Workgroup: 6lo
Updates: 4944 (if approved)
Published: 11 February 2020
Intended Status: Standards Track
Expires: 14 August 2020
Authors: P. Thubert, Ed.
Cisco Systems

6LoWPAN Selective Fragment Recovery

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 will expire on 14 August 2020.

Copyright Notice

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 (on the order of a 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 a payload of 74 bytes or more, 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 the firmware upgrade of the LLN nodes or the 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 10 kilobytes 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 reassembling 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 using a fragment forwarding (FF) technique to alleviate those undesirable effects. "LLN Minimal Fragment Forwarding" [I-D.ietf-6lo-minimal-fragment] specifies the general behavior that all FF techniques including this specification follow, and presents the associated caveats. In particular, the routing information is fully indicated in the first fragment, which is always forwarded first. A state is formed and used to forward all the next fragments along the same path. The datagram_tag is locally significant to the Layer-2 source of the packet and is swapped at each hop.

"Virtual reassembly buffers in 6LoWPAN" [I-D.ietf-lwig-6lowpan-virtual-reassembly] (VRB) proposes a FF technique 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 inherent fragility of fragmentation (see [I-D.ietf-intarea-frag-fragile]) in particular the issues of resources locked on the receiver and the wasted transmissions due to the loss of a single fragment in a whole datagram. [Kent] compares the unreliable delivery of fragments with a mechanism it calls "selective acknowledgements" that recovers the loss of a fragment individually. The paper illustrates the benefits that can be derived from such a method in figures 1, 2 and 3, on pages 6 and 7. [RFC4944] has no selective recovery and the whole datagram fails when one fragment is not delivered to the destination 6LoWPAN endpoint. Constrained memory resources are blocked on the receiver until the receiver times out, possibly causing the loss of subsequent packets that cannot be received for the lack of buffers.

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 in congestion that may in turn cause the loss of a fragment and potentially the retransmission of the full datagram.

This specification provides a method to forward fragments across a multi-hop route-over mesh, and a selective acknowledgment to recover individual fragments between 6LoWPAN 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 "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing" [RFC6606]

"LLN Minimal Fragment Forwarding" [I-D.ietf-6lo-minimal-fragment] introduces the generic concept of a Virtual Reassembly Buffer (VRB) and specifies behaviours and caveats that are common to a large family of FF techniques including this, which fully inherits from that specification.

Past experience with fragmentation has shown that misassociated or lost fragments can lead to poor network behavior and, occasionally, trouble at the 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' along a Label Switched Path (LSP). 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.

2.3. 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.

Compressed Form: This specification uses the generic term Compressed Form to refer to the format of a datagram after the action of [RFC6282] and possibly [RFC8138] for RPL [RFC6550] artifacts.

datagram_size: The size of the datagram in its Compressed Form before it is fragmented. The datagram_size is expressed in a unit that depends on the MAC layer technology, by default a byte.

datagram_tag: An identifier of a datagram that is locally unique to the Layer-2 sender. Associated with the MAC address of the sender, this becomes a globally unique identifier for the datagram.

fragment_offset: The offset of a particular fragment of a datagram in its Compressed Form. The fragment_offset is expressed in a unit that depends on the MAC layer technology and is by default a byte.

RFRAG: Recoverable Fragment

RFRAG-ACK: Recoverable Fragment Acknowledgement

RFRAG Acknowledgment Request: An RFRAG with the Acknowledgement Request flag ('X' flag) set.

NULL bitmap: Refers to a bitmap with all bits set to zero.

FULL bitmap: Refers to a bitmap with all bits set to one.

Forward: The direction of a LSP path, followed by the RFRAG.

Reverse: The reverse direction of a LSP path, taken by the RFRAG-ACK.

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 fragments is introduced and new dispatch types are defined in Section 5.

[RFC8138] allows modifying the size of a packet en route by removing the consumed hops in a compressed Routing Header. This requires that fragment_offset and datagram_size (see Section 2.3) are also modified en route, which is difficult to do in the uncompressed form. This specification expresses those fields in the Compressed Form and allows modifying them en route (see Section 4.3) easily.

Note that consistent with 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. Extending draft-ietf-6lo-minimal-fragment

This specification implements the generic FF technique specified in "LLN Minimal Fragment Forwarding" [I-D.ietf-6lo-minimal-fragment] in a fashion that enables end-to-end recovery of fragments and some degree of flow control.

4.1. Slack in the First Fragment

[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 happening 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 a sequence 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 an 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 inter-frame gap ensures that the next hop has completed processing of 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 inter-frame gap is required to protect the frame against hidden terminal collisions with the previous frame of a same flow that is still progressing along 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 in the IPv6 Header, 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 to forward individual fragments across a 6LoWPAN route-over mesh without reassembly at each hop. The datagram_tag is used as a label; it is locally unique to the node that owns 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 chosen datagram_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 2 new Dispatch types, for Recoverable Fragment (RFRAG) and for the RFRAG Acknowledgment back. The new 6LoWPAN Dispatch types are taken from Page 0 [RFC8025] as indicated in Table 1 in Section 9.

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, if the packet is compressed then the size and offset of the fragments are expressed with respect to the Compressed Form of the packet form as opposed to the uncompressed (native) packet form.

The format of the fragment header is shown in Figure 1. It is the same for all fragments. The format has a length and an offset, as well as a sequence field. This would be redundant if the offset was computed as the product of the sequence by the length, but this is not the case. The position of a fragment in the reassembly 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 subfragmenting, e.g., a fragment seq. 5 that is retried end-to-end as smaller fragments seq. 5, 13 and 14 due to a change of MTU along the path between the 6LoWPAN endpoints.

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                 |1 1 1 0 1 0 0|E|  datagram_tag |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |X| sequence|   Fragment_Size   |       fragment_offset         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                         X set == Ack-Request
```

Figure 1: RFRAG Dispatch type and Header

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 before it is fragmented, 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.

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

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. Unless overridden by a more specific specification, that unit is the octet, which allows fragments up to 512 bytes.

datagram_tag: 8 bits; an identifier of the datagram that is locally unique to the sender.

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 reassembly 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 as follows:

* 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 be stored for local reassembly. Alternatively, it may be routed based on the destination IPv6 address. In that 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 of the datagram. The fragment may be added to a local reassembly 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 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 as described in Section 6.1.2.

5.2. RFRAG Acknowledgment Dispatch type and Header

This specification also defines a 4-octet RFRAG Acknowledgment bitmap that is used by the reassembling endpoint to confirm selectively the reception of individual fragments. A given offset in the bitmap maps one-to-one with a given sequence number and indicates which fragment is acknowledged as follows:

```
                    1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |            RFRAG Acknowledgment Bitmap                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  ^                 ^
  |                 |    bitmap indicating whether:
  |                 +----- Fragment with sequence 9 was received
  +----------------------- Fragment with sequence 0 was received
```

Figure 2: RFRAG Acknowledgment Bitmap Encoding

Figure 3 shows an example Acknowledgment bitmap which indicates that all fragments from sequence 0 to 20 were received, except for fragments 1, 2 and 16 were lost and must be retried.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|0|0|1|1|1|1|1|1|1|1|1|1|1|1|1|0|1|1|1|1|0|0|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Example RFRAG Acknowledgment Bitmap

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

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |1 1 1 0 1 0 1|E|  datagram_tag |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          RFRAG Acknowledgment Bitmap (32 bits)                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: 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 2. A NULL bitmap that indicates that the fragmentation process is aborted. A FULL bitmap that indicates that the fragmentation process is complete, all fragments were received at the reassembly endpoint.

6. Fragments Recovery

The Recoverable Fragment header RFRAG is used to transport a fragment and optionally request an RFRAG Acknowledgment that will confirm the good reception of one or more fragments. An RFRAG Acknowledgment is carried as a standalone fragment header (i.e., with no 6LoWPAN payload) in a message that is propagated back to the 6LoWPAN endpoint that was the originator of the fragments. To achieve this, each hop that performed an MPLS-like operation on fragments reverses that operation for the RFRAG_ACK by sending a frame from the next hop to the previous hop as known by its MAC address in the VRB. The datagram_tag in the RFRAG_ACK is unique to the receiver and is enough information for an intermediate hop to locate the VRB that contains the datagram_tag used by the previous hop and the Layer-2 information associated to it (interface and MAC address).

The 6LoWPAN endpoint that fragments the packets at the 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. The maximum number of outstanding fragments is controlled by the Window-Size. It is configurable and may vary in case of ECN notification. When the 6LoWPAN endpoint that reassembles the packets at the 6LoWPAN level (the receiver) receives a fragment with the Ack-Request flag set, it MUST send an RFRAG Acknowledgment back to the originator to confirm reception of all the fragments it has received so far.

The Ack-Request ('X') set in an RFRAG marks the end of a window. This flag MUST be set on the last fragment if the sender wishes to protect the datagram, and it MAY be set in any intermediate fragment for the purpose of flow control.

This automatic repeat request (ARQ) process MUST be protected by a Retransmission TimeOut (RTO) timer, and the fragment that carries the 'X' flag MAY be retried upon a time out for a configurable number of times (see Section 7.1). 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 'X' flag set in order to reestablish a path and discover which fragments were received over the old path in the acknowledgment bitmap. 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 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 the sender that the reassembling endpoint aborted the processing of an individual datagram.

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.

In order to protect the datagram, the sender transfers a controlled number of fragments and flags the last fragment of a window 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. Because it might defeat the round-trip delay computation, delaying the acknowledgment should be configurable and not enabled by default.

When all the fragments are received, the receiving endpoint reconstructs the packet, passes it to the upper layer, sends an RFRAG Acknowledgment on the reverse path with a FULL bitmap, and arms a short timer, e.g., in the order of an average round-trip delay in the network. As the timer runs, the receiving endpoint absorbs the fragments that were still in flight for that datagram without creating a new state. The receiving endpoint abort the communication if it keeps going on beyond the duration of the timer.

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 [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 insert a delay between fragments of a same datagram that covers multiple transmissions 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], where nodes that communicate at Layer-2 are scheduled to send and receive respectively, and different hops operate on different channels.

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. The first fragment carries the IP header and it is routed all the way from the fragmenting endpoint to the reassembling endpoint. Upon receiving 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, the next fragments can only follow the path that was set up by the first fragment and cannot follow an alternate route. The datagram_tag is used to carry the label, which is swapped in each hop. All fragments follow the same path and fragments are delivered in the order at which they are sent.

6.1.1. Receiving the first fragment

In Route-Over mode, the source and destination MAC addresses in a frame change at each hop. The label that is formed and placed in the datagram_tag is associated with the source MAC address and only valid (and unique) for that source MAC address. Upon a first fragment (i.e., with a sequence of zero), an intermediate router creates a VRB and the associated LSP state 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], where the receiving endpoint allocates a reassembly buffer.

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 matching 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.

The first fragment may be received a second time, indicating that it did not reach the destination and was retried. In that case, it SHOULD follow the same path as the first occurrence. It is up to sending endpoint to determine whether to abort a transmission and then retry it from scratch, which may build an entirely new path.

6.1.2. Receiving the next fragments

Upon receiving a next fragment (i.e., with a non-zero sequence), an intermediate 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 is 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.

[I-D.ietf-6lo-minimal-fragment] indicates that the receiving endpoint stores "the actual packet data from the fragments received so far, in a form that makes it possible to detect when the whole packet has been received and can be processed or forwarded". How this is computed in implementation specific but relies on receiving all the bytes up to the datagram_size indicated in the first fragment. An implementation may receive overlapping fragments as the result of retries after an MTU change.

6.2. Receiving RFRAG Acknowledgments

Upon receipt of 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). Note that Path MTU Discovery is out of scope for this document.

6.3. Aborting the Transmission of 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 aborted, 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 with the datagram_tag. If an acknowledgment is requested, the receiver responds with a NULL bitmap.

The other way around, the receiver might need to abort the process of a fragmented packet for internal reasons, for instance if it is out of reassembly buffers, already uses all 256 possible values of the datagram_tag, or if it keeps receiving fragments beyond a reasonable time while it considers that this packet is already fully reassembled and was passed to the upper layer. In that case, the receiver SHOULD indicate so to the sender with a NULL bitmap in an RFRAG Acknowledgment. The RFRAG Acknowledgment is forwarded all the way back to the source of the packet and cleans up all resources on the way. Upon an acknowledgment with a NULL bitmap, the sender endpoint MUST abort the transmission of the fragmented datagram with one exception: In the particular case of the first fragment, it MAY decide to retry via an alternate next hop instead.

6.4. Applying Recoverable Fragmentation along a Diverse Path

The text above can be read with the assumption of a serial path between a source and a destination. Section 4.5.3 of the "6TiSCH Architecture" [I-D.ietf-6tisch-architecture] defines the concept of a Track that can be a complex path between a source and a destination with Packet ARQ, Replication, Elimination and Overhearing (PAREO) along the Track. This specification can be used along any subset of the complex Track where the first fragment is flooded. The last RFRAG Acknowledgment is flooded on that same subset in the reverse direction. Intermediate RFRAG Acknowledgments can be flooded on any sub-subset of that reverse subset that reach back to the source.

7. Management Considerations

This specification extends "On Forwarding 6LoWPAN Fragments over a Multihop IPv6 Network" [I-D.ietf-6lo-minimal-fragment] and requires the same parameters in the receiver and on intermediate nodes. There is no new parameter as echoing ECN is always on. This parameters typically include the reassembly time-out at the receiver and an inactivity clean-up timer on the intermediate nodes, and the number of messages that can be processed in parallel in all nodes.

The configuration settings introduced by this specification only apply to the sender, which is in full control of the transmission. LLNs vary a lot in size (there can be thousands of nodes in a mesh), in speed (from 10Kbps to several Mbps at the PHY layer), in traffic density, and in optimizations that are desired (e.g., the selection of a RPL [RFC6550] Objective Function [RFC6552] impacts the shape of the routing graph).

For that reason, only a very generic guidance can be given on the settings of the sender and on whether complex algorithms are needed to perform flow control or estimate the round-trip time. To cover the most complex use cases, this specification enables the sender to vary the fragment size, the window size and the inter-frame gap, based on the amount of losses, the observed variations of the round-trip time and the setting of the ECN bit.

7.1. Protocol Parameters

The management system SHOULD be capable of providing the parameters listed in this section.

An implementation must control the rate at which it sends packets over a same path to allow the next hop to forward a packet before it gets the next. In a wireless network that uses a same frequency along a path, more time must be inserted to avoid hidden terminal issues between fragments. This is controlled by the following parameter:

inter-frame gap: 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 inter-frame gap 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.

An implementation should consider the generic recommendations from the IETF in the matter of flow control and rate management in [RFC5033]. To control the flow, an implementation may use a dynamic value of the window size (Window_Size), adapt the fragment size (Fragment_Size) and insert an inter-frame gap that is longer than necessary. In a large network where node contend for the bandwidth, a larger Fragment_Size consumes less bandwidth but also reduces the fluidity and incurs higher chances of loss in transmission. This is controlled by the following parameters:

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

OptFragmentSize: The OptFragmentSize is the value for the Fragment_Size that the sender should use to start with. It is more than or equal to MinFragmentSize. It is less than or equal to MaxFragmentSize. On the first fragment, it must enable the expansion of the IPv6 addresses and of the Hop Limit field within MTU. On all fragments, it is a balance between the expected fluidity and the overhead of MAC and 6LoWPAN headers. For a small MTU, the idea is to keep it close to the maximum, whereas for larger MTUs, it might makes sense to keep it short enough, so that the duty cycle of the transmitter is bounded, e.g., to transmit at least 10 frames per second.

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.

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. It is more than or equal to MinWindowSize. It is less than or equal to MaxWindowSize. The Window_Size should be maintained below the number of hops in the path of the fragment to avoid stacking fragments at the bottleneck on the path. If an inter-frame gap is used to avoid interference between fragments then the Window_Size should be at most in the order of the estimation of the trip time divided by the inter-frame gap.

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

An implementation may perform its estimate of the RTO or use a configured one. The ARQ process is controlled by the following parameters:

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 RTO, that is amount of time that a sender should wait for an RFRAG Acknowledgment before it takes a next action. It is more than or equal to MinARQTimeOut. It is less than or equal to MaxARQTimeOut.

MaxARQTimeOut: The maximum amount of time a node should wait for an RFRAG Acknowledgment before it takes a next action. It must cover the longest expected round-trip time, and be several times less than the time-out that covers the recomposition buffer at the receiver, which is typically in the order of the minute. See Appendix C for recommendations on computing the round-trip time.

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

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

An implementation may be capable to perform flow control based on ECN, more in Appendix C. This is controlled by the following parameter:

UseECN: Indicates whether the sender should react to ECN. The sender may react to ECN by varying the Window_Size between MinWindowSize and MaxWindowSize, varying the Fragment_Size between MinFragmentSize and MaxFragmentSize and/or by increasing the inter-frame gap.

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 both the sender and the receiver, and may tune the optimum size of Fragment_Size and of the Window_Size, OptDatagramSize 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 inter-frame gap is another tool that can be used to increase the spacing between fragments of the same datagram and reduce the ratio of time when a particular intermediate node holds a fragment of that datagram.

8. Security Considerations

This document specifies an instantiation of a 6LoWPAN Fragment Forwarding technique. [I-D.ietf-6lo-minimal-fragment] provides the generic description of Fragment Forwarding and this specification inherits from it. The generic considerations in the Security sections of [I-D.ietf-6lo-minimal-fragment] apply equally to this document.

This specification does not recommend a particular algorithm for the estimation of the duration of the RTO that covers the detection of the loss of a fragment with the 'X' flag set; regardless, an attacker on the path may slow down or discard packets, which in turn can affect the throughput of fragmented packets.

Compared to "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944], this specification reduces the datagram_tag to 8 bits and the tag wraps faster than with [RFC4944]. But for a constrained network where a node is expected to be able to hold only one or a few large packets in memory, 256 is still a large number. Also, the acknowledgement mechanism allows cleaning up the state rapidly once the packet is fully transmitted or aborted.

The abstract Virtual Recovery Buffer inherited from [I-D.ietf-6lo-minimal-fragment] may be used to perform a Denial-of-Service (DoS) attack against the intermediate Routers since the routers need to maintain a state per flow. The particular VRB implementation technique described in [I-D.ietf-lwig-6lowpan-virtual-reassembly] allows realigning which data goes in which fragment, which causes the intermediate node to store a portion of the data, which adds an attack vector that is not present with this specification. With this specification, the data that is transported in each fragment is conserved and the state to keep does not include any data that would not fit in the previous fragment.

9. IANA Considerations

This document allocates 2 patterns for a total of 4 dispatch values in Page 0 for recoverable fragments from the "Dispatch Type Field" registry that was created by "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944] and reformatted by "6LoWPAN Paging Dispatch" [RFC8025].

The suggested patterns (to be confirmed by IANA) are indicated in Table 1.

Bit Pattern	Page	Header Type	Reference
11 10100x	0	RFRAG - Recoverable Fragment	THIS RFC
11 10100x	1-14	Unassigned	
11 10100x	15	Reserved for Experimental Use	RFC 8025
11 10101x	0	RFRAG-ACK - RFRAG Acknowledgment	THIS RFC
11 10101x	1-14	Unassigned	
11 10101x	15	Reserved for Experimental Use	RFC 8025

Table 1: Additional Dispatch Value Bit Patterns

10. Acknowledgments

The author wishes to thank Michel Veillette, Dario Tedeschi, Laurent Toutain, Carles Gomez Montenegro, Thomas Watteyne, and Michael Richardson for in-depth reviews and comments. Also many thanks to Peter Yee, Colin Perkins, Tirumaleswar Reddy Konda and Erik Nordmark for their careful reviews and for helping through the IETF Last Call and IESG review process, and to Jonathan Hui, Jay Werb, Christos Polyzois, Soumitri Kolavennu, Pat Kinney, Margaret Wasserman, Richard Kelsey, Carsten Bormann and Harry Courtice for their various contributions in the long process that lead ot this document.

11. Normative References

[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>.

[I-D.ietf-6lo-minimal-fragment]
Watteyne, T., Thubert, P., and C. Bormann, "On Forwarding 6LoWPAN Fragments over a Multihop IPv6 Network", Work in Progress, Internet-Draft, draft-ietf-6lo-minimal-fragment-10, 1 February 2020, <https://tools.ietf.org/html/draft-ietf-6lo-minimal-fragment-10>.

12. Informative References

[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>.

[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>.

[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>.

[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>.

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

[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>.

[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>.

[RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <https://www.rfc-editor.org/info/rfc6550>.

[RFC6552] Thubert, P., Ed., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, DOI 10.17487/RFC6552, March 2012, <https://www.rfc-editor.org/info/rfc6552>.

[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>.

[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>.

[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>.

[RFC5033] Floyd, S. and M. Allman, "Specifying New Congestion Control Algorithms", BCP 133, RFC 5033, DOI 10.17487/RFC5033, August 2007, <https://www.rfc-editor.org/info/rfc5033>.

[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>.

[I-D.ietf-lwig-6lowpan-virtual-reassembly]
Bormann, C. and T. Watteyne, "Virtual reassembly buffers in 6LoWPAN", Work in Progress, Internet-Draft, draft-ietf-lwig-6lowpan-virtual-reassembly-01, 11 March 2019, <https://tools.ietf.org/html/draft-ietf-lwig-6lowpan-virtual-reassembly-01>.

[I-D.ietf-intarea-frag-fragile]
Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", Work in Progress, Internet-Draft, draft-ietf-intarea-frag-fragile-17, 30 September 2019, <https://tools.ietf.org/html/draft-ietf-intarea-frag-fragile-17>.

[I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", Work in Progress, Internet-Draft, draft-ietf-6tisch-architecture-28, 29 October 2019, <https://tools.ietf.org/html/draft-ietf-6tisch-architecture-28>.

[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/>.

[Kent] Kent, C. and J. Mogul, ""Fragmentation Considered Harmful", In Proc. SIGCOMM '87 Workshop on Frontiers in Computer Communications Technology", DOI 10.1145/55483.55524, August 1987, <http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-87-3.pdf>.

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 would need to be 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 an 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 a 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 controlling 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 4 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 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 "Computing TCP's Retransmission Timer" [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
06254 MOUGINS - Sophia Antipolis
France

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