Workgroup: Network Working Group
Internet-Draft:
draft-templin-intarea-parcels-14
Updates: RFC2675 (if approved)
Published: 29 July 2022
Intended Status: Standards Track
Expires: 30 January 2023
Authors: F. L. Templin, Ed.
Boeing Research & Technology

IP Parcels

Abstract

IP packets (both IPv4 and IPv6) contain a single unit of upper layer protocol 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 upper layer protocol segments, essentially creating a "packet-of-packets". IP parcels provide an essential building block for improved performance and efficiency while supporting 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 30 January 2023.

Table of Contents

1. Introduction

IP packets (both IPv4 [RFC0791] and IPv6 [RFC8200]) contain a single unit of upper layer protocol data which becomes the retransmission unit in case of loss. Upper layer protocols such as 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 upper layer protocol segments. This essentially creates a "packet-of-packets" with the IP layer and full {TCP,UDP} headers appearing only once but with possibly multiple segments included.

Parcels are formed when an upper layer protocol entity identified by the "5-tuple" (source address, destination address, source port,

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 sub-parcels and/or individual packets using a copy of the {TCP,UDP}/IP headers if necessary. All segments except the final one must be equal in length and no larger than 65535 octets (minus headers), while the final segment must be no larger than the others but may be smaller. The upper layer protocol entity then appends a {TCP,UDP} header and delivers the buffer and non-final segment size to the IP layer, which appends an IP header plus extensions that 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 a router with a next hop link that does not support parcels or an ingress middlebox OMNI interface [I-D.templin-6man-omni] that spans intermediate Internetworks using adaptation layer encapsulation and fragmentation. In the former case, the router transforms the parcel into individual IP packets then forwards each packet via the next hop link. In the latter case, the OMNI interface breaks the parcel out into smaller sub-parcels if necessary then encapsulates each (sub-)parcel in headers suitable for traversing the Internetworks while applying any necessary fragmentation.

These sub-parcels may then be recombined into one or more larger parcels by an egress middlebox OMNI interface which either delivers them locally or forwards them over additional parcel-capable links on the path to the final destination. Reordering and even loss of individual segments due to repackaging in the network is therefore 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 improved performance and efficiency while supporting larger Maximum Transmission Units (MTUs) in the Internet. It is further expected that the parcel concept will drive future innovation in applications, operating systems, network equipment and data links.

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 "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 and a "null IP parcel" is a singleton with segment length 0. IP parcels are distinguished from ordinary packets through the special header constructions discussed in this document.

The IP parcel 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.

Where 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 IP parcels.

Where the document refers to checksum calculations, it means the standard Internet checksum unless otherwise specified. The same as for TCP [RFC0793], UDP [RFC0768] and IPv4 [RFC0791], 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".

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 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.

A first study [QUIC] involved performance enhancement of the QUIC protocol [RFC9000] using the linux Generic Segment/Receive Offload (GSO/GRO) facility. GSO/GRO provides 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 improves performance for the Licklider Transmission Protocol (LTP) [RFC5326] used for the Delay Tolerant Networking (DTN) Bundle Protocol [RFC9171] 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 are 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 option 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 the network 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 path as a whole 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

An IP parcel is formed by an upper layer protocol entity (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 UDP header or TCP header and its options, minus the length of the IPv4/IPv6 header and its options/extensions. (Note that this also satisfies the case of ingress middlebox OMNI interfaces in the path which would regard the headers as upper layer protocol payload during encapsulation/ fragmentation.) The upper layer protocol entity then presents the buffer and non-final segment size to the IP layer which appends a single IPv4/IPv6 header plus options/extensions, a full UDP header or TCP header plus options for the first segment, a 4-octet Sequence Number header for each TCP non-first segment and a 2-octet Checksum trailer for each segment. The IP layer then presents the parcel to a network interface attachment to either a parcel-capable ordinary IP link or an OMNI link that performs adaptation layer encapsulation and fragmentation (see: Section 7).

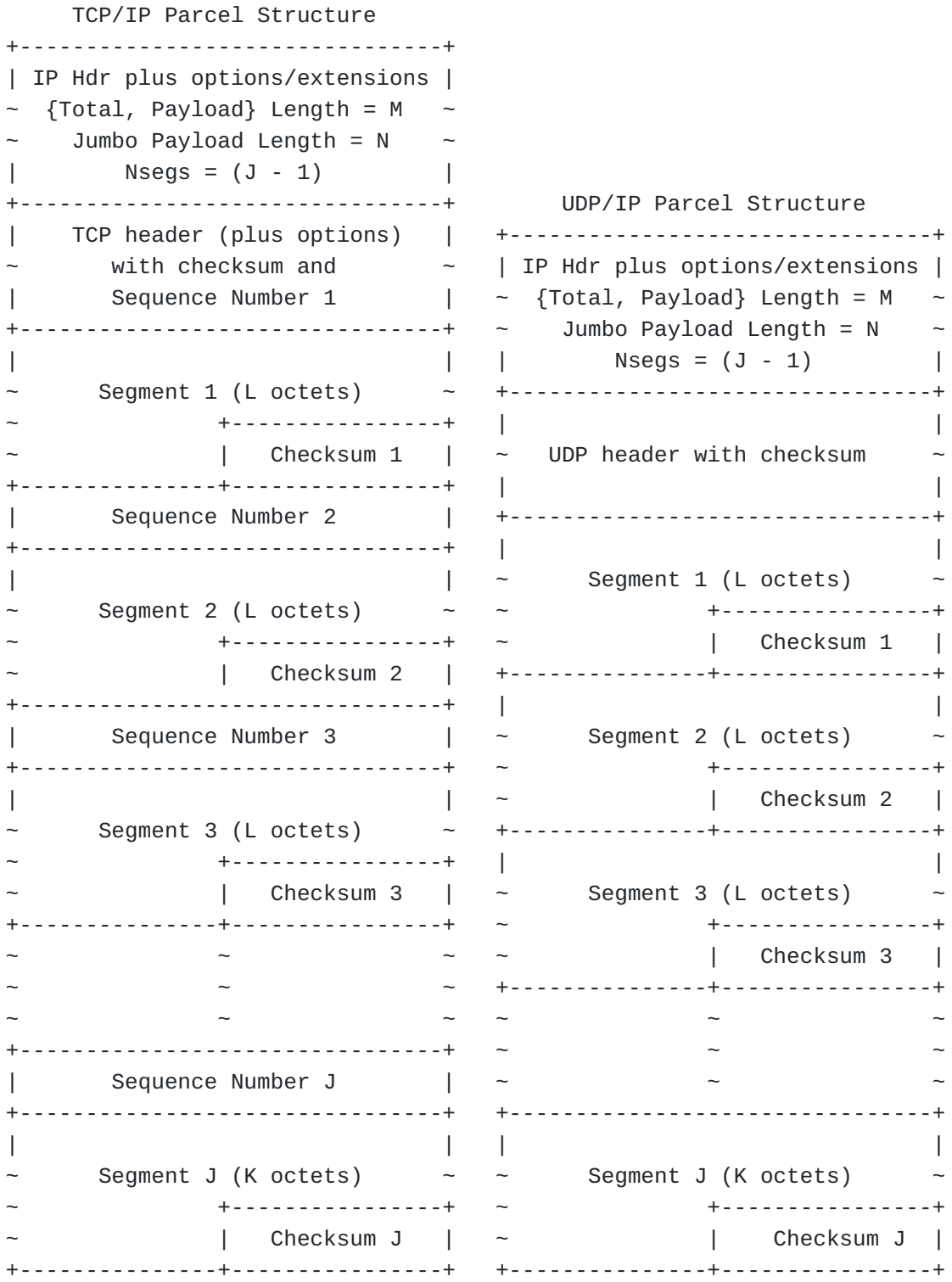
For IPv4, the IP layer prepares the parcel by appending an IPv4 header with a Jumbo Payload option formed as follows:

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 obsoleted use as the "IPv4 Probe MTU" option [RFC1063]. The IP layer then sets the "Nsegs" octet to one less than the number of included segments (see below) and sets "Jumbo Payload

Length" to a 24-bit value (in network byte order) that encodes the length of the {TCP,UDP}/IPv4 headers (plus options) plus the combined length of all concatenated segments with their headers and trailers. 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 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 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 the "Nsegs" octet to one less than the number of included segments (see below) and sets the 24-bit "Jumbo Payload Length" to the lengths of all IPv6 extension headers present plus the length of the {TCP,UDP} header (plus options) plus the combined length of all concatenated segments with their headers and trailers. The IP layer next sets the IPv6 header Payload Length field to 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]).

TCP/IP and UDP/IP parcels therefore have 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) and K is the length of the final segment which MUST NOT be larger than L. For TCP, the TCP header Sequence Number field encodes a 4-octet starting sequence number for the first segment only, while each non-first segment is preceded by its own 4-octet Sequence Number header. For both TCP and UDP, each

segment is followed by a 2-octet Checksum trailer that covers only that segment.

The {Total, Payload} Length "M" is then set to L if there are multiple segments or K if there is only a single segment. Next, 1-octet Nsegs field in the Jumbo Payload option is set to (J - 1). Finally, the 3-octet Jumbo Payload Length "N" is set to the length of the IP header plus extensions for IPv4 (or to the length of the extension headers only for IPv6) plus the length of the UDP header or TCP header plus options. Jumbo Payload Length "N" is finally incremented to include the lengths of the individual segments plus headers and trailers as follows:

For TCP: N = N + (((J-1) * (L + 6)) + (K + 2))

For UDP: N = N + (((J-1) * (L + 2)) + (K + 2))

5. TCP Parcels

A TCP Parcel is an IP Parcel that includes an IP header plus extensions including a Jumbo Payload option but with the first octet encoding the number of non-final segments included between 0 - 63 and the next three octets encoding a Jumbo Payload Length up to 4MB. The IP header is then immediately followed by a 20-octet TCP header plus any additional TCP option octets. The TCP header is then followed by J segments, where each non-first segment is preceded by a 4-octet Sequence Number header that encodes the starting (TCP) sequence number for the segment and each segment is followed by a 2-octet Checksum trailer. The length of each non-final segment is determined by the {Total, Payload} Length "M" while the overall length of the parcel as well as final segment length are determined by Nsegs and the Jumbo Payload Length "N" as discussed above.

Sources prepare TCP Parcels in a similar fashion as for TCP jumbograms [RFC2675] and include the sequence number of the first segment in the TCP header. The source sets the TCP header Checksum field to the checksum of the TCP header plus an IP pseudo-header only (see: Section 9) with calculated '0' values written as 'ffff' (i.e., according to "UDP rules"). The source then calculates the checksum of the first segment beginning with the sequence number found in the full TCP header as a 4-octet pseudo-header. The source next calculates the checksum for each non-first segment independently over the length of the segment beginning with the 4-octet Sequence Number header for that segment. As the source calculates each per-segment checksum, it writes the value into the corresponding Checksum trailer with calculated '0' values written as 'ffff'.

6. UDP Parcels

A UDP Parcel is an IP Parcel that includes an IP header plus extensions including a Jumbo Payload option but with the first octet encoding the number of non-final segments included between 0 - 63 and the next three octets encoding a Jumbo Payload Length up to 4MB. The IP header is immediately followed by an 8-octet UDP header. The UDP header is then followed by J segments prepared by the transport layer user of UDP, where each segment begins with a transport-specific start delimiter (e.g., a sequence number field included by the UDP user) and is followed by a 2-octet Checksum trailer. The length of each non-final segment is determined by the {Total, Payload} Length "M" while the overall length of the parcel as well as the final segment length are determined by Nsegs and the Jumbo Payload Length "N" as discussed above.

Sources prepare UDP Parcels in a similar fashion as for UDP jumbograms [RFC2675] and with the UDP header length field set to 0. The source sets the UDP header Checksum field to the checksum of the UDP header plus an IP pseudo-header only (see: Section 9), then calculates a separate checksum for each segment independently over the length of the segment beginning with the first octet of that segment. As the source calculates each per-segment checksum, it writes the value into the corresponding Checksum trailer with calculated '0' values written as 'ffff'.

7. Transmission of IP Parcels

The IP layer of the source next presents the parcel to the outgoing network interface. For ordinary IP interface attachments to parcel-capable links, the interface simply admits the parcel into the 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 possibly even traversing the entire forward path to the final destination itself. If any router in the path does not recognize the parcel construct, it drops the parcel and may return an ICMP "Parameter Problem" message.

If the router recognizes parcels but the next hop IP link in the path either does not support parcels or configures an MTU that is too small, the router instead opens the parcel and forwards each enclosed segment as a separate IP packet. The router prepares each segment by appending a copy of the parcel's {TCP,UDP}/IP headers to the segment but with the Jumbo Payload option removed, the per segment 4-octet Sequence Number header removed (if present), the per-segment 2-octet Checksum trailer removed and the IP {Total, Payload} length set according to the standards [RFC0791][RFC8200]. For TCP, the router then sets the TCP header Sequence Number field according to the starting sequence number of the segment and also

clears the ACK flag in all but the first packet. For both TCP and UDP, the router next calculates the checksum over the length of the entire packet according to the native TCP [RFC0793] or UDP [RFC0768] protocol specification, then writes the value in the Checksum field and finally forwards the packet.

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. Note that upper layer protocol processing procedures are specified in detail here, while lower layer encapsulation and fragmentation procedures are specified in detail in [I-D.templin-6man-omni]. The remainder of this section considers the OMNI interface case under these observations.

When the OAL source (or OMNI link ingress node) forwards a parcel, it first 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. For TCP, the OAL source then sets the TCP Sequence Number field to the value that appears in the first sub-parcel segment while omitting the first segment Sequence Number header (if present) and also clearing the ACK flag in all sub-parcels except the first. For both TCP and UDP, the OAL source finally resets the {TCP,UDP} header checksum according to ordinary parcel formation procedures (see above). The OAL source next selects a monotonically-incrementing Identification value for each sub-parcel then performs adaptation layer encapsulation and fragmentation and finally forwards them to the next OAL hop which forwards or reassembles as necessary.

If the final OAL hop is an OAL link egress node, it may retain the sub-parcels along with their Parcel ID for a brief time to support re-combining with peer sub-parcels of the same original parcel identified by the 4-tuple consisting of the adaptation layer header (source, destination, Identification, Parcel ID) fields. The re-combining entails the concatenation of 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 need not be strictly enforced, with the exception that the sub-parcel containing the final segment must occur as a final concatenation and not as an intermediate. The OAL link egress 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. For TCP, if any sub-parcels have the ACK bit set then the OAL link egress node also sets the ACK bit in the re-combined sub-parcel TCP header. The OAL link egress node then resets the {TCP,UDP}/IP header checksum for each re-combined sub-parcel and re-encapsulates each in a separate adaptation layer header.

This OAL link egress node next forwards each resulting encapsulated sub-parcel to the next hop toward the OAL destination the same as specified above. When the sub-parcel(s) arrive at the OAL destination, the OAL destination re-combines them into the largest possible sub-parcels while honoring final segment and TCP ACK bit as above then removes the encapsulation headers. If the OAL destination is also the final destination, it delivers the sub-parcels to the IP layer which processes them according to the 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: sub-dividing a larger parcel into two or more sub-parcels entails replication of the {TCP,UDP}/IP headers. For TCP, the process includes copying the full TCP/IP header from the original parcel while writing the sequence number of the first sub-parcel segment into the TCP Sequence Number field, clearing the ACK bit if necessary as discussed above and truncating the Sequence Number field for that segment (if present). For UDP, the process includes simply copying the full UDP/IP header from the original parcel into each sub-parcel. For both TCP and UDP, the process finally includes recalculating and resetting Nsegs and the Jumbo Payload Length then recalculating the {TCP,UDP} header checksum. Note that the Checksum trailer values in the sub-parcel segments themselves are still valid and need not be recalculated.

Note: re-combining two or more sub-parcels into a larger parcel entails a reverse process of the above in which the {TCP,UDP}/IP headers of non-first sub-parcels are discarded and their included segments concatenated following those of a first sub-parcel. For TCP, the process includes setting the ACK in the TCP header only if ACK was set in any of the original sub-parcels. For both TCP and UDP, the process finally includes recalculating and resetting Nsegs and the Jumbo Payload Length then recalculating the {TCP,UDP} header checksum as discussed above. Note that the Checksum trailer values

in the combined parcel segments themselves are still valid and need not be recalculated.

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 non-final segment lengths, this process could become complicated when the different parcels each have differing final segment lengths. Since this might interfere with any perceived performance advantage, the decision of whether and how to perform inter-parcel concatenation is an implementation matter.

Note: sub-dividing of IP parcels occurs only at OMNI link ingress nodes while re-combining of IP parcels occurs only at OMNI link egress nodes. Therefore, intermediate OAL nodes do not participate in any such sub-dividing or recombining.

Note: for TCP, the ACK bit must be managed as specified above to avoid confusing receivers with spurious duplicate ACK indications.

8. Parcel Path Qualification

To determine whether parcels are supported over at least a leading portion of the forward path up to and including the final destination, the original source can send IP parcels that contain Jumbo Payload options formatted as "Parcel Probes". 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 Parcel Reply, it marks the path as "parcels supported" and ignores any ordinary ICMP [RFC0792][RFC4443] and/or Packet Too Big (PTB) messages [RFC1191][RFC8201] concerning the probe. If the original source instead receives no reply, it marks the path as "parcels not supported" and may regard any ordinary ICMP and/or PTB messages concerning the probe (or its contents) as indications of a possible path MTU restriction.

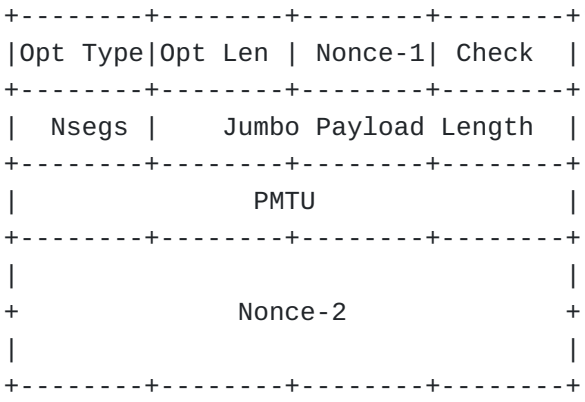
The original source can therefore send Parcel Probes in parallel with sending real data as ordinary IP packets/parcels. If the original source receives a Parcel Reply, it can continue using IP parcels.

Parcel Probes use the same Jumbo Payload option type used for ordinary parcels (see: Section 4) but set a different option length and include a 4-octet "Path MTU" field into which conformant routers write the minimum link MTU observed in a similar fashion as described in [RFC1063][I-D.ietf-6man-mtu-option]. Parcel Probes include one or more upper layer protocol segments corresponding to

the 5-tuple for the flow, which may also include {TCP,UDP} probes used per packetization layer path MTU discovery [RFC4821][RFC8899].

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 filtered in the forward path by any node that does not recognize IP parcels, but Parcel Replys must be packaged to avoid filtering since parcels may not be recognized along portions of the return path. For this reason, the Jumbo Payload options included in Parcel Probes are always packaged as IPv4 header options or IPv6 Hop-by-Hop options while Parcel Replys are returned as UDP/IP encapsulated ICMPv6 PTB messages with a "Parcel Reply" Code value (see: [I-D.templin-6man-omni]).

Original sources send Parcel Probes 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 '00010100' to distinguish this as a probe. The original source sets Nonce-1 to '1111', sets Check to the same value that will appear in the TTL of the outgoing IPv4 header, sets Nonce-2 to a 64-bit random number and sets PMTU to the MTU of the outgoing IPv4 interface. The source next includes a {TCP,UDP} header followed by upper layer protocol segments in the same format as for an ordinary parcel. The source then sets {Nsegs, Jumbo Payload Length, IPv4 Total Length} and calculates the header and per-segment checksums the same as for an ordinary parcel, then finally sends the Parcel Probe via the outbound 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 '00010010' to distinguish this as a probe. The original source sets Nonce-1 to '1111', sets Check to the same value that will appear in the Hop Limit of the outgoing IPv6 header, sets Nonce-2 to a 64-bit random number and sets PMTU to the MTU of the outgoing IPv6 interface. The source next includes a {TCP,UDP} header followed by one or more upper layer protocol segments in the same format as for an ordinary parcel. The source then sets {Nsegs, Jumbo Payload Length, IPv6 Payload Length} and calculates the header and per-segment checksums the same as for an ordinary parcel, then finally sends the Parcel Probe via the outbound 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) due to the different option length. IPv6 middleboxes that observe this specification instead MUST process the option as a Parcel Probe as specified below.

When a router that observes this specification receives either an IPv4 or IPv6 Parcel Probe it first compares Nonce-1 to '1111' and Check with the IP header TTL/Hop Limit; if the values differ, the router MUST drop the probe without returning a Parcel Reply. 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 router compares the PMTU value with the MTU of the inbound link for the probe and MUST (re)set PMTU to the lower MTU. The router then MUST return a Parcel Reply (see below) and convert the probe into an ordinary IP packet the same as was described previously for routers forwarding to non-parcel-capable links. If the next hop IP link configures a sufficiently large MTU to pass the packet(s), the router 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 router 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 router 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 or an intact Parcel Probe. If the final destination receives ordinary IP packets, it performs any necessary integrity checks then

delivers the packets to upper layers which will return an upper layer probe response if necessary. If the final destination receives a Parcel Probe, it first compares Nonce-1 with '1111' and Check with the IP header TTL/Hop Limit; if the values differ, the final destination MUST drop the probe without returning a Parcel Reply. 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 Parcel Reply and deliver the probe contents to upper layers the same as for an ordinary IP parcel.

When the router or final destination returns a Parcel Reply, it prepares an ICMPv6 PTB message [RFC4443] with Code set to "Parcel Reply" [I-D.templin-6man-omni] and with MTU set to the PMTU value reported in the Parcel Probe. The node then writes its own IP address as the Parcel Reply source and writes the source of the Parcel Probe as the Parcel Reply destination (for IPv4 Parcel Probes, the node writes the Parcel Reply addresses as IPv4-Compatible IPv6 addresses [RFC4291]). The node next copies the leading portion of the Parcel Probe beginning with the IP header into the "packet in error" field without causing the Parcel Reply to exceed 512 octets in length, then calculates the ICMPv6 header checksum. Since IPv6 packets cannot traverse IPv4 paths, and since middleboxes often filter ICMPv6 messages as they traverse IPv6 paths, the node next wraps the Parcel Reply in UDP/IP headers of the correct IP version with the IP 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]. In the process, the node either calculates or omits the UDP checksum as appropriate and (for IPv4) clears the DF bit. The node finally sends the prepared Parcel Reply to the original source of the probe.

After sending a Parcel Probe the original source may therefore receive a UDP/IP encapsulated Parcel Reply (see above) and/or an upper layer protocol probe reply. If the source receives a Parcel Reply, it first verifies the checksum(s) then matches the enclosed PTB message with the original Parcel Probe by examining the Nonce-2 field echoed in the ICMPv6 "packet in error" field containing the leading portion of the probe. If PTB does not match, the source discards the Parcel Reply; otherwise, the source marks the path as "parcels supported" and also records the PMTU value as the maximum parcel size for the forward path to this destination.

After receiving a Parcel Reply, the original source can continue 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: when a Parcel Probe forwarded into an ingress OMNI interface is broken into sub-parcels, each sub-parcel includes its own copy of the Parcel Probe header. When multiple sub-parcels of the same Parcel Probe arrive at an intermediate or egress OMNI interface, the interface re-combines the sub-parcels (if possible) while retaining the Parcel Probe header. It is therefore possible that a single Parcel Probe with multiple upper layer protocol segments could generate multiple Parcel Replys.

Note: The original source includes Nonce-1 and Check fields as the first two octets of Parcel Probes in case a router on the path overwrites the values in a wayward attempt to implement [RFC1063]. Parcel Probe recipients should therefore regard a Nonce-1 value other than '1111' as an indication that the field was either intentionally or accidentally altered during previous hop processing.

Note: The MTU value returned in a Parcel Reply determines only the maximum IP parcel size for the path, while the maximum upper layer protocol segment size may be smaller. The upper layer protocol segment size is instead determined separately according to any upper layer protocol probing.

Note: If the final destination receives a Parcel Probe but does not recognize the parcel construct, it simply drops the probe. The original source will then deem the probe as lost and parcels cannot be used.

9. Integrity

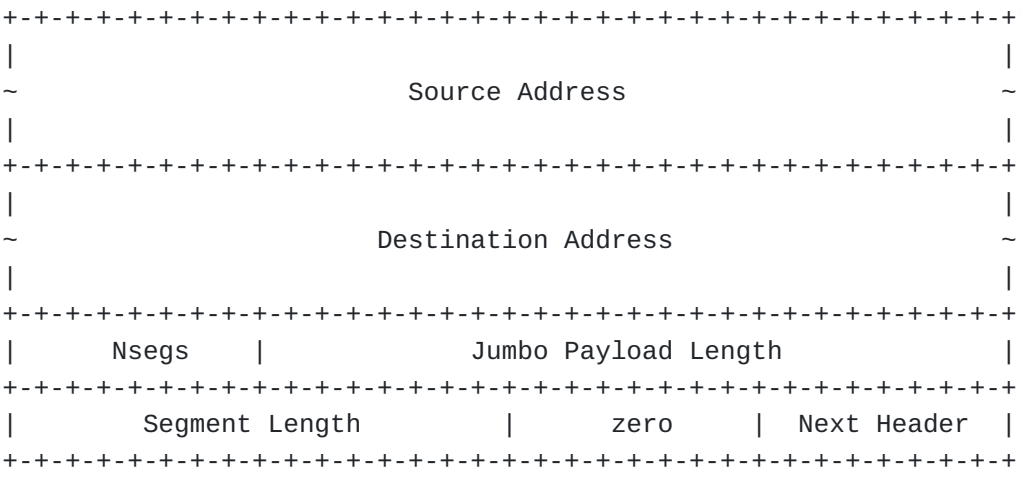
The {TCP,UDP}/IP header plus each segment of a (multi-segment) IP parcel includes its own integrity check. This means that IP parcels can support stronger integrity for the same amount of upper layer protocol data compared to an ordinary IP packet or Jumbogram. The header integrity check can be verified at each hop to ensure that parcels with errored headers are dropped to avoid mis-delivery. The header and per-segment integrity checks must then be verified at the final destination, which accepts any segments with correct integrity while discarding any with incorrect integrity and regarding them as a loss event.

IP parcels can range in length from as small as only the {TCP,UDP}/IP headers themselves to as large as the 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 link-layer integrity checks may be inadvisable for links with significantly larger MTUs and may not be possible at all for links such as tunnels over IPv4 that invoke fragmentation. Moreover, the segment contents of a received parcel may arrive in an incomplete and/or rearranged order with respect to their original packaging.

The parcel header and individual parcel segment checksums may be verified by end system lower layer hardware or software entities. If an entity detects a parcel header or per-segment checksum error, it rewrites the checksum field to 0 and may defer further action to upper layers. Upper layer entities that process received parcels can then discard any entire parcels with header checksum 0, or discard any individual parcel segments with checksum 0, i.e., without performing an explicit checksum calculation.

In order to support the parcel {TCP,UDP}/IP header checksum calculation, IP parcels use the following IP pseudo-header:



where:

- *Source Address is the 16-octet IPv6 or 4-octet IPv4 source address of the prepared parcel.

- *Destination Address is the 16-octet IPv6 or 4-octet IPv4 destination address of the prepared parcel.

- *Jumbo Payload Length is the value found in the Jumbo Payload option of the prepared parcel.

- *Segment Length is the value found in the IPv6 Payload Length or IPv4 Total Length field of the prepared parcel.

- *zero encodes the constant value '0'.

*Next Header is the IP protocol number corresponding to the upper layer protocol, i.e., TCP or UDP.

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 0. 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 reply is likely an authentic response to the probe. In environments where stronger authentication is necessary, the Automatic Extended Route Optimization (AERO) message authentication services can be applied [I-D.templin-6man-aero].

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, DOI 10.17487/RFC0791, RFC 791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

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

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", DOI 10.17487/RFC2119, BCP 14, RFC 2119, 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", DOI 10.17487/RFC4291, RFC 4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[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", DOI 10.17487/RFC8174, RFC 8174, BCP 14, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", DOI 10.17487/RFC8200, RFC 8200, STD 86, 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, March 2022, <https://www.ietf.org/archive/id/draft-ietf-tcpm-rfc793bis-28.txt>.

[I-D.templin-6man-aero]
Templin, F. L., "Automatic Extended Route Optimization (AERO)", Work in Progress, Internet-Draft, draft-templin-6man-aero-58, 23 July 2022, <https://www.ietf.org/archive/id/draft-templin-6man-aero-58.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-69, 23 July 2022, <https://www.ietf.org/archive/id/draft-templin-6man-omni-69.txt>.

[I-D.templin-dtn-ltpfrag] Templin, F. L., "LTP Fragmentation", Work in Progress, Internet-Draft, draft-templin-dtn-ltpfrag-09, 25 July 2022, <https://www.ietf.org/archive/id/draft-templin-dtn-ltpfrag-09.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 C., Kent, C A., 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 C. and S E. Deering, "Path MTU discovery", DOI 10.17487/RFC1191, RFC 1191, 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", DOI 10.17487/RFC7126, RFC 7126, BCP 186, 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", RFC 8201, STD 87, 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", DOI 10.17487/RFC8899, RFC 8899, 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., Birrane, E., and III, "Bundle Protocol Version 7", RFC 9171, DOI 10.17487/RFC9171, January 2022, <https://www.rfc-editor.org/info/rfc9171>.

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