Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 22 November 2025

F. L. Templin, Ed.
Boeing Research & Technology

21 May 2025

IPv6 Parcels and Advanced Jumbos (AJs)
draft-templin-6man-parcels2-27

Abstract

IPv6 packets contain a single unit of transport layer protocol data which becomes the retransmission unit in case of loss. Transport layer protocols including the Transmission Control Protocol (TCP) and reliable transport protocol users of the User Datagram Protocol (UDP) prepare data units known as segments which the network layer packages into individual IPv6 packets each containing a single segment. This specification presents new packet constructs termed IPv6 Parcels and Advanced Jumbos (AJs) with different properties. Parcels permit a single packet to include multiple segments as a "packet-of-packets", while AJs offer essential operational advantages over basic jumbograms for transporting singleton segments of all sizes ranging from very small to very large. Parcels and AJs provide essential building blocks for improved performance, efficiency and integrity while encouraging larger Maximum Transmission Units (MTUs) according to both the classic Internetworking link model and a new Delay Tolerant Networking (DTN) link model.

Status of This Memo

Table of Contents

1. Introduction 3
2. Terminology 4
3. Requirements 7
4. Background and Motivation 8
5. A Delay-Tolerant Networking (DTN) Link Model 10
6. IPv6 Parcel Formation 12
6.1. TCP Parcels 16
6.2. UDP Parcels 17
6.3. Calculating K 18
7. Transmission of IPv6 Parcels 18
7.1. Original Source Packetization 19
7.2. Final Destination Restoration 20
8. Advanced Jumbos (AJ) 21
9. Parcel Probing 23
10. OMNI Interface Jumbo-in-Jumbo Encapsulation 24
11. Integrity 26
12. Implementation Status 27
13. IANA Considerations 27
14. Security Considerations 28
15. Acknowledgements 29
16. References 30
16.1. Normative References 30
16.2. Informative References 32
Appendix A. TCP Extensions for High Performance 35
Appendix B. Extreme L Value Implications 36
Appendix C. GSO/GRO API 37
C.1. GSO (i.e., Parcel Packetization) 37
C.2. GRO (i.e., Parcel Restoration) 38
Appendix D. Relation to Standard RFC2675 Jumbograms 39
Appendix E. CRC128J 39
Appendix F. Change Log 39
Author's Address 40

1. Introduction

IPv6 packets [RFC8200] contain a single unit of transport layer protocol data which becomes the retransmission unit in case of loss. Transport layer protocols such as the Transmission Control Protocol (TCP) [RFC9293] and reliable transport protocol users of the User Datagram Protocol (UDP) [RFC0768] (including QUIC [RFC9000], LTP [RFC5326] and others) prepare data units known as segments which the network layer packages into individual IPv6 packets each containing only a single segment. This document presents a new construct termed the "IPv6 Parcel" which permits a single packet to include multiple segments. The parcel is essentially a "packet-of-packets" with the full {TCP,UDP}/IPv6 headers appearing only once but with possibly multiple segments included. IPv6 parcels represent a network encapsulation for the multi-segment buffers managed by Generic Segment Offload (GSO) and Generic Receive Offload (GRO); these buffers are termed "parcel buffers" or simply "parcels" which may become "IP parcels" following encapsulation in {TCP,UDP}/IP.

Transport layer protocol entities form parcels by preparing a buffer (or buffer chain) containing at most 64 consecutive transport layer protocol segments that lower layers can break out into individual packets as necessary. All non-final segments must be equal in length while the final segment must not be larger. The transport layer protocol entity then presents the parcel buffer, number of segments and non-final segment size to the network layer. The network layer next either performs packetization to forward each segment as an individual IPv6 packet or appends a single {TCP,UDP} header and a single IPv6 header plus extensions that identify this as an IP parcel. Any included {TCP,UDP} options are associated with all segments, therefore parcels may only include segments that employ compatible options.

This document further introduces an "Advanced Jumbo (AJ)" service that provides essential improvements over basic IPv6 jumbograms as defined in [RFC2675]. AJs provide a robust delivery service when transmission of singleton segments or parcels of all sizes ranging from very small to very large is necessary.

The following sections discuss rationale for adopting parcels and AJs as core elements of the Internet architecture, as well as the actual protocol constructs and operational procedures involved. Parcels and AJs provide an essential data transit service for improved performance, efficiency and integrity while supporting larger Maximum Transmission Units (MTUs). A new Delay Tolerant Networking (DTN) link service model for parcels and AJs further supports delay/disruption tolerance especially well suited for air/land/sea/space mobility applications. These services should inspire future

innovation in applications, transport protocols, operating systems, network equipment and data links for Internetworking performance maximization.

2. Terminology

The Oxford Languages dictionary defines a "parcel" as "a thing or collection of things wrapped in paper in order to be carried or sent by mail". Indeed, there are many examples of parcel delivery services worldwide that provide an essential transit backbone for efficient business and consumer transactions.

In this same spirit, an "IPv6 parcel" is simply a collection of at most 64 transport layer protocol segments wrapped in an efficient package with {TCP,UDP}/IPv6 headers appended for transmission and delivery as a "packet-of-packets". All non-final segments must be equal in length while the final segment must not be larger. IPv6 parcels and AJs are distinguished from ordinary packets and jumbograms through the constructs specified in this document.

The term "Advanced Jumbo (AJ)" refers to a packaging variation modeled from the basic IPv6 jumbogram construct defined in [RFC2675]. AJs include either a single transport layer protocol segment the same as for basic IPv6 jumbograms or a multi-segment parcel. Unlike basic IPv6 jumbograms which are never smaller than 64KB, however, AJs can range in size from as small as the headers plus a minimal or even null payload to as large as 2**32 octets minus headers.

The term "link" is defined in [RFC8200] as: "a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6. Examples are Ethernets (simple or bridged); PPP links; X.25, Frame Relay, or ATM networks; and internet-layer or higher-layer "tunnels", such as tunnels over IPv4 or IPv6 itself".

Where the document refers to "IPv6 header length", it means only the length of the base IPv6 header (i.e., 40 octets), while the length of any extension headers is referred to separately as the "IPv6 extension header length". The term "IPv6 header plus extensions" refers generically to an IPv6 header plus all included extension headers.

Where the document refers to "{TCP,UDP} header length", it means the length of either the TCP header plus options (20 or more octets) or UDP header plus options (8 or more octets). Most significantly, only a single IPv6 header and a single full {TCP,UDP} header plus options appears in each parcel regardless of the number of segments included. This distinction often provides a measurable overhead savings made possible only by parcels.

Where the document refers to checksum calculations, it means the standard Internet checksum unless otherwise specified. The same as for TCP [RFC9293] and UDP [RFC0768], the standard Internet checksum is defined as (sic) "the 16-bit one's complement of the one's complement sum of all (pseudo-)headers plus data, padded with zero octets at the end (if necessary) to make a multiple of two octets". A notional Internet checksum algorithm can be found in [RFC1071].

The term "Cyclic Redundancy Check (CRC)" is used consistently with its application in widely deployed Internetworking services. Parcels and AJs that employ end-to-end integrity checks use the CRC32C [RFC3385] or CRC64E [ECMA-182] standards or a message digest calculated according to the MD5 [RFC1321], SHA1 [RFC3174] or US Secure Hash [RFC6234] algorithms.

The terms "application layer (L5 and higher)", "transport layer (L4)", "network layer (L3)", "(data) link layer (L2)" and "physical layer (L1)" are used consistently with common Internetworking terminology, with the understanding that reliable delivery protocol users of UDP are considered as transport layer elements. The Overlay Multilink Network (OMNI) Interface specification [I-D.templin-6man-omni3] further introduces an "adaptation layer" logically positioned below the network layer but above the link layer (which may include physical links and Internet- or higher-layer tunnels). The adaptation layer is not associated with a layer number itself and is simply known as "the layer below L3 but above L2". A network interface is a node's attachment to a link (via L2), and an OMNI interface is therefore a node's attachment to an OMNI link (via the adaptation layer).

The term "5-tuple" refers to a transport layer protocol entity identifier that includes the network layer (Source Address, Destination Address, Source Port, Destination Port, Protocol Number). The term "4-tuple" refers to a network layer entity identifier that includes the adaptation layer (Source Address, Destination Address, Flow Label, Identification).

The Internetworking term "Maximum Transmission Unit (MTU)" is widely understood to mean the largest packet size that can transit a single link ("link MTU") or an entire path ("path MTU") without requiring network layer fragmentation.

The terms "packetization" and "restoration" refer to a network layer process in which the original source breaks a parcel into individual IPv6 packets that can transit the remainder of the path without loss due to a size restriction. The final destination then restores the combined packet contents into a parcel (or multiple sub-parcels) before delivery to the transport layer. In standard practice, parcel packetization and restoration are functional equivalents of the well-known GSO/GRO services.

The terms "fragmentation" and "reassembly" follow exactly from their definitions in the IPv6 standard [RFC8200], however a new Extended Fragment Header (EFH) service defined in [I-D.templin-6man-ipid-ext2] may be used in place of the standard IPv6 Fragment Header for some applications. Note that AJs are ineligible for fragmentation unless they are first presented to an OMNI interface for adaptation layer encapsulation and are no larger than 65535 octets.

"Automatic Extended Route Optimization (AERO)" [I-D.templin-6man-aero3] and the "Overlay Multilink Network (OMNI) Interface" [I-D.templin-6man-omni3] provide an adaptation layer framework for transmission of parcels/AJs over one or more concatenated Internetworks. AERO/OMNI will provide an operational environment for parcels/AJs beginning from the earliest deployment phases and extending indefinitely to accommodate continuous future growth. As more and more parcel/AJ-capable links are enabled (e.g., in data centers, wireless edge networks, space-domain optical links, etc.) AERO/OMNI will continue to provide an essential service for Internetworking performance maximization.

The terms "(original) source" and "(final) destination" refer to host systems that produce and consume IPv6 packets/parcels/AJs, respectively. The term "router" refers to a system that forwards IPv6 packets/parcels/AJs not addressed to itself while decrementing the Hop Limit. The terms "OAL source", "OAL intermediate system" and "OAL destination" refer to OMNI Adaptation Layer (OAL) nodes that (respectively) produce, forward and consume OAL-encapsulated IPv6 packets/parcels/AJs over an OMNI link.

The terms "controlled environment" and "limited domain" follow directly from [RFC8799]. All nodes within a controlled environment / limited domain are expected to honor the protocol specifications found in this document, whereas nodes on open Internetworks may exhibit varying levels of conformance.

When present, the "Parcel Integrity Block (PIB)" follows the {TCP,UDP}/IPv6 headers of each parcel and includes a separate integrity check for each parcel segment.

The "Parcel Buffer (PB)" includes the concatenated upper layer protocol segments of the parcel. The PB follows the PIB when present; otherwise it follows the {TCP,UDP}/IPv6 headers.

The "Forward Error Correction (FEC)" services specified in this document conform to the IETF FEC architecture found in [RFC5052][RFC5445]. In this FEC architecture, a source node applies FEC encoding to an original IP packet/parcel/AJ and the corresponding destination(s) in turn apply FEC decoding to obtain the original data minus any corrected errors.

The term "flow" refers to a sequence of packets sent from a particular source to a particular unicast, anycast or multicast destination that a node desires to label as a flow [RFC6437].

The parcel sizing variables "J", "K", "L" and "M" are cited extensively throughout this document. "J" denotes the number of segments included in the parcel, "K" is the length of the final segment, "L" is the length of each non-final segment and "M" is termed the "Parcel Payload Length".

3. Requirements

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.

All IPv6 nodes MUST observe their respective requirements found in the normative references, including [RFC8200].

IPv6 parcels and AJs are similar to the basic jumbogram specification found in [RFC2675], but observe the specifications in this document. All IPv6 parcels include a single Parcel Payload Destination Option and all AJs include a single Parcel Payload HBH option; if more than one of either is included, the first is processed and the others are ignored. Only those parcels/AJs intended for paths that support the new link service model and/or larger sizes include the HBH Option.

IPv6 parcels/AJs are not limited only to segment sizes that exceed 65535 octets; instead, parcels can be as small as the packet and parcel headers plus a small or even NULL singleton segment. Parcels that are no larger than 65535 octets and set the IPv6 Payload Length to a non-zero value may be subject to source network layer fragmentation the same as for ordinary IPv6 packets.

For further IPv6 HBH Option considerations, see: [RFC9673]. For IPv6 extension header limits, see: [I-D.ietf-6man-eh-limits]. For IPv4 parcel and advanced jumbo considerations, see: [I-D.templin-intarea-parcels2].

4. Background and Motivation

Studies have shown that applications can improve their performance by sending and receiving larger packets due to reduced numbers of system calls and interrupts as well as larger atomic data copies between kernel and user space. Larger packets also result in reduced numbers of network device interrupts and better network utilization (e.g., due to header overhead reduction) in comparison with smaller packets. However, the most prominent performance increases were observed by increasing the transport layer protocol segment size even if doing so invoked network layer fragmentation.

A first study [QUIC] involved performance enhancement of the QUIC protocol [RFC9000] using the linux GSO/GRO facility. GSO/GRO provides a robust service that has shown significant performance increases based on a multi-segment transfer capability between the operating system kernel and the QUIC transport. GSO/GRO performs packetization and restoration with a transport protocol segment size limited by the path MTU (typically 1500 octets or smaller in current Internetworking practices).

A second study [I-D.templin-dtn-ltpfrag] showed that GSO/GRO also improves performance for the Licklider Transmission Protocol (LTP) [RFC5326] used for the Delay Tolerant Networking (DTN) Bundle Protocol [RFC9171] for segments larger than the actual path MTU through the use of IP fragmentation. Historically, the NFS protocol also saw significant performance increases using larger (single-segment) UDP datagrams even when IP fragmentation is invoked, and LTP still follows this profile today. Moreover, LTP shows this (single-segment) performance increase profile extending to the largest possible segment size which suggests that additional performance gains are possible using (multi-segment) parcels or AJs that approach or even exceed 65535 octets in total length.

TCP also benefits from larger packet sizes and efforts have investigated TCP performance using jumbograms internally with changes to the linux GSO/GRO facilities [BIG-TCP]. The approach proposed to use the Jumbo Payload Option internally and to allow GSO/GRO to use buffer sizes that exceed 65535 octets, but with the understanding that links that support jumbograms natively are not yet widely deployed and/or enabled. Hence, parcels/AJs provide a packaging that can be considered in the near term under current deployment limitations.

A limiting consideration for sending large packets is that they are often lost at links with MTU restrictions, and the resulting Packet Too Big (PTB) messages [RFC4443][RFC8201] may be lost somewhere in the return path to the original source. This path MTU "black hole" condition can negatively impact performance unless robust path probing techniques are used, however optimal performance always occurs when loss of packets due to size restrictions is minimized.

These considerations therefore motivate a design where transport protocols can employ segment sizes as large as 65535 octets (minus headers) while parcels that carry multiple segments may themselves be significantly larger. (Transport layer protocols can also use AJs to transit even larger singleton segments.) Parcels allow the receiving transport layer protocol entity to process multiple segments in parallel instead of one at a time per existing practices. Parcels therefore support improvements in performance, integrity and efficiency for the original source, final destination and networked path as a whole. This is true even if the network or lower layers need to apply packetization/restoration and/or fragmentation/ reassembly.

An analogy: when a consumer orders 50 small items from a major online retailer, the retailer does not ship the order in 50 separate small boxes. Instead, the retailer packs as many of the small items as possible into one or a few larger boxes (i.e., parcels) then places the parcels on a semi-truck or airplane. The consumer will then find only one or a few parcels at their doorstep and not 50 separate small boxes. This flexible parcel delivery service greatly reduces shipping and handling cost for all including the retailer, regional distribution centers and finally the consumer.

5. A Delay-Tolerant Networking (DTN) Link Model

The classic Internetworking link service model [RFC3819] requires each link in the path to apply a link-layer integrity check often termed a "Frame Check Sequence (FCS)" over the entire length of the frame. The link near-end calculates and appends an FCS trailer to each packet pending transmission, and the link far-end verifies the FCS upon packet reception. If verification fails, the link far-end unconditionally discards the packet. This process is repeated for each link in the path so that only packets that pass all link-layer checks over their full lengths are delivered to the final destination. (Note that Internet- or higher-layer tunnels may traverse many underlying physical links that each apply their own FCS in series.)

While the classic link model has contributed to the unparalleled success of terrestrial Internetworks (including the global public Internet), new uses in which significant delays or disruptions can occur are not as well supported. For example, a path that transits one or more links with higher bit error rates may be unable to pass an acceptable percentage of packets since loss due to link errors can occur at any hop. Moreover, packets that incur errors at an intermediate link but somehow pass the link integrity check will be forwarded by all remaining links in the path leaving only the final destination’s integrity checks as a last resort. Especially with the advent of space-domain and wireless Internetworking in inhospitable environments where retransmissions may be onerous or even impractical, advanced end-to-end error detection and correction services not typically associated with ordinary packets are needed. This specification therefore introduces a new Delay Tolerant Networking (DTN) link model, but still with principles of operation consistent with [RFC3819].

IPv6 parcels/AJs that engage this DTN link model request a limited hop-by-hop integrity check that covers only the headers plus a leading portion of the payload. Each IPv6 parcel/AJ also includes per-segment end-to-end Cyclic Redundancy Checks (CRCs) or message digests to be verified by the final destination. For each parcel/AJ admitted under the DTN link model, the original source applies Forward Error Correction (FEC) encoding [RFC5052][RFC5445] if necessary. Each delay/disruption challenged link near-end in the path then applies its standard link-layer FCS for only the leading portion upon transmission according to the Integrity Limit specified by the source then writes the FCS as a trailer following the end of the parcel/AJ payload.

The link far-end then verifies the FCS for the leading portion upon reception and discards the parcel/AJ if an error is detected. However, each link in the path passes parcels/AJs with valid headers through to the final destination even if the unchecked portion of the payload accumulates bit errors in transit. The final destination then invokes FEC decoding [RFC5052][RFC5445] if necessary, verifies integrity using per segment end-to-end CRCs/Digests and delivers each segment to the local transport layer which may employ higher-layer integrity checks.

The ubiquitous 1500 octet link MTU had its origins in the very earliest deployments of 10Mbps Ethernet technologies, however modern wired-line link data rates in the 1Gbps range are now typical for end user devices such as laptop computers while much higher rates approaching 1Tbps commonly occur for data center servers. At these data rates, the serialization delays range from 1200usec at 10Mbps to only .12usec at 100Gbps [ETHERMTU] (still higher data rates are expected in the near future). This suggests that the legacy 1500 MTU may be too small by multiple orders of magnitude for many well-connected data centers, wide-area wired-line networked paths or even for deep space communications over optical links. For such cases, larger parcels and AJs present performance maximization constructs that support larger transport layer segment sizes.

While data centers, Internetworking backbones and deep space networks are often connected through robust fixed link services, the Internet edge is rapidly evolving into a much more mobile environment where 5G (and beyond) cellular services and WiFi radios connect a growing majority of end user systems. Although some wireless edge networks and mobile ad-hoc networks support considerable data rates, more typical rates with wireless signal disruption and link errors suggest that limiting channel contention by configuring more conservative MTU levels is often prudent. Even in such environments, a mixed link model with error-tolerant data sent in DTN parcels/AJs and error-intolerant data sent in classic packet/parcel/AJ constructs may present a more balanced profile.

IPv6 parcels and AJs therefore provide a revolutionary advancement for delay/disruption tolerance in air/land/sea/space mobile Internetworking applications. As the Internet continues to evolve from its more stable fixed terrestrial network origins to one where more and more nodes are exposed to extreme conditions, this new link service model shifts bulk error detection and correction responsibilities to end systems that are uniquely qualified to take corrective actions. This is true even for paths where only one or a few links engage the new reduced coverage link integrity service model, while all other links can continue to employ the full frame checking services as they have always done.

Note: IPv6 parcels and AJs may already be compatible with widely-deployed high data rate link types such as Gbps/Tbps Ethernet as well as lower data rate links such as wireless. For Ethernet, Each frame is identified by a preamble followed by a Start Frame Delimiter (SFD) followed by the frame data itself followed by the FCS and finally an Inter Packet Gap (IPG). Since no length field is included, however, the frame can theoretically extend as long as necessary for transmission of IPv6 parcels and AJs that are much larger than the typical 1500 octet Ethernet MTU as long as the time duration on the link media is properly bounded. Widely-deployed links may therefore already include all of the necessary features to natively support large parcels and AJs with no additional extensions, while operating systems may require extensions to post larger receive buffers.

6. IPv6 Parcel Formation

A transport protocol entity of the source identified by its 5-tuple forms a Parcel Buffer (PB) by concatenating "J" transport layer protocol segments (for J between 1 and 64) into a contiguous buffer or chain of smaller buffers. All non-final segments MUST be of equal length "L" while the final segment of length "K" MUST NOT be larger and MAY be smaller. The overall parcel length (including all segments and headers) is represented by the value "M".

The source sets L to a 16-bit non-final segment length of at least 1 but no larger than 65535 octets minus the lengths of the {TCP,UDP} header (plus options) and IPv6 header (plus extensions) (see: Appendix B). The transport layer protocol entity then presents the resulting PB and non-final segment length L to the network layer, noting that the combined PB length may exceed 65535 octets when there are sufficient segments of a large enough size.

The source then prepends a single full {TCP,UDP} header and a single full IPv6 header that includes a Parcel Payload Destination Option formatted as shown in Figure 1:

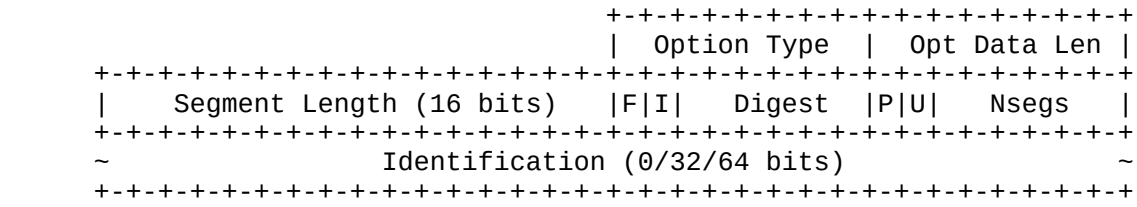

Figure 1: IPv6 Parcel Payload Destination Option

In this encoding, the source includes the Parcel Payload Option as an IPv6 Destination Option with Option Type "rest" set to ’00010’, "action" set to ’11’ and "change" set to ’0’ (i.e., as Hex Value 0xC2). Note that this is the same Option Type as for the Jumbo Payload option specified in [RFC2675] but appearing as a Destination option and not a HBH option. All destinations must therefore consistently accept or discard packets with Destination option 0xC2 according to this specification.

The source sets Opt Data Len to 4/8/12 based on the Identification length. The source may include a full 64-bit Identification only in initial parcels of a flow while including only the 32 least significant bits or omitting the Identification entirely in subsequent parcels when it has sent the full 64-bit value recently. The destination should therefore cache the most recent 64-bit value received for this source.

The source then sets Segment Length to a 16-bit non-final segment length between 0 and 65535. The source also sets the F flag to 1 if a Forward Error Correction (FEC) header follows, sets the I flag to 1 if a PIB is included and sets a 6-bit Digest field to the selected CRC/Digest type per Figure 2. The source finally sets the P flag to 1 if a Probe Reply is requested (see: Section 9, sets the U flag to 1 if a trailing UDP option length field is included and sets a 6-bit Nsegs field to the value (J-1).

The source then optionally inserts a Parcel Integrity Block (PIB) before the PB that includes J consecutive N-octet CRCs/Digests. The source includes each CRC/Digest in the PIB according to one of the CRC32, CRC64, MD5 [RFC1321], SHA1 [RFC3174] or the advanced US Secure Hash Algorithms [RFC6234] as indicated by the Parcel Payload Option Digest field per Figure 2. (A Digest value is also reserved by IANA as a non-functional placeholder for a nominal CRC128J algorithm, which may be specified in future documents; see: Appendix E.)

Type	Algorithm	CRC/Digest Length
0	NULL	0 octets
1	CRC32C	4 octets
2	CRC64E	8 octets
3	MD5	16 octets
4	SHA1	20 octets
5	SHA-224	28 octets
6	SHA-256	32 octets
7	SHA-384	48 octets
8	SHA-512	64 octets
9-63	Reserved	

Figure 2: Parcel CRC/Digest Types

If F is 1, the source then inserts an "IANA FEC Header" immediately following the {TCP,UDP} header (i.e., appearing before the PIB/PB) as shown in Figure 3:

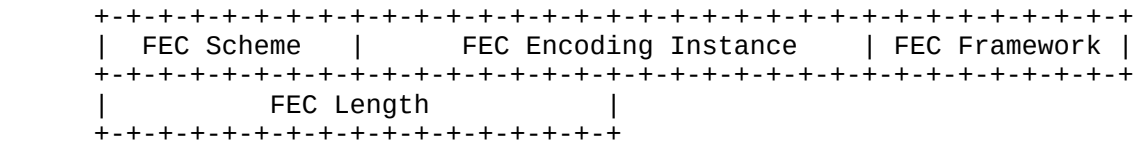

Figure 3: IANA FEC Header

The source sets FEC Scheme according to the appropriate registry values found in [IANA-FEC] and includes a 16-bit FEC Encoding Instance field (with value set according to [IANA-FEC]) only if FEC Scheme is larger than 127. The source then sets FEC Framework according to [IANA-FEC] then sets FEC Length to the length of this FEC header (i.e., either 4 or 6 octets) plus the number of padding octets to be added by the FEC encoding operation.

The source then either includes or omits a Parcel Payload HBH Option. For parcels that are no larger than 65535 octets and do not specify link layer integrity check limits, the source omits the HBH option and sets the IPv6 Payload Length field to a 16-bit value M that encodes the length of the IPv6 extension headers plus the length of the {TCP,UDP} header (plus options and option length field when present) plus the length of the PIB and FEC plus the combined lengths of all concatenated segments.

For all other parcels, the source includes a Parcel Payload HBH option as shown in Figure 4.

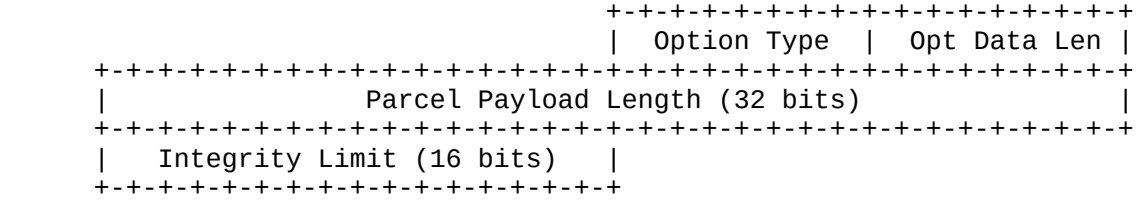

Figure 4: IPv6 Parcel Payload HBH Option

When the source includes a Parcel Payload HBH Option, it sets Option Type "rest" to ’00010’, "action" to ’00’ and "change" to ’0’ (i.e., as Hex Value 0x02) then sets Opt Data Len to 6. (Note: The destination plus all routers on the path must therefore consistently

accept, ignore or discard packets with HBH option 0x02 according to this specification. Intermediate nodes MUST NOT regard the presence of the option as a reason to submit the packet for slow path processing.)

The source then sets the IPv6 Payload Length field to 0 and sets Parcel Payload Length to a 32-bit value M that encodes the length of the IPv6 extension headers plus the length of the {TCP,UDP} header (plus options and option length field when present) plus the length of the PIB and FEC plus the combined lengths of all concatenated segments. This arrangement will cause any routers on the path that do not recognize the option to discard or truncate the parcel to only the IPv6 header due to the IPv6 Payload Length of 0.

Integrity Limit determines the leading length of the parcel subject to link layer FCS integrity checks at links that engage the new link service model while Parcel Payload Length determines the end of the parcel payload after which the link layer appends the trailing FCS itself. Integrity Limit therefore must be less than or equal to Parcel Payload Length. If Integrity Limit is set to 0, link layer FCS integrity checks instead cover the entire parcel as indicated in Parcel Payload Length.

{TCP,UDP}/IPv6 parcels produced by the transport and network layers of the source therefore have the structures shown in Figure 5:

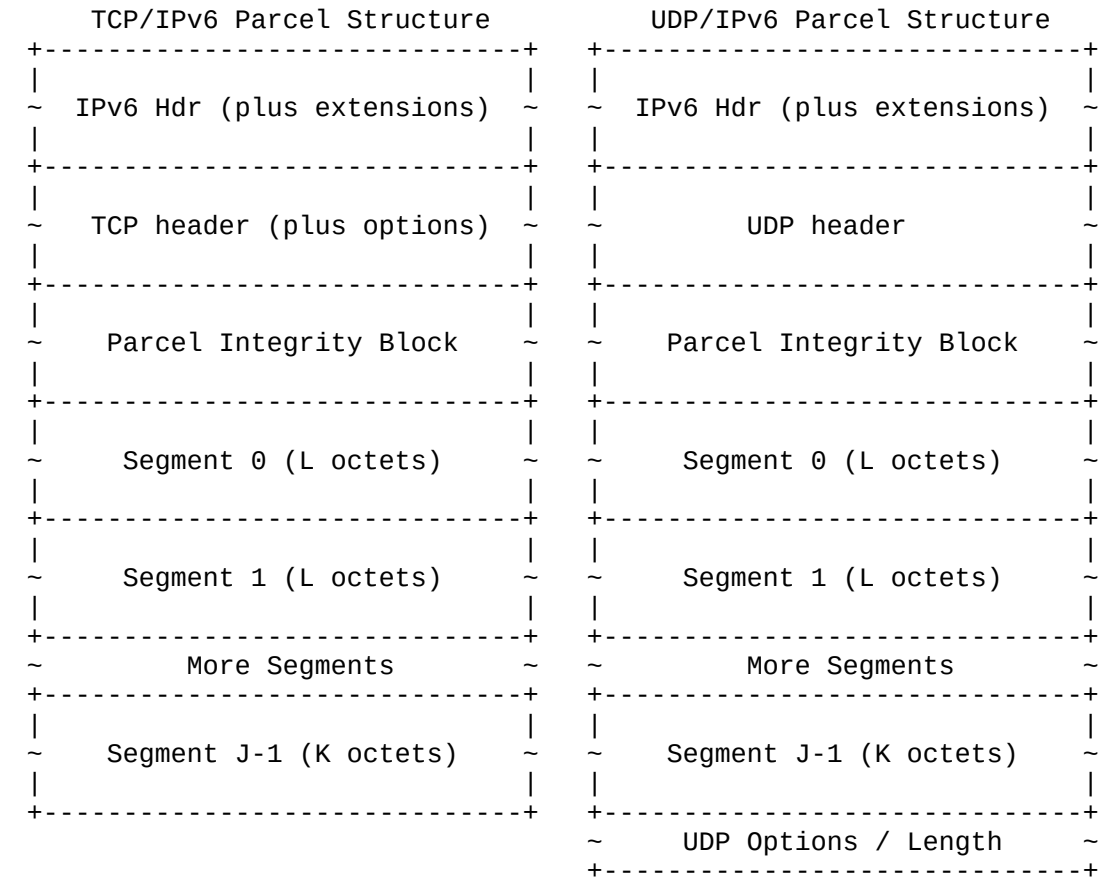

Figure 5: {TCP,UDP}/IPv6 Parcel Structure

6.1. TCP Parcels

A TCP Parcel is an IPv6 parcel that includes a TCP header plus options preceded by an IPv6 header plus extensions with a Parcel Payload Destination Option formed as specified in Section 6. The TCP header is then followed by an optional PIB followed by the J consecutive PB segments. Each non-final segment is L octets in length and the final segment is K octets in length. The value L is encoded in the Segment Length field while the overall length of the parcel is determined by the payload length M.

When the Parcel Payload HBH Option is absent, the source sets the IPv6 Payload Length the same as for an ordinary IPv6 packet. When the HBH option is included, the source instead sets the IPv6 Payload Length to 0. The source then sets the Sequence Number field in the

TCP header to identify the first sequence numbered octet of the first segment present; all additional segments present must then begin on successive sequence number offsets according to L. The destination can then determine the starting sequence number for each segment by examining the Segment Length and Index values with respect to the first segment.

When the PIB is present, the source calculates a CRC/Digest extending over the length of each Segment(i) then writes the value into the PIB CRC(i) field. The source then applies any FEC coding necessary. The source then finally calculates the Internet checksum over the entire length of the parcel the same as for an ordinary TCP packet and writes the value in the TCP checksum field.

See Appendix A for additional TCP considerations. See Section 11 for additional integrity considerations.

Note: The parcel TCP header Source Port, Destination Port and Sequence Number fields apply to each parcel segment (modulo Segment Length and Index), while the TCP control bits and all other fields apply only to the first segment (i.e., "Segment(0)"). Therefore, only parcel Segment(0) may be associated with control bit settings while all other segment(i)’s must be simple data segments.

6.2. UDP Parcels

UDP/IPv6 parcels include a UDP header preceded by an IPv6 header plus extensions with a Parcel Payload Destination Option formed as shown in Figure 1. The UDP header is followed by an optional PIB followed by a PB containing J transport layer segments followed by any UDP options followed by a trailing 2-octet length field when necessary (see below). Each PB segment must begin with a transport-specific start delimiter (e.g., a segment identifier, a sequence number, etc.) included by the transport layer user of UDP. The length of the first segment L is encoded in the Segment Length field while the overall length of the parcel is determined by the parcel payload length M as above.

The source prepares UDP Parcels in an alternative adaptation of UDP jumbograms [RFC2675] . When the Parcel Payload HBH Option is absent, the source sets the IPv6 Payload Length normally. When a Parcel Payload HBH option is present, the source instead sets the IPv6 Payload Length to 0.

The source then sets the UDP header Length field to the length of the UDP header plus the lengths of the PIB plus all PB segments. If this length exceeds 65535 octets, the source instead sets UDP Length to 0. When UDP options are present but their length cannot be determined by

comparing the UDP Length and IPv6 Payload Length values, the source also sets the U flag and includes a 2-octet trailing "UDP Option Length" field that encodes the length of the UDP options which immediately precede it plus 2 octets for the length field itself.

When the PIB is present, the source next populates the PIB by calculating the CRC/Digest over the length of each Segment(i), then writes the value into CRC(i). For the final segment, the source extends the CRC/Digest calculation beyond the length of the segment to also include the UDP options plus UDP Option Length field when either or both are present. (Note that the length of the UDP Option Length field itself is also included in the Parcel Payload Length.) The source then applies FEC coding as necessary.

Finally, when UDP checksums are disabled, the source writes the value '0' in the UDP checksum field. When UDP checksums are enabled the source instead calculates the UDP checksum the same as for an ordinary UDP packet and writes the value into the UDP checksum field while rewriting calculated 0 values as '0xffff'.

See: Section 11 for additional integrity considerations.

6.3. Calculating K

The parcel source unambiguously encodes the values J, L and M in parcel header fields as specified above. The value K is not encoded in a header and must therefore be calculated by nodes that process the parcel. A temporary value T is calculated as the payload length M minus the length of the IPv6 extension headers minus the length of the {TCP,UDP} header (plus options and option length when present) minus the length of the PIB. K is then calculated as the remainder of T divided by the Segment Length.

7. Transmission of IPv6 Parcels

When the network layer of the source assembles a {TCP,UDP}/IPv6 parcel it fully populates all IPv6 header fields including the source and destination addresses, then sets the Parcel Payload Destination Option fields as above with Segment Length L set to a value between 1 and 65535. The source then sets Hop Limit the same as for an ordinary IPv6 packet.

The source also maintains a randomly-initialized (64-bit) Identification value for each flow. For each parcel or AJ transmission, the source sets the Identification to the current cached value for this destination and increments the cached value by 1 (modulo 2**64). (The source can then reset the cached value to a new random number as necessary, e.g., to maintain an unpredictable

profile.) If the parcel/AJ includes a Parcel Payload HBH Option with an Identification field, the source writes the current Identification value into the HBH option field of the same name.

The source also populates all {TCP,UDP} header and option fields, includes a populated PIB/PB then presents the parcel to an interface for transmission to the next hop the same as for an ordinary packet. If the new link model and/or an extended payload length field are required, the source instead first inserts a Parcel Payload HBH Option, sets the IPv6 Payload Length to 0 and forwards the parcel over the parcel-capable path.

When the Parcel Payload HBH option Integrity Limit field is present, each delay/disruption challenged link in the path checks integrity of only that leading portion of the parcel/AJ even if the remainder of the payload contains accumulated link errors. This ensures that the majority of coherent data is delivered to the final destination instead of being discarded along with a minor amount of corrupted data at an intermediate hop while leaving integrity assurance for the remainder as an end-to-end service (see: Section 11).

When the path MTU is insufficient, the source can apply IPv6 fragmentation when the HBH option is not included such that the destination will be required to reassemble. This arrangement should be selected with care since loss of a single fragment would require retransmission of the entire parcel. The source can instead apply packetization to break the parcel up into individual IPv6 packets. The destination then applies restoration to submit the largest possible parcels to upper layers. These considerations are discussed in detail in the following sections.

7.1. Original Source Packetization

For transmission of individual packets when the path MTU is too small to accommodate the entire parcel, the source invokes packetization the same as for GSO.

To initiate packetization, the source first determines whether an individual packet with segment of length L can fit within the path MTU. If an individual packet would be too large the source drops the parcel and returns a Packet Too Big (PTB) message (subject to rate limiting).

For each packet(i), the source then clears the TCP control bits in all but packet(0), and includes only those {TCP,UDP} options that are permitted to appear in data segments in all but packet(0) which may also include control segment options (see: Appendix A for further discussion). The source then sets IPv6 Payload Length for each packet(i) based on the length of segment(i) according to [RFC8200].

For each packet(i), the source then inserts a single Parcel Parameters Destination Option. The option is formatted as shown in Figure 6:

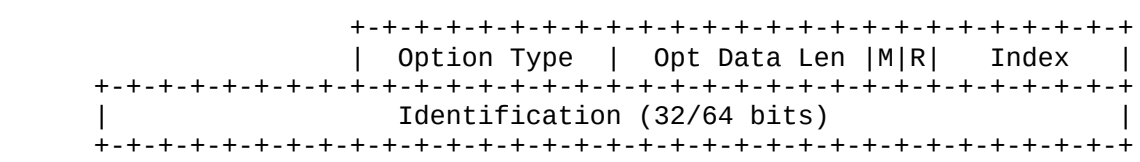

Figure 6: Parcel Parameters Destination Option

The source then sets Option Type "rest" to ’00010’, "action" to ’00’ and "change" to ’0’ (i.e., as Hex Value 0x02) then sets Opt Data Len to 5/9 based on the Identification length. The source includes Identification values corresponding to the original parcel then sets Index to ’i’ and sets M to 1 for non-final packet(i)’s or to 0 for the final packet(i) while also setting R to 0. The source should include only a single Parcel Parameters Destination Option; if multiple are included, the destination processes the first and ignores any others. Note that the source can include a 64-bit Identification in initial packets then revert to including only the 32 least significant bits in additional packets, but the destination must honor the full 64-bit value when it applies restoration.

For each IPv6 packet, the source then sets Hop Limit to the same value as for any IPv6 packet. For each TCP/IPv6 packet, the source next sets IPv6 Payload Length according to [RFC8200] then calculates/sets the checksum for the packet according to [RFC9293]. For each UDP/IPv6 packet, the node instead sets the IPv6 Payload Length and UDP length fields then calculates/sets the checksum according to [RFC0768].

7.2. Final Destination Restoration

When the original source opens a parcel and forwards its contents as individual IPv6 packets, these packets will arrive at the final destination which can hold them in a restoration buffer for a short time before restoring the original parcel the same as for GRO. The 5-tuple information plus the Parcel Parameters Option values included by the source during packetization (see: Figure 6) provide

unambiguous context for GRO restoration which practical implementations have proven as a robust service at high data rates.

The final destination concatenates segments according to ascending Index numbers to preserve segment ordering even if a small degree of reordering and/or loss may have occurred in the networked path. When the final destination performs restoration on TCP segments, it must include the one with any TCP flag bits set as the first concatenation and with the TCP options including the union of the TCP options of all concatenated packets. For both TCP and UDP, any packet containing the final segment must appear as a final concatenation. The final destination can then present the concatenated parcel contents to the transport layer with segments arranged in the same order in which they were originally transmitted.

Note: Restoration buffer management is based on a hold timer during which singleton packets are retained until all members of the same original parcel have arrived. Implementations should maintain a short hold timer (e.g., 1 second) and advance any (partial) restorations to upper layers when the hold timer expires.

Note: Restoration buffer congestion may indicate that the network layer cannot sustain the service(s) at current arrival rates. The network layer should then begin to deliver partial restorations or even individual segments to upper layers (e.g., via the socket buffer) instead of waiting for all segments to arrive. The network layer can manage restoration/ buffers, e.g., by maintaining buffer occupancy high/low watermarks.

Note: Some implementations may encounter difficulty in applying network layer restoration for packets that have already incurred lower layer reassembly. In that case, the network layer can either linearize each packet before applying restoration or deliver incomplete restorations or even individual segments to upper layers.

8. Advanced Jumbos (AJ)

This specification introduces an IPv6 Advanced Jumbo (AJ) service as a (single-segment) parcel alternative to basic jumbograms. Each AJ begins with a {TCP,UDP}/IPv6 header followed by optional FEC and PIB blocks the same as specified for parcels above.

When the source forms a single-segment AJ, it includes a Parcel Payload HBH option and omits the Parcel Payload Destination option. The HBH option format is shown in Figure 7:

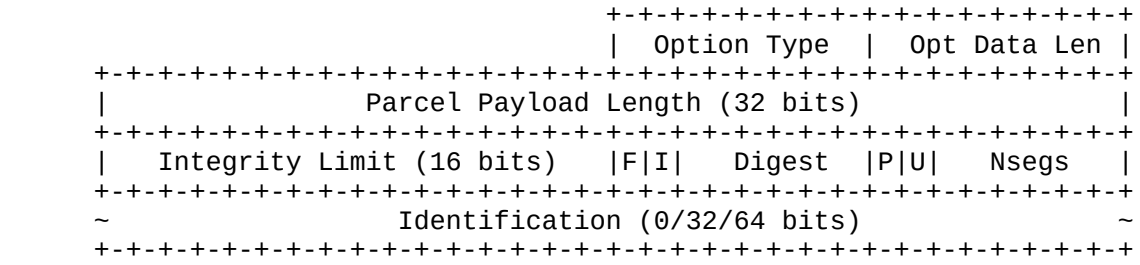

Figure 7: Parcel Payload HBH Option for Advanced Jumbos

The source sets Option Type to Hex Value 0x02 then sets Opt Data Len to 8/12/16 according to the Identification length. The source sets the F, I, P, U flags and Digest the same as for the Parcel Payload Destination option (see: Figure 1) and sets Nsegs to 1.

When I=1, the source next includes a PIB formatted the same as for the parcel PIB but with only a single CRC/Digest. When F=1 is the source includes an FEC encoding the same as for parcels.

The source then sets Parcel Payload Length to the entire AJ payload length and sets Integrity Limit to the length of the leading portion of the AJ intended for coverage by hop-by-hop FCS integrity checks. The source next forms the {TCP/UDP}/IPv6 AJ the same as for parcels as shown in Figure 5 except that the PIB is followed by only a single segment. UDP AJs set the UDP Length field the same as specified for UDP parcels, and include a trailing UDP Option Length field if U is set to 1.

The source next calculates the CRC/Digest over the length of the (single) segment and writes the value into the PIB CRC/Digest field. The source then performs FEC encoding if necessary and resets the Payload Length to include the additional length introduced by FEC. The source finally calculates the standard Internet checksum over the length of the AJ and writes the value in the TCP/UDP checksum field (or writes 0 if UDP checksums are disabled) then sends the AJ via the next hop link toward the final destination.

When the AJ arrives, the destination parses the IPv6 header and Parcel Payload Options then applies FEC decoding for the payload if necessary. The destination then rewrites the (Parcel) Payload Length to reflect the payload decrease due to FEC, then verifies the CRC/Digest if present and delivers the AJ to upper layers.

9. Parcel Probing

The original source can send parcels or AJs without risk of causing harm or triggering alerts even with no prior coordination with routers on the path or the final destination. Unless the source has operational assurance that all nodes in the networked path will correctly pass parcel options, however, this approach may result in systematic loss perceived as a black hole.

The original source should therefore send initial parcel or AJ probes into the forward path according to the probing disciplines specified in [RFC4821] and [RFC8899]. The source should thereafter occasionally send additional probes to determine whether path characteristics have changed and/or to detect black hole conditions.

The original source prepares a parcel/AJ with the P flag set in the Parcel Payload Destination or HBH option header and with a 32- or 64-bit Identification value. The parcel/AJ can be either a purpose-built probe or part of an existing transport protocol session, but it should cause the destination to return a responsive {TCP,UDP}/IPv6 packet with authenticating credentials and with a Parcel Probe Reply Destination Option (see below).

When the destination receives the probe, it returns a responsive IPv6 packet that includes a Parcel Probe Reply Destination Option formatted as shown in Figure 8.

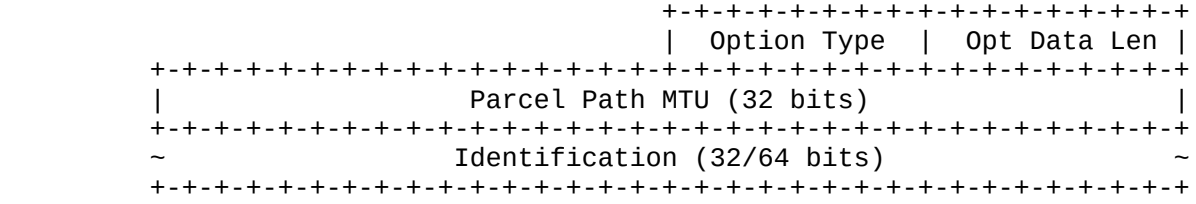

Figure 8: Parcel Probe Reply Destination Option

When the destination includes a Parcel Probe Reply Destination Option, it sets Option Type "rest" to ’00010’, "action" to ’00’ and "change" to ’0’ (i.e., as Hex Value 0x02) then sets Opt Data Len to 8/12 (based on the Identification length). The destination then sets Parcel Path MTU to the length of the probe and Identification to the value included in the probe. The destination then includes any additional identifying parameters (such as authentication codes) in the IPv6 packet and returns the packet to the source while discarding the probe. The destination should include only a single Parcel Probe Reply Destination Option; if multiple are included, the first is processed and all others ignored.

The original source can therefore send parcel probes in the same packets used to carry real data. The probes will transit all routers on the forward path possibly extending all the way to the destination. If the source does not receive a probe reply, it is likely that the path or the final destination does not recognize and correctly pass parcel options. If the source receives a probe reply, it authenticates the message and matches the Identification value with one of its previous probes. If a match is confirmed, then the Parcel Probe Reply Option will contain all information necessary for the source to use in its future parcel/AJ transmissions to this destination.

All parcels/AJs also serve as implicit probes and may cause a router in the path to return an ordinary ICMPv6 error [RFC4443] and/or Packet Too Big (PTB) message [RFC8201] concerning the parcel if the path changes. The source should treat these indications as hints that it should resume probing the forward path.

After the initial path probing, any parcels/AJs for the flow can serve as additional probes to determine whether a path change resulting in an MTU black hole may have occurred. This allows for inline probing with real protocol data and with less dependence on transmission of explicit probe data.

When the source includes a Parcel Probe in a HBH option, it can regard the receipt of an authentic Parcel Probe Reply as evidence that the probe transited the entire forward path to the destination and that the destination observes all aspects of this specification. If the source receives no probe reply, or if it only needs to determine whether the destination accepts parcels, the source can instead include the Parcel Probe as a Destination option.

10. OMNI Interface Jumbo-in-Jumbo Encapsulation

OMNI interfaces set an unlimited MTU and can process parcels and AJs as large as 65535 octets according to normal OMNI link encapsulation and fragmentation procedures. For parcels/AJs that exceed 65535 octets, the OMNI interface can instead insert OMNI and L2 encapsulations per [I-D.templin-6man-omni3] then perform "jumbo-in-jumbo" encapsulation as shown in Figure 9.

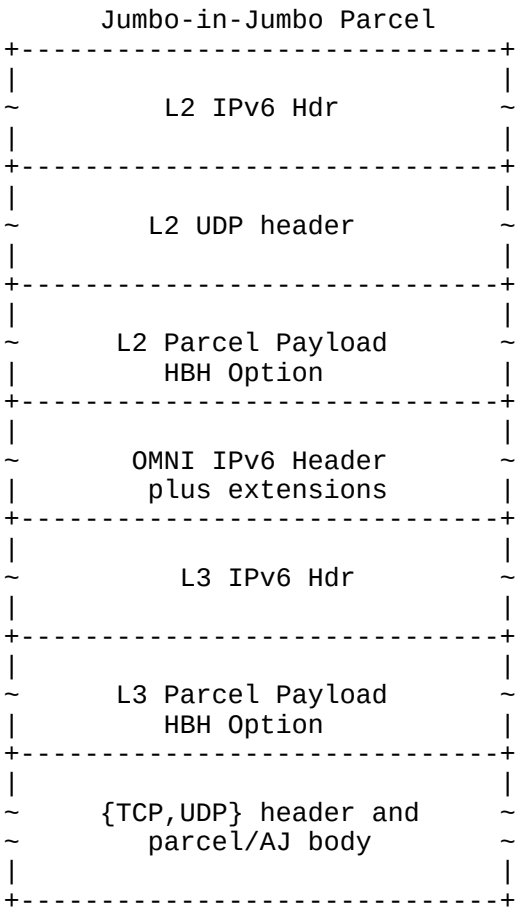

Figure 9: Jumbo-in-Jumbo Encapsulation

When the OMNI link ingress receives a parcel/AJ larger than 65535 octets, it leaves the L3 parcel/AJ headers intact then appends OMNI adaptation layer IPv6 encapsulations plus L2 encapsulations that include a Parcel Payload HBH Option as an L2 extension. The OMNI link ingress sets the Parcel Payload Length field to the length of the L2 extension headers (including the L2 UDP header, if present) plus the lengths of the OMNI IPv6 encapsulation header and the L3 packet (including all L3 headers). The OMNI link ingress sets all other OMNI and L2 encapsulation header fields as specified in [I-D.templin-6man-omni3] then forwards the parcel/AJ.

If the encapsulated parcel/AJ arrives at the OAL destination, the OMNI interface performs decapsulation and forwards the parcel/AJ to next hop toward the final destination from where it may transit

multiple additional OMNI and non-OMNI links. If the parcel/AJ traverses the entire path to the final destination, the destination will then return a probe reply to the source if necessary.

11. Integrity

IPv6 parcel/AJ integrity assurance responsibility is shared between lower layers of the protocol stack and the transport layer where more discrete compensations for lost or corrupted data recovery can be applied. In the classic link model, parcels and AJs are delivered to the final destination only if they pass the integrity checks of all links in the path over their entire length. In the DTN link model, links in the path may forward parcels/AJs with correct headers to the final destination transport layer even if the upper layer protocol data accumulates link errors. The destination is then ultimately responsible for its own end-to-end error correction and integrity assurance.

The Parcel/AJ Internet checksum provides only a rough indication of packaging integrity; an incorrect checksum does not necessarily indicate segment corruption. Parcels/AJs should therefore include a PIB when the path may not support adequate hop-by-hop integrity checks. The per-segment CRCs are set by the source and may be verified by the destination even if the Internet checksum verification fails. Note there may be many instances when the CRCs and Internet Checksum disagree [STONE].

IPv6 parcels can range in length from as small as only the {TCP,UDP}/IPv6 headers plus a single segment to as large as the headers plus (64 * 65535) octets, while AJs include only a single segment that can be as small as a null segment to as large as 2**32 octets (minus headers). IPv6 parcels and AJs with I=1 include integrity checks and use the CRC/Digest algorithm specified in the Digest field to populate the PIB.

For links that observe the DTN link model, the link far end discards the parcel/AJ if it detects an FCS error in the leading portion to avoid the possibility of misdelivery and/or corrupted FEC/PIB fields. Otherwise, the link far end unconditionally forwards the parcel/AJ to the next hop even if the upper layer protocol data incurred link errors. Following any FEC repairs, the PIB integrity checks will ensure that only good data is delivered to upper layers.

Note: Classical links often use CRC32 as their hop-by-hop integrity checking service and this specification assumes that future DTN-capable links will also use CRC32. Since the error detection resolution for CRC32 diminishes for frame sizes larger than ~9KB, implementations should select hop-by-hop integrity protection for

only the leading portions of parcels/AJs while leaving the remaining payload for end-to-end integrity checks. Hop-by-hop integrity checks should at a minimum extend to cover the {TCP,UDP}/IP headers (plus options/extensions) plus the FEC preamble and PIB.

Note: the source performs FEC encoding after calculating the PIB contents and the destination performs FEC decoding before verifying the PIB contents. This ensures that the source and destination will obtain identical copies of the original parcel provided any errors incurred in the path were corrected.

Note: the source and destination network layers can often engage hardware functions to greatly improve CRC/Checksum calculation performance.

12. Implementation Status

Common widely-deployed implementations include services such as TCP Segmentation Offload (TSO) and Generic Segmentation/Receive Offload (GSO/GRO). These services support a robust service that has been shown to improve performance in many instances.

An early prototype of UDP/IPv4 parcels (draft version -15) has been implemented relative to the linux-5.10.67 kernel and ION-DTN ion-open-source-4.1.0 source distributions. Patch distribution found at: "https://github.com/fltemplin/ip-parcels.git".

Performance analysis with a single-threaded receiver has shown that including increasing numbers of segments in a single parcel produces measurable performance gains over fewer numbers of segments due to more efficient packaging and reduced system calls/interrupts. For example, sending parcels with 30 2000-octet segments shows a 48% performance increase in comparison with ordinary packets with a single 2000-octet segment.

Since performance is strongly bounded by single-segment receiver processing time (with larger segments producing dramatic performance increases), it is expected that parcels with increasing numbers of segments will provide a performance multiplier on multi-threaded receivers in parallel processing environments.

13. IANA Considerations

The IANA is instructed to add the following new entries to the "Internet Protocol Version 6 (IPv6) Parameters Registry group:

- in the "Destination Options and Hop-by-Hop Options" Registry (registration procedure IESG Approval, IETF Review or Standards Action) assign the following new entries:

Hex Val	act	chg	rest	Description	Reference
0x02	00	0	00010	Parcel Payload HBH Option	[RFCXXXX]
0x02	00	0	00010	Parcel Param/Reply DestOpt	[RFCXXXX]
0xC2	11	0	00010	Parcel Payload Dest Option	[RFCXXXX]

Figure 10: Destination Options and Hop-by-Hop Options

Note that the "rest" value is the same as for the existing Jumbo Payload option [RFC2675] but the act/chg and resulting Hex Values differentiate.

The IANA is also instructed to create and maintain a new registry titled "IPv6 Parcels and Advanced Jumbos (AJs)" that includes an "IPv6 Advanced Jumbo Digest Types" table with the initial values given below:

Value	Jumbo Type	Reference
0	Advanced Jumbo / NULL	[RFCXXXX]
1	Advanced Jumbo / CRC32C	[RFCXXXX]
2	Advanced Jumbo / CRC64E	[RFCXXXX]
3	Advanced Jumbo / MD5	[RFCXXXX]
4	Advanced Jumbo / SHA1	[RFCXXXX]
5	Advanced Jumbo / SHA-224	[RFCXXXX]
6	Advanced Jumbo / SHA-256	[RFCXXXX]
7	Advanced Jumbo / SHA-384	[RFCXXXX]
8	Advanced Jumbo / SHA-512	[RFCXXXX]
9	Advanced Jumbo / CRC128J	[RFCXXXX]
10-15	Unassigned	[RFCXXXX]

Figure 11: IPv6 Advanced Jumbo Digest Types

14. Security Considerations

In the control plane, original sources match the Identification (and/or other identifying information) received in a Parcel Probe Reply with their earlier parcel/AJ transmissions. If the identifying information matches, the report is likely authentic. When stronger authentication is necessary, the Parcel Probe Reply can appear in the same packets that include transport layer security.

In the data plane, multi-layer security solutions may be necessary to ensure confidentiality, integrity and availability. According to [RFC8200], a full IPv6 implementation includes the Authentication Header (AH) [RFC4302] and Encapsulating Security Payload (ESP) [RFC4303] per the IPsec architecture [RFC4301] to support authentication, data integrity and (optional) data confidentiality. These AH/ESP services provide comprehensive integrity checking for parcel/AJ upper layer protocol headers and all upper layer protocol payload that follows. Since the network layer does not manipulate transport layer segments, parcels/AJs do not interfere with transport or higher-layer security services such as (D)TLS/SSL [RFC8446] which often provide greater flexibility.

IPv4 fragment reassembly is considered dangerous at high data rates where undetected reassembly buffer corruptions can result from fragment misassociations [RFC4963]. IPv6 is less subject to these concerns when the 32-bit Identification field is managed responsibly. IPv6 Parcels and AJs that include the Parcel Payload HBH Option are not subject to fragmentation unless exposed to OMNI interface encapsulation which includes a 64-bit Identification space.

For IPv6 parcels and AJs that engage the DTN link model, the destination end system is uniquely positioned to verify and/or correct the integrity of any transport layer segments received. For this reason, transport layer protocols that use parcels/AJs should include higher layer integrity checks and/or forward error correction codes in addition to the per-segment link error integrity checks.

The CRC/Digest codes included with parcels/AJs that engage the DTN link model provide integrity checks only and must not be considered as authentication codes in the absence of additional security services. Further security considerations related to IPv6 parcels and Advanced Jumbos are found in the AERO/OMNI specifications.

The Parcel Payload Destination and HBH Options support end-to-end authentication since the option contents are not permitted to change en route. The Parcel Probe Destination and HBH options permit their contents to change en route excluding them from end-to-end authentication coverage.

15. Acknowledgements

This work was inspired by ongoing AERO/OMNI/DTN investigations through Boeing Internal Research and Development (IRAD) supporting DTN operations for the International Space Station (ISS). Some of the concepts were further motivated through discussions with colleagues.

A considerable body of work over recent years has produced useful segmentation offload facilities available in widely-deployed implementations.

With the advent of networked storage, big data, streaming media and other high data rate uses the early days of Internetworking have evolved to accommodate the need for improved performance. The need fostered a concerted effort in the industry to pursue performance optimizations at all layers that continues in the modern era. All who supported and continue to support advances in Internetworking performance are acknowledged.

This work has been presented at working group sessions of the Internet Engineering Task Force (IETF). The following IETF and Boeing colleagues are acknowledged for their contributions: Roland Bless, Ron Bonica, Scott Burleigh, Madhuri Madhava Badgandi, Brian Carpenter, David Dong, Joel Halpern, Mike Heard, Tom Herbert, Bob Hinden, Andy Malis, Bill Pohlchuck, Herbie Robinson, Bhargava Raman Sai Prakash, Joe Touch and others who have provided guidance.

Honoring life, liberty and the pursuit of happiness.

16. References

16.1. Normative References

[I-D.ietf-tsvwg-udp-options]
Touch, J. D. and C. M. Heard, "Transport Options for UDP", Work in Progress, Internet-Draft, draft-ietf-tsvwg-udp-options-45, 16 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-udp-options-45>.

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

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

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

[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>.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <https://www.rfc-editor.org/info/rfc2675>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[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>.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <https://www.rfc-editor.org/info/rfc7323>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

16.2. Informative References

[BIG-TCP] Dumazet, E., "BIG TCP, Netdev 0x15 Conference (virtual), https://netdevconf.info/0x15/session.html?BIG-TCP", 31 August 2021.

[ECMA-182] ECMA, E., "European Computer Manufacturers Association (ECMA) Standard ECMA-182, https://ecma-international.org/wp-content/uploads/ECMA-182_1st_edition_december_1992.pdf", December 1992.

[ETHERMTU] Murray, D., Koziniec, T., Lee, K., and M. Dixon, "Large MTUs and Internet Performance, 2012 IEEE 13th International Conference on High Performance Switching and Routing, https://ieeexplore.ieee.org/document/6260832", 24 June 2012.

[I-D.ietf-6man-eh-limits] Herbert, T., "Limits on Sending and Processing IPv6 Extension Headers", Work in Progress, Internet-Draft, draft-ietf-6man-eh-limits-19, 27 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-6man-eh-limits-19>.

[I-D.templin-6man-aero3] Templin, F., "Automatic Extended Route Optimization (AERO)", Work in Progress, Internet-Draft, draft-templin-6man-aero3-44, 21 April 2025, <https://datatracker.ietf.org/doc/html/draft-templin-6man-aero3-44>.

[I-D.templin-6man-ipid-ext2] Templin, F. and T. Herbert, "IPv6 Extended Fragment Header (EFH)", Work in Progress, Internet-Draft, draft-templin-6man-ipid-ext2-13, 19 May 2025, <https://datatracker.ietf.org/doc/html/draft-templin-6man-ipid-ext2-13>.

[I-D.templin-6man-omni3] Templin, F., "Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces", Work in Progress, Internet-Draft, draft-templin-6man-omni3-57, 21 April 2025, <https://datatracker.ietf.org/doc/html/draft-templin-6man-omni3-57>.

[I-D.templin-dtn-ltpfrag]
Templin, F., "LTP Performance Maximization", Work in Progress, Internet-Draft, draft-templin-dtn-ltpfrag-17, 23 May 2024, <https://datatracker.ietf.org/doc/html/draft-templin-dtn-ltpfrag-17>.

[I-D.templin-intarea-parcels2]
Templin, F., "IPv4 Parcels and Advanced Jumbos (AJs)", Work in Progress, Internet-Draft, draft-templin-intarea-parcels2-16, 11 April 2025, <https://datatracker.ietf.org/doc/html/draft-templin-intarea-parcels2-16>.

[IANA-FEC] FEC, I., "Reliable Multicast Transport (RMT) FEC Encoding IDs and FEC Instance IDs, https://www.iana.org/assignments/rmt-fec-parameters", November 2002.

[QUIC] Ghedini, A., "Accelerating UDP packet transmission for QUIC, https://blog.cloudflare.com/accelerating-udp-packet-transmission-for-quic/", 8 January 2020.

[RFC1071] Braden, R., Borman, D., and C. Partridge, "Computing the Internet checksum", RFC 1071, DOI 10.17487/RFC1071, September 1988, <https://www.rfc-editor.org/info/rfc1071>.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <https://www.rfc-editor.org/info/rfc3174>.

[RFC3385] Sheinwald, D., Satran, J., Thaler, P., and V. Cavanna, "Internet Protocol Small Computer System Interface (iSCSI) Cyclic Redundancy Check (CRC)/Checksum Considerations", RFC 3385, DOI 10.17487/RFC3385, September 2002, <https://www.rfc-editor.org/info/rfc3385>.

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <https://www.rfc-editor.org/info/rfc3819>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/info/rfc4821>.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <https://www.rfc-editor.org/info/rfc4963>.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, DOI 10.17487/RFC5052, August 2007, <https://www.rfc-editor.org/info/rfc5052>.

[RFC5326] Ramadas, M., Burleigh, S., and S. Farrell, "Licklider Transmission Protocol - Specification", RFC 5326, DOI 10.17487/RFC5326, September 2008, <https://www.rfc-editor.org/info/rfc5326>.

[RFC5445] Watson, M., "Basic Forward Error Correction (FEC) Schemes", RFC 5445, DOI 10.17487/RFC5445, March 2009, <https://www.rfc-editor.org/info/rfc5445>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <https://www.rfc-editor.org/info/rfc6437>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8799] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <https://www.rfc-editor.org/info/rfc8799>.

[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>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[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>.

[RFC9673] Hinden, R. and G. Fairhurst, "IPv6 Hop-by-Hop Options Processing Procedures", RFC 9673, DOI 10.17487/RFC9673, October 2024, <https://www.rfc-editor.org/info/rfc9673>.

[STONE] Stone, J. and C. Partridge, "When the CRC and TCP Checksum Disagree, ACM SIGCOMM Computer Communication Review, Volume 30, Issue 4, October 2000, pp. 309-319, https://doi.org/10.1145/347057.347561", October 2000.

Appendix A. TCP Extensions for High Performance

TCP Extensions for High Performance are specified in [RFC7323], which updates earlier work that began in the late 1980’s and early 1990’s. These efforts determined that the TCP 16-bit Window was too small to sustain transmissions at high data rates, and a TCP Window Scale option allowing window sizes up to 2^30 was specified. The work also defined a Timestamp option used for round-trip time measurements and as a Protection Against Wrapped Sequences (PAWS) at high data rates. TCP users of IPv6 parcels/AJs are strongly encouraged to adopt these mechanisms.

Since TCP/IPv6 parcels only include control bits for the first segment ("segment(0)"), nodes must regard all other segments of the same parcel as data segments. When a node breaks a TCP/IPv6 parcel out into individual packets, only the first packet contains the original segment(0) and therefore only its TCP header retains the control bit settings from the original parcel TCP header. If the original TCP header included TCP options such as Maximum Segment Size (MSS), Window Scale (WS) and/or Timestamp, the node copies those same options into the options section of the new TCP header.

For all other packets, the note sets all TCP header control bits to 0 as data segment(s). If the original parcel contained a Timestamp option, the node then copies the Timestamp option into the options section of the new TCP header. Appendix A of [RFC7323] provides implementation guidelines for the Timestamp option format.

Appendix A of [RFC7323] also discusses Interactions with the TCP Urgent Pointer as follows: "if the Urgent Pointer points beyond the end of the TCP data in the current segment, then the user will remain in urgent mode until the next TCP segment arrives. That segment will update the Urgent Pointer to a new offset, and the user will never have left urgent mode". In the case of IPv6 parcels, however, it will often be the case that the next TCP segment is included in the same parcel as the segment that contained the urgent pointer such that the urgent pointer can be updated immediately.

Finally, if a parcel/AJ contains more than 65535 octets of data (i.e., even if spread across multiple segments), then the Urgent Pointer can be regarded in the same manner as for jumbograms as described in Section 5.2 of [RFC2675].

Appendix B. Extreme L Value Implications

For each parcel, the transport layer can specify any L value between 1 and 65535 octets.

The transport layer should also specify an L value no larger than can accommodate the maximum-sized transport and network layer headers that the source will include without causing a single segment plus headers to exceed 65535 octets. For example, if the source will include a 28 octet TCP header plus a 40 octet IPv6 header with 24 extension header octets the transport should specify an L value no larger than (65535 - 28 - 40 - 24) = 65443 octets.

The transport can specify still larger "extreme" L values up to 65535 octets, but the resulting parcels might be lost along some paths with unpredictable results. For example, a parcel with an extreme L value set as large as 65535 might be able to transit paths that can pass large parcels/AJs natively but might not be able to transit a path that includes conventional links. The transport layer should therefore carefully consider the benefits of constructing parcels with extreme L values larger than the recommended maximum due to high risk of loss compared with only minor potential performance benefits.

Appendix C. GSO/GRO API

Some modern operating systems include Generic Segment Offload (GSO) and Generic Receive Offload (GRO) services for use by Upper Layer Protocols (ULPs) that engage segmentation. For example, GSO/GRO support has been included in linux beginning with kernel version 4.18. Some network drivers and network hardware also support GSO/GRO at or below the operating system network device driver interface layer to provide benefits of delayed segmentation and/or early reassembly. The following sections discuss the linux GSO and GRO APIs.

C.1. GSO (i.e., Parcel Packetization)

GSO allows ULP implementations to present the sendmsg() or sendmmsg() system calls with parcel buffers that include up to 64 ULP segments, where each concatenated segment is distinguished by an ULP segment delimiter. The operating system kernel will in turn prepare each parcel buffer segment for transmission as an individual UDP/IP packet. ULPs enable GSO either on a per-socket basis using the "setsockopt()" system call or on a per-message basis for sendmsg()/sendmmsg() as follows:

```
/* Set GSO segment size */
unsigned integer gso_size = SEGSIZE;
...
/* Enable GSO for all messages sent on the socket */
setsockopt(fd, SOL_UDP, UDP_SEGMENT, &gso_size, sizeof(gso_size)));
...
/* Alternatively, set per-message GSO control */
cm = CMSG_FIRSTHDR(&msg);
cm->cmsg_level = SOL_UDP;
cm->cmsg_type = UDP_SEGMENT;
cm->cmsg_len = CMSG_LEN(sizeof(uint16_t));
*((uint16_t *) CMSG_DATA(cm)) = gso_size;
```

ULPs must set SEGSIZE to a value no larger than the path MTU via the underlying network interface, minus header overhead; this ensures that UDP/IP datagrams generated during GSO segmentation will not incur local IP fragmentation prior to transmission (Note: the linux kernel returns EINVAL if SEGSIZE encodes a value that exceeds the Path-MTU.)

ULPs should therefore dynamically determine SEGSIZE for paths that traverse multiple links through Packetization Layer Path MTU Discovery for Datagram Transports [RFC8899] (DPMTUD). ULPs should set an initial SEGSIZE to either a known minimum MTU for the path or to the protocol-defined minimum path MTU. The ULP may then dynamically increase SEGSIZE without service interruption if the discovered Path-MTU is larger.

C.2. GRO (i.e., Parcel Restoration)

GRO allows the kernel to return parcel buffers that contain multiple concatenated received segments to the ULP in recvmsg() or recvmmsg() system calls, where each concatenated segment is distinguished by an ULP segment delimiter. ULPs enable GRO on a per-socket basis using the "setsockopt()" system call, then optionally set up per receive message ancillary data to receive the segment length for each message as follows:

```
/* Enable GRO */
unsigned integer use_gro = 1; /* boolean */
setsockopt(fd, SOL_UDP, UDP_GRO, &use_gro, sizeof(use_gro)));
...
/* Set per-message GRO control */
cmsg->cmsg_len = CMSG_LEN(sizeof(int));
*((int *)CMSG_DATA(cmsg)) = 0;
cmsg->cmsg_level = SOL_UDP;
cmsg->cmsg_type = UDP_GRO;
...
/* Receive per-message GRO segment length */
if ((segmentLength = *((int *)CMSG_DATA(cmsg))) <= 0)
     segmentLength = messageLength;
```

ULPs include a pointer to a "use_gro" boolean indication to the kernel to enable GRO; the only interoperability requirement therefore is that each UDP/IP packet includes a parcel buffer with an integral number of properly-formed segments. The kernel and/or underlying network hardware will first coalesce multiple received segments into a larger single segment whenever possible and/or return multiple coalesced or singular segments to the ULP so as to maximize the amount of data returned in a single system call.

ULPs that invoke recvmsg( ) and/or recvmmsg() will therefore receive parcel buffers that include one or more concatenated received ULP segments. The ULP accepts all received segments and identifies any segments that may be missing. The ULP then engages segment ACK/NACK procedures if necessary to request retransmission of any missing segments.

Appendix D. Relation to Standard RFC2675 Jumbograms

This specification uses a new Parcel Payload Destination Option along with a companion HBH Option of the same name instead of the [RFC2675] Jumbo Payload HBH Option.

Standard [RFC2675] jumbograms are incompatible with UDP options, since they always set the IPv6 Payload Length field to 0 and do not otherwise encode a UDP options length. Standard jumbograms are further subject to myriad formatting rules that require routers on the path to drop packets containing the option that do not fully observe all rules and return an ICMPv6 Parameter Problem message.

Standard jumbograms are also always 64KB or larger and rely on IPv6 Path MTU Discovery (PMTUD) ICMPv6 Packet Too Big (PTB) messages to determine whether the end-to-end path supports jumbograms. But the ICMPv6 messages produced for Parameter Problem and PTB are often unreliable and/or untrustworthy in nature.

Appendix E. CRC128J

This section postulates a 128-bit Cyclic Redundancy Check (CRC) algorithm for parcels/AJs termed "CRC128J". A parcel/AJ Digest value is reserved for CRC128J, but at the time of this writing no algorithm exists. Future specifications may update this document and provide an algorithm for handling parcels/AJs with Type CRC128J.

Appendix F. Change Log

<< RFC Editor - remove prior to publication >>

Changes from version -26 to -27:

* Removed per-segment checksums. TCP and UDP checksums now calculated the same as for ordinary IPv6 packets.

Changes from version -25 to -26:

* Made "Digest" types and fields common to both parcels and AJs.

Changes from version -23 to -25:

* Removed all requirements of handshaking with intermediate systems in support of RFC9268 and reverted to path probing by only the end systems themselves per RFC4821 and RFC8899. This means that routers on the path are expected only to forward or not forward parcels/AJs with Parcel Payload options and are not required to engage in any other form of signaling. Parcels/AJs therefore become an end-to-end service with no intervention by routers.

Changes from version -22 to -23:

* Relocated full specifications of OMNI parcellation and reunification from OMNI into this document.

* Clarified inclusion of UDP Option Length field.

Changes from version -21 to -22:

* Added note to clarify that adaptation layer parcel reunification is OPTIONAL allowing routers to immediately release sub-parcels rather than hold them in a reunification buffer.

* Rearranged header fields to avoid splitting multi-bit fields across byte boundaries; also placed single-bit fields as most-significant bits.

Author’s Address

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
United States of America
Email: fltemplin@acm.org