Delay-Tolerant Networking B. Sipos
Internet-Draft JHU/APL
Intended status: Standards Track J. Deaton
Expires: 20 June 2026 SAIC
17 December 2025

Delay-Tolerant Networking UDP Convergence Layer Protocol Version 2
draft-ietf-dtn-udpcl-03

Abstract

This document describes a UDP convergence layer (UDPCL) for Delay-Tolerant Networking (DTN). This version of the UDPCL protocol clarifies requirements of the earlier experimental RFC 7122, adds discussion of multicast addressing, congestion signaling, and updates to the Bundle Protocol (BP) contents, encodings, and convergence layer requirements in BP version 7. Specifically, the UDPCL uses CBOR-encoded BPv7 bundles as its service data unit being transported and provides an unacknowledged transport of such bundles. This version of UDPCL also includes security and extensibility mechanisms.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes the UDP convergence-layer protocol for Delay-Tolerant Networking (DTN). DTN is an end-to-end architecture providing communications in and/or through highly stressed environments, including those with intermittent connectivity, long and/or variable delays, and high bit error rates. More detailed descriptions of the rationale and capabilities of these networks can be found in "Delay-Tolerant Networking Architecture" [RFC4838].

An important goal of the DTN architecture is to accommodate a wide range of networking technologies and environments. The protocol used for DTN communications is the Bundle Protocol version 7 (BPv7) [RFC9171], an application-layer protocol that is used to construct a store-and-forward overlay network. BPv7 requires the services of a "convergence-layer adapter" (CLA) to send and receive bundles using the service of some "native" link, network, or Internet protocol.

This document describes one such convergence-layer adapter that uses the well-known User Datagram Protocol (UDP). This convergence layer is referred to as UDP Convergence Layer (UDPCL). For the remainder of this document, the abbreviation "BP" without the version suffix refers to BPv7.

The locations of the UDPCL and the Bundle Protocol in the Internet Protocol Suite Model [RFC1122] [RFC1123] are shown in Figure 1. In particular, when BP is using UDP as its bearer with UDPCL as its convergence layer, both BP and UDPCL reside at the application layer of the Internet model.

```
+-------------------------+
|     DTN Application     | -\
+-------------------------+   |
|  Bundle Protocol (BP)   |   -> Application Layer
+-------------------------+   |
| UDP Conv. Layer (UDPCL) |   |
+-------------------------+   |
|     DTLS (optional)     | -/
+-------------------------+
|          UDP            | ---> Transport Layer
+-------------------------+
|       IPv4/IPv6         | ---> Network Layer
+-------------------------+
|   Link-Layer Protocol   | ---> Link Layer
+-------------------------+
```

Figure 1: The Locations of BP and UDPCL above the Internet Protocol Stack

This specification defines a mechanism to embed protocol extensions in Section 3.4 and an initial set of extensions in Section 3.5. Following IETF best practices [RFC6709] these extension items are all mandatory-to-implement, meaning an interoperable UDPCL entity needs to understand them, but none are mandatory-to-use. These initial extensions also define the canonical means of implementing BP PDU segmentation (Section 3.6), IP path characterization (Section 2.4 and Section 3.7), and congestion control feedback (Section 2.7 and Section 3.8).

This specification also defines a transport security mechanism based on Datagram Transport Layer Security (DTLS) with Public Key Infrastructure Using X.509 (PKIX) authentication credentials. Similar to extension support, the use of DTLS with PKIX is mandatory-to-implement but not mandatory-to-use and is the canonical means of providing transport security.

1.1. Scope

This document describes the format of the protocol data units passed between entities participating in UDPCL communications along with procedures for those entities. This document does not address:

* The format of protocol data units of the Bundle Protocol, as those are defined elsewhere in [RFC9171]. This includes the concept of bundle fragmentation and bundle encapsulation. The UDPCL transfers bundles as opaque BP PDU octet strings.

* Mechanisms for locating or identifying other bundle entities (peers) within a network or across an internet. The mapping of a Node ID to a potential convergence layer (CL) protocol and network address is left to implementation and configuration of the BP Agent (BPA) and its various potential routing strategies, as is the mapping of a DNS name and/or address to a choice of an end-entity certificate to authenticate a node to its peers.

* Logic for routing bundles along a path toward a bundle’s endpoint. This CL protocol is involved only in transporting bundles between adjacent entities in a routing sequence.

* Logic for performing rate control and congestion control of bundle transfers, both incoming and outgoing from a UDPCL entity. The signaling defined in Section 3.5.8 and Section 3.8 can support a congestion control mechanism, but it is an implementation matter to choose and configure such a mechanism (see Appendix B).

* Policies or mechanisms for issuing PKIX certificates; provisioning, deploying, or accessing certificates and private keys; deploying or accessing certificate revocation lists (CRLs); or configuring security parameters on an individual entity or across a network.

* Uses of DTLS which are not based on PKIX certificate authentication (see Section 5.11.2) or in which authentication of both entities is not possible (see Section 5.11.1).

Any UDPCL implementation requires a BPA to perform those above listed functions in order to perform end-to-end bundle delivery.

1.2. Use of CDDL

This document defines CBOR structure using the Concise Data Definition Language (CDDL) [RFC8610]. The entire CDDL structure can be extracted from the XML version of this document using the XPath expression:

'//sourcecode[@type="cddl"]'

The following initial fragment defines the top-level symbols of this document's CDDL.

```
start = udpcl-ext-map
```

This specification does not make use of CDDL rules from BP [RFC9171] because a BP PDU appears as an opaque octet string to the UDPCL.

1.3. Terminology

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.

The following are definitions specific to the UDPCL protocol.

UDPCL Entity: This is the notional UDPCL application that initiates UDPCL transfers. This design, implementation, configuration, and specific behavior of such an entity is outside of the scope of this document. However, the concept of an entity has utility within the scope of this document as the container and initiator of transfers. The relationship between a UDPCL entity and UDPCL conversations is defined as follows:

- A UDPCL Entity MAY actively perform any number of transfers and SHOULD do so whenever the entity has a bundle to forward to another entity in the network.

- A UDPCL Entity MAY support zero or more passive listening elements that listen for transfers from other entities in the network, including non-unicast transfers.

These relationships are illustrated in Figure 2. For the remainder of this document, the term "entity" without the prefix "UDPCL" refers to a UDPCL entity.

UDP Conversation: This refers to datagrams exchanged between two network peers, with each peer identified by a (unicast IP address, UDP port) tuple. Because UDP is connectionless, there is no notion of a conversation being "opened" or "closed" and some conversations are uni-directional, meaning UDP datagrams are exchanged in only one direction between two peers.

IP Reachable: The ability of a source IP node to send packets which are received at a destination IP node. Reachability requires support not just of the IP endpoints but of the network and middleboxes between the nodes and links along the path between those endpoints. Reachability in one direction between two IP endpoints does not imply reachability (at the same time or any other time) in the opposite direction. The UDPCL is meant to be usable, with limited capability, in situations where peers are reachable in only one direction.

Transfer: This refers to the procedures and mechanisms for conveyance of an individual bundle from one entity to one or more destinations. This version of UDPCL includes a segmentation mechanism to allow transfers which are larger than the allowable UDP datagram size.

Transmit: This refers to a transfer outgoing from an entity as seen from that transmitting entity.

Receive: This refers to a transfer incoming to an entity as seen from that receiving entity.

```
   +----------------------------------------+
   |              UDPCL Entity              |
   |                                        |      +----------------+
   |   +--------------------------------+   |      |                |-+
   |   | Actively Initiated Transfer #1 |--------->| Other          | |
   |   +--------------------------------+   |      | UDPCL Entity’s | |
   |                 ...                    |      | Passive        | |
   |   +--------------------------------+   |      | Listener       | |
   |   | Actively Initiated Transfer #n |--------->|                | |
   |   |                                |          |                | |
   |   |          Sender Listen         |<---------|                | |
   |   +--------------------------------+   |      +----------------+ |
   |                                        |       +----------------+
   |      +----------------------------+    |
   |      | +----------------------------+  |      +----------------+
   |      | | Optional Passive           |  |      |                |-+
   |      +-| Listener(s)                |<---------+ Other          | |
   |        +----------------------------+  |      | UDPCL Entity’s | |
   |                                 ^      |      | Active         | |
   |                                 |      |      | Initiator(s)   | |
   |                                 +-------------|                | |
   +----------------------------------------+      +----------------+ |
                                                    +----------------+
```

Figure 2: The relationships between UDPCL entities

2. General Protocol Description

The service of this protocol is the transmission of DTN bundles via the User Datagram Protocol (UDP). This document specifies the optional segmentation of bundles, procedures for DTLS setup and teardown, and a set of messages and entity requirements. The general operation of the protocol is as follows.

Fundamentally, the UDPCL is a (logically) unidirectional "transmit and forget" protocol which itself maintains no long-term state and provides no feedback from the receiver to the transmitter. The only long-term state related to UDPCL is used by DTLS in its session keeping (which is bound to a UDP conversation). An entity receiving a bundle from a particular source address-and-port does not imply that the transmitter is willing to accept bundle transfers on that same address-and-port. It is the obligation of a BPA and its routing schemes to determine a bundle return path (if such a path even exists).

2.1. Convergence Layer Services

This version of the UDPCL provides the following services to support the over-laying BPA. In all cases, this is not an API definition but a logical description of how the UDPCL entity can interact with the BPA. Each of these interactions can be associated with any number of additional metadata items as necessary to support the operation of the CL or BPA.

Begin Transmission: The principal purpose of the UDPCL is to allow a BPA to transmit bundle data to one or more other entities. The receiver of each transfer is identified by an (destination) IPv4 or IPv6 address, a UDP port number, and an optional local interface identifier (see Section 3 for details). The CL does not necessarily perform any transmission queueing (see Section 2.7), but might block while transmissions are being processed at the UDP layer. Any queueing of transmissions is the obligation of the BPA.

Transmission Started: The UDPCL entity indicates to the BPA when a bundle transmission begins sending UDP datagrams. Once started, there is no notion of a UDPCL transmission failure; a BPA has to rely on bundle-level status reporting to track bundle progress through the network. Because of potential queueing or DTLS setup time, this can be delayed from the BPA providing the bundle-to-transmit.

Transmission Finished: The UDPCL entity indicates to the BPA when a bundle has been fully transmitted. This is not a positive indication that any next-hop receiver has either received or processed the transfer.

Cancel Transmission: The UDPCL allows a BPA to cancel a long-running transmission, likely associated with an identified transfer (because unframed transfers occupy only a single UDPCL packet). Because UDPCL transfers are unacknowledged, canceling a transmission means simply stop sending UDPCL messages associated with that transmission. One outcome of a BPA canceling a transmission before all of its associated messages have been sent is the receiver simply timing out and failing the reception (see Section 3.6.2). Another possible outcome, if the transmission is short enough, is that all associated messages have already been sent by that time and the entity does nothing.

Reception Started: The UDPCL entity indicates to the BPA when a bundle transfer has begun, which can include information about the total size of a segmented transfer and the network parameters associated with the reception (_e.g._, IP addresses, UDP port numbers, local interface identifier).

Reception Success: The UDPCL entity indicates to the BPA when a bundle has been fully transferred from a peer entity. The transmitter of each transfer is identified by network parameters associated with the reception.

Reception Failure: The UDPCL entity indicates to the BPA on certain reasons for reception failure, notably upon an unfinished transfer timeout (see Section 3.6.2).

Attempt DTLS Session: The UDPCL allows a BPA to preemptively attempt to establish a DTLS session with a peer entity (see Section 3.5.5 and Section 3.9). Each session attempt can send a different set of session negotiation parameters as directed by the BPA.

Close DTLS Session: The UDPCL allows a BPA to preemptively close an established DTLS session with a peer entity. The closure request is on a per-session basis.

DTLS Session State Changed: The UDPCL entity indicates to the BPA when a DTLS session state changes. The possible DTLS session states are defined in [RFC9147].

Begin Sender Listen: The UDPCL allows a BPA to indicate when packets on a particular address-and-port is listened for. This includes configuring the local network stack to do the listening as well as optionally sending Sender Listen (Section 3.5.3) items to indicate this state.

End Sender Listen: The UDPCL allows a BPA to indicate when packets on a particular address-and-port are no longer be accepted.

Sender Listen Received: The UDPCL entity indicates to the BPA when a Sender Listen extension has been received from a peer. The Sender Node ID, if present, is part of this indication.

Both path characterization and congestion control are seen as behaviors of a UDPCL entity not directly controlled or observed by its BPA. These both operate in support of normal operations between UDP/IP endpoints transparently to the BPA.

2.2. PKIX Environments and CA Policy

This specification gives requirements about how to use PKIX certificates issued by a Certificate Authority (CA), but does not define any mechanisms for how those certificates come to be. The UDPCL uses the exact same mechanisms and makes the same assumptions as TCPCL in Section 3.4 of [RFC9174].

2.3. UDP Conversation Stability

Unlike the earlier experimental UDPCL definition [RFC7122], the requirements in this document ensure that traffic between two peers with stable IP addresses also use stable UDP port numbers. This enables the UDPCL to be more compatible with network address translation (NAT) middleboxes and, where necessary, enables the UDPCL to be used along with protocols such as Session Traversal Utilities for NAT (STUN) [RFC8489] and Interactive Connectivity Establishment (ICE) [RFC8445].

There is a trade-off between stability of UDP conversations and a possible need to protect from off-path data injection, as described in Section 5.8, so the conversation long-term stability (in Section 3.1 and Section 3.2) is recommended while specific cases of short-term consistency Section 3.5.2, Section 3.5.3, and Section 3.5.6 are required.

2.4. Path Characterization Policies

The earlier experimental UDPCL definition [RFC7122] assumed that BP node operators had deep visibility into the IP networks over which bundles would be transferred, specifically that the IP paths between UDPCL sender and receiver had well-characterized parameters for: path maximum transmit unit (PMTU), one-way delay time, one-way throughput limit, and packet loss statistics (both due to congestion and to link errors).

In situations where node operators do not have prior information about those parameters, this version of the UDPCL supports mechanisms to measure or estimate them. These mechanisms are defined in detail under Section 3.7.

2.5. Fragmentation Policies

It is a implementation matter for a transmitting entity to determine the PMTU, from which a target upper-bound UDPCL packet size is derived by subtracting IP- and UDP-layer header sizes.

The priority order of fragmentation is the following:

1. When possible, bundles too large to fit in one PMTU-sized packet MAY be fragmented at the BP layer. Bundle payload fragmentation does not help a large bundle if extension blocks are a major contributor to bundle size, so in some circumstances BP layer fragmentation will not reduce the bundle size sufficiently. It is outside the scope of UDPCL to manage BPA fragmentation policies; encoded bundles are received from the BPA either already fragmented or not.

2. When a PMTU is available to a peer, bundles too large to fit in one PMTU-sized packet SHALL be segmented as a UDPCL transfer (see Section 3.6). Segmentation at this level treats bundle transfers as opaque data, so it is independent of bundle block sizes or counts. When a PMTU is available, IPv4 packets sent by a UDPCL entity SHOULD have the DF bit set to allow detection of PMTU issues.

3. All IPv6 packets and all IPv4 packets which do not have the DF bit set which are larger than the link MTU will be fragmented by the transmitting entity to fit within one link MTU. Because of the issues listed in Section 3.2 of [RFC8085] and [RFC8900], it is best to avoid IP fragmentation as much as possible.

A UDPCL entity SHOULD NOT proactively drop an outgoing transfer due to datagram size. If intermediate network nodes drop IP packets it is an implementation matter to receive network feedback (e.g. ICMP _Fragmentation Needed_ [RFC792] for IPv4 or ICMPv6 _Packet Too Big_ of Section 3.2 of [RFC4443] for IPv6). The UDPCL itself provides no transfer acknowledgement mechanism or other protocol-level reception feedback other than the optional Packetization Layer Path MTU Discovery procedure, which is why an accurate PMTU for a peer is critical information.

2.6. Error Checking Policies

The core Bundle Protocol specification assumes that bundles are transferring over an erasure channel, i.e., a channel that either delivers packets correctly or not at all.

A UDP transmitter SHALL NOT disable UDP checksums. A UDP receiver SHALL NOT disable the checking of received UDP checksums.

Even when UDP checksums are enabled, a small probability of UDP packet corruption remains. In some environments, it could be acceptable for a BPA to occasionally receive corrupted input for some blocks. In general, however, a BPA is recommended to ensure the a bundle’s blocks are covered by a CRC so that next-hop receivers can verify the block contents.

2.7. Congestion Control Policies

The applications using UDPCL for bundle transport SHALL conform to the congestion control requirements of Section 3.1 of [RFC8085]. The application SHALL either perform active congestion control of bundles, behave as the Low Data-Volume application as defined in Section 3.1.3 of [RFC8085], or enable congestion control in the UDPCL layer.

When a UDPCL entity has bidirectional IP reachability with a peer, the Explicit Congestion Notification (ECN) marking and feedback mechanism of Section 3.8 can be used to provide Active Queue Management (AQM) of UDP traffic (and thus BP traffic) to the peer providing feedback. It is an implementation matter to ensure that traffic patterns which require UDPCL congestion control are paired with a UDPCL entity capable of AQM as well as IP routers supporting ECN and UDPCL peers which provide requisite ECN feedback.

When nodes have bidirectional BP transfer capability, the bundle deletion reason code "traffic pared" can be used by a receiving agent to signal to the bundle source application that throttling of bundles along that BP path needs to occur.

3. UDPCL Operation

This section defines the UDPCL protocol and its interactions with under-layers (IP and UDP) and over-layers (BP), as illustrated in Figure 1. The section is organized from the network layer up toward the BP layer. It also discusses behavior within the UDPCL layer, which is illustrated in Figure 3.

```
+--------------------------------------------+
| Identified Transfer | Extension Signaling | <- Sequencing /
+--------------------------------------------+      segmentation
 \                                          /
  ------                  -----------------
        \                /
+--------------------------------------------+
| Bundle | Extension Map |  ...   | Padding | <- Messaging
+--------------------------------------------+
|                UDPCL Packet               | <- Packetization
+--------------------------------------------+
```

Figure 3: Breakdown of sub-layers within the UDPCL

3.1. IP Header

The earlier experimental UDPCL definition [RFC7122] did not include guidance on IP addressing, interface sourcing, or potential use of multicast, though the DTN architecture [RFC4838] explicitly includes multicast and anycast as expected network modes.

For each forwarding transfer the BPA determines the mapping from destination EID to next-hop CL parameters, including next-hop destination address and source interface. Some EIDs represent singleton destinations and others non-singleton destinations as defined in Section 4.2.5.1 of [RFC9171]. The singleton-ness of an EID does not necessarily correspond with the unicast-ness of its forwarding, as some bundle routing schemes involve attempting multiple parallel paths to a singleton endpoint and some involve forwarding non-singleton-destination traffic to known individual BP neighbors along unicast paths.

For unicast transfers to a single node, the destination address SHALL be a non-multicast IPv4 or IPv6 address (which includes link-local addresses). For unicast transfers, the source interface address MAY be supplied by the BPA or otherwise determined by the operating system IP routing. When performing unicast transfers, a UDPCL entity SHOULD require DTLS use (see Section 3.9) or restrict the network to one protected by IPsec or some other under-layer security mechanism (e.g., a virtual private network).

For multicast transfers to any number of nodes, the destination address SHALL be a multicast IPv4 [IANA-IPv4-MCAST] or IPv6 [IANA-IPv6-MCAST] address. The well-known multicast addresses defined in Section 6.1 SHALL be used as a default in the absence of any network-specific configuration. For multicast transfers, the source interface address MUST be supplied by the BPA rather than inferred by the UDPCL entity. For multicast transfers the UDPCL does not define any security mechanism.

When supported by an implementation, the UDPCL SHOULD accept and provide information on the local interface from which a transmission is sent or to which a reception was received. For IPv6 this corresponds with the IPV6_PKTINFO ancillary data API of Section 6.1 of [RFC3542]; for IPv4 this is an implementation-specific detail supported by IP_PKTINFO ancillary data on some operating systems. Identifying an interface for transmission or reception allows a BPA to provide fine-grained control and visibility when UDPCL is used for IP multicast or when link-local addressing is in use.

The IPv4 header bit _don’t fragment_ (DF) is used as discussed in Section 2.5.

The IPv4 and IPv6 header field for ECN marking is used as discussed in Section 2.7 and Section 3.8.

3.2. UDP Header

UDP port number 4556 has been assigned by IANA [IANA-PORTS] as the Registered Port number for the UDPCL and SHALL be used as a default for both source and destination. Other source or destination port numbers MAY be used per local configuration. Determining a passive entity’s destination port number (if different from the registered UDPCL port number) is up to the implementation.

If the default source port is not used, typically an operating system assigned number in the UDP Ephemeral range (49152-65535) is used. For repeated messaging to the same destination address-and-port, the active entity SHOULD reuse the same source address-and-port. For feedback messaging, the passive entity SHALL respond to the same address-and-port that solicited the feedback and use the same source address-and-port which was listed on. Reusing source address-and-port allows simplifies network monitoring and analysis and also enables bi-directional messaging as defined in Section 3.5.3, Section 3.7.1, and Section 3.8.

3.3. UDPCL Packets

The lowest layer of UDPCL communication are UDPCL packets used as the payload of a UDP datagram. To exchange UDPCL data, an active entity SHALL transmit a UDP datagram to a listening passive entity in accordance with [RFC0768], typically by using the services provided by the operating system.

Each UDPCL packet SHALL contain one or more UDPCL message as defined in Section 3.4. Each type of message defines additional restrictions on how it can be used in a packet.

The following are special cases of UDPCL packet uses.

Unframed Transfer:
An unframed transfer packet SHALL consist of a single encoded BPv6 or BPv7 bundle with no padding. This provides backward compatibility with the experimental UDPCL [RFC7122] and a allows a trivial use of UDPCL which is just embedding an encoded bundle PDU in a UDP datagram.

This case has caveats to its use, including the limitation that the bundle PDU be small enough that its containing IP packet fit into the PMTU size. This version of UDPCL provides the Identified Transfer (Section 3.6) mechanism to overcome this, and other limitations, of the unframed transfer.

Keepalive:
A keepalive packet SHALL consist of exactly four octets of padding with no preceding message. This behavior maintains backward compatibility with the experimental UDPCL [RFC7122].

DTLS Record:
In addition to the UDPCL-specific messaging, immediately after a DTLS Initiation (see Section 3.5.5) the DTLS handshake sequence will begin. Packets with a leading octet value of 0x16 SHALL be treated as a DTLS handshake record in accordance with Section 4.1 of [RFC9147].

If the datagram with the DTLS Initiation extension is not received by an entity, the entity SHOULD still detect the DTLS handshake records and start the handshake sequence at that point. Packets with a leading octet value of 0x14--0x1A or 0x20--0x3F SHALL be treated as a DTLS sequencing failure; DTLS non-handshake records SHOULD never be seen by the UDPCL messaging layer.

3.3.1. Redundant Transmission

Because the UDPCL uses UDP as its transmission layer, an entity with information about the loss characteristics of the path to a peer entity can intentionally perform duplicate transmissions to improve the chances that the packet, message, or whole transfer will arrive at the receiver. It is an implementation matter to determine when to perform redundant transmission, and could be based on the interface, peer network or address, or properties of the UDPCL packet itself.

The number of duplicate transmissions for a packet is referred to as its Redundancy Factor. A Redundancy Factor of one means to perform no duplicate transmission. A larger Redundancy Factor will result in more network resources used (and possibly wasted).

The time interval between original and each duplicate transmission is referred to as its Redundancy Delay. A larger Redundancy Delay will result in more entity resources used keeping packet contents in memory for retransmission. To avoid a receiver mishandling redundant packets containing Transfer items (see Section 3.6.2) the Redundancy Delay SHOULD be no longer than the receiver’s transfer timeout, if known, or one minute (60 seconds).

For paths where losses over time are bursty, a low redundancy factor but high redundancy delay will ensure that the duplicated packets are transmitted over a longer span of time and (hopefully) do not have a correlated probability of loss.

To avoid unnecessary processing on the receiver’s BPA, when redundant transmission is used the identified transfer (Section 3.6) mechanism SHALL be used even for single-segment transfers. This allows the UDPCL entity can discard any duplicate receptions before they reach the BPA. If redundant transmission is used with an unframed transfer the UDPCL has no ability to discard any duplicate receptions and all duplicates will be seen by the receiving BPA.

3.4. UDPCL Messages

The middle layer of UDPCL communication are unframed, but self-delimited, messages. Specific message types MAY be concatenated together into a single packet, each message type indicates any restrictions on how it can be used within a packet.

For backward compatibility with the experimental UDPCL [RFC7122], this version of UDPCL has no explicit message type identifier. The message type is inferred by the inspecting the data contents according to the following rules:

Bundle: When sent outside of an explicit identified transfer (Section 3.6) an encoded bundle is treated as a separate message type. This is for compatibility with the experimental UDPCL [RFC7122] but comes at the expense of not being able to de-duplicate any redundant transfers (Section 3.3.1).

BPv6 Bundle: All encoded BP version 6 bundles begin with the version identifier octet 0x06 in accordance with [RFC5050]. A message with a leading octet value of 0x06 SHALL be treated as a BPv6 bundle. Multiple BPv6 Bundles SHOULD NOT be present in one UDPCL packet to maintain compatibility with the experimental UDPCL [RFC7122].

BPv7 Bundle: All encoded BP version 7 bundles begin with a CBOR array head in accordance with [RFC9171]. A message with a leading octet value indicating CBOR array (major type 4) SHALL be treated as a BPv7 bundle.

BPv7 bundles transmitted via UDPCL SHALL NOT include any leading CBOR tag. If the BPA provides bundles with such tags the transmitting UDPCL entity SHALL remove them.

Extension Map: All UDPCL extensions SHALL be contained in a CBOR map in accordance with the definitions of Section 3.5. The encoded Extension Map SHALL NOT have any CBOR tags. A message with a leading octet value indicating CBOR map (major type 5) SHALL be treated as an Extension Map.

Padding: Padding data SHALL be a sequence of octets all with value 0x00. A message with a leading octet value of 0x00 SHALL be treated as padding.

Padding is used to ensure a UDP datagram is exactly a desired size. Because padding has no intrinsic length indication, if present it SHALL be the last contents of any UDPCL packet. A receiving UDPCL entity SHALL ignore all padding, including any trailing non-zero octets.

A summary of how a receiving UDPCL entity can interpret the first octet of a UDPCL packet is listed in Table 1. When inspecting using CBOR major types, the range of values is caused by the CBOR head encoding [RFC8949] using only the high three bits for the major type.

Octet Value	Message Content	Message Extent
0x00	Padding	Remainder of UDPCL packet
0x06	BPv6 Bundle	Entire UDPCL packet (to be decoded by BPA)
0x14--0x1A or 0x20--0x3F	DTLS Record	Entire UDPCL packet
0x80--0x9F	BPv7 Bundle (CBOR array)	Entire UDPCL packet (to be decoded by BPA)
0xA0--0xBF	Extension Map (CBOR map)	End of map item, possibly followed by other map or padding
others	Unused by this specification	

Table 1: First-Octet Contents

3.5. UDPCL Extension Items

Extensions to UDPCL are encoded per-message in a single CBOR map as defined in Section 3.4. Each UDPCL extension item SHALL be identified by a unique Extension ID used as a key in the Extension Map. Extension ID values SHALL be an int item which fits within a signed 16-bit integer. Extension ID assignments are listed in Section 6.3.

Because of this structure, a single Extension Map can contain only zero or one of each Extension ID. Unless prohibited by particular extension type requirements, a single Extension Map MAY contain any combination of extension items. Receivers SHALL ignore extension items with unknown Extension ID and continue to process known extension items.

```
; Map structure requiring 16-bit int keys.
; CDDL cannot enforce type-specific requirements about other items
; being present (or not present) in the same map.
udpcl-ext-map = $udpcl-ext-map .within udpcl-ext-map-structure
$udpcl-ext-map /= {
    + $$udpcl-ext-item
}
udpcl-ext-map-structure = {
    + ext-key => any
}
ext-key = -32768 .. 32767
```

Figure 4: Extension Map structure CDDL

The following subsections define the initial UDPCL extension types.

3.5.1. Extension Support

This extension allows an entity to signal its support for receiving specific UDPCL extension types.

The Extension Support value SHALL be an array of items representing a range of extension type codes which are able to be handled by the entity sending the Extension Support in accordance with Section 3.5.1.1.

The minimum set of Extension Support SHOULD include values 1 through 8 inclusive, which are the extension types defined in this document. This minimum is represented, in the CBOR diagnostic notation, by [1, 7] and the largest valid range of support is represented by [-32768, 65535].

```
$$udpcl-ext-item //= (
    ; Range of extension key support
    1: int-range<ext-key,(0 .. 65535)>
)
```

Figure 5: Extension Support CDDL

This extension type is expected to be used along with Sender Listen for announcing presence to (potential) peer entities.

If an entity receives an extension type which it cannot handle, that entity MAY send an Extension Support item in response. The enveloping UDPCL packet SHALL use the source address-and-port that sent the unhandled extension type as the destination of the packet.

3.5.1.1. Integer Range Encoding

This subsection defines a general purpose structure for representing a non-empty range of disjoint, finite, integer intervals in a CDDL generic rule. This rule is is an array structure parameterized on two types: the value type is used for the least included value in the domain, and the width type is the maximum possible width of any interval in the domain.

The first item of the array is the least included value. Each other item is a width of of a finite interval, alternating between values included in the range and values excluded from the range. The last interval represented is always the included one. Each width is the difference between the largest value of the interval and the least value. Each subsequent interval has a least value one more than the largest value of the preceding interval.

```
int-range<value,width> = [
    least: value,
    * (
        ; width of included interval
        incl: width,
        ; width of excluded interval
        excl: width
    )
    ; width of final included interval
    incl: width
]
```

Figure 6: Integer Range CDDL

The interpretation of these parameters are shown in Figure 7, where the top of the diagram indicates values in the domain being covered by the range and the bottom indicates fields in the CBOR structure.

```
             _____(repeated)______
            /                     \
             Include       Exclude       Include
          min       max min       max min       max    value
        <..|.........|...|.........|...|.........|..>  space
           |         |   |         |   |         |
           |<------->|<->|<------->|<->|<------->|
          /   incl     1    excl     1    incl         encoded
       least  width         width         width        fields
```

Figure 7: Integer Range Encoding

The bounding cases are for use of this structure are a singleton, a range which includes only single value val as the notional array of

[val, 0]

and the universe, a range which includes all values in the domain from least to most as the notional array of

[least, most - least]

3.5.2. Transfer

This extension item allows identification and segmentation of bundle transfers as defined in Section 3.6.

The Transfer value SHALL be an untagged CBOR array of two or four items. The items are defined in the following order:

Transfer ID: This field SHALL be a uint item. This value is used to correlate multiple segments from the same source address-and-port. More specific requirements on the uniqueness of this field are given in Section 3.6.1.

Total Length: This optional field SHALL be a uint item, which is used to indicate the total length (in octets) of the transfer. If multiple Transfer items for the same Transfer ID are received with differing Total Length values, the receiver SHALL treat the transfer as being malformed and refuse to handle any further segments associated with the transfer. If a transfer requires only a single segment, this field SHALL be omitted and the total length treated as the size of the Segment Data.

Segment Offset: This optional field SHALL be a uint item, which is used to indicate the offset (in octets) into the transfer for the start of this segment. If a transfer requires only a single segment, this field SHALL be omitted and the segment offset treated as zero.

Segment Data: This field SHALL be a bstr item, in which the segment data is contained. The bstr item itself indicates the length of the segment data. Per requirements in Section 3.6.2 the size of the segment data will be limited so that the entire UDPCL packet (and its encapsulating UDP/IP packet) fits within a PMTU size limit.

```
$$udpcl-ext-item //= (
    2: [
        transfer-id: uint,
        ?(
            total-length: uint,
            segment-offset: uint,
        ),
        segment-data: bstr,
    ]
)
```

Figure 8: Transfer CDDL

Multiple Transfer items MAY be present in the same UDPCL packet by concatenating multiple Extension Map items into the packet.

A transmitting entity SHALL use the same source and destination address-and-port for all UDPCL packets comprising a single segmented transfer. This is more strict than the requirements of Section 3.2 to ensure consistent reassembly behavior in Section 3.6.2.

3.5.3. Sender Listen

This extension item indicates that the transmitter is listening for UDPCL packets on the source address-and-port used to transmit the message containing this extension item. This is different from simply listening on a UDP port (either the default or any other) when the entity is behind a NAT or firewall which will not allow unsolicited UDP/IP datagrams. Although the packet containing this extension is not retransmitted, the time interval is finite and the extension is sent repeatedly while the transmitter continues to listen for packets. There is no positive indication that packets are no longer accepted; the Sender Listen just stops being transmitted.

The Sender Listen value SHALL be an untagged uint value representing the interval of time (in milliseconds) that the entity is willing to accept UDPCL packets on the source address-and-port used for the associated transmitted message. After transmitting a Sender Listen, the entity SHALL listen for and accept datagrams on the source address-and-port used for the associated transmitted message. As long as the entity is still willing to accept packets, at the end of one accept interval the entity SHALL transmit another Sender Listen item. This repetition continues until the entity is no longer willing to listen for packets.

A receiving entity SHOULD treat a peer as no longer listening after an implementation-defined timeout since the last received Sender Listen item. A RECOMMENDED Sender Listen timeout is three (3) times the associated time duration; this allows a single dropped datagram to not interrupt a continuous sequence.

```
$$udpcl-ext-item //= (
    3: time-duration,
)
time-duration = uint
```

Figure 9: Sender Listen CDDL

Unlike the generic source port requirement in Section 3.2, when repeated Sender Listen are transmitted in a sequence a consistent source address-and-port SHALL be used.

The Sender Listen interval SHOULD be no shorter than 1 second and no longer than 60 seconds.

An entity SHOULD include an Extension Support and Sender Node ID item along with a Sender Listen item if the conditions of those extension types are met. An entity MAY include any other extension type along with a Sender Listen item. An entity SHALL NOT transmit a Sender Listen item before or along with a DTLS Initiate if DTLS is desired for a conversation.

This extension is not a neighbor discovery mechanism and does not indicate an entity listening generally on a particular UDP port. Sender Listen applies only to UDP datagrams from the the source address-and-port, and does not express information about any other addresses or ports used by the sending node.

3.5.4. Sender Node ID

This extension item indicates the Node ID of the transmitter. For DTLS-secured sessions (see Section 3.9.4) this extension can be used to disambiguate an end-entity certificate which has multiple NODE-ID values.

The Sender Node ID value SHALL be an untagged tstr value containing a Node ID. Every Node ID SHALL be consistent with the requirements of a URI [RFC3986] and the URI schemes of the IANA "Bundle Protocol URI Scheme Type" registry [IANA-BUNDLE].

```
$$udpcl-ext-item //= (
    4: nodeid,
)
nodeid = tstr
```

Figure 10: Sender Node ID CDDL

An entity SHOULD NOT include a Sender Node ID item if a DTLS session has already been established and the presented end-entity certificate contains a single NODE-ID. In this case there is no ambiguity about which Node ID is identified by the certificate. Otherwise, after DTLS session establishment and before initiating any transfers an entity SHALL include a Sender Node ID to allow its peer to authenticate its desired identity.

If an entity receives a peer Node ID which is not authenticated (by the procedure of Section 3.9.4) that Node ID SHOULD NOT be used by a BPA for any discovery or routing functions. Trusting an unauthenticated Node ID can lead to the threat described in Section 5.7.

3.5.5. DTLS Initiation (STARTTLS)

This extension item indicates that the transmitter is about to begin a DTLS handshake sequence in accordance with Section 3.9.

The DTLS Initiation value SHALL be an untagged null value. There are no DTLS parameters actually transmitted as part of this extension, it only serves to indicate to the recipient that the next datagram will be a DTLS ClientHello. Although the datagram containing this extension is not retransmitted, the DTLS handshake itself will retransmit ClientHello messages until confirmation is received.

```
$$udpcl-ext-item //= (
    5: null
)
```

Figure 11: DTLS Initiation CDDL

If the entity is configured to enable exchanging messages according to DTLS 1.3 [RFC9147] or any successors which are compatible with that DTLS ClientHello, the first message in any sequence to a unicast recipient SHALL be an Extension Map with the DTLS Initiation item. The RECOMMENDED policy is to enable DTLS for all unicast recipients, even if security policy does not allow or require authentication. This follows the Opportunistic Security model [RFC7435], though an active attacker could interfere with the exchange in such cases (see Section 5.3).

The Extension Map containing a DTLS Initiation item SHALL NOT contain any other items. A DTLS Initiation item SHALL NOT be present in any message transmitted within a DTLS session. A receiver of a DTLS Initiation item within a DTLS session SHALL ignore it. Between transmitting a DTLS Initiation item and finishing a DTLS handshake (either success or failure) an entity SHALL NOT transmit any other UDP datagrams in that same conversation.

3.5.6. Peer Probe

This extension item provides the Probe Packet functionality of Section 4.1 of [RFC8899] in order to support path MTU discovery mechanism of Section 3.7.1.

The Peer Probe value SHALL be an array of three items. The items are defined in the following order:

Nonce: The first item SHALL be a uint nonce for correlating with Peer Confirmation extensions. It is RECOMMENDED to use small nonce values to keep encoded sizes short.

Sequence Number: The second item SHALL be a uint sequence number for the specific Probe packet. It is RECOMMENDED that sequence numbers begin at zero and increment by one for each subsequent probe using the same nonce value.

Confirmation Delay: The third item SHALL be a uint representing the desired confirmation delay time (in milliseconds).

```
$$udpcl-ext-item //= (
    6: [
        probe-nonce,
        probe-seqno,
        confirm-delay
    ],
)
; A correlator nonce for confirmations
probe-nonce = uint
probe-seqno = uint
confirm-delay = uint
```

Figure 12: Peer Probe CDDL

When the Peer Probe item is present in a UDPCL packet, the packet SHALL contain a single Extension Map followed by padding to a specific total size. A transmitting entity SHALL use the same source and destination address-and-port for all UDPCL packets containing a Peer Probe with the same nonce value. This is more strict than the requirements of Section 3.2 to ensure consistent confirmation behavior defined below.

The nonce value SHALL be used to correlate multiple Probe packets into a single sequence. If an entity receives multiple unique nonce values from the same source address-and-port and the number of unique nonce values exceeds an upper limit, the receiving entity SHALL ignore the Probe. The upper limit is implementation defined but SHOULD be no less than one unique nonce at-a-time.

If the confirmation delay time is above an upper limit the receiving entity SHALL ignore the Probe. The upper limit is implementation defined but SHOULD be no shorter than one minute (60 seconds).

If an entity receives multiple instances of Peer Probe with the same nonce and sequence number values, the duplicates SHALL NOT be ignored. This makes the last duplicate transmission of the last probe the basis of Peer Confirmation timing.

After an entity receives and validates a packet with a Peer Probe extension item it SHALL wait for the indicated confirmation delay time before sending a Peer Confirmation item. The enveloping UDPCL packet SHALL use the source address-and-port that sent the last Probe item as the destination of the Confirmation. Once the Peer Confirmation packet is sent, the entity SHALL remove state associated with the Peer Probe items being confirmed. This bounds the state on a confirming entity and makes each Peer Confirmation with the same nonce value incremental and not additive (_i.e._, there will never be two Confirmations with the same nonce covering the same sequence numbers, except for duplicate transmissions).

3.5.7. Peer Confirmation

This extension item provides the Confirmation functionality for Section 4.2 of [RFC8899] in order to support path MTU discovery mechanism of Section 3.7.1.

The Peer Confirmation value SHALL be an array of two items. The items are defined in the following order:

Nonce: The first item SHALL be a uint nonce for correlating with Peer Probe items.

Seen Sequence Range: The second item SHALL be an array of items representing the range of sequence numbers of received Probe packets with the same nonce value in accordance with Section 3.5.1.1. This is an identical structure to the Extension Support extension encoding with different domain of integer values to cover sequence numbers.

```
$$udpcl-ext-item //= (
    7: [
        ; Correlator with all probes
        probe-nonce,
        ; Range of sequence numbers seen
        seen-range: int-range<probe-seqno,uint>
    ],
)
```

Figure 13: Peer Confirmation CDDL

When an entity receives a packet with a Peer Confirmation extension item it SHALL wait for the same confirmation delay used in each Probe item for multiple Confirmation items with the same nonce. Upon reception of all Confirmation items within the waiting delay, the entity SHALL treat the largest value of the seen range as the PMTU to the corresponding peer entity.

For example, if an entity has received three probes using nonce value 123 with sequence numbers 2, 4, 5, and 6 the Peer Confirmation will contain the same nonce and a seen range of 2..2 included, 3..3 excluded, 4..6 included. This will be encoded as the extension map value [123,[2,0, 0,0, 0,2]].

3.5.8. ECN Counts

This extension item provides the ECN feedback functionality for Section 4.2 of [RFC9331] in order to support congestion control feedback of Section 3.8.

The ECN Counts value SHALL be an array of three items. The items are defined in the following order:

ECT(0) Count: The first item SHALL be a uint count of packets marked with ECT(0) received from the peer, wrapped to 32-bits.

ECT(1) Count: The second item SHALL be a uint count of packets marked with ECT(1) received from the peer, wrapped to 32-bits.

CE Count: The third item SHALL be a uint count of packets marked with CE received from the peer, wrapped to 32-bits.

```
$$udpcl-ext-item //= (
    8: [
        ect0: ecn-count,
        ect1: ecn-count,
        ce: ecn-count
    ],
)
; Wraps around at 32-bits
ecn-count = uint .size 4
```

Figure 14: ECN Counts CDDL

Because this extension type will only be used for feedback to a specific transmitting peer, it SHALL NOT be present in any UDPCL packet with an IP multicast destination.

3.6. Identified Transfers

This version of UDPCL supports explicit identification of bundle transfers and segmentation of bundles larger than the PMTU would otherwise allow. Policies related to segmentation at or fragmentation above, or below the UDPCL layer are defined in Section 2.5. The entire segmented bundle is referred to as a Transfer and individual segments of a transfer are encoded as Transfer extension items in accordance with Section 3.5.2.

This mechanism also allows a bundle transfer to be transmitted along with additional extension items, which the unframed bundle-in-datagram data does not. This specification does not define any extension items which augment an associated transfer.

3.6.1. Bundle Transfer ID

Each Transfer item contains a Transfer ID which is used to correlate messages for a single bundle transfer from a single source. A Transfer ID does not attempt to address uniqueness of the bundle data itself and has no relation to concepts such as bundle fragmentation. Each invocation of UDPCL by the BPA requesting transmission of a bundle PDU (fragmentary or otherwise), can cause the initiation of a single UDPCL transfer.

Because UDPCL operation is connectionless, Transfer IDs from each transmitting entity SHOULD be unique for the operating duration of the entity. In practice, the ID needs only be unique for the longest receiver de-duplication and reassembly time window; but because that information is not part of the protocol there is no way for an transmitting entity to know the reassembly time window of any receiver (see Section 3.6.2). When there are bidirectional bundle transfers between UDPCL entities, an entity SHOULD NOT rely on any relation between Transfer IDs originating from each side of the conversation.

Although there is not a strict requirement for Transfer ID initial values or ordering (see Section 5.12), in the absence of any other mechanism for generating Transfer IDs an entity SHALL use the following algorithm: the initial Transfer ID from each entity is zero; subsequent Transfer ID values are incremented from the prior Transfer ID value by one; upon exhaustion of the entire 64-bit Transfer ID space, the subsequent Transfer ID value is zero.

A transmitting entity MAY use a smaller range of valid Transfer ID values to reduce encoded size, or partitions of ranges to delegate transfer logic. A receiving entity SHALL NOT assume or rely on a specific limited range of Transfer ID when correlating messages.

3.6.2. Segmentation and Reassembly

The full data content of a transfer SHALL be an unframed (BPv6 or BPv7) bundle PDU as defined in Section 3.4. A receiving entity SHALL discard any reassembled transfer which does not properly contain a bundle, based on the initial octets of the PDU, as defined in Section 3.4.

The size of each Segment Data within a transfer SHALL be chosen so that the entire encoded Extension Map and the UDPCL packet in which it is contained fits within a transfer maximum transmit unit (TMTU) size. The TMTU needs to be supplied to the transmitting entity either via direct configuration, introspecting the operating system for PMTU information associated with a destination, or can be matched to a PMTU estimated using the procedure in Section 3.7.1. It is an implementation matter to determine what TMTU to apply to each segmented transfer, including the option to deliberately choose a TMTU larger than the associated PMTU which will cause IP fragmentation (see Section 2.5).

A transmitting entity can produce a Transfer with a single segment (i.e., a Segment Data size identical to the Total Length) in which case the redundant fields Total Length and Segment Offset SHALL be omitted. A transmitting entity SHALL NOT produce Transfer segments with overlapping span. A transmitting entity SHOULD transmit Transfer segments in order of Segment Offset; this makes the behavior deterministic.

A receiving entity SHALL maintain transfer reassembly state based on the tuple of packet source address-and-port and the Transfer ID created by that source. This relies on the stability of source port numbers at least for the duration of a single transfer, in accordance with Section 3.5.2. See Section 2.3 for options and implications of source port number use.

A single UDPCL entity MAY transmit packets from multiple source address-and-port combinations but these will be treated as separate scopes for the purposes of Transfer ID correlation and reassembly. A receiving entity SHALL NOT attempt to correlate peer identity or state across multiple source address-and-port combinations.

Because of the nature of UDP transport, there is no guaranteed order or timing of received Transfer items. A receiving entity SHALL NOT assume any specific reception order of segments for the same transfer. A receiving entity SHALL consider a transport as finished when Segment Data has been received which fully covers the Total Length of the transfer.

A receiving entity SHALL discard any Transfer item containing different CBOR types than defined in this document. A receiving entity SHALL discard any Transfer item containing a segment with a span overlapping any other in the reassembly state. That includes overlapping span because of a redundant transmission (see Section 3.3.1). Because there is no feedback indication at the UDPCL layer, a transmitter has no indication when a Transfer item is discarded by the receiver.

A receiving entity SHOULD discard unfinished reassembly state after an implementation-defined timeout since the last received segment. This timeout is purely receiver-side and represents the maximum allowed time between sequential received datagrams (in any order), which will be short if the datagrams take a similar network path. A receiving entity SHALL NOT discard reassembly state upon successful completion of a transfer, in case the transmitting entity is sending redundant packets (Section 3.3.1) which need to be discarded. Entities SHOULD choose a transfer timeout interval no longer than one minute (60 seconds). Discarding an unfinished transfer causes no indication to the transmitting entity, but does indicate this to the receiving BPA.

3.7. Path Characterization Procedures

The procedures defined in this section allow pairs of UDPCL entities to characterize IP path parameters between them when those parameters are either unknown or for troubleshooting when actual network performance deviates from expectations.

These procedures not mandated to be used operationally, but when the associated characterization is desired these are the procedures which SHALL be used.

3.7.1. Packetization Layer Path MTU Discovery

This version of UDPCL is compatible with the requirements of Packetization Layer Path MTU Discovery (PLPMTUD) from [RFC8899]. Specifically the UDPCL supports the requirements of Section 6.1 of [RFC8899] by using the Peer Probe extension item to perform the probe behavior and Peer Confirmation extension item for the confirmation behavior. Because the Peer Confirmation acknowledges a possibly large range of Peer Probe packets, this makes the UDPCL less sensitive to large or time-variable path delays than if each Probe was separately acknowledged.

The use of PLPMTUD is especially helpful when DTLS is used to secure UDPCL conversations, as the DTLS confidentiality mechanism and associated record layer reduces the maximum packet size (MPS) by an amount depending upon the negotiated ciphersuite.

In case one or more Peer Probe items are sent but no corresponding Peer Confirmation is ever received, this might indicate that the peer has not implemented or enabled PLPMTUD and SHOULD NOT be treated as an indication that the peer is unreachable.

3.7.2. Delay Time Estimation

The round-trip time (RTT) between two entities can be estimated by sending a sequence of small sized Peer Probe items, each in its own UDPCL packet, with a small or zero Confirmation Delay and measuring the time taken to receive the corresponding Peer Confirmation item.

The sequence MAY be as short as a single packet to get a point estimate of RTT. Otherwise, the inter-packet interval at the sender SHALL be larger than the Confirmation Delay contained in each item. This will cause each probe packet to correspond with a single related confirmation packet allowing either a time-series estimation or an average-over-time estimation of RTT.

Because there is no explicit time tagging of the acknowledgement, the one-way delay can be assumed to be one half of the RTT total.

3.7.3. Throughput and Loss Estimation

The one-way packet loss between two entities can be estimated by sending a sequence of equally-sized Peer Probe items, each in its own UDPCL packet, over some short time interval and observing how many of the probes have been seen in one or more corresponding Peer Confirmation item. For relatively low loss rates, a BER can be estimated by assuming each lost packet is caused by an equivalent to a single random error.

Similarly, a sequence of relatively large-sized (for the expected throughput order-of-magnitude) Peer Probe items, each in its own UDPCL packet, and the total volume indicated by corresponding Peer Confirmation items over some time interval can be used to estimate the one-way throughput limit.

For both cases above, the Probe packets can be combined with ECN markings as described in Section 3.8 to ensure that losses or throughput limits are not related to intermediate queuing congestion.

3.8. Explicit Congestion Notification

Data packet AQM can be performed using "classic" ECN signaling [RFC3168] or with more advanced methods such as the ECN Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S) [RFC9331]. These mechanisms allow for endpoints to implement an internet-friendly congestion control algorithm (CCA) with a small amount of application-level feedback of ECN data. The ECN Counts extension of Section 3.5.8 allows a UDPCL entity to both signal that it is ECN capable and to report ECN feedback in a way compatible with both classic ECN and L4S. It is up to specific algorithm requirements and details to determine how and when the ECN Counts needs to be used to support each CCA.

Because the information base defined below counts ECN values based on UDP conversation, the feedback occurs at the level of the entire conversation and not an individual message or bundle transfer within. When present, any CCA sending queues of a UDPCL entity SHALL be organized around the UDP conversation.

Within the UDPCL, congestion control is applied only to packets containing data to be transferred and not to packets containing purely control information. For the structures defined in this document, data packets include unframed transfers (Section 3.3) and those containing identified transfer (Section 3.5.2) and peer probe (Section 3.5.6) extensions. This separates UDPCL entity roles for a data flow within a conversation into a transmitting role and a receiving role, and each entity can act as a transmitter for some data as well as a receiver for other data. But the congestion control of those two flows is distinct and managed by which IP packets include ECN marking.

3.8.1. Transmitting Entity

When a transmitting entity has enabled a congestion control algorithm, it SHALL mark outgoing IP packets with an ECN field of either ECT(0) or ECT(1) codepoint if they contain data (from an unframed transfer, identified transfer, or Peer Probe). This marking is the signal that the transmitting entity is ECN capable and desires to receive ECN feedback.

There is no explicit synchronization or correlation of ECN marking to ECN feedback. The feedback counters maintained by the receiver and sent in the ECN Counts extension are an accumulation of all received ECN markings within that conversation. A transmitting entity MAY use the procedure of Section 3.7.2 to estimate RTT as needed to support CCA functions. There is no restriction in the UDPCL about interleaving packets for data transfer and path characterization.

Each state represented by the ECN Counts extension is independent and supersedes earlier state, meaning there is no need for the transmitting entity to maintain a history of ECN Counts for its half of a conversation. Because the fields of ECN Counts increase monotonically (modulo the 32-bit wrap-around) instances of that extension can be ordered and earlier ECN Counts state SHOULD be discarded for a conversation. The reception of an ECN Counts extension does not need to trigger or synchronize with any CCA activity. For example, a CCA can update its state at regular intervals (_e.g._, based on estimated RTT) and update based on whatever was the latest ECN Counts for its half of a conversation.

3.8.2. Receiving Entity

When a receiving entity has enabled ECN feedback and receives a UDPCL packet with a unicast IP destination and an ECN field other than the _Not-ECT_ codepoint, the ECN field of the IP packet SHALL be used to update the entry of the ECN feedback information table corresponding to its UDP conversation. When a new entry is needed in the ECN feedback information table, the counts SHALL be initialized to zero.

When a receiving entity has ECN feedback for a peer, it will send ECN Counts extension items to that peer based on CCA-specific logic. The ECN Counts extension MAY be sent periodically by a timer, when count values are increased above some threshold, or based on other extension items received by the entity. A receiving entity SHOULD suppress sending repeated, identical ECN Counts after an upper limit of duplicates has been sent. The upper limit is implementation defined and can depend on expected loss characteristics of the path back to the transmitting entity.

If the ECN feedback is updated because of ECN marking from a Peer Probe, the receiving entity SHOULD combine any corresponding Peer Confirmation with an ECN Counts in the same extension map. This allows some level of implicit synchronization between the packets from the transmitting entity and the ECN feedback.

The information base for ECN feedback information is a logical table containing the columns of Table 2.

Name	Description
UDP Conversation	This is the four-tuple of address-and-port for the remote and local UDP endpoints.
ECT(0) Count	This is the total number of IP packets received in the UDP Conversation with the ECN field containing the ECT(0) codepoint.
ECT(1) Count	This is the total number of IP packets received in the UDP Conversation with the ECN field containing the ECT(0) codepoint.
CE Count	This is the total number of IP packets received in the UDP Conversation with the ECN field containing the CE codepoint.

Table 2: ECN Feedback Information Columns

3.9. UDPCL Security

This version of the UDPCL supports establishing a DTLS session within an existing UDP conversation. When DTLS is used within the UDPCL it affects the entire conversation. There is no concept of a plaintext message being sent in a conversation after a DTLS session is established.

Once established, the lifetime of a DTLS session SHALL be bound by the DTLS session ticket lifetime or either peer sending a Closure Alert record.

Subsequent DTLS session attempts to the same passive entity MAY attempt to use the DTLS session resumption feature. There is no guarantee that the passive entity will accept the request to resume a DTLS session, and the active entity cannot assume any resumption outcome.

3.9.1. Entity Identification

The UDPCL uses DTLS for certificate exchange in both directions to identify each entity and to allow each entity to authenticate its peer. Each certificate can potentially identify multiple entities and there is no problem using such a certificate as long as the identifiers are sufficient to meet authentication policy (as described in later sections) for the entity which presents it.

The types and priorities of identities used by DTLS in UDPCL is the same as those for TLS in TCPCL as defined in Section 4.4.1 of [RFC9174].

3.9.2. Certificate Profile for UDPCL

All end-entity certificates used by a UDPCL entity SHALL conform to the profile defined in Section 4.4.2 of [RFC9174] and any updates to that profile.

3.9.3. DTLS Handshake

The signaling for DTLS Initiation is described in Section 3.5.5. After sending or receiving an Extension Map containing a DTLS Initiation item, an entity SHALL begin the DTLS handshake procedure of Section 5 of [RFC9147]. By convention, this protocol uses the entity which sent the DTLS Initiation (the active peer) as the "client" role of the DTLS handshake request.

Upon receiving an unexpected ClientHello record outside of a DTLS session, an entity SHALL begin the DTLS handshake procedure as if a DTLS Initiation had been received. This allows recovering from a dropped packet containing DTLS Initiation.

3.9.4. DTLS Authentication

The function and mechanism of DTLS authentication in UDPCL is the same as for TLS in TCPCL as defined in Section 4.4.4 of [RFC9174], with the exception that Node ID Authentication is based on an optional Sender Node ID extension (see Section 3.5.4) used to disambiguate when an end-entity certificate contains multiple NODE-ID values.

3.9.5. Policy Recommendations

The policy recommendations given here are are the same as those for TCPCL in Section 4.4.5 of [RFC9174]. They are restated in this document for clarity.

A RECOMMENDED security policy is to enable the use of OCSP checking during DTLS handshake. A RECOMMENDED security policy is that if an Extended Key Usage is present that it needs to contain the id-kp-bundleSecurity value [IANA-SMI] to be usable with UDPCL security. A RECOMMENDED security policy is to require a validated NODE-ID and to ignore any network-level DNS-ID or IPADDR-ID.

This policy relies on and informs the certificate requirements in Section 3.9.2. This policy assumes that a DTN-aware CA (see Section 2.2) will only issue a certificate for a Node ID when it has verified that the private key holder actually controls the DTN node; this is needed to avoid the threat identified in Section 5.7. This policy requires that a certificate contain a NODE-ID and allows the certificate to also contain network-level identifiers. A tailored policy on a more controlled network could relax the requirement on Node ID validation and allow just network-level identifiers to authenticate a peer.

3.9.6. Example Secured and Bidirectional Transfers

This simple example shows a sequence of pre-transfer setup followed by a set of (unrelated) bundle transfers. All messaging in this example occurs between the same Entity A address-and-port and Entity B address-and-port.

The example Entity A has a policy to only send or receive bundles within a DTLS session, so any outgoing bundles to Entity B are queued until the DTLS session is established. Because Entity A is willing to accept transfers on its ephemeral UDP port, the first outgoing message after the DTLS handshake contains the Sender Listen extension along with a Sender Node ID indicating its identity to Entity B. Likewise, the first outgoing message from Entity B after the DTLS handshake contains a Sender Node ID indicating its identity to Entity A.

```
             Entity A                             Entity B
            active peer                         passive peer

      +------------------------+
      |    Initiate DTLS Ext.  | ->
      +------------------------+
      +------------------------+         +------------------------+
      |     DTLS Negotiation   | ->   <- |     DTLS Negotiation   |
      |       (as client)      |         |       (as server)      |
      +------------------------+         +------------------------+

              DNS-ID and IPADDR-ID authentication occurs.
                  Secured UDPCL messaging can begin.

      +------------------------+
      |   Sender Listen Ext.   | ->      +------------------------+
      |   Sender Node ID Ext.  |      <- |   Sender Node ID Ext.  |
      |  Extension Support Ext. |        |  Extension Support Ext. |
      +------------------------+         +------------------------+

                     NODE-ID authentication occurs.
                      Secured transfers can begin.

      +------------------------+
      |    Unframed Transfer   | ->      +------------------------+
      +------------------------+      <- |    Unframed Transfer   |
      +------------------------+         +------------------------+
      |   Identified Transfer  | ->
      +------------------------+
```

Figure 15: An example of the flow of protocol messages on a single UDP conversation between two entities

4. Implementation Status

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

[NOTE to the RFC Editor: please remove this section before publication, as well as the reference to [RFC7942], [github-dtn-demo-agent], and [github-dtn-wireshark].]

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 [RFC7942]. 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 can exist.

An example implementation of the this draft of UDPCL has been created as a GitHub project [github-dtn-demo-agent] and is intended to use as a proof-of-concept and as a possible source of interoperability testing. This example implementation uses D-Bus as the CL--BPA interface, so it only runs on hosts which provide the Python "dbus" library.

A wireshark dissector for UDPCL has been created as a GitHub project [github-dtn-wireshark] and has been kept in-sync with the latest encoding of this specification.

5. Security Considerations

This section separates security considerations into threat categories based on guidance of BCP 72 [RFC3552].

5.1. Threat: Passive Leak of Node Data

When used without DTLS security, the UDPCL can expose the Node ID and other configuration data to passive eavesdroppers. This can occur even if no bundle transfers are transmitted. This can be avoided by always using DTLS, even if authentication is not available (see Section 5.11).

5.2. Threat: Passive Leak of Bundle Data

UDPCL can be used to provide point-to-point unicast transport security, but does not provide multicast security, security of data-at-rest, and does not guarantee end-to-end bundle security. In those cases the bundle security mechanisms defined in [RFC9172] are to be used instead.

When used without DTLS security, the UDPCL exposes all bundle data to passive eavesdroppers. This can be avoided by always using DTLS for unicast messaging, even if authentication is not available (see Section 5.11).

5.3. Threat: Transport Security Stripping

When security policy allows non-DTLS messaging, UDPCL does not protect against active network attackers. It is possible for a on-path attacker to drop or alter packets containing Extension Map and/or DTLS handshake records, which will cause the receiver to not negotiate a DTLS session. This leads to the "SSL Stripping" attack described in [RFC7457].

When DTLS is available on an entity, it is strongly encouraged that the security policy disallow non-DTLS messaging for unicast purposes. This requires that the DTLS handshake occurs before any other UDPCL messaging, regardless of the policy-driven parameters of the handshake and policy-driven handling of the handshake outcome.

One mechanism to mitigate the possibility of DTLS stripping is the use of DNS-based Authentication of Named Entities (DANE) [RFC6698] toward the passive peer. This mechanism relies on DNS and is unidirectional, so it doesn’t help with applying policy toward the active peer, but it can be useful in an environment using opportunistic security. The configuration and use of DANE are outside of the scope of this document.

The negotiated use of DTLS is identical behavior to STARTTLS use in [RFC2595], [RFC4511], and others.

5.4. Threat: Weak DTLS Configurations

Even when using DTLS to secure the UDPCL session, the actual ciphersuite negotiated between the DTLS peers can be insecure. Recommendations for ciphersuite use are included in BCP 195 [RFC9325]. It is up to security policies within each UDPCL entity to ensure that the negotiated DTLS ciphersuite meets transport security requirements.

5.5. Threat: Untrusted End-Entity Certificate

The profile in Section 3.9.4 uses end-entity certificates chained up to a trusted root CA. During DTLS handshake, either entity can send a certificate set which does not contain the full chain, possibly excluding intermediate or root CAs. In an environment where peers are known to already contain needed root and intermediate CAs there is no need to include those CAs, but this has a risk of an entity not actually having one of the needed CAs.

5.6. Threat: Certificate Validation Vulnerabilities

Even when DTLS itself is operating properly an attacker can attempt to exploit vulnerabilities within certificate check algorithms or configuration to establish a secure DTLS session using an invalid certificate. An invalid certificate exploit could lead to bundle data leaking and/or denial of service to the Node ID being impersonated.

There are many reasons, described in [RFC5280] and [RFC6125], why a certificate can fail to validate, including using the certificate outside of its valid time interval, using purposes for which it was not authorized, or using it after it has been revoked by its CA. Validating a certificate is a complex task and can require network connectivity outside of the primary UDPCL network path(s) if a mechanism such as OCSP [RFC6960] is used by the CA. The configuration and use of particular certificate validation methods are outside of the scope of this document.

5.7. Threat: BP Node Impersonation

The certificates exchanged by DTLS enable authentication of peer DNS name and Node ID, but it is possible that a peer either not provide a valid certificate or that the certificate does not validate either the DNS-ID/IPADDR-ID or NODE-ID of the peer (see Section 2.2). Having a CA-validated certificate does not alone guarantee the identity of the network host or BP node from which the certificate is provided; additional validation procedures in Section 3.9.3 bind the DNS-ID/IPADDR-ID or NODE-ID based on the contents of the certificate.

The DNS-ID/IPADDR-ID validation is a weaker form of authentication, because even if a peer is operating on an authenticated network DNS name or IP address it can provide an invalid Node ID and cause bundles to be "leaked" to an invalid node. Especially in DTN environments, network names and addresses of nodes can be time-variable so binding a certificate to a Node ID is a more stable identity.

NODE-ID validation ensures that the peer to which a bundle is transferred is in fact the node which the BPA expects it to be. In circumstances where certificates can only be issued to DNS names, Node ID validation is not possible but it could be reasonable to assume that a trusted host is not going to present an invalid Node ID. Determining when a DNS-ID/IPADDR-ID authentication can be trusted to validate a Node ID is also a policy matter outside of the scope of this document.

One mitigation to arbitrary entities with valid PKIX certificates impersonating arbitrary Node IDs is the use of the PKIX Extended Key Usage key purpose id-kp-bundleSecurity value [IANA-SMI]. When this Extended Key Usage is present in the certificate, it represents a stronger assertion that the private key holder is trusted to operate as a DTN Node.

5.8. Threat: Off-Path Packet Injection

As described in Section 5.1 of [RFC8085], when each endpoint of a UDP conversation uses a stable port number the conversation can be more easily snooped and off-path packets can be injected to spoof UDPCL traffic. When the requirements in Section 3.2 are followed UDPCL traffic will use predictable, well-known port numbers and will be subject to this threat.

One direct mitigation of this threat is to use DTLS to ensure that off-path attackers cannot inject packets without being discovered and those packets rejected at the destination.

Another mitigation is for a UDPCL entity to use the well-known UDPCL port for passive listening but choose an ephemeral port to bind to when acting as the active peer. This will enable some amount of obfuscation while maintaining stable UDP conversations, but could also make legitimate traffic analysis and troubleshooting more difficult.

5.9. Threat: Denial of Service

The behaviors described in this section all amount to a potential denial-of-service to a UDPCL entity. The denial-of-service could be limited to an individual UDPCL entity, or could affect all entities on a host or network segment.

An entity can send a large amount of data to a UDPCL entity, requiring the receiving entity to handle the data. The victim entity can block UDP packets from network peers which are thought to be incorrectly behaving within network.

An entity can also send only one segment of a seemingly valid transfer and never send the remaining segments, which will cause resources on the receiver to be wasted on transfer reassembly state. The victim entity can either block packets from network peers or intentionally keep a short unfinished transfer timeout (see Section 3.6.2).

The keepalive mechanism can be abused to waste throughput within a network link which would otherwise be usable for bundle transmissions.

5.10. Mandatory-to-Implement DTLS

Following IETF best current practice, DTLS is mandatory to implement for all UDPCL implementations but DTLS is optional to use for a any given transfer. The recommended configuration of Section 3.5.5 is to always attempt DTLS, but entities are permitted to disable DTLS based on local configuration. The configuration to enable or disable DTLS for an entity or a session is outside of the scope of this document. The configuration to disable DTLS is different from the threat of DTLS stripping described in Section 5.3.

5.11. Alternate Uses of DTLS

This specification makes use of PKIX certificate validation and authentication within DTLS. There are alternate uses of DTLS which are not necessarily incompatible with the security goals of this specification, but are outside of the scope of this document. The following subsections give examples of alternate DTLS uses.

5.11.1. DTLS Without Authentication

In environments where PKI is available but there are restrictions on the issuance of certificates (including the contents of certificates), it can be possible to make use of DTLS in a way which authenticates only the passive entity of a UDPCL transfer or which does not authenticate either entity. Using DTLS in a way which does not successfully authenticate some claim of both peer entities of a UDPCL transfer is outside of the scope of this document but does have similar properties to the opportunistic security model [RFC7435].

5.11.2. Non-Certificate DTLS Use

In environments where PKI is unavailable, alternate uses of DTLS which do not require certificates such as pre-shared key (PSK) authentication [RFC5489] and the use of raw public keys [RFC7250] are available and can be used to ensure confidentiality within UDPCL. Using non-PKI node authentication methods is outside of the scope of this document.

5.12. Predictability of Transfer IDs

The only requirement on Transfer IDs is that they are unique from the transmitting peer only. The trivial algorithm of the first transfer starting at zero and later transfers incrementing by one causes absolutely predictable Transfer IDs. Even when UDPCL is not DTLS secured and there is a on-path attacker altering UDPCL messages, there is no UDPCL feedback mechanism to interrupt or refuse a transfer so there is no benefit in having unpredictable Transfer IDs.

6. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of code points in existing registries and creation of a new UDPCL registry in accordance with BCP 26 [RFC8126].

6.1. IP Multicast Addresses

This document allocates new routable multicast addresses for use with the UDPCL.

Within the IPv4 Multicast Address Space registry group [IANA-IPv4-MCAST], the registry titled "Internetwork Control Block" has been updated to include the following address.

Parameter	Value
Address(es):	TBA-IP4
Description:	All BP Nodes
References:	[This specification]

Table 3

Within the IPv6 Multicast Address Space registry group [IANA-IPv6-MCAST], the registry titled "Variable Scope Multicast Addresses" has been updated to include the following address.

Parameter	Value
Address(es):	TBA-IP6
Description:	All BP Nodes
References:	[This specification]

Table 4

6.2. UDP Port Number

Within the "Service Name and Transport Protocol Port Number Registry" [IANA-PORTS], UDP port number 4556 has been previously assigned as the default port for the experimental UDPCL [RFC7122]. This assignment to UDPCL is unchanged, but the assignment reference is updated to this specification. There is no UDPCL version indication on-the-wire but this specification is a superset of the experimental UDPCL [RFC7122] and is fully backward compatible.

Service Name:
dtn-bundle

Transport Protocol(s):
UDP

Assignee:
IESG <iesg@ietf.org>

Contact:
IETF Chair <chair@ietf.org>

Description:
DTN Bundle UDP CL Protocol

Reference:
[This specification]

Port Number:
4556

The related assignment for DCCP port 4556, registered by [RFC7122], is unchanged.

6.3. UDPCL Extension Types

EDITOR NOTE: sub-registry to-be-created upon publication of this specification.

IANA will create, under the "Bundle Protocol" registry group [IANA-BUNDLE], a registry titled "Bundle Protocol UDP Convergence-Layer Extension Types" and initialize it with the contents of Table 5. Negative code points are reserved for use on private networks for functions not published to IANA. For code points between 1 and 255 inclusive the registration procedure is RFC Required. For code points between 256 and 32767 inclusive the registration procedure is Specification Required.

Specifications of new extension types need to define the CBOR item structure of the extension data as well as the purpose and relationship of the new extension to existing transfer state within the baseline UDPCL sequencing. Receiving entities will ignore items with unknown Extension ID, and that behavior needs to be considered by new extension types.

Experts are encouraged to be biased towards approving registrations unless they are abusive, frivolous, or actively harmful (not merely aesthetically displeasing, or architecturally dubious).

Extension ID	Name	References
-32768 to -32641	Reserved for Experimental Use	[This specification]
-32640 to -1	Reserved for Private Use	[This specification]
0	Reserved	[This specification]
1	Extension Support	Section 3.5.1 of [this specification]
2	Transfer	Section 3.5.2 of [this specification]
3	Sender Listen	Section 3.5.3 of [this specification]
4	Sender Node ID	Section 3.5.4 of [this specification]
5	DTLS Initiation (STARTTLS)	Section 3.5.5 of [this specification]
6	Peer Probe	Section 3.5.6 of [this specification]
7	Peer Confirmation	Section 3.5.7 of [this specification]
8	ECN Counts	Section 3.5.8 of [this specification]
9 to 32767	Unassigned	

Table 5: UDPCL Extension Types

7. References

7.1. Normative References

[IANA-BUNDLE]
IANA, "Bundle Protocol",
<https://www.iana.org/assignments/bundle/>.

[IANA-PORTS]
IANA, "Service Name and Transport Protocol Port Number Registry", <https://www.iana.org/assignments/service-names-port-numbers/>.

[IANA-IPv4-MCAST]
IANA, "IPv4 Multicast Address Space Registry", <https://www.iana.org/assignments/multicast-addresses/>.

[IANA-IPv6-MCAST]
IANA, "IPv6 Multicast Address Space Registry", <https://www.iana.org/assignments/ipv6-multicast-addresses/>.

[IANA-SMI] IANA, "Structure of Management Information (SMI) Numbers", <https://www.iana.org/assignments/smi-numbers/>.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[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/info/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

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

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[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/info/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/info/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8899] Fairhurst, G., Jones, T., T端xen, M., R端ngeler, I., and T. V旦lker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <https://www.rfc-editor.org/info/rfc8899>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[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/info/rfc9171>.

[RFC9174] Sipos, B., Demmer, M., Ott, J., and S. Perreault, "Delay-Tolerant Networking TCP Convergence-Layer Protocol Version 4", RFC 9174, DOI 10.17487/RFC9174, January 2022, <https://www.rfc-editor.org/info/rfc9174>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

[RFC9331] De Schepper, K. and B. Briscoe, Ed., "The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S)", RFC 9331, DOI 10.17487/RFC9331, January 2023, <https://www.rfc-editor.org/info/rfc9331>.

7.2. Informative References

[RFC792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, DOI 10.17487/RFC2595, June 1999, <https://www.rfc-editor.org/info/rfc2595>.

[RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, DOI 10.17487/RFC3542, May 2003, <https://www.rfc-editor.org/info/rfc3542>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4511] Sermersheim, J., Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, DOI 10.17487/RFC4511, June 2006, <https://www.rfc-editor.org/info/rfc4511>.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <https://www.rfc-editor.org/info/rfc4838>.

[RFC5489] Badra, M. and I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", RFC 5489, DOI 10.17487/RFC5489, March 2009, <https://www.rfc-editor.org/info/rfc5489>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC6709] Carpenter, B., Aboba, B., Ed., and S. Cheshire, "Design Considerations for Protocol Extensions", RFC 6709, DOI 10.17487/RFC6709, September 2012, <https://www.rfc-editor.org/info/rfc6709>.

[RFC7122] Kruse, H., Jero, S., and S. Ostermann, "Datagram Convergence Layers for the Delay- and Disruption-Tolerant Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP)", RFC 7122, DOI 10.17487/RFC7122, March 2014, <https://www.rfc-editor.org/info/rfc7122>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <https://www.rfc-editor.org/info/rfc7435>.

[RFC7457] Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC 7457, DOI 10.17487/RFC7457, February 2015, <https://www.rfc-editor.org/info/rfc7457>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[RFC8489] Petit-Huguenin, M., Salgueiro, G., Rosenberg, J., Wing, D., Mahy, R., and P. Matthews, "Session Traversal Utilities for NAT (STUN)", RFC 8489, DOI 10.17487/RFC8489, February 2020, <https://www.rfc-editor.org/info/rfc8489>.

[RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <https://www.rfc-editor.org/info/rfc8900>.

[RFC9172] Birrane, III, E. and K. McKeever, "Bundle Protocol Security (BPSec)", RFC 9172, DOI 10.17487/RFC9172, January 2022, <https://www.rfc-editor.org/info/rfc9172>.

[RFC9743] Duke, M., Ed. and G. Fairhurst, Ed., "Specifying New Congestion Control Algorithms", BCP 133, RFC 9743, DOI 10.17487/RFC9743, March 2025, <https://www.rfc-editor.org/info/rfc9743>.

[github-dtn-demo-agent]
Sipos, B., "UDPCL Example Implementation", <https://github.com/BrianSipos/dtn-demo-agent/>.

[github-dtn-wireshark]
Sipos, B., "UDPCL Wireshark Dissector", <https://github.com/BrianSipos/dtn-wireshark/>.

Appendix A. Significant changes from RFC 7122

The areas in which changes from the experimental UDPCL [RFC7122] have been made to existing requirements:

* Made explicit references to UDP- and IP-related RFCs.

* Made more strict Keepalive and Padding requirements.

* Added Extension Map message type and initial extension types.

* Defined UDPCL security and made it mandatory-to-implement but optional to use.

The areas in which extensions from the experimental UDPCL [RFC7122] have been made as new behaviors are:

* Added BPv7 bundle as a possible UDPCL payload.

* Added procedures for optional explicit redundancy in transmission.

* Added procedures for optional path characterization.

* Added procedures for optional explicit congestion notification.

* Defined semantics for IP multicast addressing and allocated IPv4 and IPv6 multicast addresses.

* Defined recommendations for UDP port use and re-use (port stability).

Appendix B. Congestion Control Algorithms

The choice of which CCA to use for a particular network, entity, or conversation, and the tuning and configuration of its parameters, is outside the scope of this document. Such a choice depends upon the need, or not, to share network resources among other UDPCL and/or non-UDPCL flows and entities each with their own CCA(s). Considerations for choosing a CCA are discussed in BCP 133 [RFC9743] and congestion control is an active area of development in the IETF Congestion Control Working Group (CCWG) and the Internet Congestion Control Research Group (ICCRG).

Because transfers in the UDPCL are not acknowledged by the receiver, there are some subtle differences between a CCA used by the UDPCL and one used by an acknowledged transport protocol (_e.g._, TCP or QUIC). Because the ECN Counts include a large counter space for all ECN-marked packets these counters can be used as a surrogate for true acknowledgement, although they do not provide a synchronization mechanism for the packets that they are counting (meaning they provide no way to explicitly detect packet loss). If a CCA needs a reasonable estimate of RTT to perform properly, or needs a coarse level of ECN Counts synchronization, the procedure of Section 3.7.2 using a single small probe packet can be combined with ECN marking at the expense of additional control traffic.

Future extensions to UDPCL can be defined in order to negotiate a specific CCA, and its shared parameters, for each entity of a conversation. Implementations and extensions should consider the scope and complexity of additions to UDPCL relative to alternative technologies such as the Datagram Congestion Control Protocol (DCCP) [RFC4340].

Acknowledgments

Much pre-draft review was performed to make the document clear and readable by Sarah Heiner of JHU/APL.

Authors’ Addresses

Brian Sipos
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
United States of America
Email: brian.sipos+ietf@gmail.com

Joshua Deaton
Science Applications International Corporation
Email: joshua.e.deaton@nasa.gov