Delay-Tolerant Networking Working Group   S. Burleigh
Internet Draft   JPL, Calif. Inst. Of Technology
Intended status: Standards Track   K. Fall
Expires: December 2017   Nefeli Networks, Inc.
                                                            E. Birrane
                                          APL, Johns Hopkins University
                                                         June 22, 2017

                            Bundle Protocol
                        draft-ietf-dtn-bpbis-07.txt

Status of this Memo

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

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

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

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

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

   This Internet-Draft will expire on December 24, 2017.

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) in 2007, the Delay-Tolerant Networking
   Bundle Protocol has been implemented in multiple programming
   languages and deployed to a wide variety of computing platforms.
   This implementation and deployment experience has 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:

. Ability to use physical motility for the movement of data
. Ability to move the responsibility for error control from one
    node to another
. 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].

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-+  +-^---------+
| T1      v |   + ^  T1/T2  v |      + ^  T2/T3  v |  | ^ T3      |
+---------v-+   +-^---------v-+      +-^---------v +  +-^---------+
| N1      v |   | ^  N1/N2  v |      | ^  N2/N3  v |  | ^ N3      |
+---------v-+   +-^---------v +      +-^---------v-+  +-^---------+
|        >>>>>>>>^        >>>>>>>>>>^        >>>>>>>>^        |
+-----------+  +------------+      +------------+  +-----------+
|              |                   |              |
|<---- A network ---->|                           |<---- A network ---->|
|              |                   |              |
```
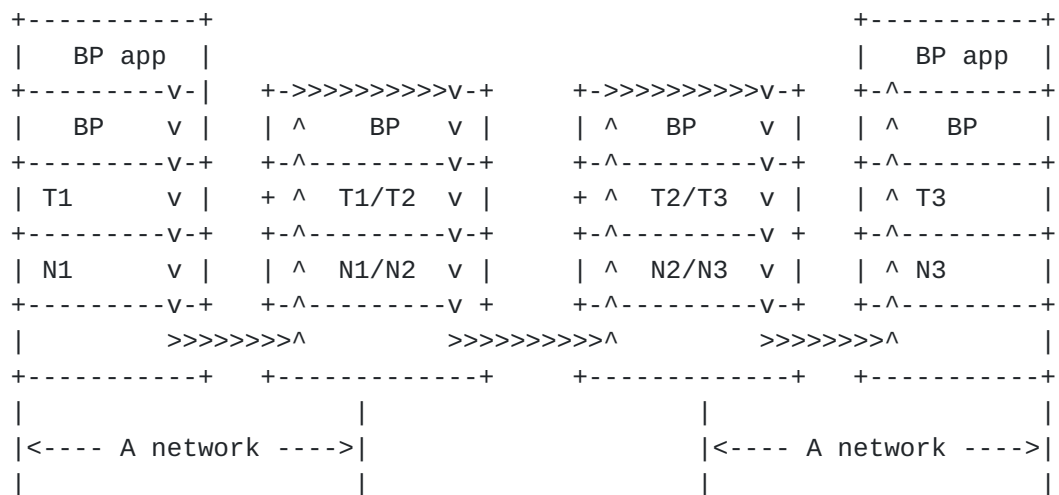
            Figure 1: The Bundle Protocol in the Protocol Stack Model

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.

Note that implementations of the specification presented in this document will not be interoperable with implementations of RFC 5050.

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. Note that every partial payload
is a payload and therefore can be further subdivided into partial
payloads.

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

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

```
+-------------------------------------------------------------+
|Node                                                         |
|                                                             |
| +---------------------------------------------------------+ |
| |Application Agent                                        | |
| |                                                         | |
| | +-------------------------+ +---------------------+     | |
| | |Administrative element   | |Application-specific |     | |
| | | |                        | |element              |     | |
| | | |                        | |                     |     | |
| | +-------------------------+ +---------------------+     | |
| |              ^                          ^              | |
| |          Admin|records        Application|data         | |
| |               |                          |             | |
| +---------------v--------------------------v-----------+ |
|                              ^                           |
|                              | ADUs                      |
|                              |                           |
| +----------------------------v------------------------+ |
| |Bundle Protocol Agent                                | |
| |                                                     | |
| |                                                     | |
| +-----------------------------------------------------+ |
|         ^                   ^                     ^      |
|         | Bundles           | Bundles       Bundles |    |
|         |                   |                     |      |
| +------v-----+     +-----v------+          +-----v-----+ |
| |CLA 1       |     |CLA 2       |          |CLA n      | |
| | |          |     |            |      . . .|           | |
| | |          |     |            |          |           | |
| +-+-----------+-----+------------+----------+-----------+-+
|         ^                   ^                     ^
```

```
   CL1|PDUs              CL2|PDUs                  CLn|PDUs
      |                     |                         |
  +------v-----+       +-----v------+          +-----v-----+
   Network 1            Network 2                Network n
```
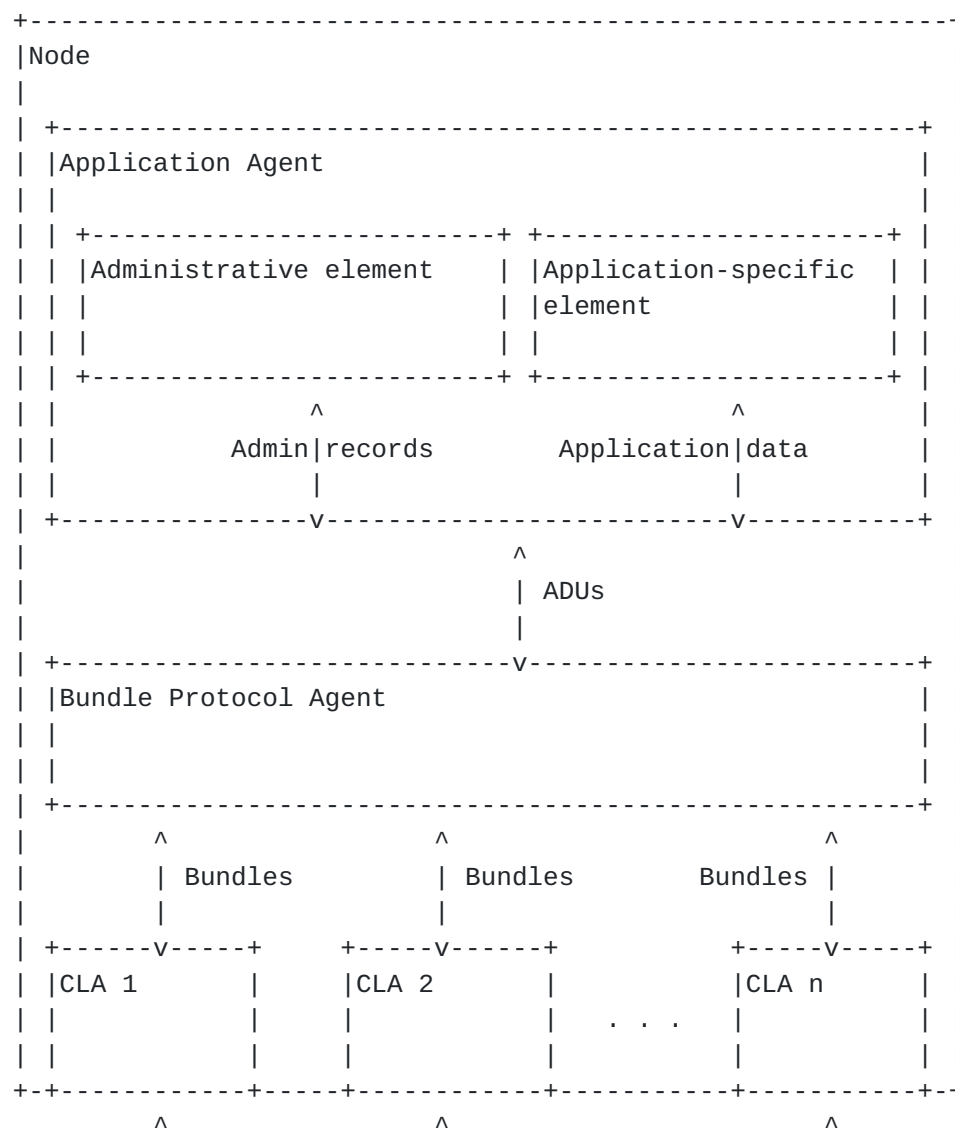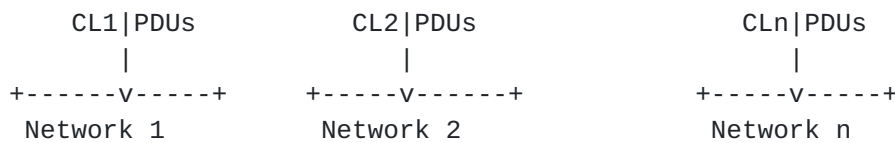
Figure 2: Components of a BP Node

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

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

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

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

Administrative element - The administrative element of an AA is the
node component that constructs and requests transmission of
administrative records (defined below), including status reports,
and accepts delivery of and processes any administrative records
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
only administrative record defined in this specification is the
status report, discussed later.

Bundle endpoint - A bundle endpoint (or simply "endpoint") is a set
of zero or more bundle nodes that all identify themselves for BP
purposes by some common identifier, called a "bundle endpoint ID"
(or, in this document, simply "endpoint ID"; endpoint IDs are
described in detail in Section 4.4.1 below).

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

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

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

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

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

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

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

Transmission - A transmission is an attempt by a node's BPA to cause copies of a bundle to be delivered to one or more of the nodes that are members 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 one or more CLAs 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.

3.2. Discussion of BP concepts

Multiple instances of the same bundle (the same unit of DTN protocol
data) might exist concurrently in different parts of a network --
possibly differing in some blocks -- in the memory local to one or
more bundle nodes and/or in transit between nodes. In the context of
the operation of a bundle node, a bundle is an instance (copy), in
that node's local memory, of some bundle that is in the network.

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

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

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

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

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

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

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

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

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

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

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

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 one or more CLAs 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.

The bundle protocol does not natively ensure delivery of a bundle to
its destination. Data loss along the path to the destination node
can be minimized by utilizing reliable convergence-layer protocols
between neighbors on all segments of the end-to-end path, but for
end-to-end bundle delivery assurance it will be necessary to develop
extensions to the bundle protocol and/or application-layer
mechanisms.

The bundle protocol is designed for extensibility. Bundle protocol
extensions, documented elsewhere, may extend this specification by:

   . defining additional blocks;
   . defining additional administrative records;
   . defining additional bundle processing flags;
   . defining additional block processing flags;
   . defining additional types of bundle status reports;
   . defining additional bundle status report reason codes;
   . defining additional mandates and constraints on processing
      that conformant bundle protocol agents must perform at
      specified points in the inbound and outbound bundle processing
      cycles.

3.3. Services Offered by Bundle Protocol Agents

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

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

4. **Bundle Format**

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

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

   An implementation of the Bundle Protocol MAY discard any sequence of
   bytes that does not conform to the Bundle Protocol specification.

   An implementation of the Bundle Protocol MAY accept a sequence of
   bytes that does not conform to the Bundle Protocol specification
   (e.g., one that represents data elements in fixed-length arrays
   rather than indefinite-length arrays) and transform it into
   conformant BP structure before processing it. Procedures for
   accomplishing such a transformation are beyond the scope of this
   specification.

4.1. **BP Fundamental Data Structures**

4.1.1. **CRC Type**

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

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

CRC type SHALL be represented as a CBOR unsigned integer.

[4.1.2](). CRC

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

When not omitted, the CRC SHALL be represented as sequence of two bytes (if CRC type is 1) or as a sequence of four bytes (if CRC type is 2).

[4.1.3](). Bundle Processing Control Flags

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

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

   . The bundle is a fragment. (Boolean)

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

   . The bundle must not be fragmented. (Boolean)

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

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

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

   . The bundle contains a "manifest" extension block. (Boolean)

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

        o Request reporting of bundle reception.

        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 all status report request flag values must be zero.

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

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

   . Bit 0 (the high-order bit, 0x8000): reserved.
   . Bit 1 (0x4000): reserved.
   . Bit 2 (0x2000): reserved.
   . Bit 3(0x1000): bundle deletion status reports are requested.
   . Bit 4(0x0800): bundle delivery status reports are requested.
   . Bit 5(0x0400): bundle forwarding status reports are requested.
   . Bit 6(0x0200): reserved.
   . Bit 7(0x0100): bundle reception status reports are requested.
   . Bit 8(0x0080): bundle contains a Manifest block.
   . Bit 9(0x0040): status time is requested in all status reports.
   . Bit 10(0x0020): user application acknowledgement is requested.
   . Bit 11(0x0010): destination is a singleton endpoint.
   . Bit 12(0x0008): reserved.
   . Bit 13(0x0004): bundle must not be fragmented.
   . Bit 14(0x0002): payload is an administrative record.
   . Bit 15 (the low-order bit, 0x0001: bundle is a fragment.

4.1.4. Block Processing Control Flags

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

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

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

   . Transmission of a status report is requested if this block
      can't be processed. (Boolean)

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

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

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

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

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

4.1.5. Identifiers

4.1.5.1. Endpoint ID

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

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

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

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

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

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

The first item of the array SHALL be the code number identifying the endpoint's URI scheme [URI], as defined in the registry of URI scheme code numbers for Bundle Protocol maintained by IANA as described in Section 10. [URIREG]. Each URI scheme code number SHALL be represented as a CBOR unsigned integer.

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

   . If the EID's URI scheme is "dtn" then the SSP SHALL be
      represented as a CBOR text string unless the EID's SSP is
      "none", in which case the SSP SHALL be represented as a CBOR
      unsigned integer with the value zero.
   . If the EID's URI scheme is "ipn" then the SSP SHALL be
      represented as a CBOR array comprising a 2-tuple. The first
      item of this array SHALL be the EID's node number represented
      as a CBOR unsigned integer. The second item of this array
      SHALL be the EID's service number represented as a CBOR
      unsigned integer.
   . Definitions of the CBOR representations of the SSPs of EIDs
      encoded in other URI schemes are included in the specifications
      defining those schemes.

4.1.5.2. Node ID

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

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

   . Every node MUST be a member of at least one singleton endpoint.
   . The EID of any singleton endpoint of which a node is a member
      MAY be used to identify that node. A "node ID" is an EID that
      is used in this way.
   . A node's membership in a given singleton endpoint MUST be
      sustained at least until the nominal operation of the Bundle

Protocol no longer depends on the identification of that node
using that endpoint's ID.

4.1.6. DTN Time

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

4.1.7. Creation Timestamp

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

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

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

4.1.8. Block-type-specific Data

Block-type-specific data in each block (other than the primary
block) SHALL be the applicable CBOR representation of the content of
the block. Details of this representation are included in the
specification defining the block type.

4.2. Bundle Representation

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

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

4.2.1. Bundle

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

4.2.2. **Primary Bundle Block**

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

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

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

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

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

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

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

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

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

Creation Timestamp: The creation timestamp is a pair of unsigned
integers that, together with the source node ID and (if the bundle
is a fragment) the fragment offset 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 shall be the time - expressed
in seconds since the start of the year 2000, on the Coordinated
Universal Time (UTC) scale [UTC] - at which the transmission request
was received that resulted in the creation of the bundle. Sequence
count shall be the latest value (as of the time at which that

transmission request was received) of a monotonically increasing
positive integer counter managed by the source node's bundle
protocol agent that may be reset to zero whenever the current time
advances by one second. For nodes that lack accurate clocks, it is
recommended that bundle creation time be set to zero and that the
counter used as the source of the bundle sequence count never be
reset to zero. Note that, in general, the creation of two distinct
bundles with the same source node ID and bundle creation timestamp
may result in unexpected network behavior and/or suboptimal
performance. 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 microseconds past the creation time. (For
high-rate deployments with very brief disruptions, fine-grained
expression of bundle lifetime may be useful.) When a bundle's age
exceeds its lifetime, bundle nodes need no longer retain or forward
the bundle; the bundle SHOULD be deleted from the network. Bundle
lifetime SHALL be represented as a CBOR unsigned integer item.

Fragment offset: If and only if the Bundle Processing Control Flags
of this Primary block indicate that the bundle is a fragment,
fragment offset SHALL be present in the primary block. Fragment
offset SHALL be represented as a CBOR unsigned integer indicating
the offset from the start of the original application data unit at
which the bytes comprising the payload of this bundle were located.

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

4.2.3. Canonical Bundle Block Format

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

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

. Block type code, an unsigned integer. Bundle block type code 1
  indicates that the block is a bundle payload block. Block type
  codes 2 through 9 are explicitly reserved as noted later in
  this specification. Block type codes 192 through 255 are not
  reserved and are available for private and/or experimental use.
  All other block type code values are reserved for future use.
. Block number, an unsigned integer. The block number uniquely
  identifies the block within the bundle, enabling blocks
  (notably bundle security protocol blocks) to explicitly
  reference other blocks in the same bundle. Block numbers need
  not be in continuous sequence, and blocks need not appear in
  block number sequence in the bundle. The block number of the
  payload block is always zero.
. Block processing control flags as discussed in Section 4.1.4
  above.
. CRC type as discussed in Section 4.1.1 above.
. If and only if the value of the CRC type field of this block is
  non-zero, a CRC. If present, the length and nature of the CRC
  SHALL be as indicated by the CRC type and the CRC SHALL be
  computed over the concatenation of all bytes of the block
  (including CBOR "break" characters) including the CRC field
  itself, which for this purpose SHALL be temporarily populated
  with the value zero.
. Block data length, an unsigned integer. The block data length
  field SHALL contain the aggregate length of all remaining
  fields of the block, i.e., the block-type-specific data fields.
  Block data length SHALL be represented as a CBOR unsigned
  integer item.
. Block-type-specific data fields, whose nature and order are
  type-specific and whose aggregate length in octets is the value
  of the block data length field. For the Payload Block in
  particular (block type 1), there SHALL be exactly one block-
  type-specific data field, termed the "payload", which SHALL be
  an application data unit, or some contiguous extent thereof,
  represented as a CBOR byte string.

4.3. Extension Blocks

"Extension blocks" are all blocks other than the primary and payload
blocks. Because not all extension blocks are defined in the Bundle
Protocol specification (the present document), not all nodes
conforming to this specification will necessarily instantiate Bundle
Protocol implementations that include procedures for processing
(that is, recognizing, parsing, acting on, and/or producing) all
extension blocks. It is therefore possible for a node to receive a
bundle that includes extension blocks that the node cannot process.

The values of the block processing control flags indicate the action to be taken by the bundle protocol agent when this is the case.

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

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

. Block Integrity Block (block type 2) and Block Confidentiality Block (block type 3) are defined in the Bundle Security Protocol specification (work in progress).
. Manifest Block (block type 4) is defined in the Manifest Extension Block specification (work in progress). The manifest block identifies the blocks that were present in the bundle at the time it was created. The bundle MUST contain one (1) occurrence of this type of block if the value of the "manifest" flag in the bundle processing control flags is 1; otherwise the bundle MUST NOT contain any Manifest block.
. The Flow Label Block (block type 6) is defined in the Flow Label Extension Block specification (work in progress). The flow label block is intended to govern transmission of the bundle by convergence-layer adapters.

The following extension blocks are defined in the current document.

[4.3.1](#). Previous Node

The Previous Node block, block type 7, identifies the node that forwarded this bundle to the local node (i.e., to the node at which the bundle currently resides); its block-type-specific data is the node ID of that forwarder node which SHALL take the form of a node ID represented as described in [Section 4.1.5.2](#). above. If the local node is the source of the bundle, then the bundle MUST NOT contain any previous node block. Otherwise the bundle SHOULD contain one (1) occurrence of this type of block.

[4.3.2](#). Bundle Age

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

integer item. (The age of the bundle is the sum of all known
intervals of the bundle's residence at forwarding nodes, up to the
time at which the bundle was most recently forwarded, plus the
summation of signal propagation time over all episodes of
transmission between forwarding nodes. Determination of these
values is an implementation matter.) If the bundle's creation time
is zero, then the bundle MUST contain exactly one (1) occurrence of
this type of block; otherwise, the bundle MAY contain at most one
(1) occurrence of this type of block. A bundle MUST NOT contain
multiple occurrences of the bundle age block, as this could result
in processing anomalies.

4.3.3. Hop Count

The Hop Count block, block type 9, contains two unsigned integers,
hop limit and hop count. A "hop" is here defined as an occasion on
which a bundle was forwarded from one node to another node. The hop
limit value SHOULD NOT be changed at any time after creation of the
Hop Count block; the hop count value SHOULD initially be zero and
SHOULD be increased by 1 on each hop.

The hop count block is mainly intended as a safety mechanism, a
means of identifying bundles for removal from the network that can
never be delivered due to a persistent forwarding error. When a
bundle's hop count exceeds its hop limit, the bundle SHOULD be
deleted for the reason "hop limit exceeded", following the bundle
deletion procedure defined in Section 5.14. Procedures for
determining the appropriate hop limit for a block are beyond the
scope of this specification. The block-type-specific data in a hop
count block SHALL be represented as a CBOR array comprising a 2-
tuple. The first item of this array SHALL be the bundle's hop
limit, represented as a CBOR unsigned integer. The second item of
this array SHALL be the bundle's hop count, represented as a CBOR
unsigned integer. A bundle MAY contain at most one (1) occurrence of
this type of block.

5. Bundle Processing

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

5.1. Generation of Administrative Records

   All transmission of bundles is in response to bundle transmission
   requests presented by nodes' application agents. When required to
   "generate" an administrative record (such as a bundle status
   report), 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 MUST be
   disabled by default and enabled only when the risk of excessive
   network traffic is deemed acceptable.

   When the generation of status reports is enabled, 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 "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.

5.2. Bundle Transmission

   The steps in processing a bundle transmission request are:

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

5.3. Bundle Dispatching

The steps in dispatching a bundle are:

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

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

5.4. Bundle Forwarding

The steps in forwarding a bundle are:

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

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

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

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

Step 4: For each node selected for forwarding, the bundle protocol
agent MUST invoke the services of the selected convergence layer
adapter(s) in order to effect the sending of the bundle to that
node. Determining the time at which the bundle protocol agent
invokes convergence layer adapter services is a BPA implementation
matter. Determining the time at which each convergence layer
adapter subsequently responds to this service invocation by sending
the bundle is a convergence-layer adapter implementation matter.
Note that:

   . If the bundle contains a flow label extension block (to be
     defined in a future document) then that flow label value MAY
     identify procedures for determining the order in which
     convergence layer adapters must send bundles, e.g., considering
     bundle source when determining the order in which bundles are
     sent. The definition of such procedures is beyond the scope of
     this specification.
   . If the bundle has a bundle age block, as defined in 4.3.2.
     above, 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 5: 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, and status
     reporting is enabled, then a bundle forwarding status report
     SHOULD be generated, destined for the bundle's report-to
     endpoint ID. The reason code on this bundle forwarding status
     report MUST be "no additional information".
   . If any applicable bundle protocol extensions mandate generation
     of status reports upon conclusion of convergence-layer data
     sending procedures, all such status reports SHOULD be generated
     with extension-mandated reason codes.
   . 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, 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: The bundle protocol agent MAY forward the bundle back to the node that sent it, as identified by the Previous Node block, if present.

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

5.5. Bundle Expiration

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

[5.6](#). Bundle Reception

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

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

Step 2: If the "request reporting of bundle reception" flag in the bundle's status report request field is set to 1, and status reporting is enabled, 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, and status reporting is enabled, then a bundle reception status report with reason code "Block unintelligible" SHOULD be generated, destined for the bundle's report-to endpoint ID.
  . If the block processing flags in that block indicate that the bundle must be deleted in this event, then the bundle protocol agent MUST delete the bundle for the reason "Block unintelligible"; the bundle deletion procedure defined in [Section 5.14](#) MUST be followed and all remaining steps of the bundle reception procedure MUST be skipped.
  . If the block processing flags in that block do NOT indicate that the bundle must be deleted in this event but do indicate that the block must be discarded, then the bundle protocol agent MUST remove this block from the bundle.
  . If the block processing flags in that block indicate neither that the bundle must be deleted nor that that the block must be discarded, then processing continues with the next extension block that the bundle protocol agent cannot process, if any; otherwise, processing proceeds from step 4.

Step 4: 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:

. An additional implementation-specific delivery deferral
  procedure MAY optionally be associated with the registration.
. If the registration is in the Active state, then the bundle
  MUST 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.
. If the registration is in the Passive state, or if delivery of
  the bundle fails for some implementation-specific reason, then
  the registration's delivery failure action MUST be taken.
  Delivery failure action MUST be one of the following:

    o 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, and also
      perform any additional delivery deferral procedure
      associated with the registration; or

    o abandon delivery of the bundle subject to this registration
      (as defined in 3.1.).

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 and bundle status reporting is enabled,
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.

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 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. NOTE, however, that some combinations of bundle
fragmentation, replication, and routing might result in unexpected
traffic patterns.

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.

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 "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. Bundle Deletion

   The steps in deleting a bundle are:

   Step 1: If the "request reporting of bundle deletion" flag in the
   bundle's status report request field is set to 1, and if status
   reporting is enabled, 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.

5.11. Discarding a Bundle

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

5.12. Canceling a Transmission

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

6. Administrative Record Processing

6.1. Administrative Records

   Administrative records are standard application data units that are
   used in providing some of the features of the Bundle Protocol. One
   type of administrative record has been defined to date: bundle
   status reports. 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. |

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

   | (other) | Reserved for future use. |

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

                 Figure 3: Administrative Record Type Codes

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

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

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

[6.1.1](#). **Bundle Status Reports**

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

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

The first item of the bundle status report array SHALL be bundle
status information represented as a CBOR array of at least 4
elements. The first four items of the bundle status information
array shall provide information on the following four status
assertions, in this order:

   . Reporting node received bundle.
   . Reporting node forwarded the bundle.
   . Reporting node delivered the bundle.
   . Reporting node deleted the bundle.

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

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

```
+---------+---------------------------------------------+
| Value   |                 Meaning                     |
+=========+=============================================+
|    0    | No additional information.                  |
+---------+---------------------------------------------+
|    1    | Lifetime expired.                           |
+---------+---------------------------------------------+
|    2    | Forwarded over unidirectional link.         |
+---------+---------------------------------------------+
|    3    | Transmission canceled.                      |
+---------+---------------------------------------------+
|    4    | Depleted storage.                           |
+---------+---------------------------------------------+
|    5    | Destination endpoint ID unintelligible.     |
+---------+---------------------------------------------+
|    6    | No known route to destination from here.    |
+---------+---------------------------------------------+
|    7    | No timely contact with next node on route.  |
+---------+---------------------------------------------+
|    8    | Block unintelligible.                       |
+---------+---------------------------------------------+
|    9    | Hop limit exceeded.                         |
+---------+---------------------------------------------+
| (other) | Reserved for future use.                    |
```

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

Figure 4: Status Report Reason Codes

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

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

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

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

6.2. 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 administrative record references a bundle and 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.

7. Services Required of the Convergence Layer

7.1. The Convergence Layer

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

defined minimal set of services to the bundle protocol agent. This
convergence layer service specification enumerates those services.

7.2. Summary of Convergence Layer Services

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

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

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

8. Implementation Status

[NOTE to the RFC Editor: please remove this section before
publication, as well as the reference to RFC 7942.]

This section records the status of known implementations of the
protocol defined by this specification at the time of posting of
this Internet-Draft, and is based on a proposal described in RFC
7942. The description of implementations in this section is
intended to assist the IETF in its decision processes in progressing
drafts to RFCs. Please note that the listing of any individual
implementation here does not imply endorsement by the IETF.
Furthermore, no effort has been spent to verify the information
presented here that was supplied by IETF contributors. This is not
intended as, and must not be construed to be, a catalog of available
implementations or their features. Readers are advised to note that
other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups
to assign due consideration to documents that have the benefit of
running code, which may serve as evidence of valuable
experimentation and feedback that have made the implemented

protocols more mature. It is up to the individual working groups to
use this information as they see fit".

At the time of this writing, the only known implementation of the
current document is microPCN (https://upcn.eu/). According to the
developers:

  The Micro Planetary Communication Network (uPCN) is a free
  software project intended to offer an implementation of Delay-
  tolerant Networking protocols for POSIX operating systems (well,
  and for Linux) plus for the ARM Cortex STM32F4 microcontroller
  series. More precisely it currently provides an implementation

    . of the Bundle Protocol (BP, RFC 5050),
    . of the Bundle Protocol version 7 specification draft (version
      6),
    . of the DTN IP Neighbor Discovery (IPND) protocol, and
    . of a routing approach optimized for message-ferry micro LEO
      satellites.

  uPCN is written in C and is built upon the real-time operating
  system FreeRTOS. The source code of uPCN is released under the
  "BSD 3-Clause License".

  The project depends on an execution environment offering link
  layer protocols such as AX.25. The source code uses the USB
  subsystem to interact with the environment.

9. Security Considerations

The bundle protocol security architecture and the available security
services are specified in an accompanying document, the Bundle
Security Protocol specification [BPSEC].

The bpsec extensions to Bundle Protocol enable each block of a
bundle (other than a bpsec extension block) to be individually
authenticated by a signature block (Block Integrity Block, or BIB)
and also enable each block of a bundle other than the primary block
(and the bpsec extension blocks themselves) to be individually
encrypted by a BCB.

Because the security mechanisms are extension blocks that are
themselves inserted into the bundle, the integrity and
confidentiality of bundle blocks are protected while the bundle is
at rest, awaiting transmission at the next forwarding opportunity,
as well as in transit.

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.

Note that, while the primary block must remain in the clear for routing purposes, the Bundle Protocol can be protected against traffic analysis to some extent by using bundle-in-bundle encapsulation to tunnel bundles to a safe forward distribution point: the encapsulated bundle forms the payload of an encapsulating bundle, and that payload block may be encrypted by a BCB.

Note that the generation of bundle status reports is disabled by default because malicious initiation of bundle status reporting could result in the transmission of extremely large numbers of bundle, effecting a denial of service attack.

The bpsec extensions accommodate an open-ended range of ciphersuites; different ciphersuites may be utilized to protect different blocks. One possible variation is to sign and/or encrypt blocks in symmetric keys securely formed by Diffie-Hellman procedures (such as EKDH) using the public and private keys of the sending and receiving nodes. For this purpose, the key distribution problem reduces to the problem of trustworthy delay-tolerant distribution of public keys, a current research topic.

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

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

Inclusion of the Bundle Security Protocol in any Bundle Protocol implementation is RECOMMENDED. Use of the Bundle Security Protocol in Bundle Protocol operations is OPTIONAL, subject to the following guidelines:

. Every block (that is not a bpsec extension block) of every bundle SHOULD be authenticated by a BIB citing the ID of the node that inserted that block. (Note that a single BIB may authenticate multiple "target" blocks.) BIB authentication MAY be omitted on (and only on) any initial end-to-end path segments on which it would impose unacceptable overhead, provided that satisfactory authentication is ensured at the

         convergence layer and that BIB authentication is asserted on
         the first path segment on which the resulting overhead is
         acceptable and on all subsequent path segments.
      . If any segment of the end-to-end path of a bundle will traverse
         the Internet or any other potentially insecure communication
         environment, then the payload block SHOULD be encrypted by a
         BCB on this path segment and all subsequent segments of the
         end-to-end path.

10. IANA Considerations

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

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

11. References

11.1. Normative References

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

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

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

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

11.2. Informative References

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

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

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

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

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

12. Acknowledgments

This work is freely adapted from RFC 5050, which was an effort of
the Delay Tolerant Networking Research Group. The following DTNRG
participants contributed significant technical material and/or
inputs to that document: Dr. Vinton Cerf of Google, Scott Burleigh,
Adrian Hooke, and Leigh Torgerson of the Jet Propulsion Laboratory,
Michael Demmer of the University of California at Berkeley, Robert
Durst, Keith Scott, and Susan Symington of The MITRE Corporation,
Kevin Fall of Carnegie Mellon University, Stephen Farrell of Trinity
College Dublin, Peter Lovell of SPARTA, Inc., Manikantan Ramadas of
Ohio University, and Howard Weiss of SPARTA, Inc.

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

13. 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.
. Migrate custody transfer to the bundle-in-bundle encapsulation
   specification.
. Introduce the concept of "node ID" as functionally distinct
   from endpoint ID, while having the same syntax.
. Restructure primary block, making it immutable. Add optional
   CRC.
. Add optional CRCs to non-primary blocks.
. Add block ID number to canonical block format (to support
   streamlined BSP).
. Add bundle age extension block, defined in this specification.
. Add previous node extension block, defined in this
   specification.
. Add flow label extension block, *not* defined in this
   specification.
. Add manifest extension block, *not* defined in this
   specification.
. Add hop count extension block, defined in this specification.
. Migrate Quality of Service markings to a new QoS extension
   block, *not* defined in this specification.

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.

   Copyright (c) 2017 IETF Trust and the persons identified as authors
   of the code. All rights reserved.

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

Appendix B.   CDDL expression

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

   start = bundle

   dtn-time = uint

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

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

   uri-code = uint

   eid = eid-choice .within eid-generic

   eid-choice /= [dtn-code, SSP: (text / 0)]

   dtn-code = 1 ; TBD

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

   ipn-code = 2 ; TBD

   bundle-control-flags = uint .bits bundleflagbits

   bundleflagbits = &(

     reserved: 15

     reserved: 14

     reserved: 13

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

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

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

     reserved: 9

```
      bundle-reception-status-reports-are-requested: 8

      bundle-contains-a-Manifest-block: 7

      status-time-is-requested-in-all-status-reports: 6

      user-application-acknowledgement-is-requested: 5

      destination-is-a-singleton-endpoint: 4

      reserved: 3

      bundle-must-not-be-fragmented: 2

      payload-is-an-administrative-record: 1

      bundle-is-a-fragment: 0

   )

   crc = bytes

   block-control-flags = uint .bits blockflagbits

   blockflagbits = &(

      reserved: 7

      reserved: 6

      reserved: 5

      reserved: 4

      bundle-must-be-deleted-if-block-cannot-be-processed: 3

      status-report-must-be-transmitted-if-block-cannot-be-processed: 2

      block-must-be-removed-from-bundle-if-it-cannot-be-processed: 1

      block-must-be-replicated-in-every-fragment: 0

   )

   bundle = [primary-block, *extension-block, payload-block]

   primary-block = [
```

```
                version: 7,

                bundle-control-flags,

                crc-type: uint,

                destination: eid,

                source-node: eid,

                report-to: eid,

                creation-timestamp,

                lifetime: uint,

                ? fragment-offset: uint,

                ? total-application-data-length: uint,

                ? crc,

   ]

   canonical-block-generic = [

                    block-type-code: uint,

                    canonical-block-common,

                    content: any

                    ]

   canonical-block-common = (

                    block-number: uint,

                    block-control-flags,

                    crc-type: uint,

                    block-data-length: uint,

                    ? crc,

                    )
```

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

canonical-block-choice /= payload-block

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

payload = bytes / bytes .cbor admin-record

canonical-block-choice /= extension-block

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

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

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

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

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

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

hop-count-block = [9, canonical-block-common,

                [hop-limit: uint,

                 hop-count: uint]]
```

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

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

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

bundle-status-report = [1, [bundle-status-information,

                          bundle-status-reason: uint,

                          admin-common]]

admin-common = (

            source-node: eid,
```

```
               creation-timestamp,

               ? fragment-offset: uint,

               ? payload-length: uint)

   bundle-status-information = [

     reporting-node-received-bundle: bundle-status-item,

     reporting-node-forwarded-the-bundle: bundle-status-item,

     reporting-node-delivered-the-bundle: bundle-status-item,

     reporting-node-deleted-the-bundle: bundle-status-item,

   ]

   bundle-status-item = [

               asserted: bool,

               ? time-of-assertion: dtn-time]
```

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
Nefeli Networks, Inc.
2150 Shattuck Ave.
Berkeley, CA 94704
US
Email: kfall@kfall.com

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