "default confidentiality",
```

- Reference to the name of the security Failure Event Set to be used
- More information on each line of the security Failure Event Sets vector are:

```
"name": "default confidentiality",
  o name/reference of the security Failure Event Set
"name": "description",
  o user description of the event set
"securityOperationEvents": [
  {
    "eventId": "sopCorruptedAtAcceptor",
    "actions": [
      "removeSecurityOperationTargetBlock"
    ]
  }
]
"name": "securityOperationEvents",
```

- Security operation events vector which has security event Ids and their corresponding actions
- The supported security events are:
 - o “sopMisconfiguredAtVerifier”, “sopMissingAtVerifier”:
 - Triggered when a bundle is received with an Endpoint Identifier (EID) that HDTN is configured to process but contains different security operations than expected.
 - Example: A bundle arrives with confidentiality applied to an extension block, but HDTN is configured to process confidentiality applied to the payload block.
 - o “sopCorruptedAtVerifier”, “sopMissingAtAcceptor”:
 - Triggered when a bundle arrives without the expected security operation.
 - Example: HDTN is configured to process confidentiality, and a bundle is received with no Block Confidentiality Block (BCB).
 - o “sopCorruptedAtAcceptor”

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 52 of 69

- Triggered when the contents of a security operation or the protected contents of a target block have been modified, or an incorrect key is used to process the operation.
 - Example: A bundle is received with integrity applied to the payload block, but a transmission error has altered the contents of the data field in that block and now it cannot be verified.
- The supported actions are:
 - “removeSecurityOperation”: Discard the affected BIB or BCB, but not the target block.
 - “removeSecurityOperationTargetBlock”: Discard the affected target block. If this is the payload block, then the entire bundle will be discarded because a bundle must contain a payload block.
 - “removeAllSecurityTargetOperations”: No effect currently.
 - “failBundleForwarding”: Discard the entire bundle.
 - “requestBundleStorage”: Place the bundle into storage instead of forwarding or discarding.
 - “reportReason-Code”: No effect currently.
 - “overrideSecurityTargetBlockBpcf”: No effect currently.
 - “overrideSecurityBlockBpcf”: No effect currently.

13.5 distributed_config

If you are running HDTN in distributed mode, you will need to add a command line argument `--hdtndistributed-config-file` as shown in [runscript distributed](#)⁴ and [hdtndistributed defaults](#)⁵.

```
"zmqIngressAddress": "localhost",
  - IP or hostname of the machine running the Ingress module of HDTN
"zmqEgressAddress": "localhost",
  - IP or hostname of the machine running the Egress module of HDTN
"zmqStorageAddress": "localhost",
  - IP or hostname of the machine running the Storage module of HDTN
"zmqRouterAddress": "localhost",
  - IP or hostname of the machine running the Router module of HDTN
"zmqBoundIngressToConnectingEgressPortPath": 10100,
  - ZMQ bound TCP port of the Ingress module for internal messages sent from Ingress to Egress
  - NOTE: This value is unused when using hdtndistributed; still needs to be defined.
  - NOTE: TCP is unidirectional in ZMQ
  - Must be an integer
"zmqConnectingEgressToBoundIngressPortPath": 10160,
  - ZMQ bound TCP port of the Ingress module for internal messages sent from Egress to Ingress
  - NOTE: This value is unused when using hdtndistributed; still needs to be defined.
  - NOTE: TCP is unidirectional in ZMQ
  - Must be an integer
"zmqBoundEgressToConnectingRouterPortPath": 10162,
  - ZMQ bound TCP port of the Router module for internal link down messages sent from Egress to Router
  - NOTE: TCP is unidirectional in ZMQ
  - Must be an integer
"zmqConnectingEgressBundlesOnlyToBoundIngressPortPath": 10161,
  - ZMQ bound TCP port of the Ingress module for internal TCPCL opportunistic bundles sent from Egress to Ingress
  - NOTE: This value is unused when using hdtndistributed; still needs to be defined.
```

⁴ https://github.com/nasa/HDTN/blob/master/tests/test_scripts/linux/runscript_distributed.sh

⁵ https://github.com/nasa/HDTN/blob/master/config_files/hdtndistributed_defaults.json

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 53 of 69

- NOTE: TCP is unidirectional in ZMQ
 - Must be an integer
- "zmqBoundIngressToConnectingStoragePortPath": 10110,
- ZMQ bound TCP port of the Ingress module for internal messages sent from Ingress to Storage
 - NOTE: This value is unused when using hdtm-one-process; still needs to be defined.
 - NOTE: TCP is unidirectional in ZMQ
 - Must be an integer
- "zmqConnectingStorageToBoundIngressPortPath": 10150,
- ZMQ bound TCP port of the Ingress module for internal messages sent from Storage to Ingress
 - NOTE: This value is unused when using hdtm-one-process; still needs to be defined.
 - NOTE: TCP is unidirectional in ZMQ
 - Must be an integer
- "zmqConnectingStorageToBoundEgressPortPath": 10120,
- ZMQ bound TCP port of the Egress module for internal messages sent from Storage to Egress
 - NOTE: This value is unused when using hdtm-one-process; still needs to be defined.
 - NOTE: TCP is unidirectional in ZMQ
 - Must be an integer
- "zmqBoundEgressToConnectingStoragePortPath": 10130,
- ZMQ bound TCP port of the Egress module for internal messages sent from Egress to Storage
 - NOTE: This value is unused when using hdtm-one-process; still needs to be defined.
 - NOTE: TCP is unidirectional in ZMQ
 - Must be an integer
- "zmqConnectingRouterToBoundEgressPortPath": 10210,
- ZMQ bound TCP port of the Egress module for internal messages sent from Router to Egress
 - NOTE: This value is unused when using hdtm-one-process; still needs to be defined.
 - NOTE: TCP is unidirectional in ZMQ
 - Must be an integer
- "zmqConnectingTelemToFromBoundIngressPortPath": 10301,
- ZMQ bound TCP port of the Ingress module for internal messages sent from Ingress to Telemetry module (GUI)
 - NOTE: This value is unused when using hdtm-one-process; still needs to be defined.
 - NOTE: TCP is unidirectional in ZMQ
 - Must be an integer
- "zmqConnectingTelemToFromBoundEgressPortPath": 10302,
- ZMQ bound TCP port of the Egress module for internal messages sent from Egress to Telemetry module
 - NOTE: This value is unused when using hdtm-one-process; still needs to be defined.
 - NOTE: TCP is unidirectional in ZMQ
 - Must be an integer
- "zmqConnectingTelemToFromBoundStoragePortPath": 10303
- ZMQ bound TCP port of the Storage module for internal messages sent from Storage to Telemetry module
 - NOTE: This value is unused when using hdtm-one-process; still needs to be defined.
 - NOTE: TCP is unidirectional in ZMQ
 - Must be an integer

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 54 of 69

14.0 CONTACT PLANS

For an HDTN node, a single contact entry describes an availability time slot for the next hop (destination) of one of its outducts. The entry contains begin and end times for when the next hop is accessible, as well as the speed and one-way trip time of the link between them.

The contact plan file lists this information for all contacts of a node. If a next hop is available at different times or with different conditions, each variation requires its own contact entry. A single contact plan may contain entries for multiple nodes in the network. Any entry will be discarded if it doesn't contain the node number of the node currently processing it. If a node has an outduct with a next hop that does not appear in the contact plan, it will never route bundles to it. This includes STCP outducts on the same host.

The contact plan is a JSON file that is only used within the HDTN bundle agent (i.e. `hdtm-one-process`). Example files are available under `HDTN/module/router/contact_plans/src`

14.1 JSON Fields

```
"contact": 1
  - Identification number of the contact.
  - Integer
"source": 10
  - Source/Sending node number for that contact.
  - Integer
"dest": 2
  - Destination/Receiving node number for that contact or next hop.
  - Integer
"startTime": 25
  - This is the time, in seconds, after which the link is UP for that contact (start transmission time)
  - Relative to the start of the router process (time 0), unless the --use-unix-timestamp option is used, in which case it represents the number of seconds since 00:00:00 UTC on 1 January 1970, the Unix epoch.
  - Integer
"endTime": 38
  - This is the time, in seconds, after which link will be DOWN for that contact (end transmission time)
  - Relative to the start of the router process (time 0), unless the --use-unix-timestamp option is used, in which case it represents the number of seconds since 00:00:00 UTC on 1 January 1970, the Unix epoch.
  - Integer
"rateBitsPerSec": 1000
  - The data rate limit in bits per second
  - Controls the rate limit HDTN will apply for this contact; this is only for LTP and UDP convergence layers
  - This option is the same that applications use with this argument --cla-rate
  - The router will prioritize a contact with a higher rate over a contact with a lower rate.
  - Integer
"owlT": 1
  - One Way Light Time which is the distance (range) expressed in light-seconds
  - The router will prioritize a contact with a lower one way light time over a contact with a higher one
  - Integer
```

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 55 of 69

"__comment__": "from node1 to node2"

- This field is not used by HDTN, but is useful for improving the readability of a contact plan file
- String

15.0 CONVERGENCE LAYERS AND ROUTING PROTOCOLS

HDTN provides the following convergence layer compatibilities: Transmission Control Protocol (TCP) Convergence Layer (TCPCL), User Datagram Protocol (UDP), Licklider Transmission Protocol (LTP), and Simple TCP (STCP).

Each convergence layer has additional “inductVector” and “outductVector” configuration fields. These additions apply to all config files that utilize “inductVector” and “outductVector”, e.g. hdtm_config, sink_config, and gen_config. The additions are listed in the sections below.

15.1 TCPCLv3

Common induct and outduct fields:

"keepAliveIntervalSeconds": 15

- This is the minimum interval, in seconds, to negotiate as the Session Keepalive. See RFC7242 Section 5.6.
- Integer

"tcpclV3MyMaxTxSegmentSizeBytes": 200000

- This is the maximum segment size, in bytes, to use for transmitting data segments.
- Integer

Induct fields:

"numRxCircularBufferBytesPerElement": 100

- This is the maximum size, in bytes, of each element in the circular receive buffer.
- Integer

Outduct fields:

"tcpclAllowOpportunisticReceiveBundle": false

- This is whether to allow receiving opportunistic bundles.
- Boolean

15.2 TCPCLv4

Common induct and outduct fields:

"keepAliveIntervalSeconds": 15

- This is the minimum interval, in seconds, to negotiate as the Session Keepalive. See RFC7242 Section 5.6.
- Integer

"tcpcl4MyMaxTxSegmentSizeBytes": 200000

- This is the maximum segment size, in bytes, to use for transmitting data segments.
- Integer

"tlsIsRequired": false

- This is whether TLS (Transport Layer Security) is required.
- Boolean

Induct fields:

"numRxCircularBufferBytesPerElement": 100

- This is the maximum size, in bytes, of each element in the circular receive buffer.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 56 of 69

- Integer

"tcpclV4MyMaxRxSegmentSizeBytes": 20000,

- This is the maximum segment size, in bytes, to use for transmitting data segments.
- Integer

"certificatePemFile": "C:hdt_n_ssl_certificatescert.pem"

- This is a path to the file containing the certificate for SSL (Secure Socket Layer).

"privateKeyPemFile": "C:hdt_n_ssl_certificatesprivatekey.pem"

- This is a path to the file containing the private key for SSL.

"diffieHellmanParametersPemFile": "C:hdt_n_ssl_certificatesdh4096.pem"

- This is a path to the file containing the Diffie-Hellman parameters for TLS.

Outduct fields:

"tryUseTls": false

- This is whether TLS is required.
- Boolean

"useTlsVersion1_3": false

- This is whether TLS version 1.3 is required. If not specified, version 1.2 will be used.
- Boolean

"doX509CertificateVerification": false

- This is whether to do X.509 certificate validation is required.
- Boolean

"verifySubjectAltNameInX509Certificate": false

- This is whether to verify the subject alternative name in the X.509 certificate is required.
- Boolean

"certificationAuthorityPemFileForVerification":
"C:hdt_n_ssl_certificatescert.pem"

- This is a path to the file containing the certificate authority.

15.3 UDPCL

Induct fields:

"numRxCircularBufferBytesPerElement": 100

- This is the maximum size, in bytes, of each element in the circular receive buffer.
- Integer

15.4 LTP

Common induct and outduct fields:

"clientServiceId": 1

- This is the ID of the client service.
- Integer

"ltpDataSegmentMtu": 1360

- This is the maximum size, in bytes, of the data portion (excluding LTP headers and UDP headers and IP headers) of an LTP sender's Red data segment being sent. Set this low enough to avoid exceeding ethernet MTU to avoid IP fragmentation.
- Integer

"ltpMaxRetriesPerSerialNumber": 500

- This is the maximum number of retries/resends of a single LTP packet with a serial number before the session is terminated.
- Integer

"ltpMaxUdpPacketsToSendPerSystemCall": 1

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 57 of 69

- This is the maximum number of UDP packets to send per system call.
 - Integer
- "ltpRandomNumberSizeBits": 64
- This is whether to use a 32 or 64 bit random number is required.
 - Integer (32 or 64)
- "oneWayLightTimeMs": 1000
- This is the one way light time. Round trip time (retransmission time) is computed by (2 * (oneWay-LightTime + oneWayMarginTime)).
 - millisecond - Integer
- "oneWayMarginTimeMs": 200
- This is the one way margin (packet processing) time. Round trip time (retransmission time) is computed by (2 * (oneWayLightTime + oneWayMarginTime)).
 - millisecond - Integer
- "remoteLtpEngineId": 20
- This is the ID of the remote LTP engine.
 - Integer
- "thisLtpEngineId": 10
- This is the ID of this LTP engine.
 - Integer
- "delaySendingOfReportSegmentsTimeMsOrZeroToDisable": 20
- Time in milliseconds to defer data retransmission in order to efficiently handle out-of-order report segments.
 - Integer
- "keepActiveSessionDataOnDisk": false
- Supports the running of LTP sessions (both for receivers and senders) from a solid-state disk drive in lieu of keeping session data-segments in memory.
 - If this feature is enabled, it also uses the added configuration values activeSessionDataOnDiskNewFileDurationMs and activeSessionDataOnDiskDirectory to determine where on the drive to temporarily store sessions.
 - As this is still experimental, if a LTP link goes down, bundles don't yet get transferred to storage and get dropped.
 - Boolean

Induct fields:

- "ltpMaxExpectedSimultaneousSessions": 500
- This is the number of expected simultaneous LTP sessions for this engine.
 - Integer
- "ltpRemoteUdpHostname": "localhost"
- This is the remote IP address or hostname.
- "ltpRemoteUdpPort": 4556
- The remote UDP port
 - Integer
- "ltpRxDataSegmentSessionNumberRecreationPreventerHistorySize": 1000
- This is the number of recent LTP receiver history of session numbers to remember. If an LTP receiver's session has been closed and it receives a session number, within the history, the receiver will refuse the session to prevent a potentially old session from being reopened, which has been known to happen with IP fragmentation enabled.
 - Integer
- "preallocatedRedDataBytes": 200000

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 58 of 69

- The number of Red data contiguous bytes to initialized on a receiver. Make this large enough to accommodate the max Red data size so that the LTP receiver does not have to reallocate, copy, and/or delete data while it is receiving Red data. Make this small enough so that the system does not have to allocate too much extra memory per receiving session.
- Integer

Outduct fields:

"ltpCheckpointEveryNthDataSegment": 0

- This enables accelerated retransmission for an LTP sender by making every Nth UDP packet a checkpoint.
- Integer

"ltpSenderBoundPort": 1113

- This is the bound port of the LTP sender.
- Integer

"ltpSenderPingSecondsOrZeroToDisable": 15

- This is the number of seconds between LTP session sender pings during times of zero data segment activity. An LTP ping is defined as a sender sending a cancel segment of a known non-existent session number to a receiver, in which the receiver shall respond with a cancel ACK to determine if the link is active.
- Integer

15.5 STCP

Common induct and outduct fields:

"keepAliveIntervalSeconds": 17

- This is the minimum interval, in seconds, to negotiate as the Session Keepalive.
- Integer

16.0 TEST CONFIGURATIONS AND INSTRUCTIONS

16.1 TCP Loopback Test

To run this simple Loopback Test as shown in Figure 16-1, from the HDTN source directory run the command:

```
./runscript.sh6
```

This works by running 3 modules for about 30 seconds:

- BPGen - Generates the bundles and sends them to the Ingress module.
- HDTN One Process - Launches the modules for HDTN as a single process. Since this is a cutthrough mode test, Storage is not used. Ingress, Egress, and Router are run.
- BPSink - Receives the bundle data from Egress.

⁶ <https://github.com/nasa/HDTN/blob/master/runscript.sh>

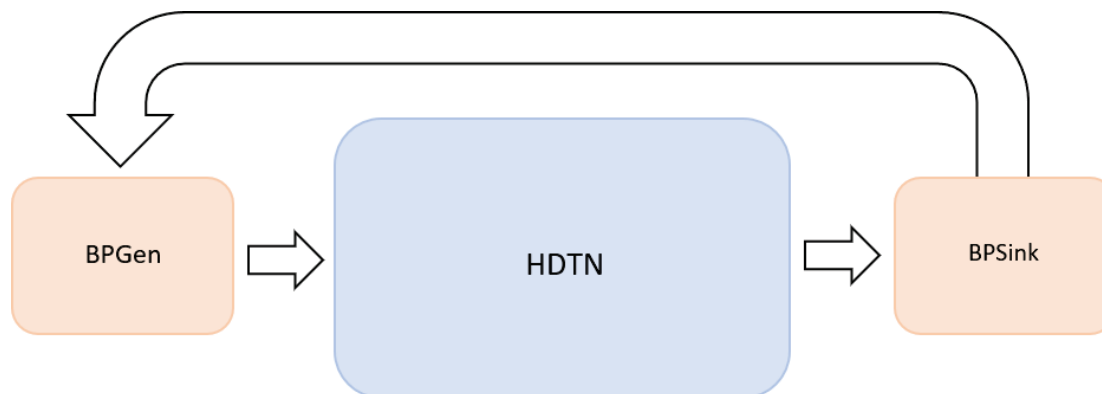


Figure 16-1: HDTN Loopback Test.

16.2 Two Node LTP Test

This test, shown in Figure 16-2, relies on having two nodes running HDTN: the sender and the receiver. The sender will run BPGen, and HDTN One Process. The receiver will run BPSink, and HDTN One Process. Example scripts for this can be found under `HDTN/tests/test_scripts_linux/LTP_2Nodes_Test`⁷.

Separate machines with HDTN installed can each run either the sender or receiver. If using these runscripts this way, users should update the `remoteHostname` field in each config file to the proper IP addresses. The config files for these scripts can be found under `HDTN/config_files/hdtn/hdtn_Node1_ltp.json`⁸ and `HDTN/config_files/hdtn/hdtn_Node2_ltp.json`⁹.

When running this test, users should start the receiver script before the sender script.

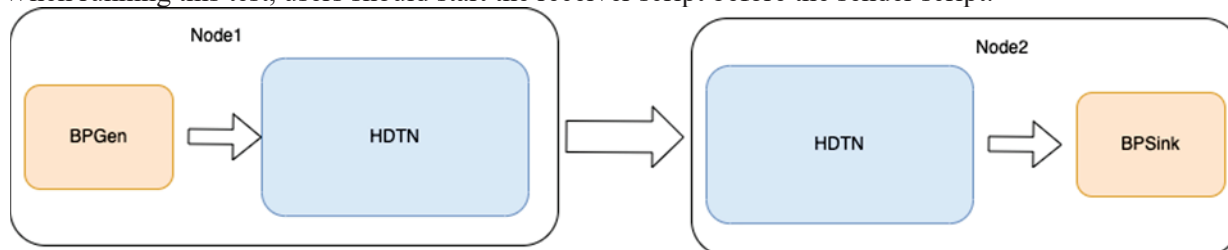


Figure 16-2: HDTN Two Nodes Test

16.3 Four Nodes STCP Test

Shown in Figure 16-3, this test relies on having four nodes running HDTN and uses the router module. Node 1 runs BPGen and HDTN One Process. Node 2 and 3 only run HDTN One Process. The final destination Node 4 will run HDTN One Process and BPSink.

⁷ https://github.com/nasa/HDTN/tree/master/tests/test_scripts_linux/LTP_2Nodes_Test

⁸ https://github.com/nasa/HDTN/blob/master/config_files/hdtn/hdtn_Node1_ltp.json

⁹ https://github.com/nasa/HDTN/blob/master/config_files/hdtn/hdtn_Node2_ltp.json

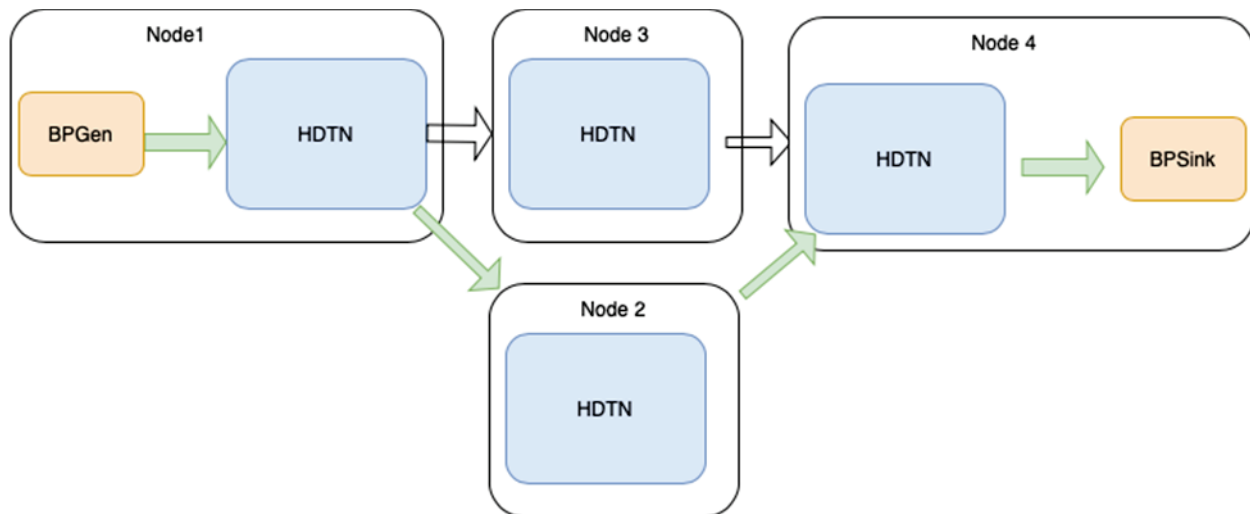


Figure 16-3: HDTN Four Nodes STCP Routing Test

In Node 1's HDTN config file, the next hop is configured to node 3 originally. After the router computes the optimal route to the final destination, the outduct will select node 2 as next hop instead. At initialization, the HDTN json config file for each node has all possible next hops. If we have multiple hops leading to the same final destination, only one Outduct should be initialized with the final destinations vector, and the other next hops Outducts should be dormant, ie having no values initialized in their final destinations json fields. Router will compute the best route and send an event to Egress to update the Outduct to use the nextHop for that optimal route leading to the final destination.

The runscripts for each node can be found under
HDTN/tests/test_scripts_linux/Routing_Test¹⁰.

The config files for Node 1 can be found at
HDTN/config_files/hdtn/hdtn_node1_cfg.json¹¹ with the other node config files immediately following it. If running on separate machines, make sure to update the remoteHostname field in each config file to the proper IP addresses. Users should start the runscript for each node ordered from receiver to sender, i.e start Node 4, then Nodes 3, 2, and 1.

16.4 File Transfer Test

This test relies on having two machines running HDTN: the sender and the receiver. The sender will run BPSendFile, and HDTN One Process. The receiver will run BPReceiveFile, and HDTN One Process. Example scripts were added to send files using LTP and TCPCL convergence layers. They can be found under HDTN/tests/test_scripts_linux/LCRD_File_Transfer_Test¹². Both BPv6 with and without custody transfer and BPv7 with and without BPsec are supported in this example. Figure 16-4 shows the file transfer test with BPsec enabled where the source encrypts the plaintext associated with the payload and the destination does the security verification and decryption of the payload. Separate machines with HDTN installed can each run either the sender or receiver. User should update the remoteHostname field in each HDTN config file in this directory to the proper IP addresses.

When running this test, users should start the receiver script before the sender script.

¹⁰ https://github.com/nasa/HDTN/tree/master/tests/test_scripts_linux/Routing_Test

¹¹ https://github.com/nasa/HDTN/blob/master/config_files/hdtn/hdtn_node1_cfg.json

¹² https://github.com/nasa/HDTN/tree/master/tests/test_scripts_linux/LCRD_File_Transfer_Test

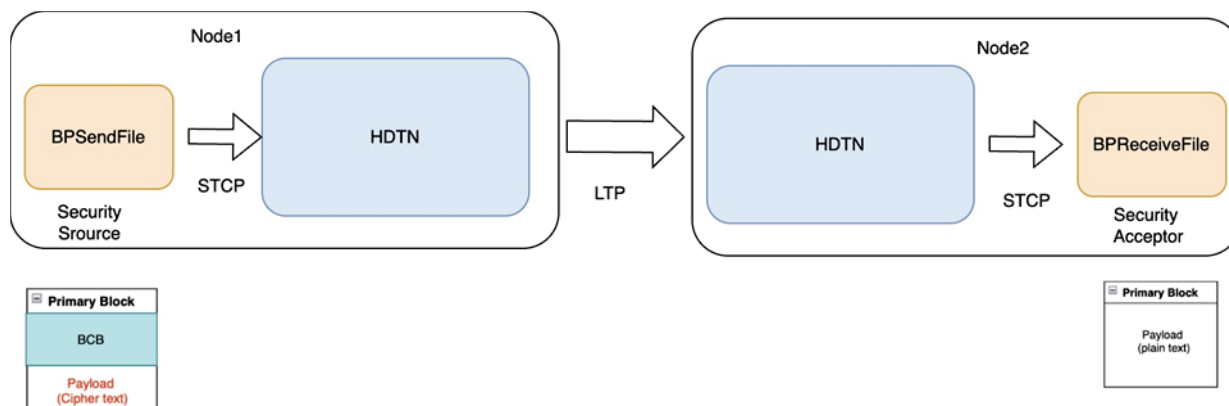


Figure 16-4: HDTN File Transfer with Confidentiality Test

16.5 Integrated Tests

A series of [integrated tests](#)¹³ were added using Boost Test Framework. These tests are automatically run as part of our CI/CD pipeline. The main tests currently included are HDTN Loopback tests in cutthrough and storage modes (using contact plans with or without link disruptions).

17.0 GETTING STARTED WITH STREAMING VIA HDTN

Streaming over HDTN requires the use of the following two HDTN applications: BPSendStream and BPrevcStream. An overview of BPSendStream and BPrevcStream are mentioned in section 11.0.

17.1 Configuration Parameters for BPSendStream

num-circular-buffer-vectors

- The number of circular buffer vector elements refers to the size of the circular buffer used by the UDP sink to store incoming RTP packets before they are processed.
- Default value: 50

max-incoming-udp-packet-size-bytes

- The max size of incoming UDP packets in bytes from the RTPstream.
- Default value: 2

incoming-rtp-stream-port

- The port that will listen for a RTP stream.
- Default value: 50000

rtp-packets-per-bundle

- The number of RTP packets placed into a bundle.
- Default value: 1

induct-type

- The type of induct to use.
- Options: appsink, udp, tcp, fd
- Default value: udp

file-to-stream

- The filepath to an AVC/H.264 encoded video file to stream.
- This parameter **ONLY** supports AVC/H.264 encoded video files. HEVC/H.265 encoded video files are currently **NOT** supported via this parameter.

17.1.1 Example JSON Configuration File for BPSendStream

```
{
```

¹³ https://github.com/nasa/HDTN/blob/master/tests/integrated_tests/src/integrated_tests.cpp

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 62 of 69

```

    "outductConfigName": "bpsendstream to node Z",
    "outductVector": [
      {
        "name": "to localhost hdtm one process",
        "convergenceLayer": "stcp",
        "nextHopNodeId": 1,
        "remoteHostname": "localhost",
        "remotePort": 5000,
        "maxNumberOfBundlesInPipeline": 10000,
        "maxSumOfBundleBytesInPipeline": 50000000,
        "keepAliveIntervalSeconds": 17
      }
    ]
  }
}

```

17.2 Configuration Parameters for BPREcvStream

max-rx-bundle-size-bytes

- The max bundle size in bytes to receive.
- Default value: 10000000 (10 MB)

outgoing-rtp-port

- The port that will transmit the RTP stream.
- Default value: 50560

outgoing-rtp-hostname

- The name of the host that will transmit the RTP stream.
- Default value: 127.0.0.1

num-circular-buffer-vectors

- The number of circular buffer vector elements refers to the size of the circular buffer used by the UDP sink to store incoming RTP packets before they are processed.
- Default value: 50

outduct-type

- The type of outduct to use.
- Options: udp
- Default value: udp

17.2.1 Example JSON Configuration File for BPREcvStream

```

{
  "inductConfigName": "from local hdtm one process to node Z",
  "inductVector": [
    {
      "name": "stcp_bpsink",
      "convergenceLayer": "stcp",
      "boundPort": 7000,
      "numRxCircularBufferElements": 10000,
      "keepAliveIntervalSeconds": 15
    }
  ]
}

```

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 63 of 69

17.3 Runscripts for Streaming Scenarios

For the following streaming scenarios, GStreamer was used to facilitate the handling of multimedia data. GStreamer is an open-source multimedia framework that provides a pipeline-base system for handling multimedia data.

17.3.1 File Streaming

GStreamer was used in this scenario to convert a video file into a RTP stream and forward the stream to BPSendStream. Example runscripts for this type of scenario can be found under the following directory: `tests\test_script_linux\Streaming\file_streaming\`

17.3.2 Live Streaming

GStreamer can be used to fetch a live RTSP stream from a camera and forward the stream to BPSendStream. Consult the GStreamer documentation to leverage this capability.

17.4 More Details

For further insights into the HDTN team's utilization of streaming via HDTN, follow the link provided: [4K High Definition Video and Audio Streaming Across High-rate Delay Tolerant Space Networks](#)¹⁴.

18.0 CONTAINERIZATION

HDTN currently supports the use of Docker and Kubernetes to deploy containers with HDTN built with all required dependencies.

18.1 Docker Instructions

First make sure docker is installed.

```
apt-get install docker
```

Check the service is running.

```
systemctl start docker
```

There are currently two Dockerfiles for building HDTN, one for building an Oracle Linux container and the other for building an Ubuntu. This command will build the Ubuntu one:

```
docker build -t hdt_n_ubuntu_containers/docker/ubuntu/ .
```

The -t sets the name of the image, in this case hdt_n_ubuntu. Check the image was built with the command:

```
docker images
```

Now to run the container use the command:

```
docker run -d -t hdt_n_ubuntu
```

Check that it is running with:

```
docker ps
```

To access it, you'll need the CONTAINER_ID listed with the ps command.

```
docker exec -it container_id bash
```

Stop the container with

```
docker stop container_id
```

¹⁴ https://ntrs.nasa.gov/api/citations/20230018166/downloads/hdt_n_aiaa_scitech_video_streaming.pdf

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 64 of 69

The same container can either be restarted or removed. To see all the containers use:

```
docker ps -a
```

These can still be restarted with the run command above. To remove one that will no longer be used:

```
docker rm container_id
```

18.2 Docker Compose Instructions

Docker compose can be used to spin-up and configure multiple nodes at the same time. This is done using the docker compose file found under HDTN/containers/docker/docker_compose.

```
cd containers/docker/docker_compose
```

This file contains instructions to spin up two containers using Oracle Linux. One is called hdt_n_sender and the other hdt_n_receiver. Start them with the following command:

```
docker compose up
```

On another bash terminal these can be accessed using the command:

```
docker exec -it hdt_n_sender bash
docker exec -it hdt_n_receiver bash
```

This setup is perfect for running a test between two hdt_n nodes. An example script for each node can be found under HDTN/tests/test_scripts_linux/LTP_2Nodes_Test/. Be sure to run the receiver script first, otherwise the sender will have nowhere to send to at launch.

18.3 Kubernetes Instructions

Download the dependencies

```
sudo apt-get install docker microk8s
```

The first step is to create a docker image to be pushed locally for kubernetes to pull:

```
docker build docker/ubuntu/. -t myhdt_n:local
```

Check that it was built:

```
docker images
```

Next we build the image locally and inject it into the microk8s image cache

```
docker save myhdt_n > myhdt_n.tar
microk8s ctr image import myhdt_n.tar
```

Confirm this with:

```
microk8s ctr images ls
```

Now we deploy the cluster, the yaml must reference the injected image name:

```
microk8s kubectl apply -f
containers/kubernetes/hdt_n_10_node_cluster.yaml
```

There should now be ten kubernetes pods running with HDTN. See them with:

```
microk8s kubectl get pods
```

To access a container in a pod, enter the following command:

```
microk8s kubectl exec -it container name -- bash
```

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 65 of 69

When you're finished working with this deployment, delete it using:

```
microk8s kubectl delete deployment hdtm-deployment
```

Use the get pods command to confirm they've been deleted:

```
microk8s kubectl get pods
```

19.0 TROUBLESHOOTING

By default HDTN is built in "Release mode". To enable "Debug mode" during build use:

```
cmake .. -DCMAKE_BUILD_TYPE=Debug
```

19.1 Logging

Logging is controlled by CMake cache variables. Build (or rebuild) HDTN after making the following changes under HDTN/build/CMakeCache.txt. By default logging to a file is turned off to reduce resource draw.

- LOG_LEVEL_TYPE controls which messages are logged. The options, from most verbose to least verbose, are TRACE, DEBUG, INFO, WARNING, ERROR, FATAL, and NONE. All log statements using a level more verbose than the provided level will be compiled out of the application. The default value is INFO.
- LOG_TO_CONSOLE controls whether log messages are sent to the console. The default value is ON.
- LOG_TO_ERROR_FILE controls whether all error messages are written to a single error.log file. The default value is OFF.
- LOG_TO_PROCESS_FILE controls whether each process writes to their own log file. The default value is OFF.
- LOG_TO_SUBPROCESS_FILE controls whether each subprocess writes to their own log file. The default value is OFF.

19.2 LTP Tuning Recommendations

There are several fields in the LTP configuration that will impact performance.

- Client service id corresponds to the LTP Client Service Identifiers as described in RFC7116. In general, select 1 for compatibility with most DTN implementations. See <https://www.iana.org/assignments/ltp-parameters/ltp-parameters.xhtml>. The following values are common for the client service id.
 - 0 - Reserved
 - 1 - Bundle Protocol
 - 2 - LTP Service Data Aggregation
 - 3 - CCSDS File Delivery Service
- The following parameters must match on the sender and receiver:
 - OneWayLightTimeMs
 - OneWayMarginTimeMs
 - LtpMaxRetriesPerSerialNumber
- To properly tune LTP for a system with appreciable delay, make sure 2 x (oneWayLightTimeMs + oneWayMarginTimeMs) is slightly larger than the expected round trip time.
- Set ltpRandomNumberSizeBits to 32 for compatibility with DTNME. For HDTN to HDTN testing use 64.
- Set ltpMaxRetriesPerSerialNumber to a larger number (around 100) on a system that has significant disruptions. For a system that does not experience significant loss, 5 or less should be acceptable.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 66 of 69

- If LTP is being used on a lower-rate communication system that does not provide flow control (for example a radio that supports several Mbps) it is important to set the LTP rate (either in the contact plan or via `--cla-rate`) to slightly lower than the expected link rate. Failure to do so may cause dropped packets since LTP is based on UDP. Alternatively, the max rate can be set per contact in the contact plan by setting `rateBitsPerSec` to a nonzero value. If both fields are set, `rateBitsPerSec` in the contact plan will take precedence.
- For a particular outduct, the max data it can hold in its sending pipeline shall not exceed, whatever comes first, either
 - More bundles than `maxNumberOfBundlesInPipeline`
 - More total bytes of bundles than `maxSumOfBundleBytesInPipeline`
- An error is thrown on startup if `(maxBundleSizeBytes * 2)` is greater than `maxSumOfBundleBytesInPipeline`.
- Worst case RAM memory usage is given by summation of all outduct `maxSumOfBundleBytesInPipeline`. The sum should not exceed the system memory.
- If using Ethernet small frames, it is recommended to set the LTP MTU to 1360 to prevent IP fragmentation.
- Please see `LtpEngineConfig.h` docstrings for specific details related to LTP configuration.
 - <https://github.com/nasa/HDTN/blob/master/common/ltp/include/LtpEngineConfig.h>

20.0 NOTES

20.1 TLS Support for TCPCL Version 4

TLS Versions 1.2 and 1.3 are supported for the TCPCL Version 4 convergence layer. The X.509 certificates must be version 3 to validate IPN URIs using the X.509 “Subject Alternative Name” field. HDTN must be compiled with `ENABLE_OPENSSL_SUPPORT` turned on in CMake. To generate (using a single command) a certificate (which is installed on both an outduct and an induct) and a private key (which is installed on an induct only), such that the induct has a Node Id of 10, use the following command:

```
openssl req -x509 -newkey rsa:4096 -nodes -keyout privatekey.pem -out cert.pem -sha256 -days 365 -extensions v3_req -extensions v3_ca -subj "/C=US/ST=Ohio/ L=Cleveland/O=NASA/OU=HDTN/CN=localhost" -addext "subjectAltName = otherName:1.3.6.1.5.5.7.8.11;IA5:ipn:10.0" -config /path/to/openssl.cnf
```

NOTE: RFC 9174 changed from the earlier -26 draft in that the Subject Alternative Name changed from a URI to an otherName with ID 1.3.6.1.5.5.7.8.11 (id-on-bundleEID).

- Therefore, do NOT use: `-addext "subjectAltName = URI:ipn:10.0"`
- Instead, use: `-addext "subjectAltName = otherName:1.3.6.1.5.5.7.8.11;IA5:ipn:10.0"`

To generate the Diffie-Hellman parameters PEM file (which is installed on an induct only), use the following command:

```
openssl dhparam -outform PEM -out dh4096.pem 4096
```

20.2 BP Version 6 and Version 7

Both versions of BP, BP version 6 (BPv6) and BP version 7 (BPv7), are supported within the bpcodex library.

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 67 of 69

APPENDIX A Acronyms

The following table contains an alphabetized list of the definitions for abbreviations and acronyms used in this document.

Acronym	Definition
ACK	Acknowledgement
ACS	Aggregate Custody Signals
AOS	Advanced Orbiting Solutions
API	Application Programming Interface
ARM	Advanced RISC Machine
AVC	Advanced Video Coding
BP	Bundle Protocol
BPSec	Bundle Protocol Security
CCSDS	Consultative Committee for Space Data Systems
CD	Continuous Delivery
CGR	Contact Graph Routing
CI	Continuous integration
CL	Convergence Layer
CLI	Command Line Interface
CM	Control Manager
COM	Communication port
CPU	Computer Processing Unit
CRC	Cyclic Redundancy Check
CSV	Comma-separated Values
DTN	Delay Tolerant Networking
DTNME	DTN Marshall Enterprise
EID	Endpoint Identifier
EOB	End of Block
GRC	Glenn Research Center
GUI	Graphical User Interface
HDTN	High-Rate Delay Tolerant Networking
HEVC	High Efficiency Video Coding
ID	Identifier
IP	Internet Protocol
IPN	InterPlanetary Network
JSON	JavaScript Object Notation
LTP	Licklider Transport Protocol
MTU	Maximum Transmission Unit
NASA	National Aeronautics Space Administration
OEL	Oracle Enterprise Linux
PEM	Privacy Enhanced Mail
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RFC	Request for Comments
RHEL	Red Hat Enterprise Linux
RTP	Real-time Transport Protocol
SCaN	Space Communications and Navigation
SDNV	Self-Delimiting Numeric Values

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 68 of 69

Acronym	Definition
SOMD	Space Operations Mission Directorate
STCP	Simple TCP
SWaP	Size, Weight and Power
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UI	User Interface
URI	Uniform Resource Identifier
WS	WebSocket
WSS	WebSocket Secure

Space Operations Mission Directorate (SOMD) High-Rate Delay Tolerant Networking (HDTN) Project		
Title: HDTN User Guide for V2.0.0	Document No.: HDTN-USER-036	Revision: 2.0.0
	Cage Code No.: 1QFP5	Page 69 of 69

APPENDIX B TBD, TBR, and TBW Table

There are currently no TBD, TBR, or TBW items.