Workgroup: Network Working Group
Internet-Draft:
draft-templin-intarea-parcels-06
Updates: RFC2675 (if approved)
Published: 22 December 2021
Intended Status: Standards Track
Expires: 25 June 2022
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 such as the Transmission Control Protocol (TCP) prepare data units known as "segments", with traditional arrangements including a single segment per 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". Parcels can be broken into smaller parcels by a middlebox on the path if necessary, then rejoined into one or more repackaged parcels to be forwarded further toward the final destination. While not desirable, reordering of segments within parcels and individual segment loss are possible. But, what matters is that the number of parcels delivered to the final destination should be kept to a minimum, and that loss or receipt of individual segments (and not parcel size) determines the retransmission unit.

Status of This Memo

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 such as the Transmission Control Protocol (TCP) [RFC0793], QUIC [RFC9000], LTP [RFC5326] and others prepare data units known as "segments", with traditional arrangements including a single segment per 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 headers appearing only once but with possibly multiple upper layer protocol segments.

Parcels are formed when an upper layer protocol entity (identified by the "5-tuple" source IP address/port number, destination IP address/port number and protocol number) prepares a buffer of data with the concatenation of up to 64 properly-formed segments that can be broken out into smaller parcels using a copy of the IP header. All segments except the final segment must be equal in size and no larger than 65535 (minus headers), while the final segment must be no 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 headers to identify this as a parcel and not an ordinary packet.

Each parcel can be opened at a first-hop middlebox on the path with its included segments broken out into smaller parcels, then rejoined into one or more parcels at a last-hop middlebox to be forwarded to the final destination. Repackaging of parcels is therefore commonplace, while reordering of segments within a parcel or loss of individual segments is possible but not desirable. But, what matters is that the number of parcels delivered to the final destination should be kept to a minimum, 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. It is 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 packets 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 packet. IP parcels are distinguished from ordinary packets through the special header constructions discussed in this document.

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 by sending and receiving larger packets applications can realize greater performance due to reduced numbers of system calls and interrupts as well as larger atomic data copies between kernel and user space. Within edge networks, large packets also result in reduced numbers of device interrupts and better network utilization in comparison with smaller packet sizes.

A first study involved performance enhancement of the QUIC protocol [RFC9000] using the Generic Segment/Receive Offload (GSO/GRO) facility [QUIC]. 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 showed that GSO/GRO also improved performance for the Licklider Transmission Protocol (LTP) [RFC5326] in a similar fashion [I-D.templin-dtn-ltpfrag]. Historically, the NFS protocol also saw dramatic performance increases when using larger UDP datagram sizes even when those sizes invoked IP fragmentation.

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 and GRO to use buffer sizes larger than just ~64KB, 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.

The issue with 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 cripple application performance unless also supplemented with robust path probing techniques, however the best case performance always occurs when no packets are lost due to size restrictions.

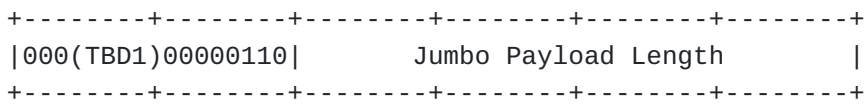
These considerations therefore motivate a design where the maximum segment size should be no larger than 65535 (minus headers), while parcels that carry the segments may themselves be significantly larger. Then, even if a middlebox needs to sub-divide the parcels into smaller sub-parcels to forward further toward the final destination, an important performance optimization for both the original source and final destination 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 boxes as possible into one or a few larger boxes (or parcels) then places the parcels on a semi-truck or airplane. The parcels arrive at a regional distribution center where they may be further redistributed into slightly smaller parcels that get delivered to the consumer. But most often, the consumer will only find one or a few parcels at his doorstep and not 50 individual boxes. This greatly reduces handling overhead for both the retailer and consumer.

4. IP Parcel Formation

IP parcel formation is invoked by an upper layer protocol (identified by the 5-tuple as above) when it produces 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 but MAY be smaller. Each non-final segment MUST be no larger than 65535 minus the length of the IP header plus extensions, minus the length of an additional IPv6 header in case encapsulation is necessary (see: Section 5). The upper layer protocol then presents the buffer and non-final segment size to the IP layer which appends a single IP header (plus any extension headers) before presenting the parcel to the adaptation layer (see: Section 5).

For IPv4, the IP layer prepares the parcel by appending an IPv4 header with a Jumbo Payload option (identified by option code TBD1) formed as follows:

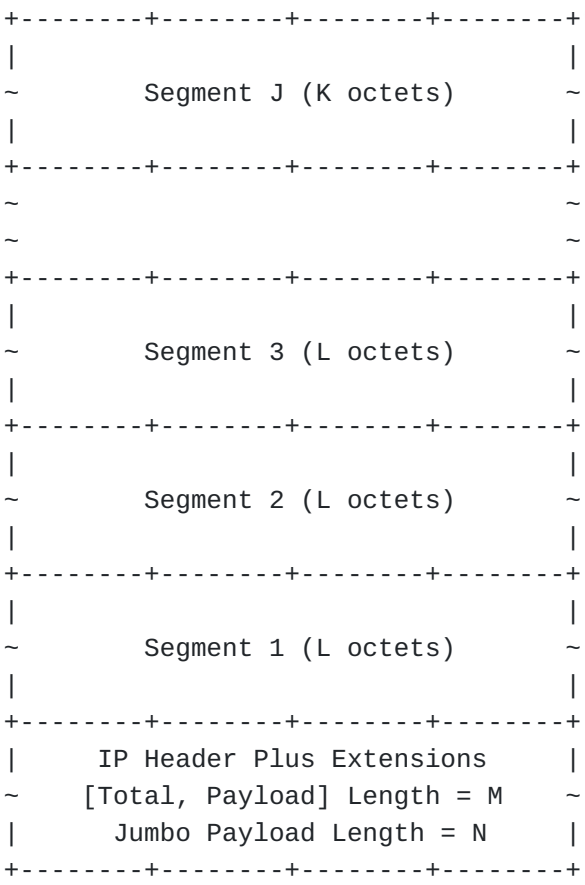


where "Jumbo Payload Length" is a 32-bit unsigned integer value (in network byte order) set to the lengths of the IPv4 header plus all concatenated segments. The IP layer next sets the IPv4 header DF bit to 1, then sets the IPv4 header Total Length field to the length of the IPv4 header plus the length 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 0 (see: Section 9).

For IPv6, the IP layer forms a parcel by appending an IPv6 header with a Jumbo Payload option [RFC2675] the same as for IPv4 above where "Jumbo Payload Length" is set to the lengths of the IPv6 Hop-by-Hop Options header and any other extension headers present plus all concatenated segments. The IP layer next sets the IPv6 header Payload Length field to the lengths of the IPv6 Hop-by-Hop Options header and any other extension headers present plus the length of

the first segment only. As with IPv4 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 (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 IP header plus extensions for IPv4 or to the length of the extensions only for IPv6, then further calculated as follows:

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

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

Note: a "singleton" parcel is one that includes only the IP header plus extensions with a single segment of length K, while a "null" parcel is a singleton with K=0, i.e., a parcel consisting of only the IP header plus extensions with no octets beyond.

5. Transmission of IP Parcels

The IP layer next presents the parcel to the outgoing network interface which invokes the OMNI Adaptation Layer (OAL) [I-D.templin-6man-omni]. The OAL then forwards the parcel to the next hop which may be either an intermediate node or the final destination itself. The OAL 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 64KB (minus the length of encapsulation 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 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 then appends an identical IP header 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 set to L for each non-final sub-parcel and with J set to the remaining number of segments for the final sub-parcel.

The OAL 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 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 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 sets P to 1 for a parcel or to 0 for a non-parcel. When P is 1, the OAL next sets S to 1 for non-final sub-parcels or to 0 if the sub-parcel contains the final segment.) The OAL then forwards each IP encapsulated packet/fragment to the next hop (i.e., an intermediate node or the final destination).

When the OAL of the next 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 not the final destination, it retains 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 important, 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 then appends a common 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.

When the next hop is an intermediate node, it next forwards the re-combined (sub-)parcel(s) to the next hop toward the final 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 parcel or sub-parcels arrive at the final destination, the OAL re-combines them into the largest possible (sub)-parcels while honoring the S flag then delivers them to upper layers which act on the enclosed 5-tuple information supplied by the original source.

Note: while the final destination may be tempted to 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 could possibly defeat any perceived performance advantages, the decision of whether and how to perform inter-parcel concatenation is an implementation matter.

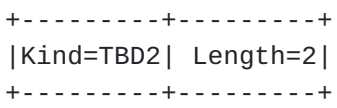
Note: some IPv6 fragmentation and reassembly implementations may require a well-formed IPv6 header to perform their operations. When the encapsulation is based on IPv4, such implementations translate the encapsulation header into an IPv6 header with IPv4-Mapped IPv6 addresses before performing the fragmentation/reassembly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

6. TCP OMNI Option

TCP peers that wish to employ IP parcels must negotiate their use upon connection establishment by including the OMNI option. This two-byte option may be sent in a SYN by a TCP that has been extended to receive (and presumably process) IP parcels once the connection has opened. It MUST NOT be sent on non-SYN segments. The TCP option has the following format:

TCP Parcel-Permitted Option:

Kind: TBD2



A TCP that includes the OMNI option need not implement the full OMNI interface abstraction but MUST implement enough of the OAL to be capable of fragmenting and reassembling maximum-length encapsulated parcels and sub-parcels (see: Section 4 and Section 5).

Note: at the time of this writing, the TCP protocol is under revision for second edition RFC publication [I-D.ietf-tcpm-rfc793bis].

7. Integrity

Parcels can range in length from as small as only the IP header sizes to as large as the IP headers plus (64 * (2**16 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 includes an integrity check when it performs IP fragmentation on a sub-parcel, with the integrity verified during reassembly at the next hop. From an end-to-end perspective, upper layers must include individual integrity checks with each segment included in the parcel with a strength compatible with the segment length. The integrity check must then be verified at the final destination on a per-segment basis, which discards any corrupted segments and considers them as a loss event.

8. 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.

9. IPv4 Jumbograms

By defining a new IPv4 Jumbo Payload option, this document also implicitly enables an IPv4 jumbogram service defined as an IPv4 packet with Total Length set to 0 and with a Jumbo Payload option in the IPv4 extension headers. All aspects of IPv4 jumbograms (including length determination for upper layer protocols) follow exactly the same as for IPv6 jumbograms as specified in [RFC2675].

10. 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.

11. IANA Considerations

The IANA is instructed to allocate a new IP option code in the 'ip option numbers' registry for 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 'TBD1 (to be assigned by IANA)'. The reference must be set to this document (RFCXXXX).

The IANA is instructed to allocate a new TCP Option Kind Number in the 'tcp-parameters' registry for the "OMNI" option. The Kind must be set to 'TBD2' (to be assigned by IANA) and the Length must be set to 2. The reference must be set to this document (RFCXXXX).

12. Security Considerations

Communications networking security is necessary to preserve confidentiality, integrity and availability.

13. Acknowledgements

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

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

.

14. References

14.1. Normative References

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

[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>.

[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>.

14.2. 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-tcpm-rfc793bis]
Eddy, W. M., "Transmission Control Protocol (TCP) Specification", Work in Progress, Internet-Draft, draft-ietf-tcpm-rfc793bis-25, 7 September 2021, <https://www.ietf.org/archive/id/draft-ietf-tcpm-rfc793bis-25.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-03, 16 December 2021, <https://www.ietf.org/archive/id/draft-templin-6man-fragrep-03.txt>.

[I-D.templin-6man-omni] Templin, F. L. and T. Whyman, "Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces", Work in Progress, Internet-Draft, draft-

templin-6man-omni-51, 15 November 2021, <https://www.ietf.org/archive/id/draft-templin-6man-omni-51.txt>.

[I-D.templin-dtn-ltpfrag] Templin, F. L., "LTP Fragmentation", Work in Progress, Internet-Draft, draft-templin-dtn-ltpfrag-06, 19 November 2021, <https://www.ietf.org/archive/id/draft-templin-dtn-ltpfrag-06.txt>.

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

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

[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>.

[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