Delay-Tolerant Networking Working Group
Internet Draft
Intended status: Standards Track
Expires: May 2017

S. Burleigh
JPL, Calif. Inst. Of Technology
K. Fall
Carnegie Mellon University / SEI
E. Birrane
APL, Johns Hopkins University

October 29, 2016

Bundle Protocol
draft-ietf-dtn-bpbis-06.txt

Status of this Memo

Copyright Notice

Abstract

This Internet Draft presents a specification for Bundle Protocol, adapted from the experimental Bundle Protocol specification developed by the Delay-Tolerant Networking Research group of the Internet Research Task Force and documented in RFC 5050.

Table of Contents

1. Introduction

Since the publication of the Bundle Protocol Specification (Experimental RFC 5050[RFC5050]) in 2007, the Delay-Tolerant Networking Bundle Protocol has been implemented in multiple programming languages and deployed to a wide variety of computing platforms for a wide range of successful exercises. This implementation and deployment experience has demonstrated the general utility of the protocol for challenged network operations.

It has also, as expected, identified opportunities for making the protocol simpler, more capable, and easier to use. The present document, standardizing the Bundle Protocol (BP), is adapted from RFC 5050 in that context.

This document describes version 7 of BP.

Delay Tolerant Networking is a network architecture providing communications in and/or through highly stressed environments. Stressed networking environments include those with intermittent connectivity, large and/or variable delays, and high bit error rates. To provide its services, BP may be viewed as sitting at the application layer of some number of constituent networks, forming a store-carry-forward overlay network. Key capabilities of BP include:

- Custodial forwarding
- Ability to cope with intermittent connectivity, including cases where the sender and receiver are not concurrently present in the network
- Ability to take advantage of scheduled, predicted, and opportunistic connectivity, whether bidirectional or unidirectional, in addition to continuous connectivity
- Late binding of overlay network endpoint identifiers to underlying constituent network addresses

For descriptions of these capabilities and the rationale for the DTN architecture, see [ARCH] and [SIGC]. [TUT] contains a tutorial-level overview of DTN concepts.

BP's location within the standard protocol stack is as shown in Figure 1. BP uses underlying "native" transport and/or network protocols for communications within a given constituent network.

The interface between the bundle protocol and a specific underlying protocol is termed a "convergence layer adapter".

Figure 1 shows three distinct transport and network protocols (denoted T1/N1, T2/N2, and T3/N3).

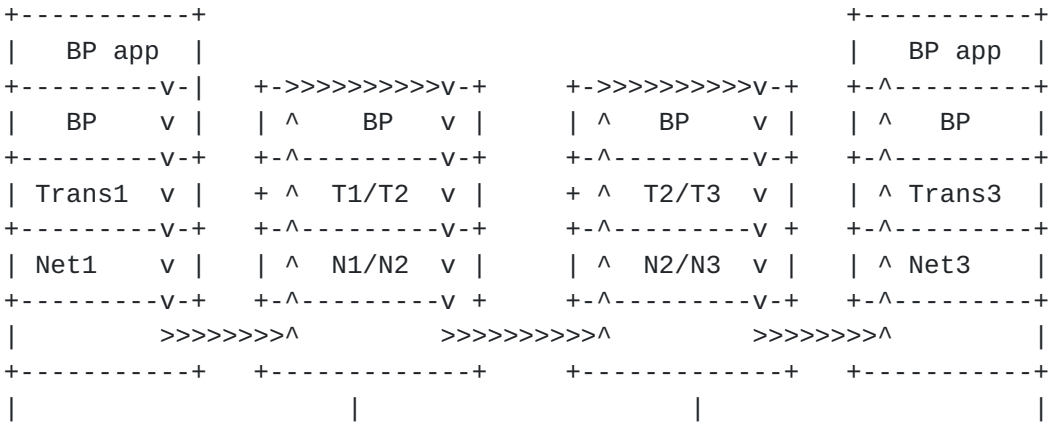


Figure 1: The Bundle Protocol in the Protocol Stack Model

This document describes the format of the protocol data units (called "bundles") passed between entities participating in BP communications.

The entities are referred to as "bundle nodes". This document does not address:

- Operations in the convergence layer adapters that bundle nodes use to transport data through specific types of internets. (However, the document does discuss the services that must be provided by each adapter at the convergence layer.)
- The bundle route computation algorithm.
- Mechanisms for populating the routing or forwarding information bases of bundle nodes.
- The mechanisms for securing bundles en route.
- The mechanisms for managing bundle nodes.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Service Description

3.1. Definitions

Bundle - A bundle is a protocol data unit of BP, so named because negotiation of the parameters of a data exchange may be impractical in a delay-tolerant network: it is often better practice to "bundle" with a unit of data all metadata that might be needed in order to make the data immediately usable when delivered to applications. Each bundle comprises a sequence of two or more "blocks" of protocol data, which serve various purposes.

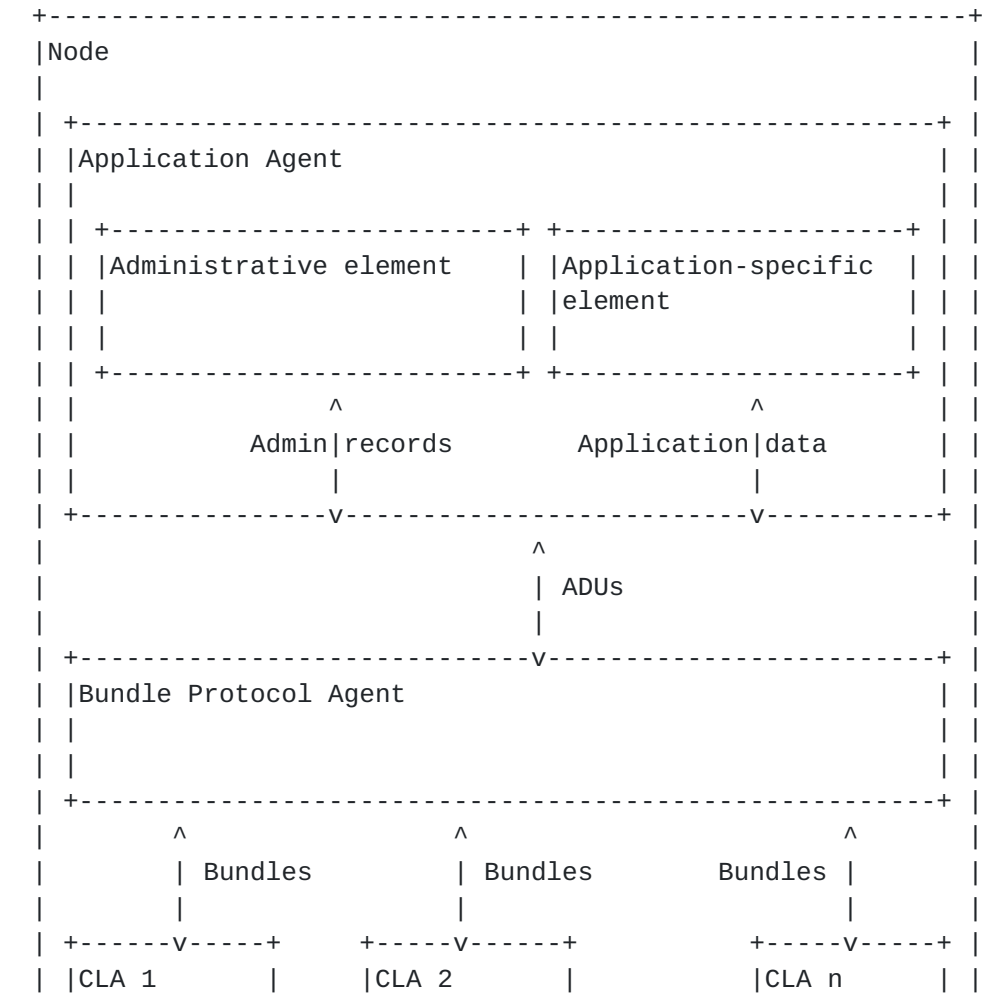
Block - A bundle protocol block is one of the protocol data structures that together constitute a well-formed bundle.

Bundle payload - A bundle payload (or simply "payload") is the application data whose conveyance to the bundle's destination is the purpose for the transmission of a given bundle; it is the content of the bundle's payload block. The terms "bundle content", "bundle payload", and "payload" are used interchangeably in this document.

Partial payload - A partial payload is a payload that comprises either the first N bytes or the last N bytes of some other payload of length M, such that 0 < N < M.

Fragment - A fragment is a bundle whose payload block contains a partial payload.

Bundle node - A bundle node (or, in the context of this document, simply a "node") is any entity that can send and/or receive bundles. Each bundle node has three conceptual components, defined below, as shown in Figure 2: a "bundle protocol agent", a set of zero or more "convergence layer adapters", and an "application agent".



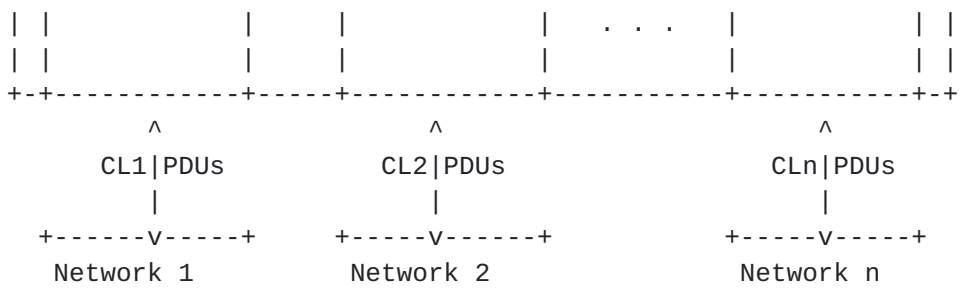


Figure 2: Components of a BP Node

Bundle protocol agent - The bundle protocol agent (BPA) of a node is the node component that offers the BP services and executes the procedures of the bundle protocol.

Convergence layer adapter - A convergence layer adapter (CLA) is a node component that sends and receives bundles on behalf of the BPA, utilizing the services of some 'native' protocol stack that is supported in one of the networks within which the node is functionally located.

Application agent - The application agent (AA) of a node is the node component that utilizes the BP services to effect communication for some user purpose. The application agent in turn has two elements, an administrative element and an application-specific element.

Application-specific element - The application-specific element of an AA is the node component that constructs, requests transmission of, accepts delivery of, and processes units of user application data.

Administrative element - The administrative element of an AA is the node component that constructs and requests transmission of administrative records (defined below), including status reports and custody signals, and accepts delivery of and processes any custody signals that the node receives.

Administrative record - A BP administrative record is an application data unit that is exchanged between the administrative elements of nodes' application agents for some BP administrative purpose. The formats of some fundamental administrative records (and of no other application data units) are defined in this specification.

Bundle endpoint - A bundle endpoint (or simply "endpoint") is a set of zero or more bundle nodes that all identify themselves for BP purposes by some common identifier, called a "bundle endpoint ID"

(or, in this document, simply "endpoint ID"; endpoint IDs are described in detail in Section 4.4.1 below).

Singleton endpoint - A singleton endpoint is an endpoint that always contains exactly one member.

Registration - A registration is the state machine characterizing a given node's membership in a given endpoint. Any single registration has an associated delivery failure action as defined below and must at any time be in one of two states: Active or Passive.

Delivery - A bundle is considered to have been delivered at a node subject to a registration as soon as the application data unit that is the payload of the bundle, together with any relevant metadata (an implementation matter), has been presented to the node's application agent in a manner consistent with the state of that registration.

Deliverability - A bundle is considered "deliverable" subject to a registration if and only if (a) the bundle's destination endpoint is the endpoint with which the registration is associated, (b) the bundle has not yet been delivered subject to this registration, and (c) the bundle has not yet been "abandoned" (as defined below) subject to this registration.

Abandonment - To abandon a bundle subject to some registration is to assert that the bundle is not deliverable subject to that registration.

Delivery failure action - The delivery failure action of a registration is the action that is to be taken when a bundle that is "deliverable" subject to that registration is received at a time when the registration is in the Passive state.

Destination - The destination of a bundle is the endpoint comprising the node(s) at which the bundle is to be delivered (as defined below).

Minimum reception group - The minimum reception group of an endpoint is the minimum number of members of the endpoint (nodes) at which the bundle must have been delivered in order for the bundle to be considered delivered to the endpoint.

Transmission - A transmission is an attempt by a node's BPA to cause copies of a bundle to be delivered at all nodes in the minimum reception group of some endpoint (the bundle's destination) in

response to a transmission request issued by the node's application agent.

Forwarding - To forward a bundle to a node is to invoke the services of a CLA in a sustained effort to cause a copy of the bundle to be received by that node.

Discarding - To discard a bundle is to cease all operations on the bundle and functionally erase all references to it. The specific procedures by which this is accomplished are an implementation matter.

Retention constraint - A retention constraint is an element of the state of a bundle that prevents the bundle from being discarded. That is, a bundle cannot be discarded while it has any retention constraints.

Deletion - To delete a bundle is to remove unconditionally all of the bundle's retention constraints, enabling the bundle to be discarded.

Custodian - A custodian of a bundle is a node that has determined that it will retain a copy of that bundle for an indefinite period of time, forwarding and possibly re-forwarding the bundle as appropriate, until it detects one of the conditions under which it may cease being a custodian of that bundle (discussed later).

Taking custody - To take custody of a bundle is to become a custodian of that bundle.

Accepting custody - To accept custody of a bundle is to take custody of the bundle, mark the bundle in such a way as to indicate this custodianship to nodes that subsequently receive copies of the bundle, and announce this custodianship to all current custodians of the bundle.

Refusing custody - To "refuse custody" of a bundle is to notify all current custodians of that bundle that an opportunity to take custody of the bundle has been declined.

Releasing custody - To release custody of a bundle is to cease to be a custodian of the bundle.

3.2. Discussion of BP concepts

Multiple instances of the same bundle (the same unit of DTN protocol data) might exist concurrently in different parts of a network --

possibly differing in some blocks -- in the memory local to one or more bundle nodes and/or in transit between nodes. In the context of the operation of a bundle node, a bundle is an instance (copy), in that node's local memory, of some bundle that is in the network.

The payload for a bundle forwarded in response to a bundle transmission request is the application data unit whose location is provided as a parameter to that request. The payload for a bundle forwarded in response to reception of a bundle is the payload of the received bundle.

In the most familiar case, a bundle node is instantiated as a single process running on a general-purpose computer, but in general the definition is meant to be broader: a bundle node might alternatively be a thread, an object in an object-oriented operating system, a special-purpose hardware device, etc.

The manner in which the functions of the BPA are performed is wholly an implementation matter. For example, BPA functionality might be coded into each node individually; it might be implemented as a shared library that is used in common by any number of bundle nodes on a single computer; it might be implemented as a daemon whose services are invoked via inter-process or network communication by any number of bundle nodes on one or more computers; it might be implemented in hardware.

Every CLA implements its own thin layer of protocol, interposed between BP and the (usually "top") protocol(s) of the underlying native protocol stack; this "CL protocol" may only serve to multiplex and de-multiplex bundles to and from the underlying native protocol, or it may offer additional CL-specific functionality. The manner in which a CLA sends and receives bundles is, again, wholly an implementation matter. The definitions of CLAs and CL protocols are beyond the scope of this specification.

Note that the administrative element of a node's application agent may itself, in some cases, function as a convergence-layer adapter. That is, outgoing bundles may be "tunneled" through encapsulating bundles:

- An outgoing bundle constitutes a byte array. This byte array may, like any other, be presented to the bundle protocol agent as an application data unit that is to be transmitted to some endpoint.
- The original bundle thus forms the payload of an encapsulating bundle that is forwarded using some other convergence-layer protocol(s).

. When the encapsulating bundle is received, its payload is delivered to the peer application agent administrative element, which then instructs the bundle protocol agent to dispatch that original bundle in the usual way.

The purposes for which this technique may be useful (such as cross-domain security) are beyond the scope of this specification.

The only interface between the BPA and the application-specific element of the AA is the BP service interface. But between the BPA and the administrative element of the AA there is a (conceptual) private control interface in addition to the BP service interface. This private control interface enables the BPA and the administrative element of the AA to direct each other to take action under specific circumstances

In the case of a node that serves simply as a BP "router", the AA may have no application-specific element at all. The application-specific elements of other nodes' AAs may perform arbitrarily complex application functions, perhaps even offering multiplexed DTN communication services to a number of other applications. As with the BPA, the manner in which the AA performs its functions is wholly an implementation matter.

Singletons are the most familiar sort of endpoint, but in general the endpoint notion is meant to be broader. For example, the nodes in a sensor network might constitute a set of bundle nodes that identify themselves by a single common endpoint ID and thus form a single bundle endpoint. *Note* too that a given bundle node might identify itself by multiple endpoint IDs and thus be a member of multiple bundle endpoints.

The destination of every bundle is an endpoint, which may or may not be singleton. The source of every bundle is a node, identified by the endpoint ID for some singleton endpoint that contains that node.

The minimum reception group of an endpoint may be any one of the following: (a) ALL of the nodes registered in an endpoint that is permitted to contain multiple nodes (in which case forwarding to the endpoint is functionally similar to "multicast" operations in the Internet, though possibly very different in implementation); (b) ANY N of the nodes registered in an endpoint that is permitted to contain multiple nodes, where N is in the range from zero to the cardinality of the endpoint; or (c) THE SOLE NODE registered in a singleton endpoint (in which case forwarding to the endpoint is functionally similar to "unicast" operations in the Internet).

The nature of the minimum reception group for a given endpoint can typically be determined from the endpoint's ID. For some endpoint ID "schemes", the nature of the minimum reception group is fixed - in a manner that is defined by the scheme - for all endpoints identified under the scheme. For other schemes, the nature of the minimum reception group is indicated by some lexical feature of the "scheme-specific part" of the endpoint ID, in a manner that is defined by the scheme.

Any number of transmissions may be concurrently undertaken by the bundle protocol agent of a given node.

When the bundle protocol agent of a node determines that a bundle must be forwarded to a node (either to a node that is a member of the bundle's destination endpoint or to some intermediate forwarding node) in the course of completing the successful transmission of that bundle, it invokes the services of a CLA in a sustained effort to cause a copy of the bundle to be received by that node.

Upon reception, the processing of a bundle that has been received by a given node depends on whether or not the receiving node is registered in the bundle's destination endpoint. If it is, and if the payload of the bundle is non-fragmentary (possibly as a result of successful payload reassembly from fragmentary payloads, including the original payload of the newly received bundle), then the bundle is normally delivered to the node's application agent subject to the registration characterizing the node's membership in the destination endpoint.

Whenever, for some implementation-specific reason, a node's BPA finds it impossible to immediately deliver a bundle that is deliverable, delivery of the bundle has failed. In this event, the delivery failure action associated with the applicable registration must be taken. Delivery failure action MUST be one of the following:

- defer delivery of the bundle subject to this registration until (a) this bundle is the least recently received of all bundles currently deliverable subject to this registration and (b) either the registration is polled or else the registration is in the Active state; or

- abandon delivery of the bundle subject to this registration.

An additional implementation-specific delivery deferral procedure MAY optionally be associated with the registration.

While the state of a registration is Passive, reception of a bundle that is deliverable subject to this registration MUST cause delivery of the bundle to be abandoned or deferred as mandated by the registration's current delivery failure action; in the latter case, any additional delivery deferral procedure associated with the registration MUST also be performed.

While the state of a registration is Active, reception of a bundle that is deliverable subject to this registration MUST cause the bundle to be delivered automatically as soon as it is the next bundle that is due for delivery according to the BPA's bundle delivery scheduling policy, an implementation matter.

Normally only registrations' registered delivery failure actions cause deliveries to be abandoned.

Custody of a bundle MAY be taken only if the destination of the bundle is a singleton endpoint. Custody MAY be released only when either (a) notification is received that some other node has accepted custody of the same bundle; (b) notification is received that the bundle has been delivered at the (sole) node registered in the bundle's destination endpoint; (c) the current custodian chooses to fragment the bundle, releasing custody of the original bundle and taking custody of the fragments instead, or (d) the bundle is explicitly deleted for some reason, such as lifetime expiration.

The custody transfer mechanism in BP provides a means of recovering from data loss along the path to the destination node. When the custodian of a bundle forwards that bundle it SHOULD set a retransmission timer; reception of a responding custody signal of any kind prior to timer expiration MUST disable that timer. Upon expiration of that timer, the custodian MUST re-forward the bundle. When a bundle for which custody has been taken arrives at a node from which it must be forwarded, and that node determines that it will forward the bundle but will not take custody, the receiving node SHOULD send a "custody delegation" signal back to the custodian indicating the next node to which the bundle will be forwarded together with an estimate of the interval that will elapse between the time the bundle was received and the time at which it will be forwarded. This mechanism is intended to facilitate accurate timeout interval calculation for this bundle.

Computation of the timeout interval for a bundle's custody transfer timer (i.e., determination of the moment at which a responding custody signal is expected) is an implementation matter and may be dynamically responsive to changes in connectivity. In some environments it may be impossible to compute this interval with

operationally satisfactory accuracy; in such environments the use of custody transfer services is contraindicated.

Alternatively, when custody transfer for a given bundle is not requested, data loss along the path to the destination node can be minimized by utilizing reliable convergence-layer protocols between neighbors on all segments of the end-to-end path. This approach may make more efficient use of links than custody transfer because a convergence-layer protocol may perform finer-grained retransmission than custody transfer does, retransmitting only the specific portions of a transmitted bundle that were not received, rather than the entire bundle. However, in some environments there may be segments of the end-to-end path for which no reliable convergence-layer protocol is available; in such environments the use of reliable convergence-layer protocols wherever possible can reduce the incidence of data loss.

3.3. Services Offered by Bundle Protocol Agents

The BPA of each node is expected to provide the following services to the node's application agent:

- commencing a registration (registering the node in an endpoint);
- terminating a registration;
- switching a registration between Active and Passive states;
- transmitting a bundle to an identified bundle endpoint;
- canceling a transmission;
- polling a registration that is in the Passive state;
- delivering a received bundle.

4. Bundle Format

The format of bundles SHALL conform to the Concise Binary Object Representation (CBOR [RFC7049]).

Each bundle SHALL be a concatenated sequence of at least two blocks, represented as a CBOR indefinite-length array. The first block in the sequence (the first item of the array) MUST be a primary bundle block in CBOR representation as described below; the bundle MUST have exactly one primary bundle block. The primary block MUST be followed by one or more canonical bundle blocks (additional array items) in CBOR representation as described below. The last such block MUST be a payload block; the bundle MUST have exactly one payload block. The last item of the array, immediately following the payload block, SHALL be a CBOR "break" stop code.

4.1. BP Fundamental Data Structures

4.1.1. CRC Type

CRC type is an unsigned integer type code for which the following values (and no others) are valid:

- 0 indicates "no CRC is present."
- 1 indicates "a CRC-16 (a.k.a., CRC-16-ANSI) is present."
- 2 indicates "a standard IEEE 802.3 CRC-32 is present."

CRC type SHALL be represented as a CBOR unsigned integer.

4.1.2. CRC

CRC SHALL be omitted from a block if and only if the block's CRC type code is zero.

When not omitted, the CRC SHALL be represented as a CBOR unsigned integer.

4.1.3. Bundle Processing Control Flags

Bundle processing control flags assert properties of the bundle as a whole rather than of any particular block of the bundle. They are conveyed in the primary block of the bundle.

The following properties are asserted by the bundle processing control flags:

- The bundle is a fragment. (Boolean)
- The bundle's payload is an administrative record. (Boolean)
- The bundle must not be fragmented. (Boolean)
- Custody transfer requested for this bundle. (Boolean)
- The bundle's destination endpoint is a singleton. (Boolean)
- Acknowledgment by the user application is requested. (Boolean)
- Status time is requested in all status reports. (Boolean)
- The bundle contains a "manifest" extension block. (Boolean)
- Flags requesting types of status reports (all Boolean):

- Request reporting of bundle reception.
- Request reporting of custody transfer request processing.
- Request reporting of bundle forwarding.
- Request reporting of bundle delivery.
- Request reporting of bundle deletion.

If the bundle processing control flags indicate that the bundle's application data unit is an administrative record, then the custody transfer requested flag value must be zero and all status report request flag values must be zero.

If the custody transfer requested flag value is 1, then the source node is requesting that every receiving node accept custody of the bundle.

If the bundle's source node is omitted (i.e., the source node ID is the ID of the null endpoint, which has no members as discussed below; this option enables anonymous bundle transmission), then the bundle is not uniquely identifiable and all bundle protocol features that rely on bundle identity must therefore be disabled: the bundle's custody transfer requested flag value must be zero, the "Bundle must not be fragmented" flag value must be 1, and all status report request flag values must be zero.

The bundle processing control flags SHALL be represented as a CBOR unsigned integer item containing a bit field of 16 bits indicating the control flag values as follows:

. Bit 0 (the high-order bit, 0x8000): reserved.
. Bit 1 (0x4000): reserved.
. Bit 2 (0x2000): reserved.
. Bit 3(0x1000): bundle deletion status reports are requested.
. Bit 4(0x0800): bundle delivery status reports are requested.
. Bit 5(0x0400): bundle forwarding status reports are requested.
. Bit 6(0x0200): custody request processing status reports are requested.
. Bit 7(0x0100): bundle reception status reports are requested.
. Bit 8(0x0080): bundle contains a Manifest block.
. Bit 9(0x0040): status time is requested in all status reports.
. Bit 10(0x0020): user application acknowledgement is requested.
. Bit 11(0x0010): destination is a singleton endpoint.
. Bit 12(0x0008): custody transfer is requested.
. Bit 13(0x0004): bundle must not be fragmented.

- Bit 14(0x0002): payload is an administrative record.
- Bit 15 (the low-order bit, 0x0001: bundle is a fragment.

4.1.4. Block Processing Control Flags

The block processing control flags assert properties of canonical bundle blocks. They are conveyed in the header of the block to which they pertain.

The following properties are asserted by the block processing control flags:

- This block must be replicated in every fragment. (Boolean)

- Status report must be transmitted if this block can't be processed. (Boolean)

- Block must be removed from the bundle if it can't be processed. (Boolean)

- Bundle must be deleted if this block can't be processed. (Boolean)

For each bundle whose bundle processing control flags indicate that the bundle's application data unit is an administrative record, or whose source node ID is the null endpoint ID as defined below, the value of the "Transmit status report if block can't be processed" flag in every canonical block of the bundle must be zero.

The block processing control flags SHALL be represented as a CBOR unsigned integer item containing a bit field of 8 bits indicating the control flag values as follows:

- Bit 0 (the high-order bit, 0x80): reserved.
- Bit 1 (0x40): reserved.
- Bit 2(0x20): reserved.
- Bit 3(0x10): reserved.
- Bit 4(0x08): bundle must be deleted if block can't be processed.
- Bit 5(0x04): status report must be transmitted if block can't be processed.
- Bit 6(0x02): block must be removed from bundle if it can't be processed.
- Bit 7(the low-order bit, 0x01): block must be replicated in every fragment.

4.1.5. Identifiers

4.1.5.1. Endpoint ID

The destinations of bundles are bundle endpoints, identified by text strings termed "endpoint IDs" (see Section 3.1). Each endpoint ID (EID) is a Uniform Resource Identifier (URI; [URI]). As such, each endpoint ID can be characterized as having this general structure:

< scheme name > : < scheme-specific part, or "SSP" >

The scheme identified by the < scheme name > in an endpoint ID is a set of syntactic and semantic rules that fully explain how to parse and interpret the SSP. The set of allowable schemes is effectively unlimited. Any scheme conforming to [URIREG] may be used in a bundle protocol endpoint ID.

Note that, although endpoint IDs are URIs, implementations of the BP service interface may support expression of endpoint IDs in some internationalized manner (e.g., Internationalized Resource Identifiers (IRIs); see [RFC3987]).

The endpoint ID "dtn:none" identifies the "null endpoint", the endpoint that by definition never has any members.

Each BP endpoint ID (EID) SHALL be represented as a CBOR array comprising a 2-tuple.

The first item of the array SHALL be the code number identifying the endpoint's URI scheme [URI], as defined in the registry of URI scheme code numbers for Bundle Protocol maintained by IANA as described in Section 9. [URIREG]. Each URI scheme code number SHALL be represented as a CBOR unsigned integer.

The second item of the array SHALL be the applicable CBOR representation of the scheme-specific part (SSP) of the EID, defined as follows:

. If the EID's URI scheme is "dtn" then the SSP SHALL be represented as a CBOR text string unless the EID's SSP is "none", in which case the SSP SHALL be represented as a CBOR unsigned integer with the value zero.
. If the EID's URI scheme is "ipn" then the SSP SHALL be represented as a CBOR array comprising a 2-tuple. The first item of this array SHALL be the EID's node number represented as a CBOR unsigned integer. The second item of this array

SHALL be the EID's service number represented as a CBOR unsigned integer.

. Definitions of the CBOR representations of the SSPs of EIDs encoded in other URI schemes are included in the specifications defining those schemes.

4.1.5.2. Node ID

For many purposes of the Bundle Protocol it is important to identify the node that is operative in some context.

As discussed in 3.1 above, nodes are distinct from endpoints; specifically, an endpoint is a set of zero or more nodes. But rather than define a separate namespace for node identifiers, we instead use endpoint identifiers to identify nodes, subject to the following restrictions:

. Every node MUST be a member of at least one singleton endpoint.
. The EID of any singleton endpoint of which a node is a member MAY be used to identify that node. A "node ID" is an EID that is used in this way.
. A node's membership in a given singleton endpoint MUST be sustained at least until the nominal operation of the Bundle Protocol no longer depends on the identification of that node using that endpoint's ID.

4.1.6. DTN Time

A DTN time is an unsigned integer indicating a count of seconds since the start of the year 2000 on the Coordinated Universal Time (UTC) scale [UTC]. Each DTN time SHALL be represented as a CBOR unsigned integer item.

4.1.7. Creation Timestamp

Each creation timestamp SHALL be represented as a CBOR array item comprising a 2-tuple.

The first item of the array SHALL be a DTN time.

The second item of the array SHALL be the creation timestamp's sequence number, represented as a CBOR unsigned integer.

4.1.8. Block-type-specific Data

Block-type-specific data in each block (other than the primary block) SHALL be the applicable CBOR representation of the content of

the block. Details of this representation are included in the specification defining the block type.

4.2. Bundle Representation

This section describes the primary block in detail and non-primary blocks in general. Rules for processing these blocks appear in Section 5 of this document.

Note that supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [BPSEC]) may require that BP implementations conforming to those protocols construct and process additional blocks.

4.2.1. Bundle

Each bundle SHALL be represented as a CBOR indefinite-length array. The first item of this array SHALL be the CBOR representation of a Primary Block. Every other item of the array except the last SHALL be the CBOR representation of a Canonical Block. The last item of the array SHALL be a CBOR "break" stop code.

4.2.2. Primary Bundle Block

The primary bundle block contains the basic information needed to forward bundles to their destinations.

Each primary block SHALL be represented as a CBOR array; the number of elements in the array SHALL be 8 (if the bundle is not a fragment and CRC type is zero) or 9 (if the bundle is not a fragment and CRC type is non-zero) or 10 (if the bundle is a fragment and CRC type is zero) or 11 (if the bundle is a fragment and CRC-type is non-zero).

The fields of the primary bundle block SHALL be as follows, listed in the order in which they MUST appear:

Version: An unsigned integer value indicating the version of the bundle protocol that constructed this block. The present document describes version 7 of the bundle protocol. Version number SHALL be represented as a CBOR unsigned integer item.

Bundle Processing Control Flags: The Bundle Processing Control Flags are discussed in Section 4.1.3. above.

CRC Type: CRC Type codes are discussed in Section 4.1.1. above.

Destination EID: The Destination EID field identifies the bundle endpoint that is the bundle's destination, i.e., the endpoint that contains the node(s) at which the bundle is to be delivered.

Source node ID: The Source node ID field identifies the bundle node at which the bundle was initially transmitted, except that Source node ID may be the null endpoint ID in the event that the bundle's source chooses to remain anonymous.

Report-to EID: The Report-to EID field identifies the bundle endpoint to which status reports pertaining to the forwarding and delivery of this bundle are to be transmitted.

Creation Timestamp: The creation timestamp is a pair of unsigned integers that, together with the source node ID and (if the bundle is a fragment) the fragment offset, serve to identify the bundle. The first of these integers is the bundle's creation time, while the second is the bundle's creation timestamp sequence number. Bundle creation time shall be the time - expressed in seconds since the start of the year 2000, on the Coordinated Universal Time (UTC) scale [UTC] - at which the transmission request was received that resulted in the creation of the bundle. Sequence count shall be the latest value (as of the time at which that transmission request was received) of a monotonically increasing positive integer counter managed by the source node's bundle protocol agent that may be reset to zero whenever the current time advances by one second. For nodes that lack accurate clocks (that is, nodes that are not at all moments able to determine the current UTC time to within 30 seconds), bundle creation time MUST be set to zero and the counter used as the source of the bundle sequence count MUST NEVER be reset to zero. In any case, a source Bundle Protocol Agent MUST NEVER create two distinct bundles with the same source node ID and bundle creation timestamp. The combination of source node ID and bundle creation timestamp serves to identify a single transmission request, enabling it to be acknowledged by the receiving application (provided the source node ID is not the null endpoint ID).

Lifetime: The lifetime field is an unsigned integer that indicates the time at which the bundle's payload will no longer be useful, encoded as a number of seconds past the creation time. When a bundle's age exceeds its lifetime, bundle nodes need no longer retain or forward the bundle; the bundle SHOULD be deleted from the network. Bundle lifetime SHALL be represented as a CBOR unsigned integer item.

Fragment offset: If and only if the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment,

fragment offset SHALL be present in the primary block. Fragment offset SHALL be represented as a CBOR unsigned integer indicating the offset from the start of the original application data unit at which the bytes comprising the payload of this bundle were located.

Total Application Data Unit Length: If and only if the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment, total application data unit length SHALL be present in the primary block. Total application data unit length SHALL be represented as aCBOR unsigned integer indicating the total length of the original application data unit of which this bundle's payload is a part.

CRC: If and only if the value of the CRC type field of this Primary block is non-zero, a CRC SHALL be present in the primary block. The length and nature of the CRC SHALL be as indicated by the CRC type. The CRC SHALL be computed over the concatenation of all bytes of the primary block including the CRC field itself, which for this purpose SHALL be temporarily populated with the value zero.

4.2.3. Canonical Bundle Block Format

Every block other than the primary block (which blocks are termed "canonical" blocks) SHALL be represented as a CBOR array; the number of elements in the array SHALL be 6 (if CRC type is zero) or 7 (otherwise).

The fields of every canonical block SHALL be as follows, listed in the order in which they MUST appear:

. Block type code, an unsigned integer. Bundle block type code 1 indicates that the block is a bundle payload block. Block type codes 2 through 9 are explicitly reserved as noted later in this specification. Block type codes 192 through 255 are not reserved and are available for private and/or experimental use. All other block type code values are reserved for future use.
. Block number, an unsigned integer. The block number uniquely identifies the block within the bundle, enabling blocks (notably bundle security protocol blocks) to explicitly reference other blocks in the same bundle. Block numbers need not be in continuous sequence, and blocks need not appear in block number sequence in the bundle. The block number of the payload block is always zero.
. Block processing control flags as discussed in Section 4.1.4 above.
. CRC type as discussed in Section 4.1.1 above.

- If and only if the value of the CRC type field of this block is non-zero, a CRC. If present, the length and nature of the CRC SHALL be as indicated by the CRC type and the CRC SHALL be computed over the concatenation of all bytes of the block including the CRC field itself, which for this purpose SHALL be temporarily populated with the value zero.
- Block data length, an unsigned integer. The block data length field SHALL contain the aggregate length of all remaining fields of the block, i.e., the block-type-specific data fields. Block data length SHALL be represented as a CBOR unsigned integer item.
- Block-type-specific data fields, whose nature and order are type-specific and whose aggregate length in octets is the value of the block data length field. For the Payload Block in particular (block type 1), there SHALL be exactly one block-type-specific data field, termed the "payload", which SHALL be an application data unit, or some contiguous extent thereof, represented as a CBOR byte string.

4.3. Extension Blocks

"Extension blocks" are all blocks other than the primary and payload blocks. Because not all extension blocks are defined in the Bundle Protocol specification (the present document), not all nodes conforming to this specification will necessarily instantiate Bundle Protocol implementations that include procedures for processing (that is, recognizing, parsing, acting on, and/or producing) all extension blocks. It is therefore possible for a node to receive a bundle that includes extension blocks that the node cannot process. The values of the block processing control flags indicate the action to be taken by the bundle protocol agent when this is the case.

(Note that, while CBOR permits considerable flexibility in the encoding of bundles, this flexibility must not be interpreted as inviting increased complexity in protocol data unit structure.)

The following extension blocks are defined in other DTN protocol specification documents as noted:

- Block Integrity Block (block type 2) and Block Confidentiality Block (block type 3) are defined in the Bundle Security Protocol specification (work in progress).
- Manifest Block (block type 4) is defined in the Manifest Extension Block specification (TBD). The manifest block identifies the blocks that were present in the bundle at the time it was created. The bundle MUST contain one (1) occurrence of this type of block if the value of the "manifest"

flag in the bundle processing control flags is 1; otherwise the bundle MUST NOT contain any Manifest block.

. The Flow Label Block (block type 6) is defined in the Flow Label Extension Block specification (TBD). The flow label block is intended to govern transmission of the bundle by convergence-layer adapters.

The following extension blocks are defined in the current document.

4.3.1. Current Custodian

The Current Custodian block, block type 5, identifies a node that is known to have accepted custody of the bundle. The block-type-specific data of this block is the node ID of a custodian, which SHALL take the form of an endpoint ID represented as described in Section 4.1.5.2. above. The bundle MAY contain one or more occurrences of this type of block.

4.3.2. Previous Node

The Previous Node block, block type 7, identifies the node that forwarded this bundle to the local node (i.e., to the node at which the bundle currently resides); its block-type-specific data is the node ID of that forwarder node which SHALL take the form of a node ID represented as described in Section 4.1.5.2. above. If the local node is the source of the bundle, then the bundle MUST NOT contain any previous node block. Otherwise the bundle MUST contain one (1) occurrence of this type of block. If present, the previous node block MUST be the FIRST block following the primary block, as the processing of other extension blocks may depend on its value.

4.3.3. Bundle Age

The Bundle Age block, block type 8, contains the number of seconds that have elapsed between the time the bundle was created and time at which it was most recently forwarded. It is intended for use by nodes lacking access to an accurate clock, to aid in determining the time at which a bundle's lifetime expires. The block-type-specific data of this block is an unsigned integer containing the age of the bundle in seconds, which SHALL be represented as a CBOR unsigned integer item. (The age of the bundle is the sum of all known intervals of the bundle's residence at forwarding nodes, up to the time at which the bundle was most recently forwarded, plus the summation of signal propagation time over all episodes of transmission between forwarding nodes. Determination of these values is an implementation matter.) If the bundle's creation time is zero, then the bundle MUST contain exactly one (1) occurrence of

this type of block; otherwise, the bundle MAY contain at most one (1) occurrence of this type of block.

4.3.4. Hop Count

The Hop Count block, block type 9, contains two unsigned integers, hop limit and hop count. A "hop" is here defined as an occasion on which a bundle was forwarded from one node to another node. The hop count block is mainly intended as a safety mechanism, a means of identifying bundles for removal from the network that can never be delivered due to a persistent forwarding error: a bundle SHOULD be deleted when its hop count exceeds its hop limit. Procedures for determining the appropriate hop limit for a block are beyond the scope of this specification. The block-type-specific data in a hop count block SHALL be represented as a CBOR array comprising a 2-tuple. The first item of this array SHALL be the bundle's hop limit, represented as a CBOR unsigned integer. The second item of this array SHALL be the bundle's hop count, represented as a CBOR unsigned integer. A bundle MAY contain at most one (1) occurrence of this type of block.

5. Bundle Processing

The bundle processing procedures mandated in this section and in Section 6 govern the operation of the Bundle Protocol Agent and the Application Agent administrative element of each bundle node. They are neither exhaustive nor exclusive. Supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [BPSEC]) may augment, override, or supersede the mandates of this document.

5.1. Generation of Administrative Records

All transmission of bundles is in response to bundle transmission requests presented by nodes' application agents. When required to "generate" an administrative record (such as a bundle status report or a custody signal), the bundle protocol agent itself is responsible for causing a new bundle to be transmitted, conveying that record. In concept, the bundle protocol agent discharges this responsibility by directing the administrative element of the node's application agent to construct the record and request its transmission as detailed in Section 6 below. In practice, the manner in which administrative record generation is accomplished is an implementation matter, provided the constraints noted in Section 6 are observed.

Under some circumstances, the requesting of status reports could result in an unacceptable increase in the bundle traffic in the network. For this reason, the generation of status reports is mandatory only in two cases:

- the reception of a bundle containing at least one block that cannot be processed, for which the value of the "transmit status report if block could not be processed" block processing flag is 1, and
- the deletion of a bundle for which custody transfer is requested.

In all other cases, the decision on whether or not to generate a requested status report is left to the discretion of the bundle protocol agent. Mechanisms that could assist in making such decisions, such as pre-placed agreements authorizing the generation of status reports under specified circumstances, are beyond the scope of this specification.

Notes on administrative record terminology:

- A "bundle reception status report" is a bundle status report with the "reporting node received bundle" flag set to 1.
- A "custody transfer status report" is a bundle status report with the "reporting node attempted custody transfer" flag set to 1.
- A "bundle forwarding status report" is a bundle status report with the "reporting node forwarded the bundle" flag set to 1.
- A "bundle delivery status report" is a bundle status report with the "reporting node delivered the bundle" flag set to 1.
- A "bundle deletion status report" is a bundle status report with the "reporting node deleted the bundle" flag set to 1.
- A "current custodian" of a bundle is a node identified in a Current Custodian extension block of that bundle.

5.2. Bundle Transmission

The steps in processing a bundle transmission request are:

Step 1: If custody transfer is requested for this bundle transmission then the destination MUST be a singleton endpoint. If, moreover, custody acceptance by the source node is required when custody is requested (an implementation matter) but the conditions under which custody of the bundle may be accepted are not satisfied, then the request cannot be honored and all remaining steps of this procedure MUST be skipped.

Step 2: Transmission of the bundle is initiated. An outbound bundle MUST be created per the parameters of the bundle transmission request, with the retention constraint "Dispatch pending". The source node ID of the bundle MUST be either the null endpoint ID, indicating that the source of the bundle is anonymous, or else the EID of a singleton endpoint whose only member is the node of which the BPA is a component.

Step 3: Processing proceeds from Step 1 of Section 5.4.

5.3. Bundle Dispatching

The steps in dispatching a bundle are:

Step 1: If the bundle's destination endpoint is an endpoint of which the node is a member, the bundle delivery procedure defined in Section 5.7 MUST be followed.

Step 2: Processing proceeds from Step 1 of Section 5.4.

5.4. Bundle Forwarding

The steps in forwarding a bundle are:

Step 1: The retention constraint "Forward pending" MUST be added to the bundle, and the bundle's "Dispatch pending" retention constraint MUST be removed.

Step 2: The bundle protocol agent MUST determine whether or not forwarding is contraindicated for any of the reasons listed in Figure 4. In particular:

- The bundle protocol agent MUST determine which node(s) to forward the bundle to. The bundle protocol agent MAY choose either to forward the bundle directly to its destination node(s) (if possible) or to forward the bundle to some other node(s) for further forwarding. The manner in which this decision is made may depend on the scheme name in the destination endpoint ID and/or on other state but in any case is beyond the scope of this document. If the BPA elects to forward the bundle to some other node(s) for further forwarding but finds it impossible to select any node(s) to forward the bundle to, then forwarding is contraindicated.
- Provided the bundle protocol agent succeeded in selecting the node(s) to forward the bundle to, the bundle protocol agent MUST select the convergence layer adapter(s) whose services will enable the node to send the bundle to those nodes. The

manner in which specific appropriate convergence layer adapters are selected is beyond the scope of this document. If the agent finds it impossible to select any appropriate convergence layer adapter(s) to use in forwarding this bundle, then forwarding is contraindicated.

Step 3: If forwarding of the bundle is determined to be contraindicated for any of the reasons listed in Figure 4, then the Forwarding Contraindicated procedure defined in Section 5.4.1 MUST be followed; the remaining steps of Section 5 are skipped at this time.

Step 4: If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1, then the custody transfer procedure defined in Section 5.10 MUST be followed.

Step 5: For each node selected for forwarding, the bundle protocol agent MUST invoke the services of the selected convergence layer adapter(s) in order to effect the sending of the bundle to that node. Determining the time at which the bundle protocol agent invokes convergence layer adapter services is a BPA implementation matter. Determining the time at which each convergence layer adapter subsequently responds to this service invocation by sending the bundle is a convergence-layer adapter implementation matter. Note that:

- If the bundle contains a flow label extension block then that flow label value MAY identify procedures for determining the order in which convergence layer adapters must send bundles, e.g., considering bundle source when determining the order in which bundles are sent. The definition of such procedures is beyond the scope of this specification.
- If the bundle has a bundle age block, then at the last possible moment before the CLA initiates conveyance of the bundle node via the CL protocol the bundle age value MUST be increased by the difference between the current time and the time at which the bundle was received (or, if the local node is the source of the bundle, created).

Step 6: When all selected convergence layer adapters have informed the bundle protocol agent that they have concluded their data sending procedures with regard to this bundle:

- If the "request reporting of bundle forwarding" flag in the bundle's status report request field is set to 1, then a bundle forwarding status report SHOULD be generated, destined for the bundle's report-to endpoint ID. If the bundle has the retention

constraint "custody accepted" and all of the nodes to which the bundle was forwarded are known to be unable to send bundles back to this node, then the reason code on this bundle forwarding status report MUST be "forwarded over unidirectional link"; otherwise, the reason code MUST be "no additional information".

. The bundle's "Forward pending" retention constraint MUST be removed.

5.4.1. Forwarding Contraindicated

The steps in responding to contraindication of forwarding are:

Step 1: The bundle protocol agent MUST determine whether or not to declare failure in forwarding the bundle. Note: this decision is likely to be influenced by the reason for which forwarding is contraindicated.

Step 2: If forwarding failure is declared, then the Forwarding Failed procedure defined in Section 5.4.2 MUST be followed.

Otherwise, (a) if the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1, then the custody transfer procedure defined in Section 5.10 MUST be followed; (b) when -- at some future time - the forwarding of this bundle ceases to be contraindicated, processing proceeds from Step 5 of Section 5.4.

5.4.2. Forwarding Failed

The steps in responding to a declaration of forwarding failure are:

Step 1: If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1, custody transfer failure must be handled. The bundle protocol agent MUST handle the custody transfer failure by generating a custody signal of type 1 (custody refusal) for the bundle, destined for the bundle's current custodian(s); the custody signal MUST contain a reason code corresponding to the reason for which forwarding was determined to be contraindicated. (Note that discarding the bundle will not delete it from the network, since each current custodian still has a copy.) In addition, if the "request reporting of custody transfer attempted" flag in the bundle's status report request field is set to 1, a custody transfer status report with the same reason code SHOULD be generated, destined for the report-to endpoint ID of the bundle.

If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 0, then the bundle protocol agent MAY forward the bundle back to the node that sent it, as identified by the Previous Node block.

Step 2: If the bundle's destination endpoint is an endpoint of which the node is a member, then the bundle's "Forward pending" retention constraint MUST be removed. Otherwise, the bundle MUST be deleted: the bundle deletion procedure defined in Section 5.14 MUST be followed, citing the reason for which forwarding was determined to be contraindicated.

5.5. Bundle Expiration

A bundle expires when the bundle's age exceeds its lifetime as specified in the primary bundle block. Bundle age MAY be determined by subtracting the bundle's creation timestamp time from the current time if (a) that timestamp time is not zero and (b) the local node's clock is known to be accurate (as discussed in section 4.5.1 above); otherwise bundle age MUST be obtained from the Bundle Age extension block. Bundle expiration MAY occur at any point in the processing of a bundle. When a bundle expires, the bundle protocol agent MUST delete the bundle for the reason "lifetime expired": the bundle deletion procedure defined in Section 5.14 MUST be followed.

5.6. Bundle Reception

The steps in processing a bundle that has been received from another node are:

Step 1: The retention constraint "Dispatch pending" MUST be added to the bundle.

Step 2: If the "request reporting of bundle reception" flag in the bundle's status report request field is set to 1, then a bundle reception status report with reason code "No additional information" SHOULD be generated, destined for the bundle's report-to endpoint ID.

Step 3: For each block in the bundle that is an extension block that the bundle protocol agent cannot process:

. If the block processing flags in that block indicate that a status report is requested in this event, then a bundle reception status report with reason code "Block unintelligible" SHOULD be generated, destined for the bundle's report-to endpoint ID.

- If the block processing flags in that block indicate that the bundle must be deleted in this event, then the bundle protocol agent MUST delete the bundle for the reason "Block unintelligible"; the bundle deletion procedure defined in Section 5.14 MUST be followed and all remaining steps of the bundle reception procedure MUST be skipped.
- If the block processing flags in that block do NOT indicate that the bundle must be deleted in this event but do indicate that the block must be discarded, then the bundle protocol agent MUST remove this block from the bundle.
- If the block processing flags in that block indicate neither that the bundle must be deleted nor that that the block must be discarded, then processing continues with the next extension block that the bundle protocol agent cannot process, if any; otherwise, processing proceeds from step 4.

Step 4: If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1 and the bundle has the same source node ID, creation timestamp, and (if the bundle is a fragment) fragment offset as another bundle that (a) has not been discarded and (b) currently has the retention constraint "Custody accepted", then custody transfer redundancy MUST be handled; otherwise, processing proceeds from Step 5. The bundle protocol agent MUST handle custody transfer redundancy by generating a custody signal of type 1 (custody refusal) for this bundle with reason code "Redundant reception", destined for this bundle's current custodian, and removing this bundle's "Dispatch pending" retention constraint.

Step 5: Processing proceeds from Step 1 of Section 5.3.

5.7. Local Bundle Delivery

The steps in processing a bundle that is destined for an endpoint of which this node is a member are:

Step 1: If the received bundle is a fragment, the application data unit reassembly procedure described in Section 5.9 MUST be followed. If this procedure results in reassembly of the entire original application data unit, processing of this bundle (whose fragmentary payload has been replaced by the reassembled application data unit) proceeds from Step 2; otherwise, the retention constraint "Reassembly pending" MUST be added to the bundle and all remaining steps of this procedure MUST be skipped.

Step 2: Delivery depends on the state of the registration whose endpoint ID matches that of the destination of the bundle:

. If the registration is in the Active state, then the bundle MUST be delivered subject to this registration (see Section 3.1 above) as soon as all previously received bundles that are deliverable subject to this registration have been delivered.
. If the registration is in the Passive state, then the registration's delivery failure action MUST be taken (see Section 3.1 above).

Step 3: As soon as the bundle has been delivered:

. If the "request reporting of bundle delivery" flag in the bundle's status report request field is set to 1, then a bundle delivery status report SHOULD be generated, destined for the bundle's report-to endpoint ID. Note that this status report only states that the payload has been delivered to the application agent, not that the application agent has processed that payload.
. If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 1, custodial delivery MUST be reported. The bundle protocol agent MUST report custodial delivery by generating a custody signal of type 0 (custody acceptance) for the bundle, destined for the bundle's current custodian(s).

5.8. Bundle Fragmentation

It may at times be advantageous for bundle protocol agents to reduce the sizes of bundles in order to forward them. This might be the case, for example, if a node to which a bundle is to be forwarded is accessible only via intermittent contacts and no upcoming contact is long enough to enable the forwarding of the entire bundle.

The size of a bundle can be reduced by "fragmenting" the bundle. To fragment a bundle whose payload is of size M is to replace it with two "fragments" -- new bundles with the same source node ID and creation timestamp as the original bundle -- whose payloads are the first N and the last (M - N) bytes of the original bundle's payload, where 0 < N < M. Note that fragments may themselves be fragmented, so fragmentation may in effect replace the original bundle with more than two fragments. (However, there is only one 'level' of fragmentation, as in IP fragmentation.)

Any bundle that has any Current Custodian extension block citing any node other than the local node MUST NOT be fragmented. This restriction aside, any bundle whose primary block's bundle processing flags do NOT indicate that it must not be fragmented MAY

be fragmented at any time, for any purpose, at the discretion of the bundle protocol agent.

Fragmentation SHALL be constrained as follows:

- The concatenation of the payloads of all fragments produced by fragmentation MUST always be identical to the payload of the fragmented bundle (that is, the bundle that is being fragmented). Note that the payloads of fragments resulting from different fragmentation episodes, in different parts of the network, may be overlapping subsets of the fragmented bundle's payload.
- The primary block of each fragment MUST differ from that of the fragmented bundle, in that the bundle processing flags of the fragment MUST indicate that the bundle is a fragment and both fragment offset and total application data unit length must be provided. Additionally, the CRC of the fragmented bundle, if any, MUST be replaced in each fragment by a new CRC computed for the primary block of that fragment.
- The payload blocks of fragments will differ from that of the fragmented bundle as noted above.
- If the fragmented bundle is not a fragment or is the fragment with offset zero, then all extension blocks of the fragmented bundle MUST be replicated in the fragment whose offset is zero.
- Each of the fragmented bundle's extension blocks whose "Block must be replicated in every fragment" flag is set to 1 MUST be replicated in every fragment.
- Beyond these rules, replication of extension blocks in the fragments is an implementation matter.
- If the local node is a custodian of the fragmented bundle, then the BPA MUST release custody of the fragmented bundle before fragmentation occurs and MUST take custody of every fragment.

5.9. Application Data Unit Reassembly

If the concatenation -- as informed by fragment offsets and payload lengths -- of the payloads of all previously received fragments with the same source node ID and creation timestamp as this fragment, together with the payload of this fragment, forms a byte array whose length is equal to the total application data unit length in the fragment's primary block, then:

- This byte array -- the reassembled application data unit -- MUST replace the payload of this fragment.
- The BPA MUST take custody of each fragmentary bundle whose payload is a subset of the reassembled application data unit,

for which custody transfer is requested but the BPA has not yet taken custody.

. The BPA MUST then release custody of every fragment whose payload is a subset of the reassembled application data unit, for which it has taken custody.

. The "Reassembly pending" retention constraint MUST be removed from every other fragment whose payload is a subset of the reassembled application data unit.

Note: reassembly of application data units from fragments occurs at the nodes that are members of destination endpoints as necessary; an application data unit MAY also be reassembled at some other node on the path to the destination.

5.10. Custody Transfer

The decision as to whether or not to accept custody of a bundle is an implementation matter that may involve both resource and policy considerations.

If the bundle protocol agent elects to accept custody of the bundle, then it must follow the custody acceptance procedure defined in Section 5.10.1.

5.10.1. Custody Acceptance

Procedures for acceptance of custody of a bundle are defined as follows.

The retention constraint "Custody accepted" MUST be added to the bundle.

If the "request reporting of custody transfer attempted" flag in the bundle's status report request field is set to 1, a custody transfer status report with reason code 0 SHOULD be generated, destined for the report-to endpoint ID of the bundle. However, if a bundle reception status report was generated for this bundle (Step 2 of Section 5.6) but has not yet been transmitted, then this report SHOULD be generated by simply turning on the "Reporting node attempted custody transfer" flag in that earlier report.

The bundle protocol agent MUST generate a custody signal of type 0 (custody acceptance) for the bundle, destined for the bundle's current custodian(s).

The bundle protocol agent MUST assert the new current custodian for the bundle. It does so by deleting all of the bundle's existing

Current Custodian extension blocks and inserting a new Current Custodian extension block whose value is the node ID of the local node.

If the value of a custody transfer timer interval for this bundle can be calculated with operationally satisfactory accuracy, then the bundle protocol agent SHOULD set a custody transfer countdown timer for the bundle; upon expiration of this timer prior to expiration of the bundle itself and prior to custody transfer success for this bundle, the custody transfer failure procedure detailed in Section 5.12 MUST be followed. The manner in which the countdown interval for such a timer is determined is an implementation matter.

The bundle SHOULD be retained in persistent storage if possible.

5.10.2. Custody Release

When custody of a bundle is released, the "Custody accepted" retention constraint MUST be removed from the bundle and any custody transfer timer that has been established for this bundle SHOULD be destroyed.

5.11. Custody Transfer Success

Upon receipt of a custody signal of type 0 (custody acceptance) at a node that is a custodial node of the bundle identified in the custody signal, custody of the bundle MUST be released as described in Section 5.10.2.

5.12. Custody Transfer Failure

Custody transfer is determined to have failed at a custodial node for a given bundle when either (a) that node's custody transfer timer for that bundle (if any) expires or (b) a custody signal of type 1 (custody refusal) for that bundle is received at that node.

Upon determination of custody transfer failure due to expiration of a custody transfer countdown timer, the bundle protocol agent MUST re-forward the bundle, possibly on a different route (Section 5.4).

Upon determination of custody transfer failure due to reception of a custody signal of type 1 (custody refusal), the action taken by the bundle protocol agent is implementation-specific and may depend on the reason code cited for the refusal. For example, if the custody signal's reason code was "Depleted storage", the bundle protocol agent might choose to re-forward the bundle, possibly on a different route (Section 5.4). If the reason code was "Redundant reception",

on the other hand, this might cause the bundle protocol agent to release custody of the bundle and to revise its algorithm for computing countdown intervals for custody transfer timers.

5.13. Custody Transfer Deferral

Upon receipt of a bundle for which custody transfer retransmission service has been requested, which the bundle protocol agent plans to forward but for which it elects not to accept custody, the bundle protocol agent SHOULD generate a custody signal of type 2 (custody delegation) for the bundle, destined for the bundle's current custodian(s).

Custody transfer is determined to have been deferred at a custodial node for given bundle when a custody signal of type 2 (custody delegation) for that bundle is received at that node. The action taken by the bundle protocol agent in this event is implementation-specific. Notionally, this is an opportunity for the bundle protocol agent to revise its retransmission timeout interval for this bundle, based on the information provided in the custody signal: the next candidate custodian for the bundle is now known, and the minimum length of time before a custody acceptance signal will arrive can now be adjusted accordingly.

5.14. Bundle Deletion

The steps in deleting a bundle are:

Step 1: If the retention constraint "Custody accepted" currently prevents this bundle from being discarded, then:

. Custody of the bundle is released as described in Section 5.10.2.
. A bundle deletion status report citing the reason for deletion MUST be generated, destined for the bundle's report-to endpoint ID.

Otherwise, if the "request reporting of bundle deletion" flag in the bundle's status report request field is set to 1, then a bundle deletion status report citing the reason for deletion SHOULD be generated, destined for the bundle's report-to endpoint ID.

Step 2: All of the bundle's retention constraints MUST be removed.

5.15. Discarding a Bundle

As soon as a bundle has no remaining retention constraints it MAY be discarded, thereby releasing any persistent storage that may have been allocated to it.

5.16. Canceling a Transmission

When requested to cancel a specified transmission, where the bundle created upon initiation of the indicated transmission has not yet been discarded, the bundle protocol agent MUST delete that bundle for the reason "transmission cancelled". For this purpose, the procedure defined in Section 5.14 MUST be followed.

6. Administrative Record Processing

6.1. Administrative Records

Administrative records are standard application data units that are used in providing some of the features of the Bundle Protocol. Two types of administrative records have been defined to date: bundle status reports and custody signals. Note that additional types of administrative records may be defined by supplementary DTN protocol specification documents.

Every administrative record consists of:

- Record type code (an unsigned integer for which valid values are as defined below).
- Record content in type-specific format.

Valid administrative record type codes are defined as follows:

Value	Meaning
1	Bundle status report.
2	Custody signal.

| (other) | Reserved for future use. |

+---------+--+

Figure 3: Administrative Record Type Codes

Each BP administrative record SHALL be represented as a CBOR array comprising a 2-tuple.

The first item of the array SHALL be a record type code, which SHALL be represented as a CBOR unsigned integer.

The second element of this array SHALL be the applicable CBOR representation of the content of the record. Details of the CBOR representation of administrative record types 1 and 2 are provided below. Details of the CBOR representation of other types of administrative record type are included in the specifications defining those records.

6.1.1. Bundle Status Reports

The transmission of "bundle status reports" under specified conditions is an option that can be invoked when transmission of a bundle is requested. These reports are intended to provide information about how bundles are progressing through the system, including notices of receipt, custody transfer, forwarding, final delivery, and deletion. They are transmitted to the Report-to endpoints of bundles.

Each bundle status report SHALL be represented as a CBOR array. The number of elements in the array SHALL be either 6 (if the subject bundle is a fragment) or 4 (otherwise).

The first item of the bundle status report array SHALL be bundle status information represented as a CBOR array of 5 elements. The five items of the bundle status information array shall provide information on the following five status assertions, in this order:

. Reporting node received bundle.
. Reporting node attempted custody transfer. When this status is asserted, a reason code value of 0 ("No additional information") SHALL indicate that custody was accepted.
. Reporting node forwarded the bundle.
. Reporting node delivered the bundle.
. Reporting node deleted the bundle.

Each item of the bundle status information array SHALL be a bundle status item represented as a CBOR array; the number of elements in each such array SHALL be either 2 (if the value of the first item of this bundle status item is 1 AND the "Report status time" flag was set to 1 in the bundle processing flags of the bundle whose status is being reported) or 1 (otherwise). The first item of the bundle status item array SHALL be a status indicator, a Boolean value indicating whether or not the corresponding bundle status is asserted, represented as a CBOR Boolean value. The second item of the bundle status item array, if present, SHALL indicate the time (as reported by the local system clock, an implementation matter) at which the indicated status was asserted for this bundle, represented as a DTN time as described in Section 4.1.6. above.

The second item of the bundle status report array SHALL be the bundle status report reason code explaining the value of the status indicator, represented as a CBOR unsigned integer. Valid status report reason codes are defined in Figure 4 below but the list of status report reason codes provided here is neither exhaustive nor exclusive; supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [BPSEC]) may define additional reason codes.

Value	Meaning
0	No additional information.
1	Lifetime expired.
2	Forwarded over unidirectional link.
3	Transmission canceled.
4	Depleted storage.

5	Destination endpoint ID unintelligible.
6	No known route to destination from here.
7	No timely contact with next node on route.
8	Block unintelligible.
(other)	Reserved for future use.

Figure 4: Status Report Reason Codes

The third item of the bundle status report array SHALL be the source node ID identifying the source of the bundle whose status is being reported, represented as described in Section 4.1.5.2. above.

The fourth item of the bundle status report array SHALL be the creation timestamp of the bundle whose status is being reported, represented as described in Section 4.1.7. above.

The fifth item of the bundle status report array SHALL be present if and only if the bundle whose status is being reported contained a fragment offset. If present, it SHALL be the subject bundle's fragment offset represented as a CBOR unsigned integer item.

The sixth item of the bundle status report array SHALL be present if and only if the bundle whose status is being reported contained a fragment offset. If present, it SHALL be the length of the subject bundle's payload represented as a CBOR unsigned integer item.

6.1.2. Custody Signals

Custody signals are administrative records that effect custody transfer operations. They are transmitted to the nodes that are the current custodians of bundles.

Each custody signal SHALL be represented as a CBOR array. The number of elements in the array SHALL be 6 (if the subject bundle is a fragment) or 4 (otherwise).

The first item of the custody signal array SHALL be a signal type code represented as a CBOR unsigned integer. Valid custody signal types are defined as follows:

Value	Meaning
0	Custody acceptance. The reporting node accepted custody of the bundle.
1	Custody refusal. The reporting node refused custody of the bundle.
2	Custody delegation: the bundle will be forwarded but custody was not taken.
(other)	Reserved for future use.

Figure 5: Custody Signal Type Codes

The second item of the custody signal array SHALL be additional information amplifying the signal type code, represented as a CBOR array. The number of elements in the array SHALL be either 2 (if the signal type is 2) or 1 (otherwise).

When signal type is 0 or 1, the sole item in the additional information array SHALL be a custody signal reason code, represented as a CBOR unsigned integer. Valid custody signal reason codes are defined as follows:

Value	Meaning
0	No additional information.
1	Reserved for future use.
2	Reserved for future use.
3	Redundant (reception by a node that is a custodial node for this bundle).
4	Depleted storage.
5	Destination endpoint ID unintelligible.
6	No known route destination from here.
7	No timely contact with next node on route.
8	Block unintelligible.
(other)	Reserved for future use.

Figure 6: Custody Signal Reason Codes

When signal type is 2, the first item in the additional information array SHALL be the node ID of the node to which the reporting node anticipates forwarding the bundle, represented as described in Section 4.1.5.2. above, and the second item in this array SHALL be an estimate of the number of seconds that will have elapsed since reception of the bundle before the anticipated forwarding begins, represented as a CBOR unsigned integer.

The third item of the custody signal array SHALL be the source node ID identifying the source of the bundle for which custodial activity is being reported, represented as described in Section 4.1.5.2. above.

The fourth item of the custody signal array SHALL be the creation timestamp of the bundle for which custodial activity is being reported, represented as described in Section 4.1.7. above.

The fifth item of the custody signal array SHALL be present if and only if the bundle for which custodial activity is being reported contained a fragment offset. If present, it SHALL be the subject bundle's fragment offset represented as a CBOR unsigned integer item.

The sixth item of the custody signal array SHALL be present if and only if the bundle for which custodial activity is being reported contained a fragment offset. If present, it SHALL be the length of the subject bundle's payload represented as a CBOR unsigned integer item.

6.2. Generation of Administrative Records

Whenever the application agent's administrative element is directed by the bundle protocol agent to generate an administrative record with reference to some bundle, the following procedure must be followed:

Step 1: The administrative record must be constructed. If the referenced bundle is a fragment, the administrative record MUST contain the fragment offset and fragment length.

Step 2: A request for transmission of a bundle whose payload is this administrative record MUST be presented to the bundle protocol agent.

6.3. Reception of Custody Signals

For each received custody signal that has signal type zero (custody acceptance), the administrative element of the application agent MUST direct the bundle protocol agent to follow the custody transfer success procedure in Section 5.11.

For each received custody signal that has signal type 1 (custody refusal), the administrative element of the application agent MUST direct the bundle protocol agent to follow the custody transfer failure procedure in Section 5.12.

For each received custody signal that has signal type 2 (custody delegation), the administrative element of the application agent MUST direct the bundle protocol agent to follow the custody delegation procedure in Section 5.13.

7. Services Required of the Convergence Layer

7.1. The Convergence Layer

The successful operation of the end-to-end bundle protocol depends on the operation of underlying protocols at what is termed the "convergence layer"; these protocols accomplish communication between nodes. A wide variety of protocols may serve this purpose, so long as each convergence layer protocol adapter provides a defined minimal set of services to the bundle protocol agent. This convergence layer service specification enumerates those services.

7.2. Summary of Convergence Layer Services

Each convergence layer protocol adapter is expected to provide the following services to the bundle protocol agent:

- sending a bundle to a bundle node that is reachable via the convergence layer protocol;
- delivering to the bundle protocol agent a bundle that was sent by a bundle node via the convergence layer protocol.

The convergence layer service interface specified here is neither exhaustive nor exclusive. That is, supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [BPSEC]) may expect convergence layer adapters that serve BP implementations conforming to those protocols to provide additional services such as reporting on the transmission and/or reception progress of individual bundles (at completion and/or incrementally), retransmitting data that were lost in

transit, discarding bundle-conveying data units that the convergence layer protocol determines are corrupt or inauthentic, or reporting on the integrity and/or authenticity of delivered bundles.

8. Security Considerations

The bundle protocol has taken security into concern from the outset of its design. It was always assumed that security services would be needed in the use of the bundle protocol. As a result, the bundle protocol security architecture and the available security services are specified in an accompanying document, the Bundle Security Protocol specification [BPSEC]; an informative overview of this architecture is provided in [SECO].

The bundle protocol has been designed with the notion that it may be run over networks with scarce resources. For example, the networks might have limited bandwidth, limited connectivity, constrained storage in relay nodes, etc. Therefore, the bundle protocol must ensure that only those entities authorized to send bundles over such constrained environments are actually allowed to do so. All unauthorized entities should be prevented from consuming valuable resources as soon as practicable.

Likewise, because of the potentially high latencies and delays involved in the networks that make use of the bundle protocol, data sources should be concerned with the integrity of the data received at the intended destination(s) and may also be concerned with ensuring confidentiality of the data as it traverses the network. Without integrity, the bundle payload data might be corrupted while in transit without the destination able to detect it. Similarly, the data source can be concerned with ensuring that the data can only be used by those authorized, hence the need for confidentiality.

Internal to the bundle-aware overlay network, the bundle nodes should be concerned with the authenticity of other bundle nodes as well as the preservation of bundle payload data integrity as it is forwarded between bundle nodes.

As a result, bundle security is concerned with the authenticity, integrity, and confidentiality of bundles conveyed among bundle nodes. This is accomplished via the use of two independent security-specific bundle blocks, which may be used together to provide multiple bundle security services or independently of one another, depending on perceived security threats, mandated security requirements, and security policies that must be enforced.

To provide end-to-end bundle authenticity and integrity, the Block Integrity Block (BIB) is used. The BIB allows any security-enabled entity along the delivery path to ensure the integrity of the bundle's payload or any other block other than a Block Confidentiality Block.

To provide payload confidentiality, the use of the Block Confidentiality Block (BCB) is available. The bundle payload, or any other block aside from the primary block and the Bundle Security Protocol blocks, may be encrypted to provide end-to-end payload confidentiality/privacy.

Additionally, convergence-layer protocols that ensure authenticity of communication between adjacent nodes in BP network topology SHOULD be used where available, to minimize the ability of unauthenticated nodes to introduce inauthentic traffic into the network.

Bundle security MUST NOT be invalidated by forwarding nodes even though they themselves might not use the Bundle Security Protocol.

In particular, while blocks MAY be added to bundles transiting intermediate nodes, removal of blocks with the 'Discard block if it can't be processed' flag set in the block processing control flags may cause security to fail.

Inclusion of the Bundle Security Protocol in any Bundle Protocol implementation is RECOMMENDED. Use of the Bundle Security Protocol in Bundle Protocol operations is OPTIONAL.

9. IANA Considerations

The "dtn" and "ipn" URI schemes have been provisionally registered by IANA. See http://www.iana.org/assignments/uri-schemes.html for the latest details.

Registries of URI scheme type numbers, extension block type numbers, and administrative record type numbers will be required.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7049] Borman, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, October 2013.

[URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, STD 66, January 2005.

[URIREG] Thaler, D., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", RFC 7595, BCP 35, June 2015.

10.2. Informative References

[ARCH] V. Cerf et al., "Delay-Tolerant Network Architecture", RFC 4838, April 2007.

[BPSEC] Birrane, E., "Bundle Security Protocol Specification", Work In Progress, October 2015.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[RFC5050] Scott, M. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, November 2007.

[SECO] Farrell, S., Symington, S., Weiss, H., and P. Lovell, "Delay-Tolerant Networking Security Overview", Work Progress, July 2007.

[SIGC] Fall, K., "A Delay-Tolerant Network Architecture for Challenged Internets", SIGCOMM 2003.

[TUT] Warthman, F., "Delay-Tolerant Networks (DTNs): A Tutorial", <http://www.dtnrg.org>.

[UTC] Arias, E. and B. Guinot, "Coordinated universal time UTC: historical background and perspectives" in "Journees systemes de reference spatio-temporels", 2004.

11. Acknowledgments

This work is freely adapted from [RFC5050], which was an effort of the Delay Tolerant Networking Research Group. The following DTNRG participants contributed significant technical material and/or inputs to that document: Dr. Vinton Cerf of Google, Scott Burleigh, Adrian Hooke, and Leigh Torgerson of the Jet Propulsion Laboratory, Michael Demmer of the University of California at Berkeley, Robert Durst, Keith Scott, and Susan Symington of The MITRE Corporation,

Kevin Fall of Carnegie Mellon University, Stephen Farrell of Trinity College Dublin, Peter Lovell of SPARTA, Inc., Manikantan Ramadas of Ohio University, and Howard Weiss of SPARTA, Inc.

This document was prepared using 2-Word-v2.0.template.dot.

12. Significant Changes from RFC 5050

Points on which this draft significantly differs from RFC 5050 include the following:

- Clarify the difference between transmission and forwarding.
- Amplify discussion of custody transfer. Move current custodian to an extension block, of which there can be multiple occurrences (possible support for the MITRE idea of multiple concurrent custodians, from several years ago); define that block in this specification.
- Introduce the concept of "node ID" as functionally distinct from endpoint ID, while having the same syntax.
- Restructure primary block, making it immutable. Add optional CRC.
- Add optional CRCs to non-primary blocks.
- Add block ID number to canonical block format (to support streamlined BSP).
- Add bundle age extension block, defined in this specification.
- Add previous node extension block, defined in this specification.
- Add flow label extension block, *not* defined in this specification.
- Add manifest extension block, *not* defined in this specification.
- Add hop count extension block, defined in this specification.
- Clean up a conflict between fragmentation and custody transfer that Ed Birrane pointed out.

Appendix A. For More Information

Please refer comments to dtn@ietf.org. The Delay Tolerant Networking Research Group (DTNRG) Web site is located at http://www.dtnrg.org.

Appendix B. CDDL expression

For informational purposes, Carsten Bormann has kindly provided an expression of the Bundle Protocol specification in the CBOR Data Definition Language (CDDL). That CDDL expression is presented below, somewhat edited by the authors. Note that wherever the CDDL expression is in disagreement with the textual representation of the BP specification presented in the earlier sections of this document, the textual representation rules.

```
start = bundle

dtn-time = uint

creation-timestamp = [dtn-time, sequence: uint]

eid-generic = [uri-code, SSP: any]

uri-code = uint

eid = eid-choice .within eid-generic

eid-choice /= [dtn-code, SSP: bytes]

dtn-code = 1 ; TBD

eid-choice /= [ipn-code, SSP: [nodenum: uint, servicenum: uint]]

ipn-code = 2 ; TBD

bundle-control-flags = uint .bits bundleflagbits

bundleflagbits = &(

  reserved: 15

  reserved: 14

  reserved: 13

  bundle-deletion-status-reports-are-requested: 12

  bundle-delivery-status-reports-are-requested: 11

  bundle-forwarding-status-reports-are-requested: 10

  custody-transfer-status-reports-are-requested: 9
```

```
    bundle-reception-status-reports-are-requested: 8
    bundle-contains-a-Manifest-block: 7
    status-time-is-requested-in-all-status-reports: 6
    user-application-acknowledgement-is-requested: 5
    destination-is-a-singleton-endpoint: 4
    custody-transfer-is-requested: 3
    bundle-must-not-be-fragmented: 2
    payload-is-an-administrative-record: 1
    bundle-is-a-fragment: 0
  )
  crc = uint
  block-control-flags = uint .bits blockflagbits
  blockflagbits = &(
    reserved: 7
    reserved: 6
    reserved: 5
    reserved: 4
    bundle-must-be-deleted-if-block-cannot-be-processed: 3
    status-report-must-be-transmitted-if-block-cannot-be-processed: 2
    block-must-be-removed-from-bundle-if-it-cannot-be-processed: 1
    block-must-be-replicated-in-every-fragment: 0
  )
  bundle = [primary-block, *extension-block, payload-block]
  primary-block = [
```

```
                version: 7,
                bundle-control-flags,
                crc-type: uint,
                destination: eid,
                source-node: eid,
                report-to: eid,
                creation-timestamp,
                lifetime: uint,
                ? fragment-offset: uint,
                ? total-application-data-length: uint,
                ? crc,
   ]
   canonical-block-generic = [
                         block-type-code: uint,
                         canonical-block-common,
                         content: any
                         ]
   canonical-block-common = (
                        block-number: uint,
                        block-control-flags,
                        crc-type: uint,
                        ? crc,
                        )
```

```
   canonical-block = canonical-block-choice .within canonical-block-
   generic

   canonical-block-choice /= payload-block

   payload-block = [1, canonical-block-common, adu-extent: payload]

   payload = bytes / bytes .cbor admin-record

   canonical-block-choice /= extension-block

   extension-block = extension-block-choice .within canonical-block

   extension-block-choice /= current-custodian-block

   current-custodian-block = [5, canonical-block-common, eid]

   extension-block-choice /= previous-node-block

   previous-node-block = [7, canonical-block-common, eid]

   extension-block-choice /= bundle-age-block

   bundle-age-block = [8, canonical-block-common, bundle-age: uint]

   extension-block-choice /= hop-count-block

   hop-count-block = [9, canonical-block-common,
                   [hop-limit: uint,
                    hop-count: uint]]

   admin-record-generic = [record-type: uint, any]

   admin-record = admin-record-choice .within admin-record-generic

   admin-record-choice /= bundle-status-report

   bundle-status-report = [1, [bundle-status-information,
                               bundle-status-reason: uint,
                               admin-common]]

   admin-common = (
```

```
                source-node: eid,
                creation-timestamp,
                ? fragment-offset: uint,
                ? payload-length: uint)
   bundle-status-information = [
     reporting-node-received-bundle: bundle-status-item,
     reporting-node-attempted-custody-transfer: bundle-status-item,
     reporting-node-forwarded-the-bundle: bundle-status-item,
     reporting-node-delivered-the-bundle: bundle-status-item,
     reporting-node-deleted-the-bundle: bundle-status-item,
   ]
   bundle-status-item = (
                asserted: Boolean,
                ? time-of-assertion: dtn-time)
   admin-record-choice /= custody-signal
   custody-signal = [2, [custody-signal-type-code: uint,
                        custody-signal-information,
                        admin-common]]
   custody-signal-information = custody-reason-code: uint / delegation-
   information
   delegation-information = (
                next-hop-node: eid,
                seconds-until-forwarding: uint)
```

Authors' Addresses

Scott Burleigh
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109-8099
US
Phone: +1 818 393 3353
Email: Scott.Burleigh@jpl.nasa.gov

Kevin Fall
Carnegie Mellon University / Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213
US
Phone: +1 412 268 3304
Email: kfall@cmu.edu

Edward J. Birrane
Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd
Laurel, MD 20723
US
Phone: +1 443 778 7423
Email: Edward.Birrane@jhuapl.edu