6lo
Internet-Draft
Updates: 4944 (if approved)
Intended status: Standards Track
Expires: January 20, 2018

P. Thubert, Ed.
Cisco Systems
J. Hui
Nest Labs

July 19, 2017

LLN Fragment Forwarding and Recovery
draft-thubert-6lo-forwarding-fragments-06

Abstract

Considering that an LLN frame can have a MAC payload below 100 bytes, an IPv6 packet might be fragmented into more than 10 fragments at the 6LoWPAN layer. In a 6LoWPAN mesh-under network, the fragments can be forwarded individually across the mesh, whereas a route-over mesh network, a fragmented 6LoWPAN packet must be reassembled at every hop, which causes latency and congestion. This draft introduces a simple protocol to forward individual fragments across a route-over mesh network, and, regardless of the type of mesh, recover the loss of individual fragments across the mesh and protect the network against bloat with a minimal flow control.

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 http://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 January 20, 2018.

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 (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. Those undesirable effects can be alleviated by a hop-by-hop fragment forwarding technique such as the one proposed in this specification, and arguably this could be achieved 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 bulk of the issues raised against it, and may create new issues like uncontrolled state in the network.

Another issue against RFC 4944 [RFC4944] is that it does not define a mechanism to first discover the loss of a fragment along a multi-hop path (e.g. having exhausted the link-layer retries at some hop on the way), 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. End-to-end transport or application-level mechanisms may require a full retransmission of the datagram, wasting resources in an already constrained network.

In that situation, the source 6LoWPAN endpoint will not be aware that a loss occurred and will continue sending all fragments for a datagram that is already doomed. The original support is 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. 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. New dispatch types are defined in Section 4.

3. Terminology and 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.

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.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

4. 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 101000	RFRAG	- Recoverable Fragment
11 101001	RFRAG-ARQ	- RFRAG with Ack Request
11 101010	RFRAG-ACK	- RFRAG Acknowledgment
11 101011	RFRAG-ECHO	- RFRAG Ack with ECN Echo

Figure 1: Additional Dispatch Value Bit Patterns

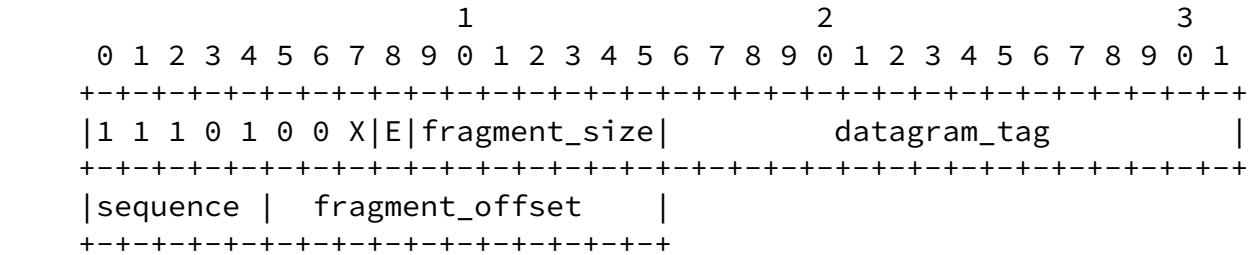
4.1. Recoverable Fragment Dispatch type and Header

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 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.



X set == Ack Requested

Figure 2: RFRAG Dispatch type and Header

X: 1 bit; Ack Requested: 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: 7 bit unsigned integer; the size of this fragment in a unit that depends on the MAC layer technology. For IEEE Std. 802.15.4, the unit is octet, and the maximum fragment size, which 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.

Sequence: 5 bit unsigned integer; the sequence number of the fragment. Fragments are sequence numbered [0..N] where N is in [0..31]. As long as the overheads enable a fragment size of 64 bits or more, this enables to fragment a packet of 2047 octets.

Fragment_offset: 11 bit unsigned integer; when set to 0, this field indicates an abort condition; else, its value depends on the value of the Sequence. When the sequence is not 0, this field indicates the offset of the fragment in the compressed form. When the sequence is 0, denoting the first fragment of a datagram, this field is overloaded to indicate the total_size of the compressed packet, to help the receiver allocate an adapted buffer for the reception and reassembly operations. This format limits the maximum MTU on a 6LoWPAN link to 2047 bytes, but 1280 bytes is the

recommended value to avoid issues with IPV6 Path MTU Discovery [RFC8201].

4.2. 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:

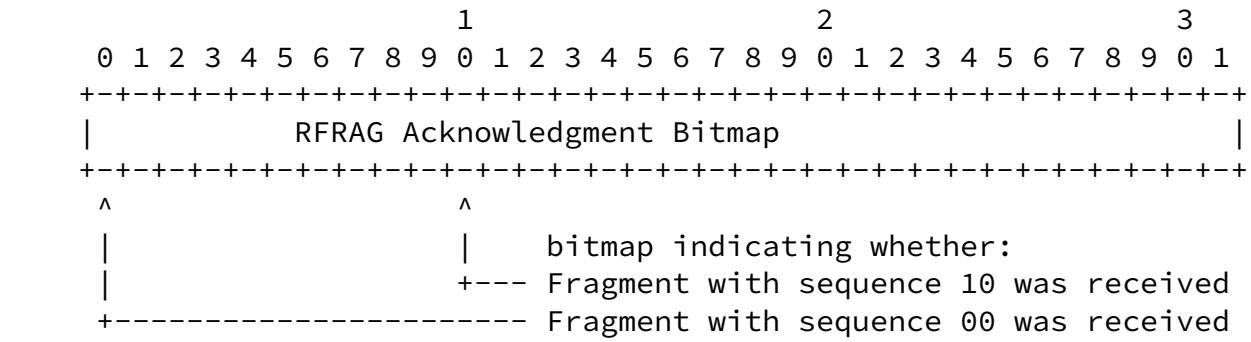


Figure 3: RFRAG Acknowledgment bitmap encoding

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 either lost or are still in the network over a slower path.

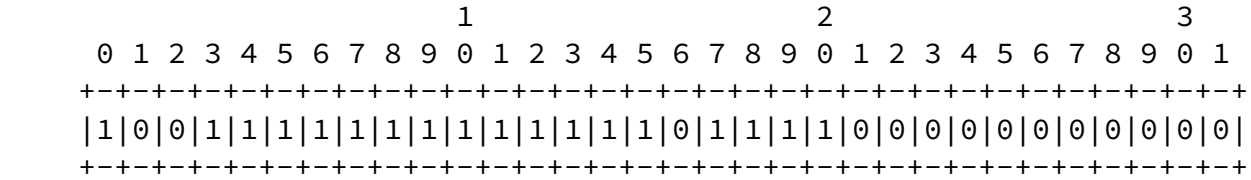


Figure 4: Expanding 3 octets encoding

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

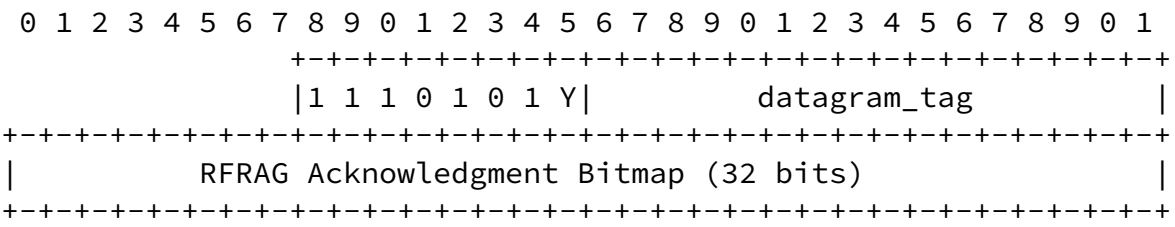


Figure 5: RFRAG Acknowledgment Dispatch type and Header

Y: 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.

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.

5. 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 can optionally carry an ECN indication; it 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 when the reassembling end point sends the RFRAG Acknowledgments by setting the Ack Requested flag in the RFRAG packets. It may set the Ack Requested 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. 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 Requested bit. When it receives a fragment with the ACK Request flag set, the 6LoWPAN endpoint that reassembles the packets at 6LoWPAN level (the receiver) sends back an RFRAG Acknowledgment to confirm reception of all the fragments it has received so far.

The sender transfers a controlled number of fragments and MAY flag the last fragment of a series with an RFRAG Acknowledgment Request. The received 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 cancelled 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.

The sender protects the transmission with a retry timer that is computed according to the method detailed in [RFC6298]. It is expected that the upper layer retries obey the same or friendly rules 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 actually retried.

When the sender decides that a packet should be dropped and the fragmentation process canceled, it sends a pseudo fragment with the fragment_offset, sequence and fragment_size all set to 0, and no data. Upon reception of this message, the receiver should clean up all resources for the packet associated to the datagram_tag. If an acknowledgment is requested, the receiver responds with a NULL bitmap.

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. Upon an acknowledgment with a NULL bitmap, the sender MUST abort the current fragmented transmission of the datagram.

6. 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 enables intermediate routers to forward fragments with no intermediate reconstruction of the entire packet. 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 sollowing 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. 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. Say the first fragment has:

Source IPv6 address = IP_A (maybe hops away)

Destination IPv6 address = IP_B (maybe hops away)

Source MAC = MAC_previous

Datagram_tag= DT_previous

The intermediate router that forwards individual fragments performs the following action:

ia route lookup to get the Next hop IPv6 towards IP_B, which resolves as IP_next.

a MAC address resolution to get the MAC address associated to IP_next, which resolves as MAC_next

Since it is a first fragment of a packet from that source MAC address MAC_previous for that tag DT_previous, the router:

cleans up any leftover resource associated to the tuple (MAC_previous, DT_previous)

allocates a new label for that flow, DT_next, from a Least Recently Used pool or some similar procedure.

allocates a label-swap entry indexed by (MAC_previous, DT_previous) that contains (MAC_next, DT_next)

allocates a label-swap structure indexed by (MAC_next, DT_next) that contains (MAC_previous, DT_previous); this enables the reverse MPLS switching operation that is used to route the RFRAG-ACK.

change the source MAC address to from MAC_prev to MAC_self

change the destination MAC address to from MAC_self to MAC_next

Swaps the datagram_tag to DT_next

At this point the router is all set and can forward the fragment to next.

6.2. Upon the next fragments

Upon next fragments (that are not first fragment), the router expects to have already installed a label-swap structure indexed by (MAC_previous, DT_previous). The router:

looks up the label-swap entry for (MAC_previous, DT_previous), which resolves as (MAC_next, DT_next)

swaps the MAC info to from self to MAC_next;

Swaps the datagram_tag to DT_next

if the label-swap entry for (MAC_previous, DT_previous) is not found, the router builds an RFRAG-ACK to indicate the error. The resulting message has the following information:

MAC info set to from self to MAC_previous as found in the fragment

Swaps the datagram_tag set to DT_previous

Numm bitmap to indicate the error

At this point the router is all set and can send the RFRAG-ACK back ot the previous router.

6.3. Upon the RFRAG Acknowledgments

Upon an RFRAG Acknowledgment, the router expects to already have label-swap structure indexed by (MAC_next, DT_next), which are respectively the source MAC address of the received frame and the received datagram_tag. DT_next should have been computed by this router and this router should have assigned it to this particular datagram. The router:

looks up the label-swap entry for (MAC_next, DT_next), which resolves as (MAC_previous, DT_previous)

swaps the MAC info to from self to MAC_previous;

Swaps the datagram_tag to DT_previous

At this point the router is all set and can forward the RFRAG-ACK to previous.

If the label-swap entry for (MAC_next, DT_next) is not found, it MUST silently drop the packet.

If the RFRAG-ACK indicates either an error (NULL bitmap) or that the fragment was entirely received (FULL bitmap), the router schedules the label-swap entries for recycling. If the RFRAG-ACK is lost on

the way back, the source may retry the last fragment, which will result as an error RFRAG-ACK from the first router on the way that has already cleaned up.

7. 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].

8. IANA Considerations

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

9. Acknowledgments

The author wishes to thank Thomas Watteyne for in-depth reviews and comments, as well as Jay Werb, Christos Polyzois, Soumitri Kolavennu, Pat Kinney, Margaret Wasserman, Richard Kelsey, Carsten Bormann and Harry Courtice for their various contributions.

10. References

10.1. Normative References

[IEEE.802.15.4]
IEEE, "IEEE Standard for Low-Rate Wireless Networks", IEEE Standard 802.15.4, DOI 10.1109/IEEESTD.2016.7460875, <http://ieeexplore.ieee.org/document/7460875/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc6282>.

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

[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, <http://www.rfc-editor.org/info/rfc8025>.

10.2. 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-11 (work in progress), January 2017.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <http://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, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc5681>.

[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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://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 4.2.

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 5 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.

Authors' Addresses

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

Jonathan W. Hui
Nest Labs
3400 Hillview Ave
Palo Alto, California 94304
USA

Email: jonhui@nestlabs.com