Bundle Protocol
draft-ietf-dtn-bpbis-01.txt

Status of this Memo

Copyright Notice

Abstract

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

Table of Contents

1. Introduction

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

implementation and deployment experience has demonstrated the
general utility of the protocol for challenged network operations.

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

This document describes version 7 of BP.

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

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

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

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

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

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

```
+-----------+                                    +-----------+
|   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-+    +-^---------+
|      >>>>>>>>^        >>>>>>>>>>^         >>>>>>>>^         |
+-----------+    +-------------+    +-------------+    +-----------+
|               |                  |                   |
|<---- A network ---->|                 |<---- A network ---->|
|               |                  |                   |
```

                Figure 1: The Bundle Protocol in the Protocol Stack Model

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

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

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

2. Conventions used in this document

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

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

[3](#). Service Description

[3.1](#). Definitions

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

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

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

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

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

Bundle node - A bundle node (or, in the context of this document,
simply a "node") is any entity that can send and/or receive bundles.
Each bundle node has three conceptual components, defined below: 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.

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

Application agent - The application agent (AA) of a node is the node
component that utilizes the BP services to effect communication for

some user purpose. The application agent in turn has two elements, an administrative element and an application-specific element.

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

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

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

Bundle endpoint - A bundle endpoint (or simply "endpoint") is a set of zero or more bundle nodes that all identify themselves for BP purposes by some common identifier, called a "bundle endpoint ID" (or, in this document, simply "endpoint ID"; endpoint IDs are described in detail in Section 4.3.1 below).

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

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

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

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

(c) the bundle has not yet been "abandoned" (as defined below) subject to this registration.

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

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

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

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

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

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

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

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

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

Custodian - A custodian of a bundle is a node that has determined that it will retain a copy of that bundle for an indefinite period

of time, forwarding and possibly re-forwarding the bundle as
appropriate, until it detects one of the conditions under which it
may cease being a custodian of that bundle (discussed later).

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

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

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

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

3.2. Discussion of BP concepts

Multiple instances of the same bundle (the same unit of DTN protocol
data) might exist concurrently in different parts of a network --
possibly 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 (copy), in that node's local memory,
of some bundle that is in the network.

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

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

The manner in which the functions of the BPA are performed is wholly
an implementation matter. For example, BPA functionality might be
coded into each node individually; it might be implemented as a
shared library that is used in common by any number of bundle nodes
on a single computer; it might be implemented as a daemon whose

services are invoked via inter-process or network communication by
any number of bundle nodes on one or more computers; it might be
implemented in hardware.

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

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

   . An outgoing bundle that has been encoded in some documented
     representation constitutes a byte array. This byte array may,
     like any other, be presented to the bundle protocol agent as an
     application data unit that is to be transmitted to some
     endpoint.
   . The original encoded bundle thus forms the payload of an
     encapsulating bundle that is forwarded using some other
     convergence-layer protocol(s).
   . When the encapsulating bundle is received, its payload is
     delivered to the peer application agent administrative element,
     which then instructs the bundle protocol agent to dispatch that
     original encoded bundle in the usual way.

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

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

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

the BPA, the manner in which the AA performs its functions is wholly
an implementation matter.

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

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

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

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

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

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

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

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

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

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

  . abandon delivery of the bundle subject to this registration.

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

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

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

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

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

The custody transfer mechanism in BP is primarily intended as a
means of recovering from forwarding failures. When a bundle arrives
at a node from which it must be forwarded, but forwarding is
impossible, BP must recover from this error. BP can "return" the
bundle back toward some node for forwarding along some other path in
the network, or else it can instead 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 nodes.

Note that custodial re-forwarding can also be initiated by
expiration of a timer prior to reception of a custody acceptance or
refusal signal. Since the absence of a custody acceptance signal
might be caused by failure to receive the bundle, rather than only a
disinclination to take custody, custody transfer can additionally
serve as an automated retransmission mechanism.

However, 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 BPAs
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 invoking retransmission protocols at the
convergence layer.

[3.3](). Implementation Architectures

The definitions of BP concepts 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.

[3.3.1](). 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 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.

[3.3.2](). Peer application nodes

Any number of bundle protocol application processes, each one
constituting a single bundle node, run 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.

[3.3.3](). 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
hardware, 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.

3.3.4. 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 specialized hardware, 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.

3.4. 3.3. Services Offered by Bundle Protocol Agents

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

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

4. Bundle Format

NOTE that only the abstract structures of blocks are defined here. The specific bitstream that is emitted when a CLA sends a bundle will be dictated by the applicable bundle representation specification to which the sending and receiving nodes conform, an implementation matter. It is important to note that not all BP implementations are required to implement all bundle representation specifications. The BP implementations used to instantiate nodes in a given network must be chosen with care in order for every node to be able to exchange bundles with every other node. Bundle representation specifications are beyond the scope of this document.

Each bundle SHALL be a concatenated sequence of at least two blocks. The first block in the sequence MUST be a primary bundle block, and the bundle MUST have exactly 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 [BSPBPSEC]. Exactly one of the blocks in the sequence MUST be a payload block, and the payload block MUST be the last block in the sequence.

4.1. Bundle Processing Control Flags

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

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

   . The bundle is a fragment. (Boolean)

   . The bundle's payload is an administrative record. (Boolean)

   . The bundle must not be fragmented. (Boolean)

   . Custody transfer is requested for this bundle. (Boolean)

   . The bundle's destination endpoint is a singleton. (Boolean)

   . Acknowledgment by the user application is requested. (Boolean)

   . The bundle is "critical" as discussed later. (Boolean)

   . Best-efforts forwarding of the bundle is requested. (Boolean)

   . Reliable forwarding of the bundle is requested. (Boolean)

   . Status time is requested in all status reports. (Boolean)

   . Type of CRC that is present in the bundle's primary block. (An
      unsigned integer CRC type code; 0 indicates that the block
      contains no CRC, other valid values TBD)

   . The bundle's priority, a numeric value from 0 to 127, with
      higher values being of higher priority (greater urgency).

   . Flags requesting types of status reports (all Boolean):

        o Request reporting of bundle reception.

        o Request reporting of custody acceptance.

        o Request reporting of bundle forwarding.

        o Request reporting of bundle delivery.

        o Request reporting of bundle deletion.

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

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

   If the bundle's source node is omitted (i.e., the source endpoint ID
   is the null endpoint, which has no members as discussed 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 value must be zero, the
   "Bundle must not be fragmented" flag value must be 1, and all status
   report request flag values must be zero.

4.2. Block Processing Control Flags

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

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

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

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

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

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

     . This block was forwarded without being processed. (Boolean)

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

4.3. Identifiers

4.3.1. Endpoint ID

   The destinations of bundles are bundle endpoints, identified by text
   strings termed "endpoint IDs" (see Section 3.1). Each endpoint ID
   (EID) 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.

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

   Note also that the representation of an EID in the bitstream that is
   emitted when a CLA sends a bundle will be dictated by the applicable
   bundle representation specification to which the sending and
   receiving nodes conform, an implementation matter.

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

4.3.2. Node ID

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

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

      . Every node must MUST be a member of at least one singleton
        endpoint.
      . The EID of any singleton endpoint of which a node is a member
        may MAY be used to identify that node. A "node ID" is an EID
        that is used in this way.

. A node's membership in a given singleton endpoint must MUST be sustained at least until the nominal operation of the Bundle Protocol no longer depends on the identification of that node using that endpoint's ID.

4.4. Contents of Bundle Blocks

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

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

4.4.1. Primary Bundle Block

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

Version: An unsigned integer value indicating the version of the bundle protocol that constructed this block. The present document describes version 7 of the bundle protocol.

Bundle Processing Control Flags: The Bundle Processing Control Flags are discussed in Section 4.1 above.

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

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

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

Creation Timestamp: The creation timestamp is a pair of unsigned integers that, together with the source node ID and (if the bundle is a fragment) the fragment offset and payload length, serve to identify the bundle. The first of these integers is the bundle's creation time, while the second is the bundle's creation timestamp

sequence number. Bundle creation time 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 (that is, nodes that are not at all moments able to determine the current UTC time to within 30 seconds), bundle creation time MUST be set to zero and the counter used as the source of the bundle sequence count MUST NEVER be reset to zero. In any case, a source Bundle Protocol Agent must never create two distinct bundles with the same source node ID and bundle creation timestamp. The combination of source node ID and bundle creation timestamp serves to identify a single transmission request, enabling it to be acknowledged by the receiving application (provided the source node ID is not the null endpoint ID).

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

Inventory: The Primary block MAY contain an accounting of all types of blocks that were in the bundle at the time it was transmitted from the source node. This feature is optional; its structure and the manner by which its presence is signaled to the receiving node are matters of representation that are beyond the scope of this document. If present, the bundle inventory SHALL comprise a sequence of block type code numbers, each an unsigned integer, one for each type of block that was present in the bundle at the time it was transmitted, and no others. The order of block types appearing in the inventory is undefined.

The CRC field of the Primary Bundle Block is present only if the CRC type field in the block's processing flags field is non-zero.

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

Total Application Data Unit Length: If and only if the Bundle Processing Control Flags of this Primary block indicate that the bundle is a fragment, then the Total Application Data Unit Length field SHALL be an unsigned integer 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 SHALL be omitted from the block.

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

4.4.2. 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, an unsigned integer. Bundle block type code 1 indicates that the block is a bundle payload block. Block type codes 2 through 9 are explicitly reserved as noted later in this specification. Block type codes 192 through 255 are not reserved and are available for private and/or experimental use. All other block type code values are reserved for future use.

. Block number, an unsigned integer. The block number uniquely identifies the block within the bundle, enabling blocks (notably bundle security protocol blocks) to explicitly reference other blocks in the same bundle. Block numbers need not be in continuous sequence, and blocks need not appear in block number sequence in the bundle. The block number of the payload block is always zero.

. Block processing control flags as discussed above.

. Block data length, an unsigned integer. 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 nature and order are type-specific and whose aggregate length in octets is the value of the block data length field. For the Payload Block in particular (block type 1), there SHALL be exactly one block-type-specific data field, the "payload", i.e., the application data carried by this bundle.

4.5. Extension Blocks

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

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
and, 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.

4.5.1. Current Custodian

The Current Custodian block, block type 5, identifies a node that is
known to have accepted custody of the bundle. The block-type-
specific data of this block is the node ID of a custodian. The
bundle MAY contain one or more occurrences of this type of block.

4.5.2. Flow Label

The Flow Label block, block type 6, indicates the flow label that is
intended to govern transmission of the bundle by convergence-layer
adapters. The syntax and semantics of BP flow labels are beyond the
scope of this document.

4.5.3. Previous Node ID

The Previous Node ID block, block type 7, identifies the node that
forwarded this bundle to the local node; its block-type-specific

data is the node ID of that node. If the local node is the source
of the bundle, then the bundle MUST NOT contain any Previous Node ID
block. Otherwise the bundle MUST contain one (1) occurrence of this
type of block. If present, the Previous Node ID block MUST be the
FIRST block following the primary block, as the processing of other
extension blocks may depend on its value.

4.5.4. Bundle Age

The Bundle Age block, block type 8, contains the number of seconds
that have elapsed between the time the bundle was created and time
at which it was most recently forwarded. It is intended for use by
nodes lacking access to an accurate clock, to aid in determining the
time at which a bundle's lifetime expires. The block-type-specific
data of this block is an unsigned integer containing the age of the
bundle (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.

4.5.5. Hop Count

The Hop Count block, block type 9, contains two unsigned integers,
hop limit and hop count. It is mainly intended as a safety
mechanism, a means of identifying bundles for removal from the
network that can never be delivered due to a persistent forwarding
error: a bundle 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.

5. Bundle Processing

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

5.1. Generation of Administrative Records

All transmission of bundles is in response to bundle transmission
requests presented by nodes' application agents. When required to

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

Under some circumstances, the requesting of status reports could
result in an unacceptable increase in the bundle traffic in the
network. For this reason, the generation of status reports is
mandatory only in 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.
   . 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 destination MUST be a singleton endpoint. If,
moreover, custody acceptance by the source node is required but the
conditions under which custody of the bundle may be accepted are not
satisfied, then the request cannot be honored and all remaining
steps of this procedure MUST be skipped.

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

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

5.3. Bundle Dispatching

The steps in dispatching a bundle are:

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

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

5.4. Bundle Forwarding

The steps in forwarding a bundle are:

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

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

  . The bundle protocol agent MUST determine which node(s) to
     forward the bundle to. The bundle protocol agent MAY choose
     either to forward the bundle directly to its destination
     node(s) (if possible) or to forward the bundle to some other
     node(s) for further forwarding. The manner in which this
     decision is made may depend on the scheme name in the
     destination endpoint ID and/or on other state but in any case
     is beyond the scope of this document. If the BPA elects to

forward the bundle to some other node(s) for further forwarding
but finds it impossible to select any node(s) to forward the
bundle to, then forwarding is contraindicated.:
  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 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 proposed Bundle Streaming Service
  Protocol. 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 any appropriate convergence layer adapter(s) to use in
  forwarding this bundle, then forwarding is contraindicated.
. Provided the bundle protocol agent succeeded in selecting the
  node(s) to forward the bundle to and additionally succeeded in
  selecting the appropriate convergence layer adapter(s), the
  bundle protocol agent MUST determine the applicable bundle
  representation by which the bundle must be encoded when sent to
  each such node so that the bundle will be intelligible when
  received by that node. The manner in which applicable bundle
  representations are selected is beyond the scope of this
  document. If the agent finds that there are no applicable
  bundle representations for any of the nodes to which the bundle
  is to be sent, 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: For each node selected for forwarding, the bundle protocol agent MUST encode the bundle in the selected applicable representation(s) and then invoke the services of the selected convergence layer adapter(s) in order to effect the sending of the bundle to that node. Determining the time at which the bundle protocol agent invokes convergence layer adapter services is a BPA implementation matter. Determining the time at which the bundle is to be sent by each convergence layer adapter subsequently responds to this service invocation by sending the bundle is an convergence-layer adapter 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 127 sent from any single source should be sent before
      all bundles of priority 126 sent from the same source and so
      on.
   . But iIf the bundle contains a flow label extension block then
      that flow label value MAY identify overriding procedures for
      determining the order in which convergence layer adapters must
      send bundles, e.g., considering bundle source when determining
      the order in which bundles are sent. The definition of such
      procedures is beyond the scope of this specification.
   . If the bundle has a bundle age block, then at the last possible
      moment before the CLA initiates conveyance of the bundle node
      via the CL protocol the bundle age value MUST be increased by
      the difference between the current time and the time at which
      the bundle was received (or, if the local node is the source of
      the bundle, created).

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

   . If the "request reporting of bundle forwarding" flag in the
      bundle's status report request field is set to 1, then a bundle
      forwarding status report SHOULD be generated, destined for the
      bundle's report-to endpoint ID. If the bundle has the retention
      constraint "custody accepted" and all of the nodes to which the
      bundle was forwarded are known to be unable to send bundles
      back to this node, then the reason code on this bundle
      forwarding status report MUST be "forwarded over unidirectional

       link"; otherwise, the reason code MUST be "no additional
       information".
     . The bundle's "Forward pending" retention constraint MUST be
       removed.

5.4.1. Forwarding Contraindicated

   The steps in responding to contraindication of forwarding are:

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

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

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

5.4.2. Forwarding Failed

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

   Step 1: If the bundle's custody transfer requested flag (in the
   bundle processing flags field) is set to 1, custody transfer failure
   must be handled. The bundle protocol agent MUST handle the custody
   transfer failure by generating a "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
   MAY forward the bundle back to the node that sent it, as identified
   by the Previous Node ID block.

   Step 2: If the bundle's destination endpoint is an endpoint of which
   the node is a member, then the bundle's "Forward pending" retention
   constraint MUST be removed. Otherwise, the bundle MUST be deleted:
   the bundle deletion procedure defined in Section 5.13 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 (as discussed in section 4.5.1 above);
otherwise bundle age MUST be obtained from the Bundle Age extension
block. Bundle expiration MAY occur at any point in the processing
of a bundle. When a bundle expires, the bundle protocol agent MUST
delete the bundle for the reason "lifetime expired": the bundle
deletion procedure defined in Section 5.13 MUST be followed.

5.6. Bundle Reception

The steps in processing a bundle that has been received from another
node and decoded from its serialized representation 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 ID, 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: 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 advantageous for bundle protocol agents to reduce
the sizes of bundles in order to forward them. This might be the
case, for example, if a node to which a bundle is to be forwarded is
accessible only via intermittent contacts and no upcoming contact is
long enough to enable the forwarding of the entire bundle.

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

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

Fragmentation SHALL be constrained as follows:

. The concatenation of the payloads of all fragments produced by
   fragmentation MUST always be identical to the payload of the
   fragmented bundle (that is, the bundle that is being
   fragmented). Note that the payloads of fragments resulting from
   different fragmentation episodes, in different parts of the

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

5.9. Application Data Unit Reassembly

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

. This byte array -- the reassembled application data unit --
  MUST replace the payload of this fragment.
. The BPA MUST take custody of each fragmentary bundle whose
  payload is a subset of the reassembled application data unit,
  for which custody transfer is requested but the BPA has not yet
  taken custody.
. The BPA MUST then release custody of every fragment whose
  payload is a subset of the reassembled application data unit,
  for which it has taken custody.
. The "Reassembly pending" retention constraint MUST be removed
  from every other fragment whose payload is a subset of the
  reassembled application data unit.

Note: reassembly of application data units from fragments occurs at
the nodes that are members of destination endpoints as necessary; an

application data unit MAY also be reassembled at some other node on
the path to the destination.

5.10. Custody Transfer

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

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

5.10.1. Custody Acceptance

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

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

If the "request reporting of custody 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)
but has not yet been transmitted, then this report SHOULD be
generated by simply turning on the "Reporting node accepted custody
of bundle" flag in that earlier report.

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 ID of the local node or by
changing the value of an existing Current Custodian extension block
to the local node ID.

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 MAY be followed. The manner in which the countdown interval for
such a timer is determined is an implementation matter.

The bundle SHOULD be retained in persistent storage if possible.

5.10.2. Custody Release

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

5.11. Custody Transfer Success

   Upon receipt of a "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.

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.

6. Administrative Record Processing

6.1. Administrative Records

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

Every administrative record consists of:

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

Valid administrative record type codes are defined as follows:

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

| Value   |                 Meaning                   |

+=========+===========================================+

|    1    | Bundle status report.                     |

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

```
|    2   | Custody signal.                          |

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

| (other) | Reserved for future use.                 |

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

Figure 2: Administrative Record Type Codes

The contents of the two types of administrative records defined in
the present document 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.

Every bundle status report comprises the following fields, in this
order:

- . Status flags. The following conditions are asserted by the
     bundle status report status flags (all Boolean):
        o Reporting node received bundle.
        o Reporting node accepted custody of bundle.
        o Reporting node forwarded the bundle.
        o Reporting node delivered the bundle.
        o Reporting node deleted the bundle.
- . Reason code, an unsigned integer explaining the values of the
     status flags. Status report reason codes are as defined below,
     but the list of status report reason codes provided here is
     neither exhaustive nor exclusive; supplementary DTN protocol
     specifications (including, but not restricted to, the Bundle
     Security Protocol [BPSECBSP]) may define additional reason
     codes.
- . Status times, a pair ofone unsigned integers for each condition
     asserted by any status flag, that indicatinge the time (as
     reported by the local system clock, an implementation matter)
     at which the indicated condition became true for this bundle
     whose status is being reported entered that status. Theseis
     fields are is included in the status report if and only if the
     "Report status time" flag was set to 1 in the subject bundle's
```

         bundle processing flags.  Status time is expressed as:in
         seconds since the start of the year 2000, on the Coordinated
         Universal Time (UTC) scale [UTC].
            o Nanoseconds within the indicated second.
      . Source node, the node ID of the source of the bundle whose
         status is being reported.
      . Creation timestamp, the creation timestamp of the bundle whose
         status is being reported.
      . Fragment offset, the fragment offset of the bundle whose status
         is being reported (omitted if omitted from the subject bundle's
         primary block).
      . Fragment length, the length of the payload of the bundle whose
         status is being reported (omitted if fragment offset is omitted
         from the subject bundle's primary block).

      Valid status report reason codes are defined as follows:

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

      | Value   |                 Meaning                    |

      +=========+============================================+

      |    0    | No additional information.                 |

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

      |    1    | Lifetime expired.                          |

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

      |    2    | Forwarded over unidirectional link.        |

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

      |    3    | Transmission canceled.                     |

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

      |    4    | Depleted storage.                          |

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

      |    5    | Destination endpoint ID unintelligible.    |

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

| 6 | No known route to destination from here. |

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

| 7 | No timely contact with next node on route. |

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

| 8 | Block unintelligible. |

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

| (other) | Reserved for future use. |

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

Figure 3: Status Report Reason Codes

### 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.

Every custody signal comprises the following fields, in this order:

- . "Custody transfer succeeded" flag (Boolean).
- . Reason code, an unsigned integer explaining the value of the
  "Custody transfer succeeded" flag. Custody signal reason codes
  are as defined below.
- . Source node, the node ID of the source of the bundle for which
  custodial activity is being reported.
- . Creation timestamp, the creation timestamp of the bundle for
  which custodial activity is being reported.
- . Fragment offset, the fragment offset of the bundle for which
  custodial activity is being reported (omitted if omitted from
  the subject bundle's primary block).
- . Fragment length, the length of the payload of the bundle for
  which custodial activity is being reported (omitted if fragment
  offset is omitted from the subject bundle's primary block).

Valid custody signal reason codes are defined as follows:

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

| Value   |                  Meaning                    |

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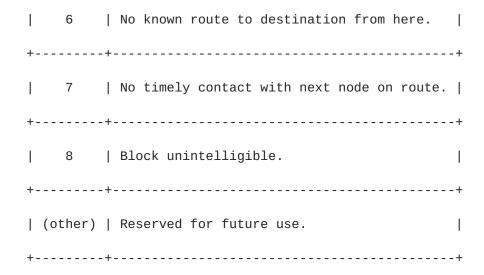+=========+=========================================+
|    0    | No additional information.              |
+---------+-----------------------------------------+
|    1    | Reserved for future use.                |
+---------+-----------------------------------------+
|    2    | Reserved for future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

Figure 4: Custody Signal Reason Codes

6.2. Generation of Administrative Records

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

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

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

6.3. Reception of Custody Signals

   For each received custody signal that has 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.

7. Services Required of the Convergence Layer

7.1. The Convergence Layer

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

7.2. Summary of Convergence Layer Services

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

      . sending a bundle to a bundle node that is reachable via the
         convergence layer protocol;

      . delivering to the bundle protocol agent a bundle that was sent
        by a bundle node via the convergence layer protocol.

   The convergence layer service interface specified here is neither
   exhaustive nor exclusive. That is, supplementary DTN protocol
   specifications (including, but not restricted to, the Bundle
   Security Protocol [BSPBPSEC]) may expect convergence layer adapters
   that serve BP implementations conforming to those protocols to
   provide additional services such as retransmitting data that were
   lost in transit, discarding bundle-conveying data units that the
   convergence layer protocol determines are corrupt or inauthentic, or
   reporting on the integrity and/or authenticity of delivered bundles.

8. Security Considerations

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

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

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

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

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

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

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

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

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

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

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

9. IANA Considerations

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

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

10. References

10.1. Normative References

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

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

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

10.2. Informative References

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

[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.

[BSPBPSEC] SymingtonBirrane, ES., "Bundle Security Protocol
Specification", Work In Progress, October 2015.

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

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

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

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

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

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

11. Acknowledgments

This work is freely adapted from [RFC5050], which was an effort of the Delay Tolerant Networking Research Group. The following DTNRG participants contributed significant technical material and/or inputs to that document: Dr. Vinton Cerf of Google, Scott Burleigh, Adrian Hooke, and Leigh Torgerson of the Jet Propulsion Laboratory, Michael Demmer of the University of California at Berkeley, Robert Durst, Keith Scott, and Susan Symington of The MITRE Corporation, Kevin Fall of Intel ResearchCarnegie Mellon University, Stephen Farrell of Trinity College Dublin, Peter Lovell of SPARTA, Inc., Manikantan Ramadas of Ohio University, and Howard Weiss of SPARTA, Inc.

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

12. Significant Changes from RFC 5050

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

. Clarify the difference between transmission and forwarding.
. Amplify discussion of custody transfer. Move current custodian to an extension block, of which there can be multiple occurrences (possible support for the MITRE idea of multiple concurrent custodians, from several years ago); define that block in this specification.
. Introduce the concept of "node ID" as functionally distinct from endpoint ID, while having the same syntax.
. Add ECOS features to primary block.
. Restrict the scope of bundle prioritization to all bundles from the same source.
. Restructure primary block, making it immutable. Add optional CRC and inventory.
. Add optional CRCs to non-primary blocks.
. Add block ID number to canonical block format (to support streamlined BSP).
. Add bundle age extension block, defined in this specification.
. Add previous node ID extension block, defined in this specification.
. Add flow label block, *not* defined in this specification.
. Add hop count extension block, defined in this specification.
. Clean up a conflict between fragmentation and custody transfer that Ed Birrane pointed out.
. Clarify that the class of service field indicates priority and increase its maximum value to 127.

          . Remove representation specifications from the document, making
             the protocol specification representation-neutral.

13. Open Issues

13.1. Application Agent

   Need to add a diagram explaining how the various components of the
   BPA interact.

13.2. Alignment with ICN

   Is it necessary to modify the bundle transmission procedure to
   enable BP to be used for information-centric networking, i.e.,
   delivering data to a node who requests that data after it has
   already been transmitted? Specifically, would a DTN ICN cache point
   "transmit" data to a client (i.e., source a new bundle) or would it
   merely "forward" a previously transmitted bundle of which it has
   retained a copy?

13.3. Implementation Architectures

   Should the BP spec be divided into two documents? One to talk about
   conops and context and one that focuses specifically on the
   protocol?

13.4. Extended class of service features

   Should these features (critical bundle, best-efforts forwarding
   requested, reliable forwarding requested) be omitted from the
   primary block? If they are omitted, should these application-
   selected CoS markings be supported in some other way? If the
   "critical" CoS feature is retained, should it have a different name?

   Note: a node selection (route computation) procedure might consider
   the availability of CLAs that match the bundle's CoS when selecting
   a node to forward to, and that is entirely the business of the route
   computation procedure. (Not all route computation procedures will
   do so.)

13.5. 13.2. Primary block CRC type

   What are the best CRC options to support here? CRC-16-ARINC, CRC-
   16-CCITT, CRC-16-CDMA2000, CRC-16-DECT, etc.?

[13.6](). Inventory

Is a list of all types of blocks in the bundle as forwarded by the source node a good implementation of the requested "inventory" feature? If not, what would be better?

[13.7](). Clearing flag

Should a node that is able to process a given extension block be permitted to clear block's "Block was forwarded without being processed" flag?

[13.8](). Time of forwarding

Should the BPA control the time at which a bundle is to be forwarded to another node, or should that determination be left to the selected convergence-layer protocol adapter(s)?

[13.9](). Block multiplicity

Would it be good to restrict BP extensions to one extension block per extension per bundle? That is, should we require that all information needed to implement a given BP extension for a given bundle be contained in a single extension block?

This would entail encapsulating any necessary multiplicity for a given extension (for example, multiple Metadata records) within a single block.

Among the advantages: no need for block numbers (block type would always be sufficient to identify the block), therefore no need for a block number generation mechanism; shorter and simpler inventory; simpler extension implementation (all information is in one block, no need to search through extension blocks for additional relevant information).

Among the disadvantages: very different from [RFC 5050](); would in some cases require that security blocks operate on data structures that are internal to extension blocks rather than always operate on entire extension blocks.

Appendix A.   For More Information

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

Authors' Addresses

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

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

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