Workgroup: Network Working Group
Internet-Draft:
draft-templin-intarea-parcels-11
Updates: RFC2675 (if approved)
Published: 11 July 2022
Intended Status: Standards Track
Expires: 12 January 2023
Authors: F. L. Templin, Ed.
Boeing Research & Technology

IP Parcels

Abstract

IP packets (both IPv4 and IPv6) are understood to contain a unit of data which becomes the retransmission unit in case of loss. Upper layer protocols including the Transmission Control Protocol (TCP) and transports over the User Datagram Protocol (UDP) prepare data units known as "segments", with traditional arrangements including a single segment per IP packet. This document presents a new construct known as the "IP Parcel" which permits a single packet to carry multiple segments, essentially creating a "packet-of-packets". IP parcels provide an essential building block for accommodating larger Maximum Transmission Units (MTUs) in the Internet as discussed in this document.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on 12 January 2023.

Table of Contents

1. Introduction

IP packets (both IPv4 [RFC0791] and IPv6 [RFC8200]) are understood to contain a unit of data which becomes the retransmission unit in case of loss. Upper layer protocols including the Transmission Control Protocol (TCP) [RFC0793] and transports over the User Datagram Protocol (UDP) [RFC0768] (including QUIC [RFC9000], LTP [RFC5326] and others) prepare data units known as "segments", with traditional arrangements including a single segment per IP packet. This document presents a new construct known as the "IP Parcel" which permits a single packet to carry multiple segments. This essentially creates a "packet-of-packets" with the IP layer and full TCP/UDP headers appearing only once but with possibly multiple upper layer protocol segments included.

Parcels are formed when an upper layer protocol entity identified by the "5-tuple" (source IP, source port, destination IP, destination port, protocol number) prepares a data buffer with the concatenation of up to 64 properly-formed segments that can be broken out into

smaller parcels using a copy of the IP and TCP/UDP header. All segments except the final segment must be equal in size and no larger than 65535 octets (minus headers), while the final segment must not be larger than the others but may be smaller. The upper layer protocol entity then delivers the buffer and non-final segment size to the IP layer, which appends the necessary IP header plus extensions to identify this as a parcel and not an ordinary packet.

Parcels can be forwarded over consecutive parcel-capable IP links in the path until arriving at an ingress middlebox at the edge of an intermediate Internetwork. Each such ingress middlebox may break the parcel out into smaller (sub-)parcels and encapsulate them in headers suitable for traversing the Internetwork. These smaller parcels may then be rejoined into one or more larger parcels at an egress middlebox which either delivers them locally or forwards them further over parcel-capable IP links toward the final destination. Middlebox repackaging of parcels is therefore possible, making reordering and even loss of individual segments possible. But, what matters is that the number of parcels delivered to the final destination should be kept to a minimum for the sake of efficiency, and that loss or receipt of individual segments (and not parcel size) determines the retransmission unit.

The following sections discuss rationale for creating and shipping parcels as well as the actual protocol constructs and procedures involved. IP parcels provide an essential building block for accommodating larger Maximum Transmission Units (MTUs) in the Internet. It is further expected that the parcel concept may drive future innovation in applications, operating systems, network equipment and data links.

2. Terminology

A "parcel" is defined 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 "IP parcel" is simply a collection of up to 64 upper layer protocol segments wrapped in an efficient package for transmission and delivery (i.e., a "packet-of-packets") while a "singleton IP parcel" is simply a parcel that contains a single segment. IP parcels are distinguished from ordinary packets through the special header constructions discussed in this document.

The IP parcels construct is defined for both IPv4 and IPv6. Where the document refers to "IPv4 header length", it means the total length of the base IPv4 header plus all included options, i.e., as

determined by consulting the Internet Header Length (IHL) field. Where the document refers to "IPv6 header length", however, 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 "extension header length". Finally, the term "IP header plus extensions" refers generically to an IPv4 header plus all included options or an IPv6 header plus all included extension headers.

When the document refers to "upper layer header length", it means the length of either the UDP header (8 octets) or the TCP header plus options (20 octets or more). It is important to note that only a single IP header and a single (full) TCP/UDP header appears in each parcel regardless of the number of segments included. This distinction often provides a significant savings in overhead made possible only by parcels.

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.

3. Background and Motivation

Studies have shown that applications can realize greater 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. Large packets also result in reduced numbers of network device interrupts and better network utilization in comparison with smaller packet sizes.

A first study [QUIC] involved performance enhancement of the QUIC protocol [RFC9000] using the linux Generic Segment/Receive Offload (GSO/GRO) facility. GSO/GRO provide a robust (but non-standard) service very similar in nature to the IP parcel service described here, and its application has shown significant performance increases due to the increased transfer unit size between the operating system kernel and QUIC application.

A second study [I-D.templin-dtn-ltpfrag] showed that GSO/GRO also improved performance for the Licklider Transmission Protocol (LTP) [RFC5326] for small- to medium-sized segments. 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 may be possible using (multi-segment) IP parcels that exceed 65535 octets.

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 idea is to use the jumbo payload internally and to allow GSO/GRO to use buffer sizes larger than 65535 octets, but with the understanding that links that support jumbos natively are not yet widely available. Hence, IP parcels provides 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 smaller Maximum Transmission Units (MTUs), and the resulting Packet Too Big (PTB) message may be lost somewhere in the path back to the original source. This "Path MTU black hole" condition can degrade performance unless robust path probing techniques are used, however the best case performance always occurs when no packets are lost due to size restrictions.

These considerations therefore motivate a design where transport protocols should employ a maximum segment size no larger than 65535 octets (minus headers), while parcels that carry the segments may themselves be significantly larger. Then, even if a middlebox needs to sub-divide the parcels into smaller sub-parcels to forward further toward the final destination, an important performance optimization for the original source, final destination and network middleboxes can be realized.

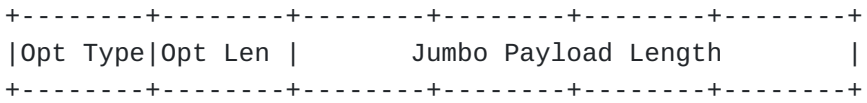
An analogy: when a consumer orders 50 small items from a major online retailer, the retailer does not ship the order in 50 separate small boxes. Instead, the retailer puts as many of the small items as possible into one or a few larger boxes (i.e., parcels) then places the parcels on a semi-truck or airplane. The parcels may then pass through one or more regional distribution centers where they may be repackaged into different parcel configurations and forwarded further until they are finally delivered to the consumer. But most often, the consumer will only find 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.

4. IP Parcel Formation

IP parcel formation is invoked by an upper layer protocol (identified by the 5-tuple described above) when it prepares a data buffer containing the concatenation of up to 64 segments. All non-final segments MUST be equal in length while the final segment MUST NOT be larger and MAY be smaller. Each non-final segment MUST NOT be larger than 65535 octets minus the length of the IPv4 header or IPv6 extension headers, minus the length of an additional IPv6 header in

case an encapsulation middlebox is visited on the path (see: Section 7). The upper layer protocol then presents the buffer and non-final segment size to the IP layer which appends a single IP header (plus extensions) and a single (full) TCP/UDP header before presenting the parcel either to an adaptation layer interface or directly to an ordinary network interface without engaging the adaptation layer (see: Section 7).

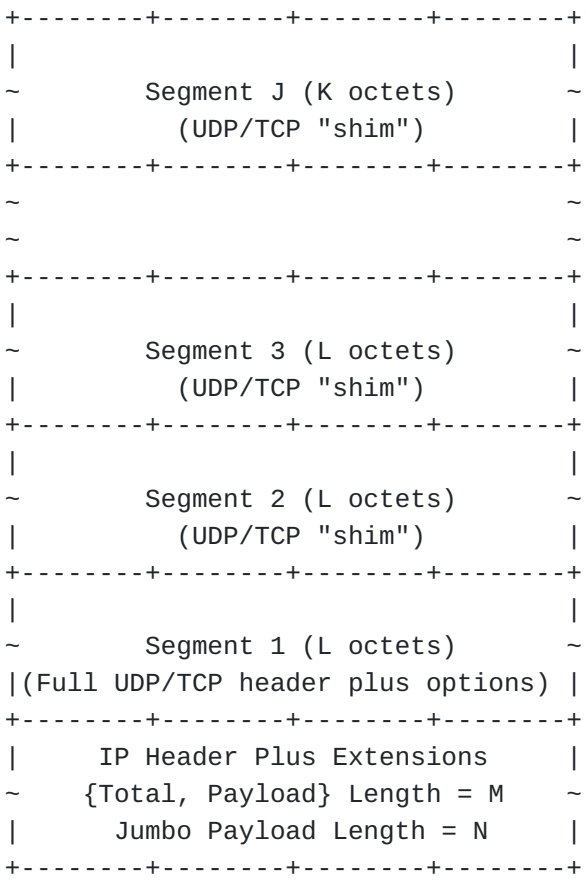
For IPv4, the IP layer prepares the parcel by appending an IPv4 header with a Jumbo Payload option formed as follows:



The IPv4 Jumbo Payload option format is identical to that defined in [RFC2675], except that the IP layer sets option type to '00001011' and option length to '00000110' noting that the length distinguishes this type from its deprecated use as the IPv4 "Probe MTU" option [RFC1063]. The IP layer then sets "Jumbo Payload Length" to the lengths of the IPv4 header plus the combined length of all concatenated segments (i.e., as a 32-bit value in network byte order). The IP layer next sets the IPv4 header DF bit to 1, then sets the IPv4 header Total Length field to the length of the IPv4 header plus the length of the first segment only. Note that the IP layer can form true IPv4 jumbograms (as opposed to parcels) by instead setting the IPv4 header Total Length field to the length of the IPv4 header only (see: Section 11).

For IPv6, the IP layer forms a parcel by appending an IPv6 header with a Hop-by-Hop Options extension header containing a Jumbo Payload option formatted the same as for IPv4 above, but with option type set to '11000010' and option length set to '00000100'. The IP layer then sets "Jumbo Payload Length" to the lengths of all IPv6 extension headers present plus the combined length of all concatenated segments. The IP layer next sets the IPv6 header Payload Length field to the lengths of all IPv6 extension headers present plus the length of the first segment only. Note that the IP layer can form true IPv6 jumbograms (as opposed to parcels) by instead setting the IPv6 header Payload Length field to 0 (see: [RFC2675]).

An IP parcel therefore has the following structure:



where J is the total number of segments (between 1 and 64), L is the length of each non-final segment which MUST NOT be larger than 65535 octets (minus headers as above) and K is the length of the final segment which MUST NOT be larger than L. The values M and N are then set to the length of the full UDP or TCP header (plus options), plus the length of the IP header for IPv4 or to the length of the extension headers only for IPv6. The values M and N are then further calculated as follows:

```
M = M + ((J-1) ? L : K)

N = N + (((J-1) * L) + K) + ((J-1) * shim_length))
```

Note: a "singleton" parcel is one that includes only the {TCP, UDP}/IP headers plus extensions with J=1 and a single segment of length K, while a "null" parcel is a singleton with (J=1; K=0), i.e., a parcel consisting of only the IP header plus extensions with no octets beyond.

5. UDP Parcels

A UDP Parcel is an IP Parcel that includes a full UDP header immediately following the IP header plus extensions. The UDP header is then followed by J segments prepared by the transport layer user

of UDP, where the first segment begins with a transport-specific start delimiter (e.g., a sequence number field) and each non-first segment begins with a "shim" UDP header including only the 2-octet checksum field followed by the start delimiter. The length of each segment is determined by the IP header {Total, Payload} length field as discussed above, noting that only the first segment includes the full UDP header and only the final segment may be of a different length no larger than the others.

The UDP Parcel is prepared in a similar fashion as for UDP jumbograms [RFC2675], except that the UDP checksum for each segment is calculated independently and written into the full/shim UDP header checksum fields (while using the full UDP header for checksum calculation for all segments). The same as for UDP jumbograms, the full UDP header length field is set to 0.

6. TCP Parcels

A TCP Parcel is an IP Parcel that includes a full TCP header (plus options) immediately following the IP header plus extensions. The TCP header is then followed by J segments, where each non-first segment begins with a "shim" TCP header including only the 2-octet checksum field followed by a 4-octet sequence number field that encodes the starting (TCP) sequence number for this segment. The length of each segment is determined by the IP header {Total, Payload} length field as discussed above, noting that only the first segment includes the full TCP header and only the final segment may be of a different length no larger than the others.

The TCP Parcel is prepared in a similar fashion as for TCP jumbograms [RFC2675], except that the TCP checksum for each segment is calculated independently and written into the full/shim TCP header checksum fields (while using the full TCP header for checksum calculation for all segments).

7. Transmission of IP Parcels

The IP layer next presents the parcel to the outgoing network interface. For ordinary IP interfaces, the interface simply forwards the parcel over the underlying link the same as for any IP packet after which it may then be forwarded by any number of routers over additional consecutive parcel-capable IP links. If any next hop IP link in the path either does not support parcels or configures an MTU that is too small to transit the parcel without fragmentation, the router instead opens the parcel and forwards each enclosed segment as a separate IP packet. The router forwards each segment by appending a copy of the parcel's IP header to each segment but with the Jumbo Payload option removed according to the standards [RFC0791][RFC8200]) and also replacing the "shim" TCP/UDP header

with a copy of the full TCP/UDP header (while updating the sequence number and checksum as necessary). Or, if the router does not recognize parcels at all, it drops the parcel and may return an ICMP "Parameter Problem" message.

If the outgoing network interface is an OMNI interface [I-D.templin-6man-omni], the OMNI Adaptation Layer (OAL) of this First Hop Segment (FHS) OAL source node forwards the parcel to the next OAL hop which may be either an OAL intermediate node or a Last Hop Segment (LHS) OAL destination node (which may also be the final destination itself). The OAL source assigns a monotonically-incrementing (modulo 127) "Parcel ID" and subdivides the parcel into sub-parcels no larger than the maximum of the path MTU to the next hop or 65535 octets (minus headers) by determining the number of segments of length L that can fit into each sub-parcel under these size constraints. For example, if the OAL source determines that a sub-parcel can contain 3 segments of length L, it creates sub-parcels with the first containing segments 1-3, the second containing segments 4-6, etc. and with the final containing any remaining segments. The OAL source then appends identical {TCP, UDP}/IP headers plus extensions to each sub-parcel while resetting M and N in each according to the above equations with J set to 3 (and K = L) for each non-final sub-parcel and with J set to the remaining number of segments for the final sub-parcel.

The OAL source next performs IP encapsulation on each sub-parcel with destination set to the next hop IP address then inserts an IPv6 Fragment Header after the IP encapsulation header, i.e., even if the encapsulation header is IPv4, even if no actual fragmentation is needed and/or even if the Jumbo Payload option is present. The OAL source then assigns a randomly-initialized 32-bit Identification number that is monotonically-incremented for each consecutive sub-parcel, then performs IPv6 fragmentation over the sub-parcel if necessary to create fragments small enough to traverse the path to the next OAL hop while writing the Parcel ID and setting or clearing the "Parcel (P)" and "(More) Sub-Parcels (S)" bits in the Fragment Header of the first fragment (see: [I-D.templin-6man-fragrep]). (The OAL source sets P to 1 for a parcel or to 0 for a non-parcel. When P is 1, the OAL source next sets S to 1 for non-final sub-parcels or to 0 if the sub-parcel contains the final segment.) The OAL source then forwards each IP encapsulated packet/fragment to the next OAL hop.

When the next OAL hop receives the encapsulated IP fragments or whole packets, it reassembles if necessary. If the P flag in the first fragment is 0, the next hop then processes the reassembled entity as an ordinary IP packet; otherwise it continues processing as a sub-parcel. If the next hop is an OAL intermediate node, it may retain the sub-parcels along with their Parcel ID and Identification

values for a brief time in hopes of re-combining with peer sub-parcels of the same original parcel identified by the 4-tuple consisting of the IP encapsulation source and destination, Identification and Parcel ID. The combining entails the concatenation of the segments included in sub-parcels with the same Parcel ID and with Identification values within 64 of one another to create a larger sub-parcel possibly even as large as the entire original parcel. Order of concatenation is not critical, with the exception that the final sub-parcel (i.e., the one with S set to 0) must occur as the final concatenation before transmission. The OAL intermediate node then appends a common {TCP, UDP}/IP header plus extensions to each re-combined sub-parcel while resetting M and N in each according to the above equations with J, K and L set accordingly.

This OAL intermediate node next forwards the re-combined sub-parcel(s) to the next hop toward the OAL destination using encapsulation the same as specified above. (The intermediate node MUST ensure that the S flag remains set to 0 in the sub-parcel that contains the final segment.) When the sub-parcel(s) arrive at the OAL destination, the OAL destination re-combines them into the largest possible sub-parcels while honoring the S flag as above. If the OAL destination is also the final destination, it delivers the sub-parcels to the IP layer which acts on the enclosed 5-tuple information supplied by the original source. Otherwise, the OAL destination forwards each sub-parcel toward the final destination the same as for an ordinary IP packet the same as discussed above.

Note: while the OAL destination and/or final destination could theoretically re-combine the sub-parcels of multiple different parcels with identical upper layer protocol 5-tuples and with non-final segments of identical length, this process could become complicated when the different parcels each have final segments of diverse lengths. Since this might interfere with any perceived performance advantages, the decision of whether and how to perform inter-parcel concatenation is an implementation matter.

Note: some IPv6 fragmentation and reassembly implementations may require a well-formed IPv6 header to perform their operations. When the encapsulation is based on IPv4, such implementations translate the encapsulation header into an IPv6 header with IPv4-Mapped IPv6 addresses before performing the fragmentation/reassembly operation, then restore the original IPv4 header before further processing.

Note: sub-dividing a larger parcel into two or more sub-parcels entails the translation of the {TCP,UDP} "shim" header of the first segment in each sub-parcel into a full {TCP, UDP} header. For TCP, the translation is based on copying the full TCP header from the original parcel while replacing the sequence number and checksum

values with the shim header information. For UDP, the translation is based on copying the full UDP header from the original parcel while replacing only the checksum value. Note that the checksum values found in the shim headers are still valid and need not be recalculated.

Note: combining two or more sub-parcels into a larger parcel entails the translation of each former first segment's full {TCP, UDP} headers into a {TCP, UDP} "shim" header. For TCP, the translation is based on copying the sequence number and checksum values into the shim header while discarding the full header. For UDP, the translation is based on copying only the checksum value into the shim header while discarding the full header. Note as above that the checksum values need not be recalculated.

8. Parcel Path Qualification

To determine whether parcels are supported over at least a leading portion of the forward path toward the final destination, the original source can send a singleton IP parcel formatted as a "Parcel Probe" that may include an upper layer protocol probe segment (e.g., a data segment, an ICMP Echo Request message, etc.). The purpose of the probe is to elicit a "Parcel Reply" and possibly also an ordinary upper layer protocol probe reply from the final destination.

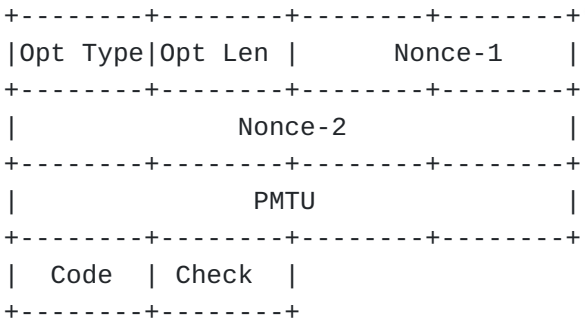
If the original source receives a positive Parcel Reply, it marks the path as "parcels supported" and ignores any ICMP [RFC0792] [RFC4443] and/or Packet Too Big (PTB) messages [RFC1191][RFC8201] concerning the probe. If the original source instead receives a negative Parcel Reply or no reply, it marks the path as "parcels not supported" and may regard any ICMP and/or PTB messages concerning the probe (or its contents) as indications of a possible path MTU restriction.

The original source can therefore send Parcel Probes in parallel with sending real data as ordinary IP packets. If the original source receives a positive Parcel Reply, it can begin using IP parcels.

Parcel Probes use the Jumbo Payload option type (see: Section 4) but set a different option length and replace the option value with control information plus a 4-octet "Path MTU" value into which conformant middleboxes write the minimum link MTU observed in a similar fashion as described in [RFC1063][I-D.ietf-6man-mtu-option]. Parcel Probes can also include an upper layer protocol probe segment, e.g., per [RFC4821][RFC8899]. When an upper layer protocol probe segment is included, it appears immediately after the IP header plus extensions.

The original source sends Parcel Probes unidirectionally in the forward path toward the final destination to elicit a Parcel Reply, since it will often be the case that IP parcels are supported only in the forward path and not in the return path. Parcel Probes may be dropped in the forward path by any node that does not recognize IP parcels, but a Parcel Reply must not be dropped even if IP parcels are not recognized along portions of the return path. For this reason, Parcel Probes are packaged as IPv4 (header) options or IPv6 Hop-by-Hop options while Parcel Replys are always packaged as IPv6 Destination Options (i.e., regardless of the IP protocol version).

Original sources send Parcel Probes and Replys that include a Jumbo Payload option coded in an alternate format as follows:



For IPv4, the original source includes the option as an IPv4 option with Type set to '00001011' the same as for an ordinary IPv4 parcel (see: Section 4) but with Length set to '00001110' to distinguish this as a probe/reply. The original source sets Nonce-1 to 0xffff, sets Nonce-2 to a (pseudo)-random 32-bit value and sets PMTU to the MTU of the outgoing IPv4 interface. The original source then sets Code to 0, sets Check to the same value that will appear in the TTL of the outgoing IPv4 header, then finally sets IPv4 Total Length to the lengths of the IPv4 header plus the upper layer protocol probe segment (if any) and sends the Parcel Probe via the outgoing IPv4 interface. According to [RFC7126], middleboxes (i.e., routers, security gateways, firewalls, etc.) that do not observe this specification SHOULD drop IP packets that contain option type '00001011' ("IPv4 Probe MTU") but some might instead either attempt to implement [RFC1063] or ignore the option altogether. IPv4 middleboxes that observe this specification instead MUST process the option as a Parcel Probe as specified below.

For IPv6, the original source includes the probe option as an IPv6 Hop-by-Hop option with Type set to '11000010' the same as for an ordinary IPv6 parcel (see: Section 4) but with Length set to '00001100' to distinguish this as a probe. The original source sets the concatenation of Nonce-1 and Nonce-2 to a (pseudo)-random 48-bit value and sets PMTU to the MTU of the outgoing IPv6 interface. The original source then sets Code to 0, sets Check to the same value

that will appear in the Hop Limit of the outgoing IPv6 header, then finally sets IPv6 Payload Length to the lengths of the IPv6 extension headers plus the upper layer protocol probe segment (if any) and sends the Parcel Probe via the outgoing IPv6 interface. According to [RFC2675], middleboxes (i.e., routers, security gateways, firewalls, etc.) that recognize the IPv6 Jumbo Payload option but do not observe this specification SHOULD return an ICMPv6 Parameter Problem message (and presumably also drop the packet). IPv6 middleboxes that observe this specification instead MUST process the option as a Parcel Probe as specified below.

When a middlebox that observes this specification receives a Parcel Probe it first compares the Check value with the IP header Hop Limit/TTL; if the values differ, the middlebox MUST return a negative Parcel Reply (see below) and drop the probe. Otherwise, if the next hop IP link either does not support parcels or configures an MTU that is too small to pass the probe, the middlebox compares the PMTU value with the MTU of the inbound link for the probe and MUST (re)set PMTU to the lower MTU. The middlebox then MUST return a positive Parcel Reply (see below) and convert the probe into an ordinary IP packet by removing the probe option according to [RFC0791] or [RFC8200]. If the next hop IP link configures a sufficiently large MTU to pass the packet, the middlebox then MUST forward the packet to the next hop; otherwise, it MUST drop the packet and return a suitable PTB. If the next hop IP link both supports parcels and configures an MTU that is large enough to pass the probe, the middlebox instead compares the probe PMTU value with the MTUs of both the inbound and outbound links for the probe and MUST (re)set PMTU to the lower MTU. The middlebox then MUST reset Check to the same value that will appear in the TTL/Hop Limit of the outgoing IP header, and MUST forward the Parcel Probe to the next hop.

The final destination may therefore receive either an ordinary IP packet containing an upper layer protocol probe or a Parcel Probe. If the final destination receives an ordinary IP packet, it performs any necessary integrity checks then delivers the packet to upper layers which will return an upper layer probe response. If the final destination instead receives a Parcel Probe, it first compares the Check value with the IP header Hop Limit/TTL; if the values differ, the final destination MUST drop the probe and return a negative Parcel Reply (see below). Otherwise, the final destination compares the probe PMTU value with the MTU of the inbound link and MUST (re)set PMTU to the lower MTU. The final destination then MUST return a positive Parcel Reply (see below) and convert the probe into an ordinary IP packet by removing the Parcel Probe option according to the standards [RFC0791][RFC8200].The final destination then performs any necessary integrity checks and delivers the packet to upper layers.

When the middlebox or final destination returns a Parcel Reply, it prepares an IP header of the same protocol version that appeared in the Parcel Probe with source and destination addresses reversed, with {Protocol, Next Header} set to the value '60' (i.e., "IPv6 Destination Option") and with an IPv6 Destination Option header with Next Header set to the value '59' (i.e., "IPv6 No Next Header") [RFC8200]. The node next copies the body of the Parcel Probe option as the sole Parcel Reply Destination Option (and for IPv4 resets Type to '11000010' and Length to '00001100') and includes no other octets beyond the end of the option. The node then MUST (re)set Check to 1 for a positive or to 0 for a negative Parcel Reply, then MUST finally set the IP header {Total, Payload} Length field according to the length of the included Destination Option and return the Parcel Reply to the source. (Since filtering middleboxes may drop IPv4 packets with Protocol '60' the destination MUST wrap an IPv4 Parcel Reply in UDP/IPv4 headers with the IPv4 source and destination addresses copied from the Parcel Reply and with UDP port numbers set to the UDP port number for OMNI [I-D.templin-6man-omni].)

After sending a Parcel Probe the original source may therefore receive a Parcel Reply (see above) and/or an upper layer protocol probe reply. If the source receives a Parcel Reply, it first matches Nonce-2 (and for IPv6 only also matches Nonce-1) with the values it had included in the Parcel Probe. If the values do not match, the source discards the Parcel Reply. Next, the source examines the Check value and marks the path as "parcels supported" if the value is 1 or "parcels not supported" otherwise. If the source marks the path as "parcels supported", it also records the PMTU value as the maximum parcel size for the forward path to this destination.

After receiving a positive Parcel Reply, the original source can begin sending IP parcels addressed to the final destination up to the size recorded in the PMTU. Any upper layer protocol probe replies will determine the maximum segment size that can be included in the parcel, but this is an upper layer consideration. The original source should then periodically re-initiate Parcel Path Qualification as long as it continues to forward parcels toward the final destination (i.e., in case the forward path fluctuates). If at any time performance appears to degrade, the original source should cease sending IP parcels and/or re-initiate Parcel Path Qualification.

Note: For IPv4, the original source sets the Parcel Probe Nonce-1 field to 0xffff on transmission and ignores the Nonce-1 field value in any corresponding Parcel Replys. This avoids any possible confusion in case an IPv4 router on the path rewrites the Nonce-1 field in a wayward attempt to implement [RFC1063].

Note: The PMTU value returned in a positive Parcel Reply determines only the maximum IP parcel size for the path, while the maximum upper layer protocol segment size may be significantly smaller. The upper layer protocol segment size is instead determined separately according to any upper layer protocol probes and must be assumed to be no larger than 1/64th of the maximum IP parcel size unless a larger size is discovered by probing.

Note: Parcel probes should include an (expendable) segment of the same upper layer protocol 5-tuple that would be used to transport ordinary data packets. This ensures that the probes will travel over the same paths as for ordinary data packets.

9. Integrity

Each segment of a (multi-segment) IP parcel includes its own upper layer protocol integrity check. This means that IP parcels can support stronger integrity for the same amount of upper layer protocol data in comparison with an ordinary IP packet or Jumbogram containing only a single segment. The integrity checks must then be verified at the final destination, which accepts any segments with correct integrity while discarding all other segments and counting them as a loss event.

IP parcels can range in length from as small as only the IP headers themselves to as large as the IP headers plus (64 * (65535 minus headers)) octets. Although link layer integrity checks provide sufficient protection for contiguous data blocks up to approximately 9KB, reliance on the presence of link-layer integrity checks may not be possible over links such as tunnels. Moreover, the segment contents of a received parcel may arrive in an incomplete and/or rearranged order with respect to their original packaging.

For these reasons, the OAL at each hop of an OMNI link includes an integrity check when it performs IP fragmentation on a sub-parcel, with the integrity verified during reassembly at the next hop.

10. RFC2675 Updates

Section 3 of [RFC2675] provides a list of certain conditions to be considered as errors. In particular:

> error: IPv6 Payload Length != 0 and Jumbo Payload option present
>
> error: Jumbo Payload option present and Jumbo Payload Length < 65,536

Implementations that obey this specification ignore these conditions and do not consider them as errors.

11. IPv4 Jumbograms

By defining a new IPv4 Jumbo Payload option, this document also implicitly enables a true IPv4 jumbogram service defined as an IPv4 packet with a Jumbo Payload option included and with Total Length set to the length of the IPv4 header only. All other aspects of IPv4 jumbograms are the same as for IPv6 jumbograms [RFC2675].

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 (but not standardized) service that has been shown to improve performance in many instances. Implementation of the IP parcel service is a work in progress.

13. IANA Considerations

The IANA is instructed to change the "MTUP - MTU Probe" entry in the 'ip option numbers' registry to the "JUMBO - IPv4 Jumbo Payload" option. The Copy and Class fields must both be set to 0, and the Number and Value fields must both be set to 11'. The reference must be changed to this document [RFCXXXX].

14. Security Considerations

Original sources match the Nonce values in received Parcel Replies with their corresponding Parcel Probes. If the values match, the Parcel Reply is likely an authentic response to the Parcel Probe. In environments where stronger authentication is necessary, the message authentication services of OMNI can be applied [I-D.templin-6man-omni].

Multi-layer security solutions may be necessary to ensure confidentiality, integrity and availability in some environments.

15. Acknowledgements

This work was inspired by ongoing AERO/OMNI/DTN investigations. The concepts were further motivated through discussions on the intarea and 6man lists.

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

.

16. References

16.1. Normative References

[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>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[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>.

[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>.

[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>.

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.

[I-D.ietf-6man-mtu-option]
Hinden, R. M. and G. Fairhurst, "IPv6 Minimum Path MTU Hop-by-Hop Option", Work in Progress, Internet-Draft, draft-ietf-6man-mtu-option-15, 10 May 2022, <https://www.ietf.org/archive/id/draft-ietf-6man-mtu-option-15.txt>.

[I-D.ietf-tcpm-rfc793bis]
Eddy, W. M., "Transmission Control Protocol (TCP) Specification", Work in Progress, Internet-Draft, draft-ietf-tcpm-rfc793bis-28, 7 March 2022, <https://www.ietf.org/archive/id/draft-ietf-tcpm-rfc793bis-28.txt>.

[I-D.templin-6man-fragrep]
Templin, F. L., "IPv6 Fragment Retransmission and Path MTU Discovery Soft Errors", Work in Progress, Internet-Draft, draft-templin-6man-fragrep-07, 29 March 2022, <https://www.ietf.org/archive/id/draft-templin-6man-fragrep-07.txt>.

[I-D.templin-6man-omni]
Templin, F. L., "Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces", Work in Progress, Internet-Draft, draft-templin-6man-omni-68, 1 July 2022, <https://www.ietf.org/archive/id/draft-templin-6man-omni-68.txt>.

[I-D.templin-dtn-ltpfrag] Templin, F. L., "LTP Fragmentation", Work in Progress, Internet-Draft, draft-templin-dtn-ltpfrag-08, 1 February 2022, <https://www.ietf.org/archive/id/draft-templin-dtn-ltpfrag-08.txt>.

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

[RFC1063] Mogul, J., Kent, C., Partridge, C., and K. McCloghrie, "IP MTU discovery options", RFC 1063, DOI 10.17487/RFC1063, July 1988, <https://www.rfc-editor.org/info/rfc1063>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[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>.

[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>.

[RFC7126] Gont, F., Atkinson, R., and C. Pignataro, "Recommendations on Filtering of IPv4 Packets Containing IPv4 Options", BCP 186, RFC 7126, DOI 10.17487/RFC7126, February 2014, <https://www.rfc-editor.org/info/rfc7126>.

[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>.

[RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "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>.

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