

Delay/Disruption Tolerant Networking
Internet-Draft
Intended status: Standards Track
Expires: 21 August 2026

R. Taylor
Aalyria Technologies
17 February 2026

Bundle Transfer Protocol - Unidirectional
draft-ietf-dtn-btpu-02

Abstract

This document defines a protocol for the unidirectional transfer of large binary objects, typically Bundle Protocol version 7 bundles, between two nodes connected by a unidirectional, unreliable, frame-based link-layer protocol, without requiring IP services.

The protocol does not require a return path for acknowledgements, but instead supports data repetition as a mechanism to protect against data loss. It fully supports the disaggregation of flows of binary objects of different priority, preventing head-of-line blocking impacting performance.

The wire format of the protocol is designed to enable performant implementation in hardware or software, with the aim of enabling protocol implementations to run at the line-rate of the underlying link-layer protocol.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at <https://ricktaylor.github.io/btpu/draft-ietf-dtn-btpu.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-dtn-btpu/>.

Discussion of this document takes place on the Delay/Disruption Tolerant Networking Working Group mailing list (<mailto:dtn@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/dtn/>. Subscribe at <https://www.ietf.org/mailman/listinfo/dtn/>.

Source for this draft and an issue tracker can be found at <https://github.com/ricktaylor/btpu>.

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on 21 August 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. Conventions and Definitions	5
2.1. Terminology	5
3. Protocol Overview	5
3.1. Messages	6
3.2. Padding	6
4. Segmentation and Transfers	7
4.1. Interleaving Transfers	8
4.2. Cancelling Transfers	8
5. Transfer Window	8
6. Handling Link-layer PDU Loss	10
7. Message Format	11
7.1. Message Flags	11
7.2. Hint Items	12
8. Message Definitions	12
8.1. Bundle Message	13
8.2. Transfer Segment Message	13
8.3. Transfer End Message	14
8.4. Transfer Cancel Message	14
8.5. Definite Padding Message	15

8.6. Indefinite Padding Message	15
9. Standard Hint Items	16
9.1. Bundle Length Hint	16
10. Security Considerations	17
11. Deployment Considerations	17
12. IANA Considerations	17
12.1. BTPU Message Types Registry	17
12.2. BTPU Hint Types Registry	19
12.3. BTPU Message Flags Registry	19
13. References	20
13.1. Normative References	20
13.2. Informative References	20
Appendix A. Examples	21
A.1. Segmentation of a sequence of Bundles of equal priority	21
A.2. Segmentation of a sequence of Bundles of different priority	22
A.3. Repetition of Bundle Segments	23
Acknowledgments	24
Author's Address	24

1. Introduction

Bundle Protocol version 7 (BPv7) is defined in terms a layered logical architecture, detailed in [RFC9171], wherein the responsibility for the storing and routing of bundles lies with the Bundle Processing Agent (BPA), and the BPA relies upon Convergence Layer Protocols (CLAs) to provide bundle transport between nodes. CLAs provide a unified interface to the BPA, allowing BPAs to be link-layer agnostic, but still use a diverse range of Convergence Layer Protocols (CLPs) to transfer bundles between BPAs over underlying link-layer protocols.

In the realm of near- and deep-space communication there are a number of standardized link-layer protocols, including [USLP], [TM], [AOS], [DVB-S2X], that share a set of common properties:

- * They are unidirectional: data transfer occurs in one direction only, there is no in-band return path for data.
- * They are frame-based: the link-layer protocol will guarantee that a frame of data is either delivered to a receiver in its entirety or not at all. Frames can be of fixed or variable length.

- * They provide a single logical channel: the communication between a sender and one or more receivers of frames can be logically separated from other communication over the link-layer by an implementation, perhaps by the use of channel identifiers, circuit identifiers, or tuples of source and destination address information.

These characteristics provide a common baseline that allows the definition of a lightweight mechanism for transferring BPv7 bundles meeting the requirements of a BPv7 Convergence Layer Protocol, and this document describes such a protocol: Bundle Transfer Protocol - Unidirectional (BTPU), suitable for implementation over any link-layer protocol that shares these characteristics. The protocol is applicable to other link-layer technologies which share these characteristics beyond those commonly used for space communication, for example 5G Unstructured PDUs [_5G], or Ethernet [IEEE.802.3], without requiring underlying IP services. Although designed for any link-layer protocol that shares the characteristics above, additional specification or profiling might be required to map the constructs of the link-layer protocol to the mechanisms defined in this specification.

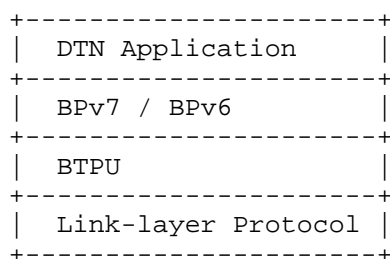


Figure 1: The location of BTPU in relation to the Bundle Protocol and a Link- layer protocol

The driving use-case of the protocol has been the transfer of BPv7 Bundles, however it is equally capable of transferring any kind of binary data, but includes no explicit discriminator of the format of a particular block of binary data. If multiple different types of binary data are to be transferred by a single implementation, this specification considers the differentiation between different types of binary data to be a matter for the implementation. For example, both BPv6 Bundles ([RFC5050]) and BPv7 Bundles can be multiplexed without issue, as the different formats can be distinguished by simple examination of the initial octets of a received bundle by an implementation. Additionally, the segmentation mechanism enables the use of this protocol with binary data formats that do not natively support some form of fragmentation.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.1. Terminology

Within the scope of this document, the following terms have specific meaning:

Bundle: A higher-layer protocol data unit, typically a BPv7 Bundle as defined in [RFC9171].

Link-layer PDU: The protocol data unit, excluding any link-layer protocol specific headers or metadata, that makes up a complete transmission unit or frame, as defined by the link-layer protocol specification.

Message: A single protocol data item, see Section 8.

Segment: In order to transfer a Bundle larger than a Link-layer PDU, Bundles can be subdivided into Segments in order to fit within a Link-layer PDU, see Section 4.

Transfer: The context in which the transmission of the Segments of a single Bundle occurs, see Section 4.

3. Protocol Overview

The purpose of the protocol is to transfer a series of Bundles between two nodes. Because a Bundle is of variable length, which is unlikely to be exactly the same size as a Link-layer PDU, the protocol defines a mechanism to divide Bundles into Segments as required, such that each Link-layer PDU is efficiently filled with data, and one or more Bundles can be transferred in a minimal number of Link-layer PDUs, described in more detail in Section 4.

This segmentation is unrelated to BPv7 bundle fragmentation as defined in Section 5.8 of [RFC9171]. Although BPv7 bundle fragmentation can be used to sub-divide oversized BPv7 bundles, the required duplication of metadata blocks can result in inefficiencies or fail to generate BPv7 bundle fragment small enough to fit in a single Link-layer PDU.

As a sender can prioritize the transfer of each Bundle differently, the protocol allows for the multiplexing of Bundle transfers, so that the transfer of higher priority Bundles can interrupt the transfer of other Bundles, avoiding "head of line blocking", see Section 4.1 for more detail. The bundle transfers are expected to occur over the same logical channel, rather than using a separate channel for each bundle or group of bundles that share priority. This does not preclude using multiple logical channels, but each channel is expected to be an independent instance of the protocol.

3.1. Messages

The basic primitive of the protocol is the Message, a self-describing unit of protocol control information of variable length. The sender is responsible for composing one or more Messages as required, and packing them into a Link-layer PDU, such that a single PDU is optimally filled. The receiving node parses the contained Messages from each received Link-layer PDU, and then processes them as individual control signals. This sequence of Messages is the logical control-plane used by the protocol. This document uses the verb "emit" to describe to the writing of a new Message to a Link-layer PDU ready for transmission, to differentiate from the the transmission of the Link-layer PDU itself, as many Messages can be emitted prior to the transmission of the containing PDU.

See Section 8 for detail of each type of Message.

3.2. Padding

Because the size of a Bundle is not expected to exactly match the size of a Link-layer PDU, an implementation will likely need to add padding to the PDU so that the Link-layer PDU size requirements are met. Two Messages are available for this purpose: The Definite Padding Message (Section 8.5) and the Indefinite Padding Message (Section 8.6). Padding Messages are valid at any point within a Link-layer PDU.

It is RECOMMENDED that implementations use the Definite Padding Message to add padding to a Link-layer PDU, except when less than four octets of padding are required, as the minimum length of the Definite Padding Message is four octets.

When the link-layer protocol provides variable length Link-layer PDUs, implementations SHOULD take into account the mechanisms used by the link-layer protocol to support variable length Link-layer PDUs, and emit Link-layer PDUs of a suitable size for the underlying protocol. For example, if variable length Link-layer PDUs are implemented by the link-layer protocol using a sub-framing mechanism, then emitting Link-layer PDUs of a single, or whole number of sub-frames can increase reliability.

The algorithm used to pad and pack Messages efficiently into Link-layer PDUs is an implementation matter.

4. Segmentation and Transfers

As described in the Protocol Overview (Section 3), in order to transfer Bundles larger than a single Link-layer PDU into multiple PDUs, Bundles are be divided into a sequence of Segments by the sender and each Segment is emitted in its own a Message. However, if a complete Bundle can fit in the next Link-layer PDU, then the Bundle SHOULD be transferred without segmentation, see the Bundle Message (Section 8.1).

Each Segment is assigned a monotonically increasing integral Segment Index that indicates the relative position of the Segment within the total sequence of Segments, with zero (0) indicating the first segment; i.e. Segments are ordered 0 to N, where N is the Segment Index in the Transfer End Message.

The Segments of a Bundle MUST be emitted by the sender as a series of Transfer Segment Messages (Section 8.2) carrying the same Transfer identifier, starting with Segment Index zero (0). The end of a transfer MUST be indicated by emitting a Transfer End Message (Section 8.3), including the index and data of the final Segment.

In addition to a Segment Index, every Segment is associated with a Transfer that provides context to the sequence of Segments to enable the correct reassembly of the original Bundle. Each Transfer is assigned a number as an identifier, with each identified Transfer mapping to the segmentation of a single Bundle.

Transfer numbers are expressed as 32-bit unsigned integers. A sending implementation SHOULD choose a random value between 0 and $2^{32}-1$ for the first Transfer number, and each subsequent Transfer MUST use the next numeric value in the sequence. To avoid placing a limit on the total number of Transfers between peers, numbers are allowed to "roll-over" to zero and repeat, i.e. the next number in the sequence is the previous number incremented by one, modulo 2^{32} .

A receiver reassembles the transferred Bundle by concatenating the Segments that share a common Transfer number in the ascending order of their Segment Index. The Segment Index in the Transfer End Message indicates the final Segment Index (N); a transfer is complete when Segments with indices 0 through N have been received. Once all the Segments have been received and concatenated, a receiver is expected to pass the recombined Bundle to an upper layer for further processing.

4.1. Interleaving Transfers

In order to support the transmission of Bundles with different priorities, Transfer Messages associated with different Transfers, i.e. with different Transfer numbers, MAY be interleaved. This allows senders to interrupt the emission of a sequence of Segments associated with one Transfer with one or more Segments of another Transfer, preventing a large lower priority Transfer blocking a higher priority Transfers.

4.2. Cancelling Transfers

A Transfer can be aborted by the sender while a Transfer is in progress by emitting a Transfer Cancel Message (Section 8.4) containing the identifier of the Transfer to cancel. A receiver of a Transfer Cancel Message SHOULD discard any segments already received and MUST ignore any further Messages associated with the Transfer.

5. Transfer Window

Because Messages can be lost in transmission due to the loss of Link-layer PDUs, and a sender MAY emit duplicate Messages as a defense against loss, see Section 6, a sender MUST maintain a sliding Transfer Window that defines the maximum number of Transfers that can be simultaneously in progress. As Transfers are identified by a monotonically increasing number, the size of the Transfer Window also strictly defines the range of identifiers of Transfers in progress.

The sender MUST maintain a reference to the greatest Transfer number used in any emitted Message, and MUST NOT emit any Message with a Transfer number less than or equal to the latest minus the size of the Transfer Window, taking into account the modulo 2^{32} roll-over.

Each receiver MUST maintain a reference to the greatest Transfer number received in any Message. When a Transfer Message is received with a Transfer number greater than the greatest previously received, the new Transfer number is considered the greatest Transfer number, and Transfers with number less than or equal to the latest minus the size of the Transfer Window MUST be considered Section 4.2. Because

of Transfer number roll-over, half the number space of 2^{32} and window size is used to determine if a number is older or newer than the latest Transfer number. Pseudocode for the algorithm is given in Figure 2.

The size of the Transfer Window SHOULD be the same at the sender and all receivers, and MUST be configured via some out-of-band mechanism. The Transfer Window size MUST be at least 4, MUST be less than 2^{12} , and is RECOMMENDED to be 16.

// 16 is an arbitrary number, and needs discussing by the WG.

```
const WINDOW_SIZE # Configured transfer window size
var GREATEST = NIL # Greatest received transfer number, initially NIL

# Function to check if a transfer is valid within the current window
FUNCTION isValidTransfer(T):
    # Ensure Transfer T is within the
    # sliding window defined by WINDOW_SIZE
    RETURN ((GREATEST - T +  $2^{32}$ ) MOD  $2^{32}$ ) < WINDOW_SIZE

# Function to check if the transfer is considered a "new" transfer
FUNCTION isNewTransfer(T):
    IF GREATEST IS NIL THEN
        # The first transfer is always considered new
        RETURN TRUE
    # Check if the transfer is within the valid window range
    # (half of the number space + window size)
    RETURN ((T - GREATEST +  $2^{32}$ ) MOD  $2^{32}$ ) <
        ( $2^{32}$  / 2) + (WINDOW_SIZE / 2)

# Main function to process a transfer and manage the sliding window
FUNCTION processTransfer(T):
    IF isNewTransfer(T) THEN
        # New transfer, update the greatest received transfer number
        GREATEST ← T
        # Cancel transfers that are now outside the window
        CANCEL_OUTDATED_TRANSFERS()
    ELSE IF isValidTransfer(T) THEN
        # Transfer is in progress, continue handling it
        CONTINUE_PROCESSING(T)
    ELSE
        # Transfer is invalid (outside the window), ignore it
        IGNORE_MESSAGE(T)
```

Figure 2: A receiver's algorithm for determining Transfer number validity and sliding window

6. Handling Link-layer PDU Loss

Due to the unreliable nature of the link-layer protocol, Link-layer PDUs can be lost in transmission, resulting in the loss of the contained Messages. Because the underlying link-layer is assumed to be unidirectional, the protocol does not include a mechanism to trigger the retransmission of lost Messages; instead the protocol allows the sender to repeat the transmission of Messages.

A sender MAY emit any Message multiple times in different Link-layer PDUs. Although every Link-layer PDU transmitted MAY contain different Messages, any repeated Message MUST be an exact copy of an already emitted Message. A receiver MAY ignore any duplicate Message already received for an in-progress Transfer. When segmenting bundles, not all Messages in a Transfer need be repeated the same number of times, and different Transfers can repeat Messages differently.

Although it is RECOMMENDED that segments are emitted in ascending order of Segment Index; once emitted, any Message MAY be repeated any number of times, in any order. The number of repetitions of a particular Message is an implementation matter that can be influenced by many factors, including:

- * Offline analysis of the deployed environment might require a certain amount of Message repetition to reach some required certainty of transfer.
- * A higher 'reliability' factor associated with a particular Bundle could result in more copies of each associated Transfer Message being emitted.
- * Signalling from the link-layer protocol, or some other out-of-band mechanism, might trigger increased repetition of a subset of Messages, to protect against some temporary spike in Link-layer PDU loss rate.

Message replication is logically separate from any facilities the underlying link-layer protocol might have to protect against information loss through redundancy and/or erasure coding, and MAY be used as required by a deployment. If a link-layer protocol receives a duplicate Link-layer PDU, it SHOULD be delivered to this protocol only once.

7. Message Format

All protocol Messages except the Indefinite Padding Message (Section 8.6) follow the common "Type-Length-Value" formatting pattern, with each Message being identified by a four octet header that encodes the type of the Message, and the length of the content of the Message.

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      Type      | Flags |      Length (20-bit unsigned int)      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     ... Optional Hint Items ...   :
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     ... Content ...               :
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Type: The type of the Message, allocated from IANA "BTPU Message Types" registry, see Section 12, encoded as a 8-bit unsigned integer in network byte order.

Flags: A 4-bit field for flags, see Section 7.1.

Length: The length of the Message in octets, excluding the 4 octets of the header itself, encoded as a 20-bit unsigned integer in network byte order. This length includes both the optional Hint Items (Section 7.2) and the Content.

Content: A sequence of octets of data. Its length is the main Message Length minus the total length of any present Hint Items. The content is encoded according to the Type of the Message.

7.1. Message Flags

The Message Flags field is a 4-bit
 // 4 bits is considered enough, as it allows a 20-bit Message Length
 // field, and many additional flags could be better expressed with
 // Hint Items. field, formatted as follows:

```

    0 1 2 3
+---+---+---+
|H| RFU |
+---+---+---+

```

H: The 'H' (Hint) flag. If this bit is set to 1, it indicates that one or more Hint Items are present immediately following the Message header.

RFU (Reserved for Future Use): These 3 bits are unassigned. They MUST be set to zero by the sender and MUST be ignored by the receiver.

7.2. Hint Items

To allow the transfer of optional additional information, Messages can carry extra information in the form of Hint Items. Because Messages can be lost in transmission, this metadata provides additional information about the Transfer itself, rather than being particular to the containing Message, and MAY be repeated in multiple different or repeated Messages.

If the 'H' flag is set in the Message header, the header is followed by one or more Hint Items. Each item is encoded in a Type-Length-Value format, with a 2-octet header followed by a variable-length value. This format intentionally omits a flags field to prevent nested metadata.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Hint Type | H | Length | ... Value ... |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Hint Type: A 7-bit identifier for the Hint Item, allocated from the "BTPU Hint Types" IANA registry, see Section 12, encoded as a 7-bit unsigned integer in network byte order.

H: The 'H' (Hint) flag. If this bit is set to 1, it indicates that another Hint Item immediately follows this Hint Item. If this bit is set to 0, then the Message content immediately follows the Hint Item.

Length: An 8-bit unsigned integer specifying the length of the Value field in octets. This limits the value of a single Hint Item to 255 octets.

Value: The payload of the Hint Item.

8. Message Definitions

The following standard protocol Messages are defined:

8.1. Bundle Message

The Bundle Message is used to encapsulate an entire Bundle, and SHOULD be used by an implementation when a Bundle will fit in its entirety in a single Link-layer PDU to avoid the overhead of segmentation, and reducing the risk of the total loss of a Bundle if one or more unnecessary segments of a Bundle is lost.

A Bundle Message has a type of 2. The Message Content MUST be a valid Bundle.

Emitting a Bundle Message with a Length field value that indicates no Bundle content (e.g., a length of 0 if no metadata is present) only adds control-plane overhead and SHOULD NOT be used as an alternative form of padding.

8.2. Transfer Segment Message

The Transfer Segment Message is used to encapsulate a segment of a multi-segment Bundle Transfer.

A Transfer Segment Message has a type of 3. The Message Content field is formatted as follows:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Transfer Number                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Segment Index                                       |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     ... Segment Data ...                               :
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Transfer Number: The numeric identifier of the Transfer that this Segment is part of, encoded as a 32-bit unsigned integer in network byte order.

Segment Index: The position of the Segment within the sequence of all Segments of a Transfer, encoded as a 32-bit unsigned integer in network byte order.

Segment Data: The octets of a Segment of the Transfer, with the length calculated as the Message content length excluding the eight (8) octets of the Transfer Number and Segment Index.

Transfer Segment Messages SHOULD NOT have zero octets of Segment Data, i.e. the total length of the Message SHOULD be greater than 12 octets. Such Messages only add control-plane overhead and SHOULD NOT be used as an alternative form of padding.

8.3. Transfer End Message

The Transfer End Message is used to indicate the completion of a multi-segment Bundle Transfer, and encapsulate the final segment.

A Transfer End Message has a type of 4. The Message Content field is formatted as follows:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Transfer Number                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Segment Index                                       |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     ... Segment Data ...                               :
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Transfer Number: The numeric identifier of the in-progress (Section 5) Transfer that is completing, encoded as a 32-bit unsigned integer in network byte order.

Segment Index: The Segment Index of the final Segment, encoded as a 32-bit unsigned integer in network byte order.

Segment Data: The octets of the final Segment of the Transfer, with the length calculated as the Message content length excluding the eight (8) octets of the Transfer Number and Segment Index.

Transfer End Messages SHOULD NOT have zero octets of Segment Data, i.e. the total length of the Message SHOULD be greater than 12 octets. Such Messages only add control-plane overhead and SHOULD NOT be used as an alternative form of padding.

8.4. Transfer Cancel Message

The Transfer Cancel Message is used to indicate that the indicated Transfer is being aborted, and any prior or later received Segments associated with the Transfer MUST be discarded by a receiver.

A Transfer Cancel Message has a type of 5. The Message Content field is formatted as follows:

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Transfer Number                                                                 |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Transfer Number: The numeric identifier of the in-progress (Section 5) Transfer that is cancelled, encoded as a 32-bit unsigned integer in network byte order.

The Transfer Cancel Message has no content, and hence has a fixed length of 4 octets.

A peer that receives a Transfer Cancel Message with a Transfer Number field value that does not match the numeric identifier of an in-progress (Section 5) Transfer MUST ignore the Message.

8.5. Definite Padding Message

The Definite Padding Message is used to add padding to a Link-layer PDU.

A Definite Padding Message has a type of 1. Any content it contains has no semantic meaning, and a sender SHOULD set the content to a sequence of zero (0) octets. Receivers MUST ignore any Message content.

It is valid for this Message to have no content, i.e. a Length field value of zero (0), adding a total of four (4) octets of padding to the Link-layer PDU.

8.6. Indefinite Padding Message

An Indefinite Padding Message has a type of zero (0), and in order to support padding with a minimum total length of one octet, the Message does not include an explicit Length or Content field, and hence has the following layout:

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Type (0) | ... Padding ... |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Type: The type of the Message: zero (0).

The Indefinite Padding Message type field is followed by a sequence of zero or more zero (0) octets, ending at the first non-zero octet, or the end of the fixed-length Link-layer PDU. The content of the Message has no meaning, and MUST be ignored by a receiver.

Note: When an Indefinite Padding Message terminates with a non-zero octet, the non-zero octet is the first octet of the subsequent Message.

9. Standard Hint Items

The following Hint Items are defined in this document:

9.1. Bundle Length Hint

The Bundle Length Hint Item is used to signal the total length of a bundle that is being transferred in segments. This allows a receiver to pre-allocate the necessary memory to reassemble the complete bundle.

This Hint Item MAY be included in a Transfer Segment Message or a Transfer End Message. Receivers SHOULD ignore this Hint Item if encountered in other message types.

The Hint Item has a Hint Type of 0, and its layout is as follows:

[illegible]

Length: The length of the Bundle Length field. This MUST be one of 1, 2, 4, or 8.

Bundle Length: The total length of the bundle being transferred, encoded as an unsigned integer in network byte order, with a size corresponding to the Length field.

A sender SHOULD choose the smallest possible length that can accommodate the total bundle length. For example, a bundle of 2000 octets should be encoded with a Length of 2, and a Value of 2000 encoded as a 16-bit unsigned integer.

10. Security Considerations

This protocol does not define any measures to protect Messages or their content. There might be link-layer mechanisms to protect the transmission of frames against over-hearing and interference, and upper layer mechanisms, such as BPSec defined in [RFC9172], can be used to provide integrity and protection at a higher layer. Therefore transport-layer security is considered out of scope for the protocol, and lower and/or upper layer mechanisms **MUST** be used to provide security.

11. Deployment Considerations

The following caveats are to be considered before deploying instances of this protocol:

1. It is unreliable. Although there might be a link-layer protocol mechanism for a receiver to be notified that a frame has been lost in transmission, due to the unidirectional nature of the link-layer there is no in-band return path suitable for higher-layer acknowledgement of transfer success. Any acknowledgement system designed to provide reliability **MUST** use a logically separate path from receiver back to sender.
2. It does not provide congestion control or signalling. The underlying link-layer is expected to provide an uncontested channel between sender and receivers, and hence such mechanisms are considered to be out of scope. The protocol **MUST NOT** be deployed in environments where congestion might occur, such as the public Internet, when the underlying link-layer does not provide suitable congestion control.
3. It requires an out-of-band mechanism for configuration. This can either be via pre-placed static configuration, a parallel dynamic control-plane protocol, or some other mechanism beyond the scope of this specification.

12. IANA Considerations

12.1. BTPU Message Types Registry

IANA is requested to create a new registry entitled "BTPU Message Types", in the existing "Bundle Protocol" registry group. The registration procedures for this registry, using terms defined in [RFC8126], is:

Values	Registration Procedure
0..0x6F	Standards Action
0x70..0x7F	Private Use
0xA0..0xFF	Reserved for future expansion

Table 1: BTPU Message Types registration policies

The initial values for the registry are:

Type	Message	Reference
0	Indefinite Padding Message (Section 8.6)	This document
1	Definite Padding Message (Section 8.5)	This document
2	Bundle Message (Section 8.1)	This document
3	Transfer Segment Message (Section 8.2)	This document
4	Transfer End Message (Section 8.3)	This document
5	Transfer Cancel Message (Section 8.4)	This document
6	Reserved to avoid clash with BPv6	This document
0x80..0x9F	Reserved to avoid clash with BPv7 Bundle (CBOR array)	This document

Table 2: BTPU Message Types initial values

The initial value of 6 is reserved to ensure that a Link-layer PDU containing a single Bundle Protocol version 6 bundle can be distinguished from BTP-U Messages.

12.2. BTPU Hint Types Registry

IANA is requested to create a new registry entitled "BTPU Hint Types", in the existing "Bundle Protocol" registry group. The registration procedures for this registry, using terms defined in [RFC8126], is:

Specifications defining new Hint Types MUST specify which Message types the Hint is applicable to, and SHOULD specify receiver behavior when the Hint is encountered in other Message types.

Values	Registration Procedure
0..0x6F	Standards Action
0x70..0x7F	Private Use

Table 3: BTPU Hint Types
registration policies

The initial values for the registry are:

Type	Hint	Reference
0	Bundle Length (Section 9.1)	This document

Table 4: BTPU Hint Types initial values

12.3. BTPU Message Flags Registry

IANA is requested to create a new registry entitled "BTPU Message Flags", in the existing "Bundle Protocol" registry group. This registry governs the 4-bit Message Flags field. The registration procedures for this registry, using terms defined in [RFC8126], is Standards Action.

The initial values for the registry are:

Bit	Name	Description	Reference
0 (0x8)	H (Hint)	Indicates presence of Hint Items	This document
1-3	Unassigned		

Table 5: BTPU Message Flags initial values

Bit 0 is the most significant bit.

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/rfc/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC9171] Burleigh, S., Fall, K., and E. Birrane, III, "Bundle Protocol Version 7", RFC 9171, DOI 10.17487/RFC9171, January 2022, <<https://www.rfc-editor.org/rfc/rfc9171>>.
- [RFC9172] Birrane, III, E. and K. McKeever, "Bundle Protocol Security (BPsec)", RFC 9172, DOI 10.17487/RFC9172, January 2022, <<https://www.rfc-editor.org/rfc/rfc9172>>.

13.2. Informative References

- [AOS] "Advanced Orbiting Systems (AOS) Space Data Link Protocol", CCSDS 732.0-B-4, October 2021, <<https://ccsds.org/Pubs/732x0b4.pdf>>.
- [DVB-S2X] "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and

other broadband satellite applications; Part 2: DVB-S2 Extensions (DVB-S2X)", ETSI EN 302 307-2, August 2024, <https://www.etsi.org/deliver/etsi_en/302300_302399/30230702/01.04.01_60/en_30230702v010401p.pdf>.

[IEEE.802.3]

"IEEE Standard for Ethernet", IEEE, DOI 10.1109/ieeestd.2022.9844436, ISBN ["9781504487252"], July 2022, <<https://doi.org/10.1109/ieeestd.2022.9844436>>.

[RFC5050]

Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, DOI 10.17487/RFC5050, November 2007, <<https://www.rfc-editor.org/rfc/rfc5050>>.

[TM]

"Telemetry (TM) Space Data Link Protocol", CCSDS 132.0-B-3, October 2021, <<https://ccsds.org/Pubs/132x0b3.pdf>>.

[USLP]

"Unified Space Data Link Protocol (USLP)", CCSDS 732.1-B-3, June 2024, <<https://ccsds.org/Pubs/732x1b3e1.pdf>>.

[_5G]

3GPP and C. Devaki, "System architecture for the 5G System (5GS)", 21 December 2022, <https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/23501-i00.zip>.

Appendix A. Examples

The following sections has examples of an implementation emitting Bundles into Link-layer PDUs. Although the examples demonstrate the core principles, for example Bundle Segmentation with priority, the algorithm used to pack Messages into Link-layer PDUs is just for example purposes. An implementation can use an alternate algorithm that meets this specification but suits its overall architecture, and where this is applicable is noted in each example.

A.1. Segmentation of a sequence of Bundles of equal priority

An example of the transmission of three Bundles of varying sizes and equal priority in three Link-layer PDUs is shown in Figure 3.

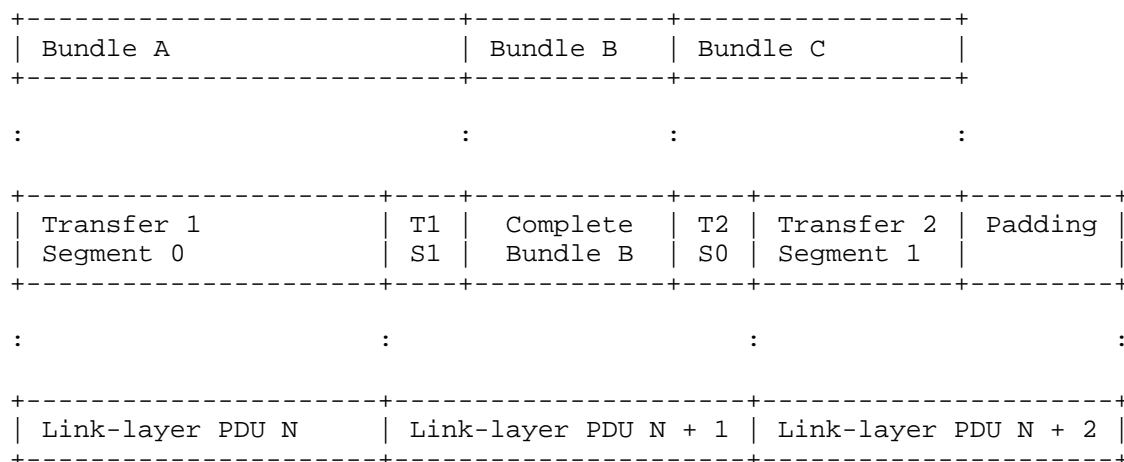


Figure 3: Segmentation of a sequence of Bundles of equal priority

Bundle A is emitted as two Segments, included in the first and second Link-layer PDU, as Transfer 1. Bundle B fits completely in the second Link-layer PDU, and is therefore emitted without segmentation. Bundle C is emitted as two Segments split between the second and third PDU, but padding is required to fill the third PDU. An alternative algorithm could have selected to not segment Bundle C, but to pad the second PDU and include Bundle C without segmentation in the third PDU, without changing the semantics, as an implementation preference.

A.2. Segmentation of a sequence of Bundles of different priority

An example of the transmission of three Bundles of varying sizes and different priority in three Link-layer PDUs is shown in Figure 4.

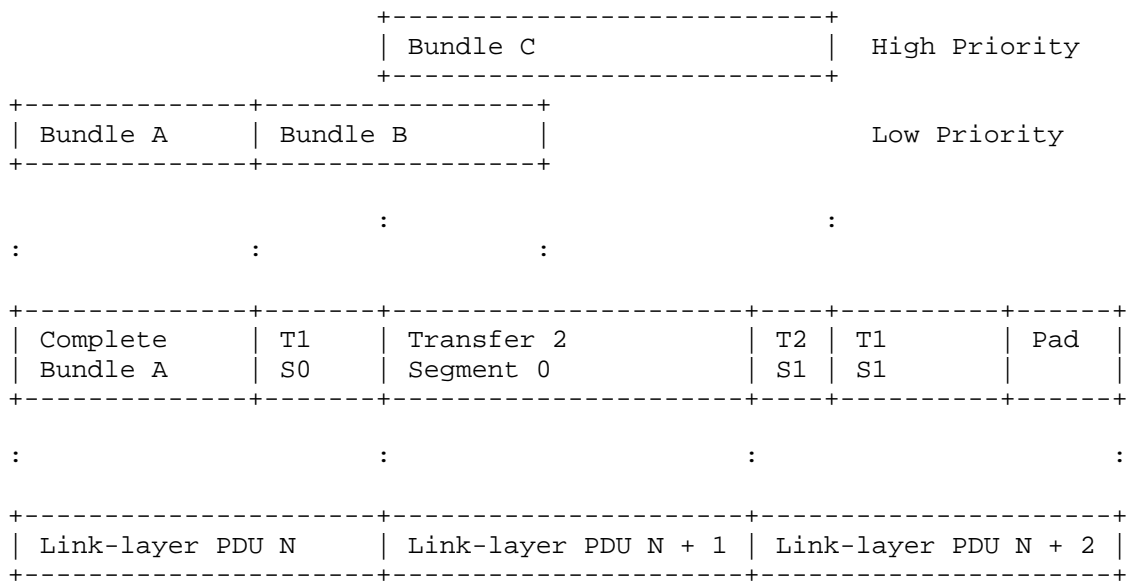


Figure 4: Interleaved segmentation of a sequence of Bundles of different priority

Bundle A is emitted without segmentation, and the Bundle B is segmented to fill the first Link-layer PDU. During the preparation of the next Link-layer PDU high priority Bundle C is queued for emission. Therefore the further emission of Bundle B is paused, and Bundle C is emitted as two Segments, with the third Link-layer PDU containing the second Segments of both Bundle B and C, plus padding. The order of emission of the second Segments of Bundle B and C makes no semantic difference.

A.3. Repetition of Bundle Segments

An example of the transmission of two Bundles of differing required repetition in three Link-layer PDUs is shown in Figure 5.

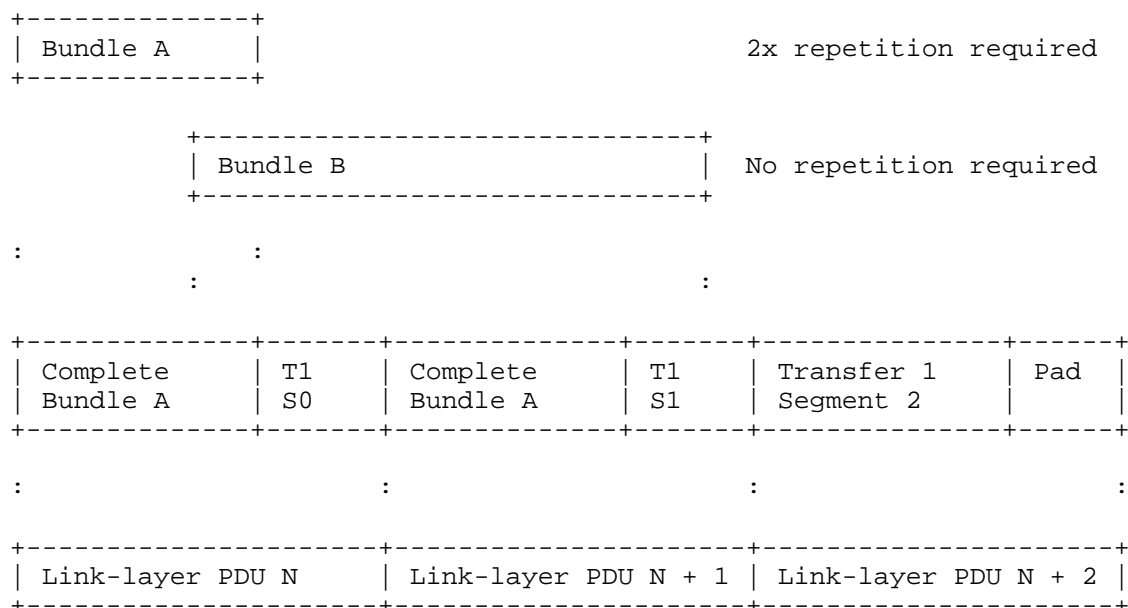


Figure 5: Repetition of some Bundles within a sequence of Segments

Bundle A is required by some higher-layer loss protection policy to be repeated twice in two separate Link-layer PDUs. Bundle A does not require additional loss protection, and can therefore be transmitted once. As Bundle A can fit in its entirety in a single Link-layer PDU, it is emitted as a Complete Bundle Message, with Bundle B emitted as a Segment Messages sized to fill the remainder of each PDU. The Complete Bundle Message of Bundle A is repeated in the second Link-layer PDU, with the remainder of the second PDU filled with the next Segment of Bundle B. The third Link-layer PDU contains the last Segment of Bundle B and padding. An alternate implementation could have segmented both Bundles, and repeated the Segments of Bundle A whilst interleaving the Segments of Bundle B.

Acknowledgments

The author would like to thank Erik Kline, Brian Sipos, and Chloe He for their invaluable feedback and discussion of the protocol, and this work would not exist without the excellent prior work by the TCP-CLv4 authors.

Author's Address

Rick Taylor
Aalyria Technologies

Email: rtaylor@aalyria.com