Delay-Tolerant Networking Research Group S. Burleigh
Internet Draft JPL, Calif. Inst. Of Technology
Intended status: Informational June 6, 2014
Expires: December 2014

Proposed Revised Bundle Protocol
draft-burleigh-bpv7-00.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on December 6, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This Internet Draft presents a proposed modification to the Bundle Protocol Specification, including notes on the rationale underlying some of the proposed changes.

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 utility of the protocol, to the extent that it has been judged suitable for some network operations and eventual commercialization.

[It has also, as expected, exposed a number of opportunities for making the protocol simpler, more capable, and easier to use. This Internet Draft presents a proposed revision of the Bundle Protocol Specification, including notes on the rationale underlying some of the proposed changes.

[Amended original "Introduction" text from RFC-5050 follows. Notes on rationale are enclosed in square brackets, like these last three paragraphs. Appendix A contains a summary of the more significant differences between this specification and RFC-5050.]

This document describes version 7 of the Delay Tolerant Networking (DTN) "bundle" protocol (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 sits at the application layer of some number of constituent internets, forming a store-and-forward overlay network. Key capabilities of BP include:

- Custody-based retransmission
- Ability to cope with intermittent connectivity
- Ability to take advantage of scheduled, predicted, and opportunistic connectivity (in addition to continuous connectivity)
- Late binding of overlay network endpoint identifiers to underlying internet 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 the "native" internet protocols for

communications within a given internet. Note that "internet" in the preceding is used in a general sense and does not necessarily refer to TCP/IP.

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

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

```
+-----------+                                        +-----------+
|   BP app  |                                        |   BP app  |
+---------v-|   +->>>>>>>>>>v-+     +->>>>>>>>>>v-+   +-^---------+
|   BP v    |   | ^    BP   v |     | ^   BP    v |   | ^   BP    |
+---------v-+   +-^---------v-+     +-^---------v-+   +-^---------+
| Trans1 v  |   + ^  T1/T2  v |     + ^  T2/T3  v |   | ^ Trans3  |
+---------v-+   +-^---------v-+     +-^---------v +   +-^---------+
| Net1    v |   | ^  N1/N2  v |     | ^  N2/N3  v |   | ^ Net3    |
+---------v-+   +-^---------v +     +-^---------v-+   +-^---------+
|         >>>>>>>>^        >>>>>>>>>>^        >>>>>>>>^         |
+-----------+   +-------------+     +-------------+   +-----------+
|                     |                     |                     |
|<--- An internet --->|                     |<--- An internet --->|
|                     |                     |                     |
```

Figure 1: The Bundle Protocol Sits at the Application Layer of the Internet 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.

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 the DTN bundle protocol, 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 usable. Each bundle comprises a sequence of two or more "blocks" of protocol data, which serve various purposes. Multiple instances of the same bundle (the same unit of DTN protocol data) might exist concurrently in different parts of a network -- possibly in different representations and/or 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 of some bundle in the network that is in that node's local memory.

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. The terms "bundle content", "bundle payload", and "payload" are used interchangeably in this document. The "nominal" 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 nominal payload for a bundle forwarded in response to reception of that bundle is the payload of the received bundle.

Fragment - A fragment is a bundle whose payload block contains a fragmentary payload. A fragmentary payload is either the first N bytes or the last N bytes of some other payload -- either a nominal payload or a fragmentary payload -- of length M, such that $0 < N < M$.

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.

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. Each bundle node has three conceptual components, defined below: a "bundle protocol agent", a set of zero or more "convergence layer adapters", and an "application agent".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. The manner in which it does so 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 interprocess or network communication by any number of bundle nodes on one or more computers; it might be implemented in hardware.

Convergence layer adapters - A convergence layer adapter (CLA) sends and receives bundles on behalf of the BPA, utilizing the services of some 'native' internet protocol stack that is supported in one of the internets within which the node is functionally located. As such, every CLA implements its own thin layer of protocol, interposed between BP and the "top" protocol of the underlying native internet protocol; this "CL protocol" may only serve to multiplex and de-multiplex bundles to and from the underlying top protocol, or it may offer additional CL-specific functionality. The manner in which a CLA sends and receives bundles is wholly an implementation matter, exactly as described for the BPA. The definitions of CLAs and CL protocols are beyond the scope of this specification.

Application agent - The application agent (AA) of a node is the node component that utilizes the BP services to effect communication for some purpose. The application agent in turn has two elements, an administrative element and an application-specific element. The application-specific element of an AA constructs, requests transmission of, accepts delivery of, and processes application-specific application data units; the only interface between the BPA and the application-specific element of the AA is the BP service interface. The administrative element of an AA constructs and requests transmission of administrative records (including status reports and custody signals), and it accepts delivery of and processes any custody signals that the node receives. In addition to the BP service interface, there is a (conceptual) private control interface between the BPA and the administrative element of the AA that enables each to direct the other to take action under specific

circumstances. In the case of a node that serves simply as a "router" in the overlay network, 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; in particular, the administrative element of an AA might be built into the library or daemon or hardware that implements the BPA and the application-specific element of an AA might be implemented either in software or in hardware.

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 single text string, called a "bundle endpoint ID" (or, in this document, simply "endpoint ID"; endpoint IDs are described in detail in Section 4.4.4 below). The special case of an endpoint that never contains more than one node is termed a "singleton" endpoint. 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 final destination of every bundle is an endpoint, which may or may not be singleton. The original source of every bundle is a singleton endpoint.

Transmission - A transmission is an attempt by a node's bundle protocol agent 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. 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 (in which case forwarding to the endpoint is functionally similar to "anycast"

operations in the Internet); 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 be determined from the endpoint's ID (again, see Section 4.4 below): 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.

Forwarding - When the bundle protocol agent of a node determines that a bundle must be "forwarded" to a node (either a node that is a member of the bundle's destination endpoint or 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.

Registration - A registration is the state machine characterizing a given node's membership in a given endpoint. Any number of registrations may be concurrently associated with a given endpoint, and any number of registrations may be concurrently associated with a given node. Any single registration must at any time be in one of two states: Active or Passive. A registration always has an associated "delivery failure action", the action that is to be taken when a bundle that is "deliverable" (see below) subject to that registration is received at a time when the registration is in the Passive state. Delivery failure action must be one of the following:

- defer "delivery" (see below) 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" (see below) 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 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 least recently received bundle that is currently deliverable subject to 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.

Delivery - Upon reception, the processing of a bundle that has been sent to 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 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. 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 the value of the bundle's "Acknowledgement by application is requested" flag and any other relevant metadata (an implementation matter), has been presented to the node's application agent in a manner consistent with the state of that registration and, as applicable, the registration's delivery failure action.

Deliverability, Abandonment - 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) delivery of the bundle subject to this registration has not been abandoned. To "abandon" delivery of a bundle subject to a registration is simply to declare it no longer deliverable subject to that registration; normally only registrations' registered delivery failure actions cause deliveries to be abandoned.

Deletion, Discarding - A bundle protocol agent "discards" a bundle by simply ceasing all operations on the bundle and functionally erasing all references to it; the specific procedures by which this is accomplished are an implementation matter. Bundles are discarded silently; i.e., the discarding of a bundle does not result in generation of an administrative record. "Retention constraints" are elements of the bundle state that prevent a bundle from being discarded; a bundle cannot be discarded while it has any retention constraints. A bundle protocol agent "deletes" a bundle in response to some anomalous condition by notifying the bundle's report-to node of the deletion (provided such notification is warranted; see

Section 5.13 for details) and then arbitrarily removing all of the bundle's retention constraints, enabling the bundle to be discarded.

Custody - A node "takes custody" of a bundle when it retains a copy of the bundle for some period, forwarding and possibly re-forwarding the bundle as appropriate, and destroys that copy only when custody of that bundle is formally "released". Custody of a bundle may only be taken if the destination of the bundle is a singleton endpoint. A "custodial node" (or "custodian") of a bundle is a node that has taken custody of the bundle and has not yet released that custody. To "accept custody" upon receiving a bundle is to take custody of the bundle and announce the new custodian to all current custodians of the bundle. Custody may only be released 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. 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.

The custody transfer mechanism in BP is primarily intended as a means of optimizing recovery from forwarding failures. When a route takes a bundle to a node from which it cannot be forwarded to the application, BP must recover from this error: it can "return" the bundle back toward some node that can forward it along some different path in the network, or else it can just send a small "signal" bundle back to such a node, in the event that this node has retained a copy of the bundle ("taken custody") and is therefore able to re-forward the bundle without receiving a copy. Custody transfer sharply reduces the network traffic required for recovery from forwarding failures, at the cost of increased buffer occupancy and state management at the custodial node.

Note that custodial re-forwarding can also be initiated by expiration of a timer prior to reception of a custody acceptance signal. Since the absence of a custody acceptance signal might be caused by failure to receive the bundle, custody transfer can additionally serve as an automated retransmission mechanism. Because custody transfer's only remedy for loss of any part of a bundle is retransmission of the entire bundle (not just the lost portion), custody transfer is a less efficient automated retransmission mechanism than the reliable transport protocols that are typically available at the convergence layer; configuring the

network to use reliable convergence-layer protocols between nodes is generally the best means of ensuring bundle delivery at the destination node(s). But there are some use cases (typically involving unidirectional links) in which custody transfer in BP may be a more cost-effective solution for reliable transmission between two BP agents than operating a retransmission protocol at the convergence layer.

Embargo - Forwarding failures are not just operational anomalies; they may also convey information about the network, i.e., a forwarding failure may indicate a sustained lapse in forwarding capability. Since forwarding a bundle to a dead end wastes time and bandwidth, the bundle protocol agent may choose to manage such a lapse by imposing a temporary "embargo" on subsequent forwarding activity that is similar to the forwarding attempt that has been seen to fail. Because the lapse may indeed be temporary, the agent may elect to test it periodically by forwarding a single "probe" bundle in deliberate violation of the embargo; upon notification that a probe bundle was accepted for forwarding, the agent may lift the embargo.

For bundles that are subject to custody transfer, custody transfer failure provides a basis for imposing an embargo. For non-custodial bundles, forwarding failure is detected upon arrival of a "returned" bundle containing a Forwarding Anomaly block that explains why the bundle could not be forwarded. The resulting embargo may be tested by attachment of a "Probe" Forwarding Anomaly block to a bundle that is forwarded in deliberate violation of the embargo. Notice that the probe bundle was accepted for forwarding is provided in a "Reopen signal" administrative record, enabling the embargo to be lifted.

Embargoes are an optional mechanism. Procedures for determining when and how to impose, enforce, and lift an embargo are an implementation matter.

3.2. Implementation Architectures

The above definitions are intended to enable the bundle protocol's operations to be specified in a manner that minimizes bias toward any particular implementation architecture. To illustrate the range of interoperable implementation models that might conform to this specification, four example architectures are briefly described below.

3.2.1. Bundle protocol application server

A single bundle protocol application server, constituting a single bundle node, runs as a daemon process on each computer. The daemon's functionality includes all functions of the bundle protocol agent, all convergence layer adapters, and both the administrative and application-specific elements of the application agent. The application-specific element of the application agent functions as a server, offering bundle protocol service over a local area network: it responds to remote procedure calls from application processes (on the same computer and/or remote computers) that need to communicate via the bundle protocol. The server supports its clients by creating a new (conceptual) node for each one and registering each such node in a client-specified endpoint. The conceptual nodes managed by the server function as clients' bundle protocol service access points.

3.2.2. Peer application nodes

Any number of bundle protocol application processes, each one constituting a single bundle node, run in ad-hoc fashion on each computer. The functionality of the bundle protocol agent, all convergence layer adapters, and the administrative element of the application agent is provided by a library to which each node process is dynamically linked at run time. The application-specific element of each node's application agent is node-specific application code.

3.2.3. Sensor network nodes

Each node of the sensor network is the self-contained implementation of a single bundle node. All functions of the bundle protocol agent, all convergence layer adapters, and the administrative element of the application agent are implemented in simplified form in Application-Specific Integrated Circuits (ASICs), while the application-specific element of each node's application agent is implemented in a programmable microcontroller. Forwarding is rudimentary: all bundles are forwarded on a hard-coded default route.

3.2.4. Dedicated bundle router

Each computer constitutes a single bundle node that functions solely as a high-performance bundle forwarder. Many standard functions of the bundle protocol agent, the convergence layer adapters, and the administrative element of the application agent are implemented in ASICs, but some functions are implemented in a high-speed processor to enable reprogramming as necessary. The node's application agent

has no application-specific element. Substantial non-volatile storage resources are provided, and arbitrarily complex forwarding algorithms are supported.

3.3. Services Offered by Bundle Protocol Agents

The bundle protocol agent 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

Each bundle shall be a concatenated sequence of at least two block structures. The first block in the sequence must be a primary bundle block, and no bundle may have more than one primary bundle block. Additional bundle protocol blocks of other types may follow the primary block to support extensions to the bundle protocol, such as the Bundle Security Protocol [SBSP]. Exactly one of the blocks in the sequence must be a payload block. The last block in the sequence must have the "last block" flag (in its block processing control flags) set to 1; for every other block in the bundle after the primary block, this flag must be set to zero.

4.1. Self-Delimiting Numeric Values (SDNVs)

The design of the bundle protocol attempts to reconcile minimal consumption of transmission bandwidth with:

. extensibility to address requirements not yet identified, and
. scalability across a wide range of network scales and payload sizes.

A key strategic element in the design is the use of self-delimiting numeric values (SDNVs). The SDNV encoding scheme is closely adapted from the Abstract Syntax Notation One Basic Encoding Rules for sub-identifiers within an object identifier value [ASN1]. An SDNV is a numeric value encoded in N octets, the last of which has its most significant bit (MSB) set to zero; the MSB of every other octet in the SDNV must be set to 1. The value encoded in an SDNV is the

unsigned binary number obtained by concatenating into a single bit string the 7 least significant bits of each octet of the SDNV. The following examples illustrate the encoding scheme for various hexadecimal values.

```
0xABC : 1010 1011 1100

      is encoded as

      {1 00 10101} {0 0111100}

      = 10010101 00111100

0x1234 : 0001 0010 0011 0100

      = 1 0010 0011 0100

      is encoded as

      {1 0 100100} {0 0110100}

      = 10100100 00110100

0x4234 : 0100 0010 0011 0100

      = 100 0010 0011 0100

      is encoded as

      {1 000000 1} {1 0000100} {0 0110100}

      = 10000001 10000100 00110100

0x7F : 0111 1111

      = 111 1111

      is encoded as

      {0 1111111}

      = 01111111
```

Figure 2: SDNV Example

Note: Care must be taken to make sure that the value to be encoded is (in concept) padded with high-order zero bits to make its bitwise length a multiple of 7 before encoding. Also note that, while there is no theoretical limit on the size of an SDNV field, the overhead of the SDNV scheme is 1:7, i.e., one bit of overhead for every 7 bits of actual data to be encoded. Thus, a 7-octet value (a 56-bit quantity with no leading zeroes) would be encoded in an 8-octet SDNV; an 8-octet value (a 64-bit quantity with no leading zeroes) would be encoded in a 10-octet SDNV (one octet containing the high-order bit of the value padded with six leading zero bits, followed by nine octets containing the remaining 63 bits of the value). 148 bits of overhead would be consumed in encoding a 1024-bit RSA encryption key directly in an SDNV. In general, an N-bit quantity with no leading zeroes is encoded in an SDNV occupying ceil(N/7) octets, where ceil is the integer ceiling function.

Implementations of the bundle protocol may handle as an invalid numeric value any SDNV that encodes an integer that is larger than (2^64 - 1).

An SDNV can be used to represent both very large and very small integer values. However, SDNV is clearly not the best way to represent every numeric value. For example, an SDNV is a poor way to represent an integer whose value typically falls in the range 128 to 255. In general, though, we believe that SDNV representation of numeric values in bundle blocks yields the smallest block sizes without sacrificing scalability.

4.2. Bundle Processing Control Flags

The bundle processing control flags field in the primary bundle block of each bundle is an SDNV; the value encoded in this SDNV is a string of bits used to invoke selected bundle processing control features. The significance of the value in each currently defined position of this bit string is described here. Note that in the figure and descriptions, the bit label numbers denote position (from least significant ('0') to most significant) within the decoded bit string, and not within the representation of the bits on the wire. This is why the descriptions in this section and the next do not follow standard RFC conventions with bit 0 on the left; if fields are added in the future, the SDNV will grow to the left, and using this representation allows the references here to remain valid.

```
                 2                   1                   0
       7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
          |Status Report|Class of Svc.|          General          |
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Bundle Processing Control Flags Bit Layout

The bits in positions 0 through 13 of the value of the bundle processing control flags SDNV are flags that characterize the bundle as follows:

0 -- Bundle is a fragment.

1 -- Application data unit is an administrative record.

2 -- Bundle must not be fragmented.

3 -- Custody transfer is requested.

4 -- Destination endpoint is a singleton.

5 -- Acknowledgement by application is requested.

6 -- Bundle is critical.

7 -- Best-efforts forwarding is requested.

8 -- Reliable forwarding is requested.

9 -- Flow label is present.

10 - Payload CRC block is present.

11-13 -- Reserved for future use.

The bits in positions 15 through 20 are used to indicate the bundle's class of service. They constitute a seven-bit priority field indicating the bundle's priority, a value from 0 to 127, with higher values being of higher priority (greater urgency). Within this field, bit 20 is the most significant bit.

The bits in positions 21 through 27 are status report request flags. These flags are used to request status reports as follows:

21 -- Request reporting of bundle reception.

22 -- Request reporting of custody acceptance.

23 -- Request reporting of bundle forwarding.

24 -- Request reporting of bundle delivery.

25 -- Request reporting of bundle deletion.

26 -- Reserved for future use.

27 -- Reserved for future use.

If the bundle processing control flags indicate that the bundle's application data unit is an administrative record, then the custody transfer requested flag must be zero and all status report request flags must be zero. If the custody transfer requested flag is 1, then the source node requests that every receiving node accept custody of the bundle. If the bundle's source endpoint ID is "dtn:none" (see below), 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 must be zero, the "Bundle must not be fragmented" flag must be 1, and all status report request flags must be zero.

4.3. Block Processing Control Flags

The block processing control flags field in every block other than the primary bundle block is an SDNV; the value encoded in this SDNV is a string of bits used to invoke selected block processing control features. The significance of the values in all currently defined positions of this bit string, in order from least significant position in the decoded bit string (labeled '0') to most significant (labeled '5'), is described here.

```
                                0

                  6 5 4 3 2 1 0

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

                 |   Flags     |

                 +-+-+-+-+-+-+-+
```

Figure 4: Block Processing Control Flags Bit Layout

0 - Block must be replicated in every fragment.

1 - Transmit status report if block can't be processed.

2 - Delete bundle if block can't be processed.

3 - Last block.

4 - Discard block if it can't be processed.

5 - Block was forwarded without being processed.

6 - Reserved for future use.

For each bundle whose primary block's bundle processing control flags (see above) indicate that the bundle's application data unit is an administrative record, the "Transmit status report if block can't be processed" flag in the block processing flags field of every other block in the bundle must be zero.

The 'Block must be replicated in every fragment' bit in the block processing flags must be set to zero on all blocks that follow the payload block.

4.4. Identifiers

4.4.1. Node number

Each bundle node has - as a permanent, immutable property of that node - an assigned number which uniquely identifies the node within the network in which it operates. Every node must have exactly one node number. Every BP node number is an integer in the range 0 to (2**63 - 1). No node may be identified by node number zero; node number zero is used to signify "no node".

Note that a node's number is not an address; node numbers have no intrinsic topological significance. A node number is simply a name that is written in numerals, for bit-efficient (binary) representation in transmission and processor-efficient representation in protocol operations. No node number can occupy more than nine (9) octets when represented as an SDNV, and any node number can always be stored in a 64-bit integer for the purposes of a BP implementation.

A registry of node numbers ("CBHE node numbers") is managed by IANA per RFC-7116 [RFC7116].

4.4.2. Service number

A BP service number notionally functions as a de-multiplexing token. When the bundle payload is a protocol data unit of some protocol

that has its own de-multiplexing identifiers, the service number may function in a manner similar to that of the protocol number in an IP packet, characterizing the encapsulated protocol; alternatively, the service number may function in a manner similar to that of the port number in a UDP datagram.

Service numbers enable inbound bundles' application data units to be de-multiplexed to instances of application functionality that are designed to process them, so that effective communication relationships can be developed between bundle producers and consumers.

Every service number is an integer in the range 0 to (2**63 - 1). Node number zero is used to signify "BP administrative records".

A registry of service numbers ("CBHE service numbers") is managed by IANA per RFC-7116 [RFC7116].

4.4.3. Group number

Each BP multicast group is identified by a multicast group number. Multicast group numbers may be used as the destination numbers of bundles for which the destination endpoint is not a singleton endpoint, as indicated by a value of zero in the "Destination endpoint is a singleton" bundle processing flag.

Every group number is an integer in the range 0 to (2**63 - 1). No BP multicast group may be identified by group number zero; group number zero is used to signify that the destination endpoint ID of the bundle is explicitly encoded in the bundle's destination-EID extension block.

4.4.4. Endpoint ID

The sources and destinations of bundles are bundle endpoints, identified by text strings termed "endpoint IDs" (see Section 3.1). Each endpoint ID conveyed in any bundle block takes the form of 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. In addition, a single additional scheme is defined by the present document:

- The "dtn" scheme, which is used at minimum in the representation of the null endpoint ID "dtn:none". The forwarding of a bundle to the null endpoint is never contraindicated, and the minimum reception group for the null endpoint is the empty set. The other syntactic and semantic rules that explain how to construct, parse and interpret the SSP of a URI in the "dtn" scheme are beyond the scope of this specification and, at the time of this writing, are not formally defined.

Note that, although the endpoint IDs conveyed in bundle blocks are expressed as 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]).

While EIDs, being human-readable, are convenient for the purposes of the human users and managers of a delay-tolerant network, they consume appreciable transmission bandwidth and are not convenient for the purposes of computer software:

- As they may be arbitrarily lengthy, they must be stored in memory regions of appropriate size which typically must be dynamically allocated. As such, operations on them always risk failure due to insufficiency of remaining available memory. Moreover, failure to release such a memory region when the EID occupying it is no longer needed (however this is determined) may result in a "memory leak" that permanently reduces remaining available memory for future operations on EIDs.
- Storing, retrieving, and comparing character strings is significantly more time-consuming than storing, retrieving, and comparing numeric values in most computer architectures.

Fortunately, for most purposes it is not necessary for BP implementations to operate directly on EIDs. This is because the text of nearly every EID string is implicit in the information carried in other fields of the bundle. This implicit text can easily be rendered explicit when EIDs must be exposed to users and network managers.

The EID that identifies the source of a bundle is a URI in the "ipn" scheme defined in RFC-6260 [RFC6260]:

```
ipn:NN.SS
```

where NN is the bundle's source node number and SS is the bundle's source service number. Note that when NN is zero, the bundle's source is anonymous.

The EID that identifies the "report-to" endpoint of a bundle (the endpoint that is the destination of all bundle status reports produced in the course of conveying the bundle to its destination) is similarly a URI in the "ipn" scheme:

    ipn:NN.0

where NN is the bundle's report-to node number. The service number in this EID is zero, which is the service number reserved for all bundle administrative record exchange. Note that when NN is zero, the report-to EID is "ipn:0.0" which is functionally equivalent to "dtn:none".

The EID that identifies a "current custodian" endpoint of a bundle (an endpoint that is a destination of Custody Accepted and Custody Refused signals produced in the course of conveying the bundle to its destination) is similarly a URI in the "ipn" scheme:

    ipn:NN.0

where NN is the bundle's current custodian node number as noted in a current custodian extension block. The service number in this EID is again zero, for the same reason. NN should never be zero in this case: if there is no current custodian then the current custodian extension block should be omitted from the bundle.

The EID that identifies the destination of a bundle is more complex to compose, as it depends on the nature of the bundle's destination.

- If the value of the bundle's "Destination endpoint is a singleton" bundle processing flag is 1, then the bundle's "forwarding mode" is "basic unicast". The EID that identifies the destination of the bundle is a URI in the "ipn" scheme:

        ipn:NN.SS

where NN is the bundle's destination number interpreted as a node number and SS is the bundle's destination service number. Note that when NN is zero the destination EID indicates that the bundle is to be delivered at no nodes; such an EID is functionally equivalent to "dtn:none".

. Otherwise, the bundle's destination is not a singleton endpoint. If the bundle's destination number is zero, then the bundle's forwarding mode is undefined and the EID that identifies the destination of the bundle is carried explicitly, as a text string, in the bundle's destination-EID extension block. In this case the bundle's destination service number is irrelevant and must be set to zero.

. Otherwise, the bundle's forwarding mode is "basic multicast". The EID that identifies the destination of the bundle is a URI in the "imc" scheme defined later in this document:

    imc:NN.SS

where NN is the bundle's destination number interpreted as a multicast group number and SS is the bundle's destination service number.

4.5. Formats of Bundle Blocks

This section describes the formats of the primary block and payload block. 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 [SBSP]) may require that BP implementations conforming to those protocols construct and process additional blocks.

The format of these two basic BP blocks is shown in Figure 5 below.

Primary Bundle Block

```
+----------------+----------------+----------------+---------------+

|    Version     |  Block length  |               CRC              |

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

|   Bundle Processing flags (*)   |        [Flow label (*)]        |

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

|       Destination number (*)    | Destination service number (*) |

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

|       Source node number (*)    |    Source service number (*)   |
```

```
+----------------+----------------+----------------+---------------+
|    Report-to node number (*)    |    Creation timestamp time (*) |
+----------------+----------------+----------------+---------------+
|                Creation Timestamp sequence number (*)            |
+---------------------------------+--------------------------------+
|            Lifetime (*)         |      [Fragment offset (*)]     |
+----------------+----------------+--------------------------------+
|              [Total application data unit length (*)]            |
+----------------+----------------+--------------------------------+

Bundle Payload Block

+----------------+----------------+----------------+---------------+
|   Block type   |Block number (*)| Proc. flags (*)| Blk length(*) |
+----------------+----------------+----------------+---------------+
/ Bundle payload (variable) /
+------------------------------------------------------------------+
```

Figure 5: Basic Bundle Block Formats

(*) Notes:

The bundle processing control ("Proc.") flags field in the Primary Bundle Block is an SDNV and is therefore of variable length. A two-octet SDNV is shown here for convenience in representation.

The block length field of the Primary Bundle Block is an SDNV and is therefore of variable length. A four-octet SDNV is shown here for convenience in representation.

The destination number, destination service number, source node number, source service number, report-to node number, creation timestamp time, and lifetime fields in the Primary Bundle Block are SDNVs and are therefore of variable length. Two-octet SDNVs are shown here for convenience in representation.

The Creation Timestamp sequence number field in the Primary Bundle Block is an SDNV and is therefore of variable length. A four-octet SDNV is shown here for convenience in representation.

The flow label field of the Primary Bundle Block is present only if the Flow Label flag in the block's processing flags byte is set to 1. It is an SDNV and is therefore of variable length; a two-octet SDNV is shown here for convenience in representation.

The fragment offset field of the Primary Bundle Block is present only if the Fragment flag in the block's processing flags byte is set to 1. It is an SDNV and is therefore of variable length; a two-octet SDNV is shown here for convenience in representation.

The total application data unit length field of the Primary Bundle Block is present only if the Fragment flag in the block's processing flags byte is set to 1. It is an SDNV and is therefore of variable length; a four-octet SDNV is shown here for convenience in representation.

The block processing control flags ("Proc. flags") field of the Payload Block is an SDNV and is therefore of variable length. A one-octet SDNV is shown here for convenience in representation.

The block length ("Blk length") field of the Payload Block is an SDNV and is therefore of variable length. A one-octet SDNV is shown here for convenience in representation.

4.5.1. Primary Bundle Block

The primary bundle block contains the basic information needed to route bundles to their destinations. The fields of the primary bundle block are:

Version: A 1-byte field indicating the version of the bundle protocol that constructed this block. The present document describes version 0x07 of the bundle protocol.

Block Length: a 1-byte field that contains the aggregate length (in bytes) of all remaining fields of the primary block. Note that, although most fields of the primary bundle block are variable-length SDNVs, the lengths of all of these SDNVs are in practice limited; this ensures that the total length of the primary block cannot exceed 255.

CRC: a 2-byte field that contains a CRC16 value computed over the concatenation of all bytes of the primary block (for this purpose

the CRC field is itself populated with the value zero), the block-type-specific data of the Destination EID extension block (if present), and the block-type-specific data of the Payload CRC extension block (if present), in that order.

[The proposed limit on the maximum length of the primary bundle block simplifies parsing: bundle parsing logic can be certain that all critical information needed to begin bundle ingestion - possibly even to forward the bundle immediately - can be acquired by reading at most the first 255 bytes of the bundle. The fixed locations and lengths of the first three fields also simplify CRC computation somewhat, making BP implementation in FPGAs or ASICs less expensive. Finally, note that all fields of the primary block are immutable; none will ever change at any point on the bundle's end-to-end path. This is intended to simplify canonicalization in the management of bundle security protocol blocks.]

Bundle Processing Control Flags: The Bundle Processing Control Flags field is an SDNV that contains the bundle processing control flags discussed in Section 4.2 above.

Destination number: The Destination number field contains information that (as noted in 4.4.4 above) may form a part of the endpoint ID of the bundle's destination, i.e., the endpoint containing the node(s) at which the bundle is to be delivered.

Destination service number: The Destination service number field contains information that (as noted in 4.4.4 above) may form a part of the endpoint ID of the bundle's destination.

Source node number: The Source node number field contains the node number of the node from which the bundle was initially transmitted.

Source service number: The Source service number field contains the service number of the endpoint ID of the bundle's nominal source.

Report-to node number: The Report-to node number field contains the node number of the node 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 SDNVs that, together with the source node number and (if the bundle is a fragment) the fragment offset and payload length, serve to identify the bundle. The first SDNV of the timestamp is the bundle's creation time, while the second is the bundle's creation timestamp sequence number. Bundle creation time is 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 is 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, 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 either case, a source Bundle Protocol Agent must never create two distinct bundles with the same source endpoint ID and bundle creation timestamp. The combination of source node number and bundle creation timestamp therefore serves to identify a single transmission request, enabling it to be acknowledged by the receiving application (provided the source node number is not zero).

Lifetime: The lifetime field is an SDNV 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 bundle's age exceeds its lifetime, bundle nodes need no longer retain or forward the bundle; the bundle may be deleted from the network.

Fragment Offset: If the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment, then the Fragment Offset field is an SDNV indicating the offset from the start of the original application data unit at which the bytes comprising the payload of this bundle were located. If not, then the Fragment Offset field is omitted from the block.

Total Application Data Unit Length: If the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment, then the Total Application Data Unit Length field is an SDNV indicating the total length of the original application data unit of which this bundle's payload is a part. If not, then the Total Application Data Unit Length field is omitted from the block.

4.5.2. Canonical Bundle Block Format

Every bundle block of every type other than the primary bundle block comprises the following fields, in this order:

- Block type code, expressed as an 8-bit unsigned binary integer. Bundle block type code 1 indicates that the block is a bundle payload block. Block type codes 2 through 11 are defined as noted later in this specification. Block type codes 192 through 255 are not defined in this specification and are

available for private and/or experimental use. All other values of the block type code are reserved for future use.

. Block number, an unsigned integer expressed as an SDNV. 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, an unsigned integer expressed as an SDNV. The individual bits of this integer are used to invoke selected block processing control features.

. Block data length, an unsigned integer expressed as an SDNV. The Block data length field contains the aggregate length of all remaining fields of the block, i.e., the block-type-specific data fields.

. Block-type-specific data fields, whose format and order are type-specific and whose aggregate length in octets is the value of the block data length field. All multi-byte block-type-specific data fields are represented in network byte order.

```
+----------------+----------------+----------------+---------------+

|   Block type   |Block number (*)| Proc. Flags (*)| Blk length(*) |

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

/ Block body data (variable) /

+------------------------------------------------------------------+
```

Figure 6: Block Layout

4.5.3. Bundle Payload Block

The fields of the bundle payload block are:

Block Type: The Block Type field is a 1-byte field that indicates the type of the block. For the bundle payload block, this field contains the value 1.

Block Number: The Block Number field is an SDNV that contains the unique identifying number of the block. The block number of the bundle payload block is always zero.

Block Processing Control Flags: The Block Processing Control Flags field is an SDNV that contains the block processing control flags discussed in Section 4.3 above.

Block Length: The Block Length field is an SDNV that contains the aggregate length of all remaining fields of the block - which is to say, the length of the bundle's payload.

Payload: The Payload field contains the application data carried by this bundle.

That is, bundle payload blocks follow the canonical format of the previous section. The block body data for payload blocks is the application data carried by the bundle.

4.6. 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.

Whenever a bundle is forwarded that contains one or more extension blocks that could not be processed, the "Block was forwarded without being processed" flag must be set to 1 within the block processing flags of each such block. For each block flagged in this way, the flag may optionally be cleared (i.e., set to zero) by another node that subsequently receives the bundle and is able to process that block; the specifications defining the various extension blocks are expected to define the circumstances under which this flag may be cleared, if any.

The extension blocks of the Bundle Security Protocol (block types 2, 3, and 4) are defined separately in the Bundle Security Protocol specification (work in progress).

The following extension blocks are defined in the current document.

4.6.1. 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 an SDNV containing the node number of

the custodian. The bundle MAY contain one or more occurrences of this type of block.

4.6.2. Destination EID

The Destination EID block, block type 6, contains the bundle's explicit destination EID. This EID may be a URI of any registered scheme. Procedures for forwarding a block to the endpoint identified in the destination EID block are beyond the scope of this specification; it is expected that they will be defined in future specifications. If the bundle's "Destination endpoint is a singleton" bundle processing flag is zero and the bundle's destination number is zero, then the bundle MUST contain exactly one (1) occurrence of this type of block; otherwise, the bundle MUST NOT contain any Destination EID block.

4.6.3. Previous Node Number

The Previous Node Number block, block type 7, identifies the node that forwarded this bundle to the local node; its block-type-specific data is an SDNV containing the number of that node. If the local node is the source of the bundle, then the bundle MUST NOT contain any Previous Node Number block. Otherwise the bundle MUST contain one (1) occurrence of this type of block if it contains no Bundle Authentication Block (block type 2, as described in the Bundle Security Protocol specification) and otherwise MUST NOT contain any Previous Node Number 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.

4.6.4. Payload CRC

The Payload CRC block, block type 8, contains a CRC32 value computed over the entire payload of an original, non-fragmented block. The bundle may contain at most one (1) occurrence of this type of block.

4.6.5. Bundle Age

The Bundle Age block, block type 9, 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 SDNV containing the age of the bundle (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) in seconds. 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.

4.6.6. Hop Count

The Hop Count block, block type 10, contains two SDNVs, hop limit and hop count, in that order. It is mainly intended as a safety mechanism, a means of removing bundles from the network that can never be delivered due to an error in network configuration: a bundle may 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. A bundle MAY contain at most one (1) occurrence of this type of block.

4.6.7. Forwarding Anomaly

The Forwarding Anomaly block, block type 11, contains a one-byte reason code indicating something unusual about the forwarding of this block. The bundle may contain at most one (1) occurrence of this type of block. The reason codes that may be conveyed in a Forwarding Anomaly block are given in Figure 7 below.

Value	Meaning
0x00	Bundle is a Probe, testing an embargo.
0x01	Reserved for future use.
0x02	Reserved for future use.
0x03	Reserved for future use.
0x04	Depleted storage.

0x05	Destination endpoint ID unintelligible.
0x06	No known route to destination from here.
0x07	No timely contact with next node on route.
0x08	Block unintelligible.
(other)	Reserved for future use.

Figure 7: Forwarding Anomaly Reason Codes

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. That is, supplementary DTN protocol specifications (including, but not restricted to, the Bundle Security Protocol [SBSP]) may require that additional measures be taken at specified junctures in these procedures. Such additional measures shall not override or supersede the mandated bundle protocol procedures, except that they may in some cases make these procedures moot by requiring, for example, that implementations conforming to the supplementary protocol terminate the processing of a given incoming or outgoing bundle due to a fault condition recognized by that protocol.

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 one case, 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 acceptance status report" is a bundle status report with the "reporting node accepted custody of bundle" 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. o A "bundle deletion status report" is a bundle status report with the "reporting node deleted the bundle" flag set to 1.
. A "Succeeded" custody signal is a custody signal with the "custody transfer succeeded" flag set to 1.
. A "Failed" custody signal is a custody signal with the "custody transfer succeeded" flag set to zero.
. 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 forwarding mode of the bundle must be basic unicast as described in 4.4 above. If, moreover, custody acceptance by the source node is required, then either the bundle protocol

agent must commit to accepting custody of the bundle -- in which case processing proceeds from Step 2 -- or else the request cannot be honored and all remaining steps of this procedure must be skipped. The bundle protocol agent must not commit to accepting custody of a bundle if the conditions under which custody of the bundle may be accepted are not satisfied.

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 number of the bundle must be either the node number assigned to the node of which the BPA is a component or else zero, indicating that the source of the node is anonymous.

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.

If the bundle has a Hop Count block:

- If the current value of the hop count in this block is greater than or equal to the value of the hop limit, then forwarding failure MAY be declared. In that case, the Forwarding Failed procedure defined in Section 5.4.2 MUST be followed; the remaining steps of Section 5 are skipped at this time.
- Otherwise, the value of the hop count in this block MUST be increased by 1.

Step 2: The bundle protocol agent must determine whether or not forwarding is contraindicated for any of the reasons listed in Figure 12. In particular:

. The bundle protocol agent must determine which node(s) to forward the bundle to. If the forwarding mode of the bundle is basic multicast then the bundle is to be forwarded to all neighboring nodes that are members of the multicast group identified by the bundle's destination number, excluding the node from which the bundle was received (if not created locally). Otherwise 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 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:

- o If the "Bundle is critical" flag (in the bundle processing flags) is set to 1, then ALL nodes that have some plausible prospect of forwarding the bundle to its destination node(s) SHOULD be selected for this purpose.
- o Any node on which an "embargo" has been imposed that would apply to the source node and destination endpoint of this bundle normally SHOULD NOT be selected. However, this restriction SHOULD be relaxed in the event that the agent determines that it is time to revisit the possible suitability of this node as a forwarder; when this is the case, a Forwarding Anomaly block with reason code Probe MUST be attached to the bundle.
- o If the agent 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. If both the "Best-efforts forwarding requested" and the "Reliable forwarding is requested" bundle processing flags are set to 1, then all selected CLAs MUST be for bundle streaming CL protocols such as the Bundle Streaming Service Protocol (work in progress). Otherwise, if only the "Reliable forwarding is requested" bundle processing flag is set to 1, then all selected CLAs MUST be for reliable protocols such as TCP/IP. Otherwise, if only the "Best-efforts forwarding requested" bundle processing flag is set to 1, then all selected CLAs MUST be for best-efforts protocols such as UDP/IP. Otherwise, any available CLAs may be selected. 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 appropriate convergence layer adapters 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 12, 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 5.10.2 must be followed.

Step 5: If the bundle has a Forwarding Anomaly block with reason code indicating that the bundle is a Probe:

- The bundle protocol agent MUST generate a Reopen signal citing the bundle's source and destination endpoints, destined for the sender of the bundle (as noted in the Previous Node block or Bundle Authentication Block).
- If an embargo has been imposed at the local node that would apply to the source node and destination endpoint of this bundle, then the "Probe" Forwarding Anomaly block SHOULD be retained in the bundle; otherwise the Probe block SHOULD be removed from the bundle.

Step 6: 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 is to be sent by each convergence layer adapter is an implementation matter. Note that:

- The order in which convergence layer adapters send bundles SHOULD normally conform to the priority indicated in each bundle's bundle processing control flags field: all bundles of priority 255 should be sent before all bundles of priority 254 and so on.
- But if the "Flow label is present" flag in the bundle processing control flags is set to 1 then (a) the SDNV immediately following the bundle processing control flags MUST be interpreted as a flow label value and (b) that flow label value may identify overriding procedures for determining the order in which convergence layer adapters must send bundles. 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 7: 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 for some reason are:

Step 1: The bundle protocol agent must determine whether or not to declare failure in forwarding the bundle for this reason. 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 for some reason 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 "Failed" custody signal 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.)

If the bundle's custody transfer requested flag (in the bundle processing flags field) is set to 0, then the bundle protocol agent SHOULD forward the bundle back to the node that sent it, as identified by the Previous Node block or Bundle Authentication Block. If the bundle protocol agent elects to do this, a Forwarding Anomaly block MUST be inserted into the bundle, containing a reason code corresponding to the reason for which forwarding was determined to be contraindicated.

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 5.13 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; 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 5.13 MUST be followed.

5.6. Bundle Reception

The steps in processing a bundle 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 5.13 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 the block must be discarded, then the bundle protocol agent must set to 1 the "Block was forwarded without being processed" flag in the block processing flags of the block.

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 number, creation timestamp, and (if the bundle is a fragment) fragment offset and payload length as another bundle that (a) has not been discarded and (b) currently has the retention constraint "Custody accepted", custody transfer redundancy must be handled. Otherwise, processing proceeds from Step 5. The bundle protocol agent must handle custody transfer redundancy by generating a "Failed" custody signal 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: If the bundle has a Forwarding Anomaly block with any reason code other than Probe, then the bundle has been returned to sender as impossible to forward, for the indicated reason. The bundle protocol agent MAY impose an "embargo" on the forwarding of bundles to the sending node that are determined (in an implementation-specific manner) to have source node and destination endpoint that are similar to those of the received bundle.

Step 6: 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 "Succeeded" custody signal for the bundle, destined for the bundle's current custodian(s).

5.8. Bundle Fragmentation

It may at times be necessary 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 number 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 bundle that was fragmented. Note that the payloads of fragments resulting from different fragmentation episodes, in different parts of the network, may be overlapping subsets of the original bundle's payload.
. The bundle processing flags in the primary block of each fragment must differ from those of the bundle that is being fragmented, in that they must indicate that the bundle is a fragment, and both fragment offset and total application data unit length must be provided at the end of each fragment's primary bundle block. The CRC computed for the primary block of each fragment will necessarily be different from that of the bundle that is being fragmented.
. The primary blocks of the fragments will differ from that of the fragmented bundle as noted above.
. The payload blocks of fragments will differ from that of the fragmented bundle as noted above.
. If the bundle being fragmented is not a fragment or is the fragment with offset zero, then all extension blocks of the bundle being fragmented MUST be replicated in the fragment whose offset is zero.
. Each extension block whose "Block must be replicated in every fragment" flag, in the block processing flags, 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 had taken custody 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 number 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.
- For 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 take custody of that bundle.
- The BPA must then release custody of all fragments 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 route 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; however, if the bundle protocol agent has committed to accepting custody of the bundle (as described in Step 1 of Section 5.2), then custody must be accepted.

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 acceptance" flag in the bundle's status report request field is set to 1, a custody acceptance status report 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 1 of Section 5.6), then this report should be generated by simply turning on the "Reporting node accepted custody of bundle" flag in that earlier report's status flags byte.

The bundle protocol agent must generate a "Succeeded" custody signal 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 inserting a new Current Custodian extension block whose value is the node number of the local node or by changing the value of an existing Current Custodian extension block to the local node number.

The bundle protocol agent may set a custody transfer countdown timer for this 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 must be destroyed.

5.11. Custody Transfer Success

Upon receipt of a "Succeeded" custody signal 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 that bundle when either (a) that node's custody transfer timer for that bundle (if any) expires or (b) a "Failed" custody signal for that bundle is received at that node.

Upon determination of custody transfer failure, the action taken by the bundle protocol agent is implementation-specific and may depend on the nature of the failure. For example, if custody transfer failure was inferred from expiration of a custody transfer timer or was asserted by a "Failed" custody signal with the "Depleted storage" reason code, the bundle protocol agent might choose to re-forward the bundle, possibly on a different route (Section 5.4). Receipt of a "Failed" custody signal with the "Redundant reception" reason code, on the other hand, 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. In any case, the bundle protocol agent MAY impose an "embargo" (as in Step 5 of 5.6 above) on the forwarding of bundles to the node to which the bundle had been sent.

5.13. 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 node 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.

5.14. Discarding a Bundle

As soon as a bundle has no remaining retention constraints it may be discarded.

5.15. 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 5.13 must be followed.

5.16. Polling

When requested to poll a specified registration that is in the Passive state, the bundle protocol agent must immediately deliver the least recently received bundle that is deliverable subject to the indicated registration, if any.

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. Four types of administrative records have been defined to date: bundle status reports, multicast petitions, reopen signals, and custody signals.

Note that supplementary DTN protocol specifications may require that BP implementations conforming to those protocols construct and process additional administrative records.

Every administrative record consists of a four-bit record type code followed by four bits of administrative record flags, followed by record content in type-specific format. Record type codes are defined as follows:

Value	Meaning
0001	Bundle status report.

0010	Custody signal.
0011	Reopen signal.
0100	Multicast petition.
(other)	Reserved for future use.

Figure 8: Administrative Record Type Codes

Value	Meaning
0001	Record is for a fragment; fragment offset and length fields are present.
(other)	Reserved for future use.

Figure 9: Administrative Record Flags

The contents of the various types of administrative records are described below.

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.

```
+----------------+----------------+----------------+---------------+

| Status Flags   | Reason code    | Fragment offset (*) (if

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

  present)       | Fragment length (*) (if present)                |

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

| Source node number of bundle X (*)                               |

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

| Copy of bundle X's Creation Timestamp time (*)                   |

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

| Copy of bundle X's Creation Timestamp sequence number (*)        |

+----------------+----------------+----------------+---------------+
```

Figure 10: Bundle Status Report Format

(*) Notes:

The Fragment Offset field, if present, is an SDNV and is therefore variable length. A three-octet SDNV is shown here for convenience in representation.

The Fragment Length field, if present, is an SDNV and is therefore variable length. A three-octet SDNV is shown here for convenience in representation.

The Source Node Number and Creation Timestamp fields replicate the Source Node Number and Creation Timestamp fields in the primary block of the subject bundle. As such they are SDNVs (see Section 4.5.1 above) and are therefore variable length. Four-octet SDNVs are shown here for convenience in representation.

The fields in a bundle status report are:

Status Flags: A 1-byte field containing the following flags:

Value	Meaning
00000001	Reporting node received bundle.
00000010	Reporting node accepted custody of bundle.
00000100	Reporting node forwarded the bundle.
00001000	Reporting node delivered the bundle.
00010000	Reporting node deleted the bundle.
00100000	Unused.
01000000	Unused.
10000000	Unused.

Figure 11: Status Flags for Bundle Status Reports

Reason Code: A 1-byte field explaining the value of the flags in the status flags byte. 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 [SBSP]) may define additional reason codes. Status report reason codes are defined as follows:

Value	Meaning
0x00	No additional information.
0x01	Lifetime expired.
0x02	Forwarded over unidirectional link.
0x03	Transmission canceled.
0x04	Depleted storage.
0x05	Destination endpoint ID unintelligible.
0x06	No known route to destination from here.
0x07	No timely contact with next node on route.
0x08	Block unintelligible.
(other)	Reserved for future use.

Figure 12: Status Report Reason Codes

Fragment Offset: If the bundle fragment bit is set in the status flags, then the offset (within the original application data unit) of the payload of the bundle that caused the status report to be generated is included here.

Fragment length: If the bundle fragment bit is set in the status flags, then the length of the payload of the subject bundle is included here.

Source Node Number of Subject Bundle: A copy of the source node number of the bundle that caused the status report to be generated.

Creation Timestamp of Subject Bundle: A copy of the creation timestamp of the bundle that caused the status report to be generated.

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.

Custody signals have the following format.

Custody signal regarding bundle 'X':

```
+----------------+----------------+----------------+---------------+

| Status         | Fragment offset (*) (if present)                |

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

| Fragment length (*) (if present)                                 |

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

| Source node number of bundle X (*)                               |

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

| Copy of bundle X's Creation Timestamp time (*)                   |

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

| Copy of bundle X's Creation Timestamp sequence number (*)        |

+----------------+----------------+----------------+---------------+
```

Figure 13: Custody Signal Format

(*) Notes:

The Fragment Offset field, if present, is an SDNV and is therefore variable length. A three-octet SDNV is shown here for convenience in representation.

The Fragment Length field, if present, is an SDNV and is therefore variable length. A four-octet SDNV is shown here for convenience in representation.

The Source Node Number and Creation Timestamp fields replicate the Source Node Number and Creation Timestamp fields in the primary block of the subject bundle. As such they are SDNVs (see Section 4.5.1 above) and are therefore variable length. Four-octet SDNVs are shown here for convenience in representation.

The fields in a custody signal are:

Status: A 1-byte field containing a 1-bit "custody transfer succeeded" flag followed by a 7-bit reason code explaining the value of that flag. Custody signal reason codes are defined as follows:

Value	Meaning
0x00	No additional information.
0x01	Reserved for future use.
0x02	Reserved for future use.
0x03	Redundant (reception by a node that is a custodial node for this bundle).

0x04	Depleted storage.
0x05	Destination endpoint ID unintelligible.
0x06	No known route destination from here.
0x07	No timely contact with next node on route.
0x08	Block unintelligible.
(other)	Reserved for future use.

Figure 14: Custody Signal Reason Codes

Fragment offset: If the bundle fragment bit is set in the status flags, then the offset (within the original application data unit) of the payload of the bundle that caused the custody signal to be generated is included here.

Fragment length: If the bundle fragment bit is set in the status flags, then the length of the payload of the subject bundle is included here.

Source Node Number of Subject Bundle: A copy of the source node number of the bundle that caused the custody signal to be generated.

Creation Timestamp of Subject Bundle: A copy of the creation timestamp of the bundle to which the signal applies.

6.1.3. Reopen Signals

Reopen signals are administrative records that enable the lifting of forwarding embargoes at upstream nodes. They are transmitted to the nodes that were the senders of bundles containing Forwarding Anomaly blocks with reason code "Probe".

Reopen signals have the following format.

Reopen signal signaling acceptance of Probe bundle 'X':

```
+----------------+----------------+----------------+---------------+
|                  Source node number of bundle X (*)              |
+----------------+----------------+----------------+---------------+
/ Bundle X's Destination EID (variable; NULL-terminated) /
+----------------+----------------+----------------+---------------+
```

Figure 15: Reopen Signal Format

(*) Notes:

The Source Node Number field replicates the Source Node Number field in the primary block of the subject bundle. As such it is an SDNV (see Section 4.5.1 above) and therefore variable length. A four-octet SDNV is shown here for convenience in representation.

The fields in a Reopen signal are:

Source Node Number of Subject Bundle: A copy of the source node number of the probe bundle that caused the Reopen signal to be generated.

Destination EID of Subject Bundle: An explicit representation (a NULL-terminated character string) of the destination endpoint ID of the probe bundle that caused the Reopen signal to be generated.

6.1.4. Multicast Petitions

Multicast petitions are administrative records that govern the propagation of bundles within multicast groups. They are expressions of interest, or lapse of interest, in bundles destined for specified multicast groups, identified by "imc"-scheme endpoint IDs.

IPN multicast forwarding is configured by the propagation of petitions through a single spanning tree structured as an overlay upon the nodes of a single DTN-based network. Each node must have accurate current information identifying all of its "kin" in the tree, i.e., the nodes that are its parent and all of its children. Mechanisms for propagating information about nodes' kinship

relations in a multicast spanning tree are beyond the scope of this specification.

Every immediate relative of a given node in the multicast tree MUST be a "neighbor" of that node in the topology of the underlying BP-based network; that is, each node must be able to exchange bundles directly, by means of some convergence-layer protocol, with each of its immediate relatives. This is because loop-free BP basic multicast petition forwarding procedures requires that the identity of the immediate relative that sends each received bundle be known with certainty; no such information can be provided when a bundle is forwarded by an intermediary non-kin node.

Each multicast petition is transmitted to all (and only) nodes that are the "kin" of the local node (that is, parent (if any) and all children (if any)) within the global basic multicast spanning tree that encompasses all nodes of the network - except that a petition is never sent back to a node from which it was received.

Multicast petitions have the following format.

```
+----------------+----------------+----------------+---------------+

| Interest flag  |          Multicast group number (*)             |

+----------------+----------------+----------------+---------------+
```

Figure 16: Multicast Petition Format

(*) Notes:

The Multicast Group Number field is an SDNV (see Section 4.5.1 above) and therefore variable length. A three-octet SDNV is shown here for convenience in representation.

The fields in a multicast petition are:

Interest Flag: An indication of the nature of the petition. A value of 1 indicates that the node sending the petition is now interested in bundles destined for the indicated multicast group. A value of 0 indicates that the node sending the petition is now no longer interested in bundles destined for the indicated multicast group.

Multicast Group Number: The number that identifies the multicast group in which a new expression of interest or disinterest is being transmitted.

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 have the Fragment flag set and must contain the fragment offset and fragment length fields. The value of the fragment offset field must be the value of the referenced bundle's fragment offset, and the value of the fragment length field must be the length of the referenced bundle's payload.

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 the "custody transfer succeeded" flag set to 1, 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 the "custody transfer succeeded" flag set to 0, the administrative element of the application agent must direct the bundle protocol agent to follow the custody transfer failure procedure in Section 5.12.

6.4. Reception of Reopen Signals

For each received reopen signal, the administrative element of the application agent must direct the bundle protocol agent to lift any existing embargo pertaining to the forwarding node, source node, and destination endpoint identified in the signal.

6.5. Generation and Handling of Multicast Petitions

Node join and leave BP basic multicast groups by registering and unregistering in endpoints formed in the "imc" URI scheme: registering in any "imc" endpoint for multicast group N (regardless of service number) causes the node to be a member of multicast group N, and unregistering from all "imc" endpoints for group N terminates the nodes membership in group N.

When a node joins an IMC multicast group, it MUST send a petition with interest flag value 1, citing the number of that group, to each of its immediate relatives (parent and children) in the multicast tree.

When a node leaves an IMC multicast group, it MUST send a petition with interest flag value 0, citing the number of that group, to each of its immediate relatives in the multicast tree.

When the administrative element of a node's application agent receives a petition in a bundle whose bundle ID timestamp (creation time and sequence number) is greater than the bundle ID timestamp of the most recently accepted petition regarding the same multicast group sent by the same immediate relative in the multicast tree, the petition is considered a "current" petition. Otherwise, the petition is considered not current and MUST be ignored. This prevents the propagation of obsolete petition information when bundles arrive out of transmission order.

When the administrative element of a node's application agent receives a current petition with interest flag value 1 it MUST:

- Note that the node from which the petition was received has an interest in bundles destined for the indicated group.
- Forward the petition to each of its immediate relatives in the multicast tree except the node from which the petition was received, UNLESS either the receiving node is a member of the indicated group or one or more other immediate relatives' interest in bundles destined for the indicated group is currently noted. In the latter case, the petition MAY (but need not) be forwarded.

When the administrative element of a node's application agent receives a current petition with interest flag value 0 it MUST:

- Note that the node from which the petition was received now has no interest in bundles destined for the indicated group.
- Forward the petition to each of its immediate relatives in the multicast tree except the node from which the petition was received, UNLESS either the receiving node is a member of the indicated group or one or more other immediate relatives' continued interest in bundles destined for the indicated group is currently noted. In the latter case, the petition MUST NOT be forwarded.

When a new immediate relative (parent or child) in the multicast tree is added for some node, that node MUST send to the new relative

a petition with interest value 1 for each multicast group in which either (a) the node itself is a member or (b) interest is currently noted by one or more of the node's other immediate relatives.

When any one of a node's immediate relatives in the multicast tree is removed, that node MUST send to every remaining immediate relative a petition with interest flag value 0 for each multicast group in which (a) the node itself is not a member and (b) the removed relative was interested but no other immediate relative is currently interested.

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;
- discarding each bundle-conveying data unit of the convergence layer protocol that the convergence layer protocol determines was corrupted in transit; and
- 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 [SBSP]) may expect convergence layer adapters that serve BP implementations conforming to those protocols to provide additional services.

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 [SBSP]; an informative overview of this architecture is provided in [SECO].

The bundle protocol has been designed with the notion that it will 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.

Likewise, because of the potentially long 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 three 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.

The Bundle Authentication Block (BAB) ensures the authenticity and integrity of bundles on a hop-by-hop basis between bundle nodes. The BAB allows each bundle node to verify a bundle's authenticity before processing or forwarding the bundle. In this way, entities that are not authorized to send bundles will have unauthorized transmissions blocked by security-aware bundle nodes.

Additionally, 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.

Finally, 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.

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 as they transit intermediate nodes, removal of blocks that do not have their 'Discard block if it can't be processed' flag in the block processing control flags set to 1 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.

8.1. Security considerations of BP basic multicast

Reliability and consistency: none of the BP endpoints identified by the URIs of the IMC scheme are guaranteed to be reachable at any time, and the identity of the processing entities operating on those endpoints is never guaranteed by the Bundle Protocol itself. Bundle authentication as defined by the Bundle Security Protocol is required for this purpose.Malicious construction: malicious construction of a conformant IMC-scheme URI is limited to malicious selection of group number. That is, a maliciously constructed IMC-scheme URI could be used to direct a bundle to an endpoint whose member nodes might be damaged by the arrival of that bundle. In that case (and indeed in all bundle processing) the node that receives a bundle should verify its authenticity and validity before operating on it in any way.

Back-end transcoding: the limited expressiveness of URIs of the IMC scheme effectively eliminates the possibility of threat due to errors in back-end transcoding.

Rare IP address formats: not relevant, as IP addresses do not appear anywhere in conformant IMC-scheme URIs.

Sensitive information: because IMC-scheme URIs are used only to represent the identities of Bundle Protocol endpoints, the risk of disclosure of sensitive information due to interception of these

URIs is minimal. Examination of IMC-scheme URIs could be used to support traffic analysis; where traffic analysis is a plausible danger, bundles should be conveyed by secure convergence-layer protocols that don't expose endpoint IDs.

Semantic attacks: the simplicity of IMC-scheme URI syntax minimizes the possibility of misinterpretation of a URI by a human user.

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.

Provisional registration (per [URIREG]) for a URI scheme for CBHE is requested, with the string "imc" as the suggested scheme name, as follows.

URI scheme name: "imc".

Status: provisional.

URI scheme syntax:

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234], including the core ABNF syntax rule for DIGIT defined by that specification.

imc-uri = "imc:" ipn-hier-part

imc-hier-part = group-nbr nbr-delim service-nbr ; a path-rootless

group-nbr = 1*DIGIT

nbr-delim = "."

service-nbr = 1*DIGIT

None of the reserved characters defined in the generic URI syntax are used as delimiters within URIs of the IPN scheme.

URI scheme semantics: URIs of the IPN scheme are used as endpoint identifiers in the Delay-Tolerant Networking (DTN) Bundle Protocol (BP) [RFC5050] as described in 2.1 above.

Encoding considerations: URIs of the IMC scheme are encoded exclusively in US-ASCII characters.

Applications and/or protocols that use this URI scheme name: the Delay-Tolerant Networking (DTN) Bundle Protocol (BP) [RFC5050].

Interoperability considerations: as noted above, URIs of the IPN scheme are encoded exclusively in US-ASCII characters.

10. Conclusions

This document is offered as a "strawman" first draft for a Bundle Protocol specification standard. Comments are welcome.

11. References

11.1. Normative References

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

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

[URIREG] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", RFC 4395, BCP 115, February 2006.

11.2. Informative References

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

[ASN1] "Abstract Syntax Notation One (ASN.1), "ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)," ITU-T Rec. X.690 (2002) | ISO/IEC 8825- 1:2002", 2003.

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

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

[RFC5234] Crocker, D., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC6260] Burleigh, S., "Compressed Bundle Header Encoding (CBHE)", RFC 6260, May 2011.

[RFC7116] Scott, K., and M. Blanchet, "Licklider Transmission Protocol (LTP), Compressed Bundle Header Encoding (CBHE), and Bundle Protocol IANA Registries", RFC 7116, February 2014.

[SBSP] Birrane, E., "Streamlined Bundle Security Protocol Specification", draft-irtf-dtnrg-sbsp-01, May 2014.

[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.

12. Acknowledgments

This work is 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 Intel Research, Stephen Farrell of Trinity College Dublin, Peter Lovell of SPARTA, Inc., Manikantan Ramadas of Ohio University, and Howard Weiss of SPARTA, Inc. New material in the adapted specification was contributed by Ed Birrane of the Johns Hopkins University Applied Physics Laboratory.

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

Appendix A. Summary of Revisions

This specification differs from RFC-5050 in a number of ways. The revisions that seem to the author to be most significant are listed below:

- Amplify the discussion of custody transfer. Move current custodian to an extension block, of which there can be multiple occurrences (providing possible support for the MITRE idea of multiple concurrent custodians, from several years ago); define that block in this spec.
- Add the notion of "embargoes", i.e., what do you do when a route unexpectedly goes bad for a while? This entails adding another extension block (Forwarding Anomaly) and another administrative record (Reopen Signal).
- Incorporate the Compressed Bundle Header Encoding [RFC6260] concepts into the base specification: nodes are explicitly identified by node numbers, and operations that pertain to nodes are described in terms of node numbers rather than endpoint IDs.
- Add basic ("imc") multicast to the BP spec. This entails adding another administrative record, Multicast Petition.
- Add Destination EID extension block for destinations that can't be expressed in "ipn"-scheme and "imc"-scheme URIs. Define it in this spec.
- Incorporate the "Extended Class of Service" features into the base specification.
- Restructure the primary block, making it immutable. Add CRC. Remove the dictionary.
- Add optional Payload CRC extension block, defined in this spec.
- Add block ID number to canonical block format (to support streamlined Bundle Security Protocol).
- Add bundle age extension block, defined in this spec.
- Define two other extension blocks in this spec: previous node number, hop count.
- Clean up a conflict between fragmentation and custody transfer that Ed Birrane pointed out.
- Remove "DTN time" values from administrative records. Nanosecond precision will not be meaningful among nodes whose clocks are not closely synchronized, and absent that feature the administrative record's bundle creation time suffices to indicate the time of occurrence of the reported event.
- Note that CL protocols are supposed to discard data that they find to have been corrupted.

Appendix B. 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.

Copyright (c) 2014 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info).

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