6lo
Internet-Draft
Updates: 4944 (if approved)
Intended status: Standards Track
Expires: November 21, 2019

P. Thubert, Ed.
Cisco Systems

May 20, 2019

6LoWPAN Selective Fragment Recovery
draft-ietf-6lo-fragment-recovery-03

Abstract

This draft updates RFC 4944 with a simple protocol to recover individual fragments across a route-over mesh network, with a minimal flow control to protect the network against bloat.

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 November 21, 2019.

Table of Contents

1. Introduction

In most Low Power and Lossy Network (LLN) applications, the bulk of the traffic consists of small chunks of data (in the order few bytes to a few tens of bytes) at a time. Given that an IEEE Std. 802.15.4 [IEEE.802.15.4] frame can carry 74 bytes or more in all cases, fragmentation is usually not required. However, and though this happens only occasionally, a number of mission critical applications do require the capability to transfer larger chunks of data, for instance to support a firmware upgrades of the LLN nodes or an extraction of logs from LLN nodes. In the former case, the large

chunk of data is transferred to the LLN node, whereas in the latter, the large chunk flows away from the LLN node. In both cases, the size can be on the order of 10Kbytes or more and an end-to-end reliable transport is required.

"Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944] defines the original 6LoWPAN datagram fragmentation mechanism for LLNs. One critical issue with this original design is that routing an IPv6 [RFC8200] packet across a route-over mesh requires to reassemble the full packet at each hop, which may cause latency along a path and an overall buffer bloat in the network. The "6TiSCH Architecture" [I-D.ietf-6tisch-architecture] recommends to use a hop-by-hop fragment forwarding technique to alleviate those undesirable effects. "LLN Minimal Fragment Forwarding" [I-D.ietf-6lo-minimal-fragment] proposes such a technique, in a fashion that is compatible with [RFC4944] without the need to define a new protocol.

However, adding that capability alone to the local implementation of the original 6LoWPAN fragmentation would not address the issues of resources locked and wasted transmissions due to the loss of a fragment. [RFC4944] does not define a mechanism to first discover a fragment loss, and then to recover that loss. With RFC 4944, the forwarding of a whole datagram fails when one fragment is not delivered properly to the destination 6LoWPAN endpoint. Constrained memory resources are blocked on the receiver until the receiver times out.

That problem is exacerbated when forwarding fragments over multiple hops since a loss at an intermediate hop will not be discovered by either the source or the destination, and the source will keep on sending fragments, wasting even more resources in the network and possibly contributing to the condition that caused the loss to no avail since the datagram cannot arrive in its entirety. RFC 4944 is also missing signaling to abort a multi-fragment transmission at any time and from either end, and, if the capability to forward fragments is implemented, clean up the related state in the network. It is also lacking flow control capabilities to avoid participating to a congestion that may in turn cause the loss of a fragment and trigger the retransmission of the full datagram.

This specification proposes a method to forward fragments across a multi-hop route-over mesh, and to recover individual fragments between LLN endpoints. The method is designed to limit congestion loss in the network and addresses the requirements that are detailed in Appendix B.

2. Terminology

2.1. BCP 14

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.

2.2. References

In this document, readers will encounter terms and concepts that are discussed in the following documents:

- "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing" [RFC6606]

2.3. 6LoWPAN Acronyms

This document uses the following acronyms:

6BBR: 6LoWPAN Backbone Router

6LBR: 6LoWPAN Border Router

6LN: 6LoWPAN Node

6LR: 6LoWPAN Router

LLN: Low-Power and Lossy Network

2.4. Referenced Work

Past experience with fragmentation has shown that miss-associated or lost fragments can lead to poor network behavior and, occasionally, trouble at application layer. The reader is encouraged to read "IPv4 Reassembly Errors at High Data Rates" [RFC4963] and follow the references for more information.

That experience led to the definition of "Path MTU discovery" [RFC8201] (PMTUD) protocol that limits fragmentation over the Internet.

Specifically in the case of UDP, valuable additional information can be found in "UDP Usage Guidelines for Application Designers" [RFC8085].

Readers are expected to be familiar with all the terms and concepts that are discussed in "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919] and "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

"The Benefits of Using Explicit Congestion Notification (ECN)" [RFC8087] provides useful information on the potential benefits and pitfalls of using ECN.

Quoting the "Multiprotocol Label Switching (MPLS) Architecture" [RFC3031]: with MPLS, "packets are "labeled" before they are forwarded. At subsequent hops, there is no further analysis of the packet's network layer header. Rather, the label is used as an index into a table which specifies the next hop, and a new label". The MPLS technique is leveraged in the present specification to forward fragments that actually do not have a network layer header, since the fragmentation occurs below IP.

"LLN Minimal Fragment Forwarding" [I-D.ietf-6lo-minimal-fragment] introduces the concept of a Virtual Reassembly Buffer (VRB) and an associated technique to forward fragments as they come, using the Datagram_tag as a label in a fashion similar to MLPS. This specification reuses that technique with slightly modified controls.

2.5. New Terms

This specification uses the following terms:

6LoWPAN endpoints

: The LLN nodes in charge of generating or expanding a 6LoWPAN header from/to a full IPv6 packet. The 6LoWPAN endpoints are the points where fragmentation and reassembly take place.

3. Updating RFC 4944

This specification updates the fragmentation mechanism that is specified in "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944] for use in route-over LLNs by providing a model where fragments can be forwarded end-to-end across a 6LoWPAN LLN, and where fragments that are lost on the way can be recovered individually. A new format for fragment is introduces and new dispatch types are defined in Section 5.

[RFC8138] allows to modify the size of a packet en-route by removing the consumed hops in a compressed Routing Header. It results that the fragment_offset and datagram_size cannot be signaled in the

uncompressed form. This specification expresses those fields in the compressed form and allows to modify them en-route (see Section 4.3.

Note that consistently with in Section 2 of [RFC6282] for the fragmentation mechanism described in Section 5.3 of [RFC4944], any header that cannot fit within the first fragment MUST NOT be compressed when using the fragmentation mechanism described in this specification.

4. Updating draft-ietf-6lo-minimal-fragment

This specification updates the fragment forwarding mechanism specified in "LLN Minimal Fragment Forwarding" [I-D.ietf-6lo-minimal-fragment] by providing additional operations to improve the management of the Virtual Reassembly Buffer (VRB).

4.1. Slack in the First Fragment

At the time of this writing, [I-D.ietf-6lo-minimal-fragment] allows for refragmenting in intermediate nodes, meaning that some bytes from a given fragment may be left in the VRB to be added to the next fragment. The reason for this to happen would be the need for space in the outgoing fragment that was not needed in the incoming fragment, for instance because the 6LoWPAN Header Compression is not as efficient on the outgoing link, e.g., if the Interface ID (IID) of the source IPv6 address is elided by the originator on the first hop because it matches the source MAC address, but cannot be on the next hops because the source MAC address changes.

This specification cannot allow this operation since fragments are recovered end-to-end based on the fragment number. This means that the fragments that contain a 6LoWPAN-compressed header MUST have enough slack to enable a less efficient compression in the next hops that still fits in one MAC frame. For instance, if the IID of the source IPv6 address is elided by the originator, then it MUST compute the fragment_size as if the MTU was 8 bytes less. This way, the next hop can restore the source IID to the first fragment without impacting the second fragment.

4.2. Gap between frames

This specification introduces a concept of Inter-Frame Gap, which is a configurable interval of time between transmissions to a same next hop. In the case of half duplex interfaces, this InterFrameGap ensures that the next hop has progressed the previous frame and is capable of receiving the next one.

In the case of a mesh operating at a single frequency with omnidirectional antennas, a larger InterFrameGap is required protect the frame against hidden terminal collisions with the previous frame of a same flow that is still progressing alon a common path.

The Inter-Frame Gap is useful even for unfragmented datagrams, but it becomes a necessity for fragments that are typically generated in a fast sequence and are all sent over the exact same path.

4.3. Modifying the First Fragment

The compression of the Hop Limit, of the source and destination addresses, and of the Routing Header may change en-route in a Route-Over mesh LLN. If the size of the first fragment is modified, then the intermediate node MUST adapt the datagram_size to reflect that difference.

The intermediate node MUST also save the difference of datagram_size of the first fragment in the VRB and add it to the datagram_size and to the fragment_offset of all the subsequent fragments for that datagram.

5. New Dispatch types and headers

This specification enables the 6LoWPAN fragmentation sublayer to provide an MTU up to 2048 bytes to the upper layer, which can be the 6LoWPAN Header Compression sublayer that is defined in the "Compression Format for IPv6 Datagrams" [RFC6282] specification. In order to achieve this, this specification enables the fragmentation and the reliable transmission of fragments over a multihop 6LoWPAN mesh network.

This specification provides a technique that is derived from MPLS in order to forward individual fragments across a 6LoWPAN route-over mesh. The Datagram_tag is used as a label; it is locally unique to the node that is the source MAC address of the fragment, so together the MAC address and the label can identify the fragment globally. A node may build the Datagram_tag in its own locally-significant way, as long as the selected tag stays unique to the particular datagram for the lifetime of that datagram. It results that the label does not need to be globally unique but also that it must be swapped at each hop as the source MAC address changes.

This specification extends RFC 4944 [RFC4944] with 4 new Dispatch types, for Recoverable Fragment (RFRAG) headers with or without Acknowledgment Request (RFRAG vs. RFRAG-ARQ), and for the RFRAG Acknowledgment back, with or without ECN Echo (RFRAG-ACK vs. RFRAG-ECHO).

(to be confirmed by IANA) The new 6LoWPAN Dispatch types use the Value Bit Pattern of 11 1010xx from page 0 [RFC8025], as follows:

Pattern	Header Type
11 10100x	RFRAG - Recoverable Fragment
11 10101x	RFRAG-ACK - RFRAG Acknowledgment

Figure 1: Additional Dispatch Value Bit Patterns

In the following sections, a "Datagram_tag" extends the semantics defined in [RFC4944] Section 5.3."Fragmentation Type and Header". The Datagram_tag is a locally unique identifier for the datagram from the perspective of the sender. This means that the datagram-tag identifies a datagram uniquely in the network when associated with the source of the datagram. As the datagram gets forwarded, the source changes and the Datagram_tag must be swapped as detailed in [I-D.ietf-6lo-minimal-fragment].

5.1. Recoverable Fragment Dispatch type and Header

In this specification, the size and offset of the fragments are expressed on the compressed packet form as opposed to the uncompressed - native - packet form.

The format of the fragment header is the same for all fragments. The format indicates both a length and an offset, which seem be redundant with the sequence field, but is not. The position of a fragment in the recomposition buffer is neither correlated with the value of the sequence field nor with the order in which the fragments are received. This enables out-of-sequence and overlapping fragments, e.g., a fragment 5 that is retried as smaller fragments 5, 13 and 14 due to a change of MTU.

There is no requirement on the receiver to check for contiguity of the received fragments, and the sender MUST ensure that when all fragments are acknowledged, then the datagram is fully received. This may be useful in particular in the case where the MTU changes and a fragment sequence is retried with a smaller fragment_size, the remainder of the original fragment being retried with new sequence values.

The first fragment is recognized by a sequence of 0; it carries its fragment_size and the datagram_size of the compressed packet, whereas the other fragments carry their fragment_size and fragment_offset.

The last fragment for a datagram is recognized when its fragment_offset and its fragment_size add up to the datagram_size.

Recoverable Fragments are sequenced and a bitmap is used in the RFRAG Acknowledgment to indicate the received fragments by setting the individual bits that correspond to their sequence.

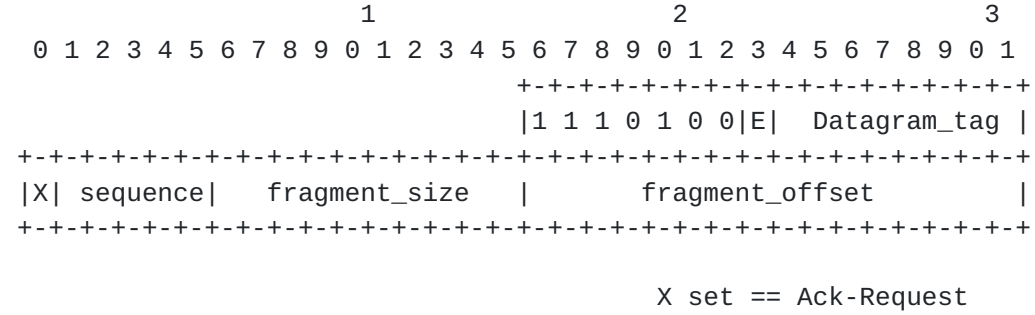

Figure 2: RFRAG Dispatch type and Header

E: 1 bit; Explicit Congestion Notification; the "E" flag is reset by the source of the fragment and set by intermediate routers to signal that this fragment experienced congestion along its path.

Fragment_size: 10 bit unsigned integer; the size of this fragment in a unit that depends on the MAC layer technology. By default, that unit is the octet which allows fragments up to 512 bytes. For IEEE Std. 802.15.4, the unit is octet, and the maximum fragment size, when it is constrained by the maximum frame size of 128 octet minus the overheads of the MAC and Fragment Headers, is not limited by this encoding.

X: 1 bit; Ack-Request: when set, the sender requires an RFRAG Acknowledgment from the receiver.

Sequence: 5 bit unsigned integer; the sequence number of the fragment in the acknowledgement bitmap. Fragments are numbered [0..N] where N is in [0..31]. A Sequence of 0 indicates the first fragment in a datagram, but non-zero values are not indicative of the position in the recomposition buffer.

Fragment_offset: 16 bit unsigned integer;

* When the Fragment_offset is set to a non-0 value, its semantics depend on the value of the Sequence field.

  + For a first fragment (i.e. with a Sequence of 0), this field indicates the datagram_size of the compressed datagram, to help the receiver allocate an adapted buffer for the

reception and reassembly operations. The fragment may stored for local recomposition, or it may be routed based on the destination IPv6 address, in which case a VRB state must be installed as described in Section 6.1.1.

+ When the Sequence is not 0, this field indicates the offset of the fragment in the compressed form. The fragment may be added to a local recomposition buffer or forwarded based on an existing VRB as described in Section 6.1.2.

* A Fragment_offset that is set to a value of 0 indicates an abort condition and all state regarding the datagram should be cleaned up once the processing of the fragment is complete; the processing of the fragment depends on whether there is a VRB already established for this datagram, and the next hop is still reachable:

  + if a VRB already exists and is not broken, the fragment is to be forwarded along the associated Label Switched Path (LSP) as described in Section 6.1.2, but regardless of the value of the Sequence field;

  + else, if the Sequence is 0, then the fragment is to be routed as described in Section 6.1.1 but no state is conserved afterwards. In that case, the session if it exists is aborted and the packet is also forwarded in an attempt to clean up the next hops as along the path indicated by the IPv6 header (possibly including a routing header).

  If the fragment cannot be forwarded or routed, then an abort RFRAG-ACK is sent back to the source.

5.2. RFRAG Acknowledgment Dispatch type and Header

This specification also defines a 4-octet RFRAG Acknowledgment bitmap that is used by the reassembling end point to confirm selectively the reception of individual fragments. A given offset in the bitmap maps one to one with a given sequence number.

The offset of the bit in the bitmap indicates which fragment is acknowledged as follows:

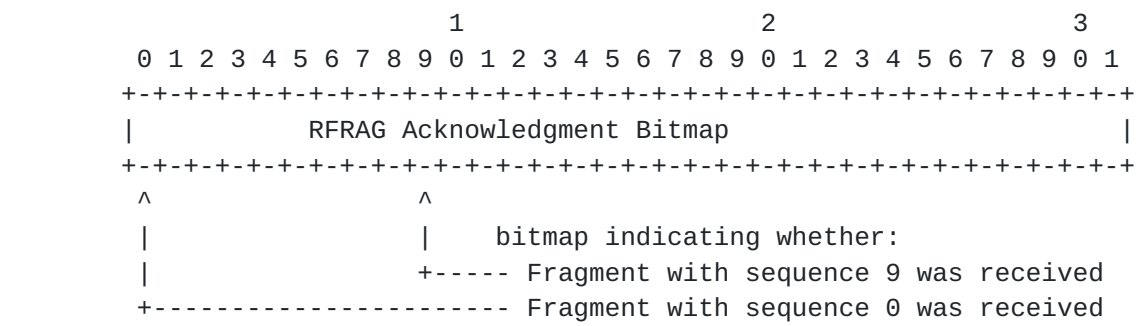

Figure 3: RFRAG Acknowledgment bitmap encoding

Figure 4 shows an example Acknowledgment bitmap which indicates that all fragments from sequence 0 to 20 were received, except for fragments 1, 2 and 16 that were lost and must be retried.

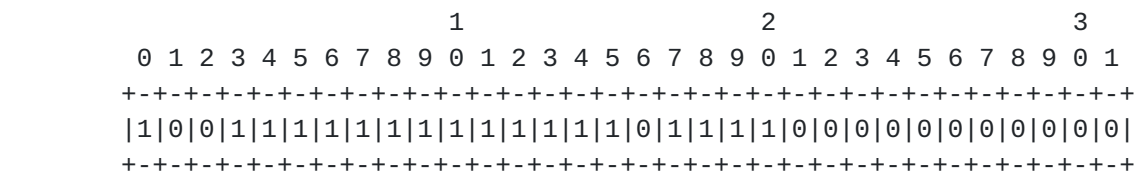

Figure 4: Example RFRAG Acknowledgment Bitmap

The RFRAG Acknowledgment Bitmap is included in a RFRAG Acknowledgment header, as follows:

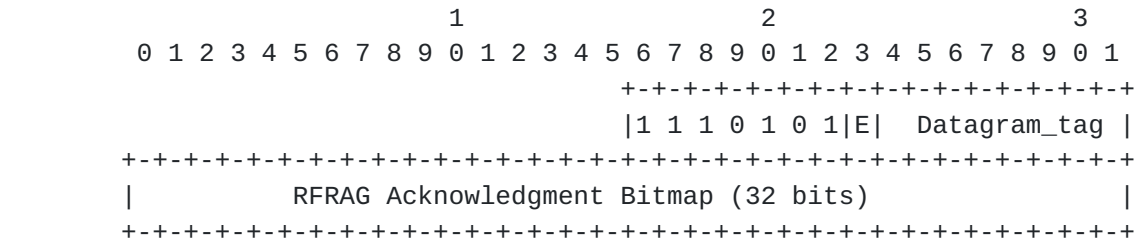

Figure 5: RFRAG Acknowledgment Dispatch type and Header

E: 1 bit; Explicit Congestion Notification Echo

When set, the sender indicates that at least one of the acknowledged fragments was received with an Explicit Congestion Notification, indicating that the path followed by the fragments is subject to congestion. More in Appendix C.

RFRAG Acknowledgment Bitmap

An RFRAG Acknowledgment Bitmap, whereby setting the bit at offset x indicates that fragment x was received, as shown in Figure 3. All 0's is a NULL bitmap that indicates that the fragmentation process is aborted. All 1's is a FULL bitmap that indicates that the fragmentation process is complete, all fragments were received at the reassembly end point.

6. Fragments Recovery

The Recoverable Fragment headers RFRAG and RFRAG-ARQ are used to transport a fragment and optionally request an RFRAG Acknowledgment that will confirm the good reception of a one or more fragments. An RFRAG Acknowledgment is carried as a standalone header in a message that is sent back to the 6LoWPAN endpoint that was the source of the fragments, as known by its MAC address. The process ensures that at every hop, the source MAC address and the Datagram_tag in the received fragment are enough information to send the RFRAG Acknowledgment back towards the source 6LoWPAN endpoint by reversing the MPLS operation.

The 6LoWPAN endpoint that fragments the packets at 6LoWPAN level (the sender) also controls the amount of acknowledgments by setting the Ack-Request flag in the RFRAG packets. The sender may set the Ack-Request flag on any fragment to perform congestion control by limiting the number of outstanding fragments, which are the fragments that have been sent but for which reception or loss was not positively confirmed by the reassembling endpoint. Te maximum number of outstanding fragments is the Window-Size. It is configurable and may vary in case of ECN notification. When it receives a fragment with the Ack-Request flag set, the 6LoWPAN endpoint that reassembles the packets at 6LoWPAN level (the receiver) MUST send back an RFRAG Acknowledgment to confirm reception of all the fragments it has received so far.

The Ack-Request bit marks the end of a window. It SHOULD be set on the last fragment to protect the datagram, and MAY be used in intermediate fragments for the purpose of flow control. This ARQ process MUST be protected by a ARQ timer, and the fragment that carries the Ack-Request flag MAY be retried upon time out a configurable amount of times. Upon exhaustion of the retries the sender may either abort the transmission of the datagram or retry the datagram from the first fragment with an Ack-Request in order to reestablish a path and discover which fragments were received over the old path. When the sender of the fragment knows that an underlying link-layer mechanism protects the fragments, it may refrain from using the RFRAG Acknowledgment mechanism, and never set the Ack-Request bit.

The RFRAG Acknowledgment can optionally carry an ECN indication for flow control (see Appendix C). The receiver of a fragment with the 'E' (ECN) flag set MUST echo that information by setting the 'E' (ECN) flag in the next RFRAG Acknowledgment.

The sender transfers a controlled number of fragments and MAY flag the last fragment of a series with an RFRAG Acknowledgment Request. The receiver MUST acknowledge a fragment with the acknowledgment request bit set. If any fragment immediately preceding an acknowledgment request is still missing, the receiver MAY intentionally delay its acknowledgment to allow in-transit fragments to arrive. Delaying the acknowledgment might defeat the round trip delay computation so it should be configurable and not enabled by default.

The receiver MAY issue unsolicited acknowledgments. An unsolicited acknowledgment signals to the sender endpoint that it can resume sending if it had reached its maximum number of outstanding fragments. Another use is to inform that the reassembling endpoint has canceled the process of an individual datagram. Note that acknowledgments might consume precious resources so the use of unsolicited acknowledgments should be configurable and not enabled by default.

An observation is that streamlining forwarding of fragments generally reduces the latency over the LLN mesh, providing room for retries within existing upper-layer reliability mechanisms. The sender protects the transmission over the LLN mesh with a retry timer that is computed according to the method detailed in [RFC6298]. It is expected that the upper layer retries obey the recommendations in "UDP Usage Guidelines" [RFC8085], in which case a single round of fragment recovery should fit within the upper layer recovery timers.

Fragments are sent in a round robin fashion: the sender sends all the fragments for a first time before it retries any lost fragment; lost fragments are retried in sequence, oldest first. This mechanism enables the receiver to acknowledge fragments that were delayed in the network before they are retried.

When a single frequency is used by contiguous hops, the sender should wait a reasonable amount of time between fragments so as to let a fragment progress a few hops and avoid hidden terminal issues. This precaution is not required on channel hopping technologies such as Time Slotted Channel Hopping (TSCH) [RFC6554]

6.1. Forwarding Fragments

It is assumed that the first Fragment is large enough to carry the IPv6 header and make routing decisions. If that is not so, then this specification MUST NOT be used.

This specification extends the Virtual Reassembly Buffer (VRB) technique to forward fragments with no intermediate reconstruction of the entire packet. It inherits operations like Datagram_tag Switching and using a timer to clean the VRB when the traffic dries up. In more details, the first fragment carries the IP header and it is routed all the way from the fragmenting end point to the reassembling end point. Upon the first fragment, the routers along the path install a label-switched path (LSP), and the following fragments are label-switched along that path. As a consequence, alternate routes not possible for individual fragments. The Datagram_tag is used to carry the label, that is swapped at each hop. All fragments follow the same path and fragments are delivered in the order at which they are sent.

6.1.1. Upon the first fragment

In Route-Over mode, the source and destination MAC addressed in a frame change at each hop. The label that is formed and placed in the Datagram_tag is associated to the source MAC and only valid (and unique) for that source MAC. Upon a first fragment (i.e. with a sequence of zero), a VRB and the associated LSP state are created for the tuple (source MAC address, Datagram_tag) and the fragment is forwarded along the IPv6 route that matches the destination IPv6 address in the IPv6 header as prescribed by [I-D.ietf-6lo-minimal-fragment]. The LSP state enables to match the (previous MAC address, Datagram_tag) in an incoming fragment to the tuple (next MAC address, swapped Datagram_tag) used in the forwarded fragment and points at the VRB. In addition, the router also forms a Reverse LSP state indexed by the MAC address of the next hop and the swapped Datagram_tag. This reverse LSP state also points at the VRB and enables to match the (next MAC address, swapped_Datagram_tag) found in an RFRAG Acknowledgment to the tuple (previous MAC address, Datagram_tag) used when forwarding a Fragment Acknowledgment (RFRAG-ACK) back to the sender endpoint.

6.1.2. Upon the next fragments

Upon a next fragment (i.e. with a non-zero sequence), the router looks up a LSP indexed by the tuple (MAC address, Datagram_tag) found in the fragment. If it is found, the router forwards the fragment using the associated VRB as prescribed by [I-D.ietf-6lo-minimal-fragment].

if the VRB for the tuple is not found, the router builds an RFRAG-ACK to abort the transmission of the packet. The resulting message has the following information:

- The source and destination MAC addresses are swapped from those found in the fragment
- The Datagram_tag set to the Datagram_tag found in the fragment
- A NULL bitmap is used to signal the abort condition

At this point the router is all set and can send the RFRAG-ACK back to the previous router. The RFRAG-ACK should normally be forwarded all the way to the source using the reverse LSP state in the VRBs in the intermediate routers as described in the next section.

6.2. Upon the RFRAG Acknowledgments

Upon an RFRAG-ACK, the router looks up a Reverse LSP indexed by the tuple (MAC address, Datagram_tag), which are respectively the source MAC address of the received frame and the received Datagram_tag. If it is found, the router forwards the fragment using the associated VRB as prescribed by [I-D.ietf-6lo-minimal-fragment], but using the Reverse LSP so that the RFRAG-ACK flows back to the sender endpoint.

If the Reverse LSP is not found, the router MUST silently drop the RFRAG-ACK message.

Either way, if the RFRAG-ACK indicates that the fragment was entirely received (FULL bitmap), it arms a short timer, and upon timeout, the VRB and all the associated state are destroyed. until the timer elapses, fragments of that datagram may still be received, e.g. if the RFRAG-ACK was lost on the way back and the source retried the last fragment. In that case, the router forwards the fragment according to the state in the VRB.

This specification does not provide a method to discover the number of hops or the minimal value of MTU along those hops. But should the minimal MTU decrease, it is possible to retry a long fragment (say sequence of 5) with first a shorter fragment of the same sequence (5 again) and then one or more other fragments with a sequence that was not used before (e.g., 13 and 14).

6.3. Cancelling a Fragmented Packet

A reset is signaled on the forward path with a pseudo fragment that has the fragment_offset, sequence and fragment_size all set to 0, and no data.

When the sender or a router on the way decides that a packet should be dropped and the fragmentation process canceled, it generates a reset pseudo fragment and forwards it down the fragment path.

Each router next along the path the way forwards the pseudo fragment based on the VRB state. If an acknowledgment is not requested, the VRB and all associated state are destroyed.

Upon reception of the pseudo fragment, the receiver cleans up all resources for the packet associated to the Datagram_tag. If an acknowledgment is requested, the receiver responds with a NULL bitmap.

The other way around, the receiver might need to cancel the process of a fragmented packet for internal reasons, for instance if it is out of reassembly buffers, or considers that this packet is already fully reassembled and passed to the upper layer. In that case, the receiver SHOULD indicate so to the sender with a NULL bitmap in a RFRAG Acknowledgment. Upon an acknowledgment with a NULL bitmap, the sender endpoint MUST abort the transmission of the fragmented datagram.

7. Management Considerations

7.1. Protocol Parameters

There is no particular configuration on the receiver, as echoing ECN should always be on. The configuration only applies to the sender that is in control of the transmission. The management system SHOULD be capable of providing the parameters below:

MinFragmentSize: The MinFragmentSize is the minimum value for the Fragment_Size.

OptFragmentSize: The MinFragmentSize is the value for the Fragment_Size that the sender should use to start with.

MaxFragmentSize: The MaxFragmentSize is the maximum value for the Fragment_Size. It MUST be lower than the minimum MTU along the path. A large value augments the chances of buffer bloat and transmission loss. The value MUST be less than 512 if the unit that is defined for the PHY layer is the octet.

UseECN: Indicates whether the sender should react to ECN. When the sender reacts to ECN the Window_Size will vary between MinWindowSize and MaxWindowSize.

MinWindowSize: The minimum value of Window_Size that the sender can use.

OptWindowSize: The OptWindowSize is the value for the Window_Size that the sender should use to start with.

MaxWindowSize: The maximum value of Window_Size that the sender can use. The value MUSt be less than 32.

UseECN: Indicates whether the sender should react to ECN. When the sender reacts to ECN the sender SHOULD adapt the Window_Size between MinWindowSize and MaxWindowSize and it MAY adapt the Fragment_Size if that is supported.

InterFrameGap: Indicates a minimum amount of time between transmissions. All packets to a same destination, and in particular fragments, may be subject to receive while transmitting and hidden terminal collisions with the next or the previous transmission as the fragments progress along a same path. The InterFrameGap protects the propagation of one transmission before the next one is triggered and creates a duty cycle that controls the ratio of air time and memory in intermediate nodes that a particular datagram will use.

MinARQTimeOut: The maximum amount of time a node should wait for an RFRAG Acknowledgment before it takes a next action.

OptARQTimeOut: The starting point of the value of the amount that a sender should wait for an RFRAG Acknowledgment before it takes a next action.

MaxARQTimeOut: The maximum amount of time a node should wait for an RFRAG Acknowledgment before it takes a next action.

MaxFragRetries: The maximum number of retries for a particular Fragment.

MaxDatagramRetries: The maximum number of retries from scratch for a particular Datagram.

7.2. Observing the network

The management system should monitor the amount of retries and of ECN settings that can be observed from the perspective of the both the sender and the receiver, and may tune the optimum size of Fragment_Size and of the Window_Size, OptWindowSize and OptWindowSize respectively, at the sender. The values should be bounded by the expected number of hops and reduced beyond that when the number of

datagrams that can traverse an intermediate point may exceed its capacity and cause a congestion loss. The InterFrameGap is another tool that can be used to increase the spacing between fragments of a same datagram and reduce the ratio of time when a particular intermediate node holds a fragment of that datagram.

8. Security Considerations

The process of recovering fragments does not appear to create any opening for new threat compared to "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

9. IANA Considerations

Need extensions for formats defined in "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

10. Acknowledgments

The author wishes to thank Michel Veillette, Dario Tedeschi, Laurent Toutain, Thomas Watteyne and Michael Richardson for in-depth reviews and comments. Also many thanks to Jonathan Hui, Jay Werb, Christos Polyzois, Soumitri Kolavennu, Pat Kinney, Margaret Wasserman, Richard Kelsey, Carsten Bormann and Harry Courtice for their various contributions.

11. References

11.1. Normative References

[I-D.ietf-6lo-minimal-fragment]
Watteyne, T., Bormann, C., and P. Thubert, "LLN Minimal Fragment Forwarding", draft-ietf-6lo-minimal-fragment-01 (work in progress), March 2019.

[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>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <https://www.rfc-editor.org/info/rfc6282>.

[RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <https://www.rfc-editor.org/info/rfc6554>.

[RFC8025] Thubert, P., Ed. and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch", RFC 8025, DOI 10.17487/RFC8025, November 2016, <https://www.rfc-editor.org/info/rfc8025>.

[RFC8138] Thubert, P., Ed., Bormann, C., Toutain, L., and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, DOI 10.17487/RFC8138, April 2017, <https://www.rfc-editor.org/info/rfc8138>.

[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>.

11.2. Informative References

[I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-20 (work in progress), March 2019.

[IEEE.802.15.4]
IEEE, "IEEE Standard for Low-Rate Wireless Networks", IEEE Standard 802.15.4, DOI 10.1109/IEEE P802.15.4-REVd/D01, <http://ieeexplore.ieee.org/document/7460875/>.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <https://www.rfc-editor.org/info/rfc2914>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

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

[RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, DOI 10.17487/RFC4919, August 2007, <https://www.rfc-editor.org/info/rfc4919>.

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

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <https://www.rfc-editor.org/info/rfc6298>.

[RFC6606] Kim, E., Kaspar, D., Gomez, C., and C. Bormann, "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing", RFC 6606, DOI 10.17487/RFC6606, May 2012, <https://www.rfc-editor.org/info/rfc6606>.

[RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <https://www.rfc-editor.org/info/rfc7554>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <https://www.rfc-editor.org/info/rfc7567>.

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

[RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <https://www.rfc-editor.org/info/rfc8087>.

[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>.

[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>.

Appendix A. Rationale

There are a number of uses for large packets in Wireless Sensor Networks. Such usages may not be the most typical or represent the largest amount of traffic over the LLN; however, the associated functionality can be critical enough to justify extra care for ensuring effective transport of large packets across the LLN.

The list of those usages includes:

Towards the LLN node:

- Firmware update: For example, a new version of the LLN node software is downloaded from a system manager over unicast or multicast services. Such a reflashing operation typically involves updating a large number of similar LLN nodes over a relatively short period of time.

- Packages of Commands: A number of commands or a full configuration can be packaged as a single message to ensure consistency and enable atomic execution or complete roll back. Until such commands are fully received and interpreted, the intended operation will not take effect.

From the LLN node:

- Waveform captures: A number of consecutive samples are measured at a high rate for a short time and then transferred from a sensor to a gateway or an edge server as a single large report.

- Data logs: LLN nodes may generate large logs of sampled data for later extraction. LLN nodes may also generate system logs to assist in diagnosing problems on the node or network.

Large data packets: Rich data types might require more than one fragment.

Uncontrolled firmware download or waveform upload can easily result in a massive increase of the traffic and saturate the network.

When a fragment is lost in transmission, the lack of recovery in the original fragmentation system of RFC 4944 implies that all fragments are resent, further contributing to the congestion that caused the initial loss, and potentially leading to congestion collapse.

This saturation may lead to excessive radio interference, or random early discard (leaky bucket) in relaying nodes. Additional queuing and memory congestion may result while waiting for a low power next hop to emerge from its sleeping state.

Considering that RFC 4944 defines an MTU is 1280 bytes and that in most incarnations (but 802.15.4g) a IEEE Std. 802.15.4 frame can limit the MAC payload to as few as 74 bytes, a packet might be fragmented into at least 18 fragments at the 6LoWPAN shim layer. Taking into account the worst-case header overhead for 6LoWPAN Fragmentation and Mesh Addressing headers will increase the number of required fragments to around 32. This level of fragmentation is much higher than that traditionally experienced over the Internet with IPv4 fragments. At the same time, the use of radios increases the probability of transmission loss and Mesh-Under techniques compound that risk over multiple hops.

Mechanisms such as TCP or application-layer segmentation could be used to support end-to-end reliable transport. One option to support bulk data transfer over a frame-size-constrained LLN is to set the Maximum Segment Size to fit within the link maximum frame size. Doing so, however, can add significant header overhead to each 802.15.4 frame. In addition, deploying such a mechanism requires that the end-to-end transport is aware of the delivery properties of the underlying LLN, which is a layer violation, and difficult to achieve from the far end of the IPv6 network.

Appendix B. Requirements

For one-hop communications, a number of Low Power and Lossy Network (LLN) link-layers propose a local acknowledgment mechanism that is enough to detect and recover the loss of fragments. In a multihop environment, an end-to-end fragment recovery mechanism might be a good complement to a hop-by-hop MAC level recovery. This draft introduces a simple protocol to recover individual fragments between 6LoWPAN endpoints that may be multiple hops away. The method addresses the following requirements of a LLN:

Number of fragments

The recovery mechanism must support highly fragmented packets, with a maximum of 32 fragments per packet.

Minimum acknowledgment overhead

Because the radio is half duplex, and because of silent time spent in the various medium access mechanisms, an acknowledgment consumes roughly as many resources as data fragment.

The new end-to-end fragment recovery mechanism should be able to acknowledge multiple fragments in a single message and not require an acknowledgment at all if fragments are already protected at a lower layer.

Controlled latency

The recovery mechanism must succeed or give up within the time boundary imposed by the recovery process of the Upper Layer Protocols.

Optional congestion control

The aggregation of multiple concurrent flows may lead to the saturation of the radio network and congestion collapse.

The recovery mechanism should provide means for controlling the number of fragments in transit over the LLN.

Appendix C. Considerations On Flow Control

Considering that a multi-hop LLN can be a very sensitive environment due to the limited queuing capabilities of a large population of its nodes, this draft recommends a simple and conservative approach to Congestion Control, based on TCP congestion avoidance.

Congestion on the forward path is assumed in case of packet loss, and packet loss is assumed upon time out. The draft allows to control the number of outstanding fragments, that have been transmitted but for which an acknowledgment was not received yet. It must be noted that the number of outstanding fragments should not exceed the number of hops in the network, but the way to figure the number of hops is out of scope for this document.

Congestion on the forward path can also be indicated by an Explicit Congestion Notification (ECN) mechanism. Though whether and how ECN [RFC3168] is carried out over the LoWPAN is out of scope, this draft

provides a way for the destination endpoint to echo an ECN indication back to the source endpoint in an acknowledgment message as represented in Figure 5 in Section 5.2.

It must be noted that congestion and collision are different topics. In particular, when a mesh operates on a same channel over multiple hops, then the forwarding of a fragment over a certain hop may collide with the forwarding of a next fragment that is following over a previous hop but in a same interference domain. This draft enables an end-to-end flow control, but leaves it to the sender stack to pace individual fragments within a transmit window, so that a given fragment is sent only when the previous fragment has had a chance to progress beyond the interference domain of this hop. In the case of 6TiSCH [I-D.ietf-6tisch-architecture], which operates over the TimeSlotted Channel Hopping [RFC7554] (TSCH) mode of operation of IEEE802.14.5, a fragment is forwarded over a different channel at a different time and it makes full sense to transmit the next fragment as soon as the previous fragment has had its chance to be forwarded at the next hop.

From the standpoint of a source 6LoWPAN endpoint, an outstanding fragment is a fragment that was sent but for which no explicit acknowledgment was received yet. This means that the fragment might be on the way, received but not yet acknowledged, or the acknowledgment might be on the way back. It is also possible that either the fragment or the acknowledgment was lost on the way.

From the sender standpoint, all outstanding fragments might still be in the network and contribute to its congestion. There is an assumption, though, that after a certain amount of time, a frame is either received or lost, so it is not causing congestion anymore. This amount of time can be estimated based on the round trip delay between the 6LoWPAN endpoints. The method detailed in [RFC6298] is recommended for that computation.

The reader is encouraged to read through "Congestion Control Principles" [RFC2914]. Additionally [RFC7567] and [RFC5681] provide deeper information on why this mechanism is needed and how TCP handles Congestion Control. Basically, the goal here is to manage the amount of fragments present in the network; this is achieved by to reducing the number of outstanding fragments over a congested path by throttling the sources.

Section 6 describes how the sender decides how many fragments are (re)sent before an acknowledgment is required, and how the sender adapts that number to the network conditions.

Author's Address

Pascal Thubert (editor)
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
FRANCE

Phone: +33 497 23 26 34
Email: pthubert@cisco.com