6LoWPAN	P. Thubert, Ed.	
Internet-Draft	Cisco	
Intended status: Standards Track	J. Hui	
Expires: December 6, 2010	Arch Rock Corporation	
	June 4, 2010	

TOC

LoWPAN fragment Forwarding and Recovery
draft-thubert-6lowpan-simple-fragment-recovery-07

Abstract

Considering that the IPv6 minimum MTU is 1280 bytes and that an an 802.15.4 frame can have a payload limited to 74 bytes in the worst case, a packet might end up fragmented into as many as 18 fragments at the 6LoWPAN shim layer. If a single one of those fragments is lost in transmission, all fragments must be resent, further contributing to the congestion that might have caused the initial packet loss. This draft introduces a simple protocol to forward and recover individual fragments that might be lost over multiple hops between 6LoWPAN endpoints.

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 December 6, 2010.

Copyright Notice

Table of Contents

---

1. Introduction

TOC

In many 6LoWPAN applications, the majority of traffic is spent sending small chunks of data (order few bytes to few tens of bytes) per packet. Given that an 802.15.4 frame can carry 74 bytes or more in all cases, fragmentation is often not needed for most application traffic. However, many applications also require occasional bulk data transfer capabilities to support firmware upgrades of 6LoWPAN devices or extraction of logs from 6LoWPAN devices. In the former case, bulk data is transferred to the 6LoWPAN device, and in the latter, bulk data flows away from the 6LoWPAN device. In both cases, the bulk data size is often on the order of 10K bytes or more and end-to-end reliable transport is required.
Mechanisms such as TCP or application-layer segmentation will be used to support end-to-end reliable transport. One option to support bulk data transfer over 6LoWPAN links is to set the Maximum Segment Size to fit within the 802.15.4 MTU. Doing so, however, can add significant header overhead to each 802.15.4 frame. This causes the end-to-end transport to be aware of the delivery properties of 6LoWPAN networks, which is a layer violation.

An alternative mechanism combines the use of 6LoWPAN fragmentation in addition to transport or application-layer segmentation. Increasing the Maximum Segment Size reduces header overhead by the end-to-end transport protocol. It also encourages the transport protocol to reduce the number of outstanding datagrams, ideally to a single datagram, thus reducing the need to support out-of-order delivery common to 6LoWPAN networks. [RFC4944] (Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," September 2007.) defines a datagram fragmentation mechanism for 6LoWPAN networks. However, because [RFC4944] (Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," September 2007.) does not define a mechanism for recovering fragments that are lost, datagram forwarding fails if even one fragment is not delivered properly to the next IP hop. End-to-end transport mechanisms will require retransmission of all fragments, wasting resources in an already resource-constrained network.

Past experience with fragmentation has shown that missassociated or lost fragments can lead to poor network behavior and, eventually, trouble at application layer. The reader is encouraged to read [RFC4963] (Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates," July 2007.) and follow the references for more information. That experience led to the definition of the Path MTU discovery (Mogul, J. and S. Deering, "Path MTU discovery," November 1990.) [RFC1191] protocol that limits fragmentation over the Internet.

For one-hop communications, a number of media propose a local acknowledgement mechanism that is enough to protect the 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. Specifically in the case of UDP, valuable additional information can be found in UDP Usage Guidelines for Application Designers (Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers," November 2008.) [RFC5405].

---

2. Terminology

TOC

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] (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.).

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" (Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," August 2007.) [RFC4919] and "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" (Montenegro, G.,

Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," September 2007.) [RFC4944].

ERP Error Recovery Procedure.

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

---

3. Rationale

TOC

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 LoWPAN; however, the associated functionality can be critical enough to justify extra care for ensuring effective transport of large packets across the LoWPAN.
The list of those usages includes:

Towards the LoWPAN node:

Packages of Commands: A number of commands or a full configuration can by 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.

Firmware update: For example, a new version of the LoWPAN 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 6LoWPAN nodes over a relatively short period of time.

From the LoWPAN 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: 6LoWPAN nodes may generate large logs of sampled data for later extraction. 6LoWPAN 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, 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.
To demonstrate the severity of the problem, consider a fairly reliable 802.15.4 frame delivery rate of 99.9% over a single 802.15.4 hop. The expected delivery rate of a 5-fragment datagram would be about 99.5% over a single 802.15.4 hop. However, the expected delivery rate would drop to 95.1% over 10 hops, a reasonable network diameter for 6LoWPAN applications. The expected delivery rate for a 1280-byte datagram is 98.4% over a single hop and 85.2% over 10 hops.
Considering that the IPv6 minimum MTU is 1280 bytes and that a 802.15.4 frame can limit the MAC payload to as little as 74 bytes, a packet might be fragmented into at least 18 fragments at the 6LoWPAM 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.

---

4. Requirements

TOC

This paper proposes a method to recover individual fragments between LoWPAN endpoints. The method is designed to fit the following requirements of a LoWPAN (with or without a Mesh-Under routing protocol):

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

Minimum acknowledgement 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 recovery mechanism should be able to acknowledge multiple fragments in a single message and not require an acknowledgement 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.

Support for out-of-order fragment delivery
A Mesh-Under load balancing mechanism such as the ISA100 Data Link Layer can introduce out-of-sequence packets.

The recovery mechanism must account for packets that appear lost but are actually only delayed over a different path.

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 LoWPAN.

---

5. Overview

TOC

Considering that a multi-hop LoWPAN 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.
From the standpoint of a source LoWPAN 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.
Because a meshed LoWPAN might deliver frames out of order, it is virtually impossible to differentiate these situations. In other words, 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 LoWPAN endpoints. The method detailed in [RFC2988] (Paxson, V. and M. Allman, “Computing TCP's Retransmission Timer,” November 2000.) is recommended for that computation.

---

6. New Dispatch types and headers

TOC

This specification extends "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" (Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” September 2007.) [RFC4944] with 4 new dispatch types, for Recoverable

Fragments (RFRAG) headers with or without Acknowledgment Request, and for the Acknowledgment back.

```
  Pattern    Header Type
+------------+-----------------------------------------------+
| 11  101000 | RFRAG      - Recoverable Fragment             |
| 11  101001 | RFRAG-AR   - RFRAG with Ack Request           |
| 11  10101x | RFRAG-ACK  - RFRAG Acknowledgment             |
+------------+-----------------------------------------------+
```

Figure 1: Additional Dispatch Value Bit Patterns

In the following sections, the semantics of "datagram_tag," "datagram_offset" and "datagram_size" and the reassembly process are changed from [RFC4944] (Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” September 2007.) Section 5.3. "Fragmentation Type and Header." The size and offset are expressed on the compressed packet as opposed to the uncompressed form.

6.1. Recoverable Fragment Dispatch type and Header

TOC

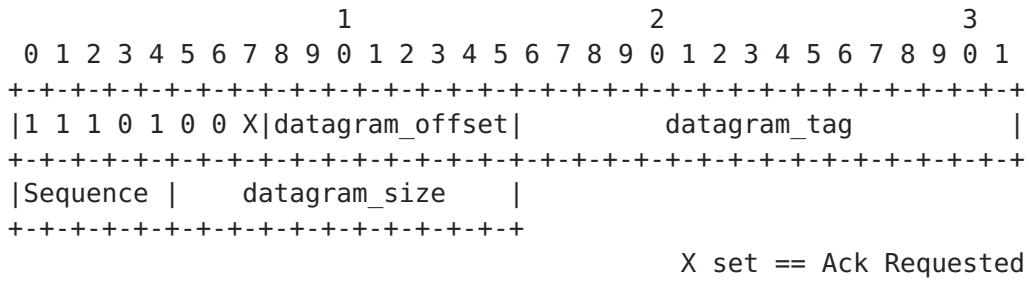

Figure 2: Recoverable Fragment Dispatch type and Header

X bit When set, the sender requires an Acknowledgment from the receiver

Sequence The sequence number of the fragment. Fragments are numbered [0..N] where N is in [0..31].

6.2. Fragment Acknowledgement Dispatch type and Header

TOC

The specification also defines an acknowledgement bitmap that is used to carry selective acknowlegements for the received fragments. A given offset in the bitmap maps one to one with a given sequence number.
The bitmap is compressed as a variable length field formed by control bits and acknowledgement bits. The leftmost bits of the compressed form are control bits up to the first 0. The rest is ack bits encoded right to left:

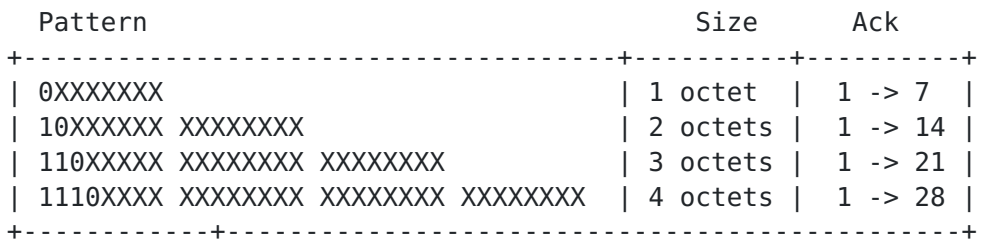

Figure 3: Compressed acknowledgement bitmap encoding

The highest sequence number to be acknowledged determines the pattern to be used. The format can be extended for more fragments in the future but this specification only requires the support of up to 4 octets encoding, which enables to acknowledge up to 28 fragments.
A 32 bits uncompressed bitmap is obtained by prepending zeroes to the XXX in the pattern above. For instance:

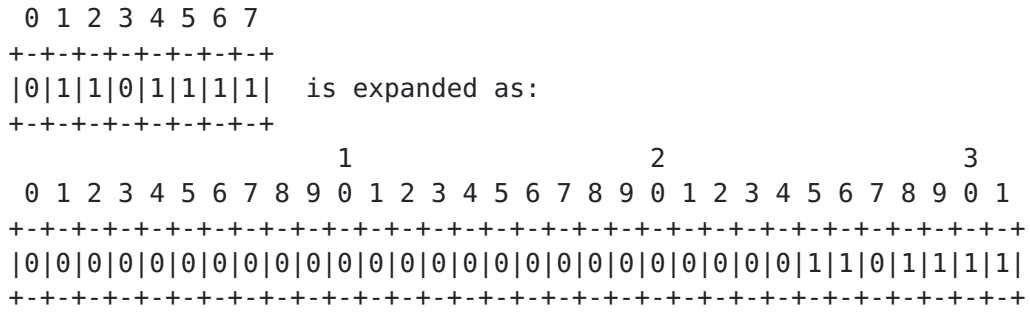

Figure 4: Expanding 1 octet encoding

and

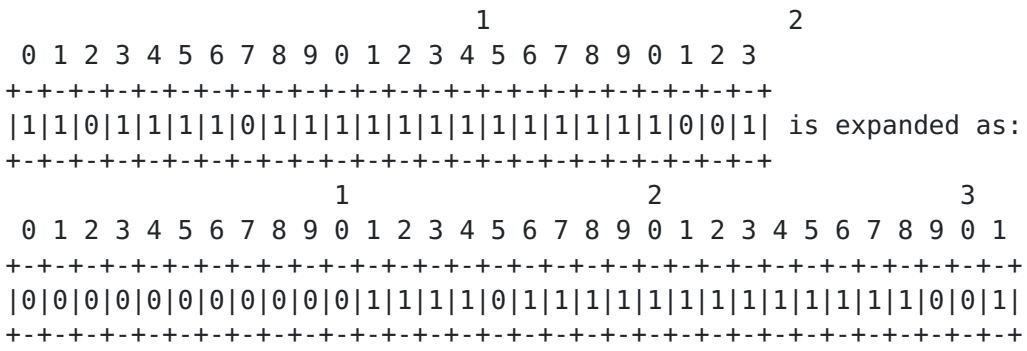

Figure 5: Expanding 3 octets encoding

whereas the 4 octets encoding is expanded by simply setting the first 3 bits to 0. The 32 bits uncompressed bitmap is written and read as follows:

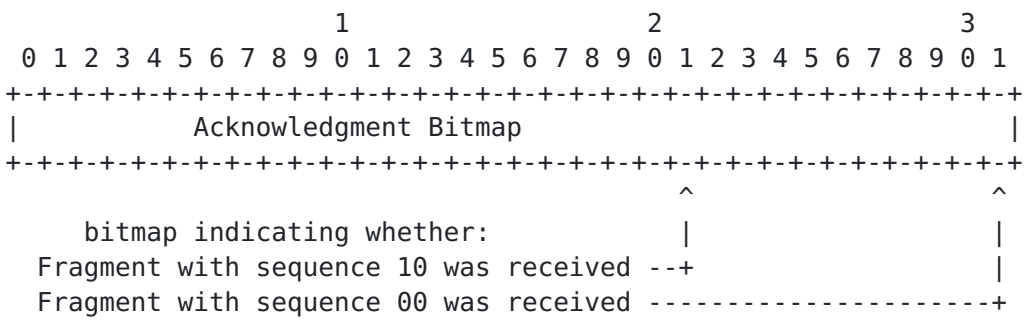

Figure 6: Expanded bitmap encoding

So in the example in Figure 5 (Expanding 3 octets encoding) it appears that all fragments from sequence 0 to 20 were received but for sequence 1, 2 and 16 that were either lost or are still in the network over a slower path.
The compressed form of the acknowledgement bitmap is carried in a Fragment Acknowledgement as follows:

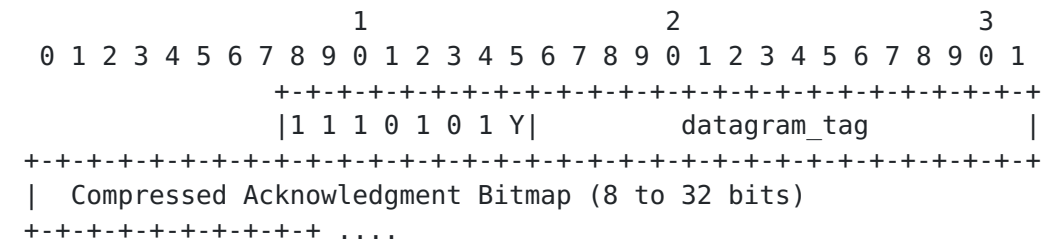

Figure 7: Fragment Acknowledgement Dispatch type and Header

Y bit Reserved for Explicit Congestion Notification (ECN) signalling

Compressed Acknowledgement Bitmap An encoded form of an acknowledgement bitmap.

---

7. Fragments Recovery

TOC

The Recoverable Fragments header RFRAG and RFRAG-AR deprecate the original fragment headers from [RFC4944] (Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” September 2007.) and replace them in the fragmented packets. The Fragment Acknowledgement RFRAG-ACK is introduced as a standalone header in message that is sent back to the fragment source endpoint as known by its MAC address. This assumes that the source MAC address in the fragment (is any) and datagram_tag are enough information to send the Fragment Acknowledgement back to the source fragmentation endpoint.

The node that fragments the packets at 6LoWPAN level (the sender) controls the Fragment Acknowledgements. If may do that at any fragment to implement its own policy or perform congestion control which is out of scope for this document. When the sender of the fragment knows that an underlying mechanism protects the Fragments already it MAY refrain from using the Acknowledgement mechanism, and never set the Ack Requested bit. The node that recomposes the packets at 6LoWPAN level (the receiver) MUST acknowledge the fragments it has received when asked to, and MAY slightly defer that acknowledgement.

The sender transfers a controlled number of fragments and MAY flag the last fragment of a series with an 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 acknowledgement might defeat the round trip delay computation so it should be configurable and not enabled by default.

The receiver interacts with the sender using an Acknowledgment message with a bitmap that indicates which fragments were actually received. The bitmap is a 32bit SWORD, which accommodates up to 32 fragments and is sufficient for the 6LoWPAN MTU. For all n in [0..31], bit n is set to 1 in the bitmap to indicate that fragment with sequence n was received, otherwise the bit is set to 0. All zeroes is a NULL bitmap that indicates that the fragmentation process was cancelled by the receiver for that datagram.

The receiver MAY issue unsolicited acknowledgments. An unsolicited acknowledgment enables the sender endpoint to resume sending if it had reached its maximum number of outstanding fragments or indicate that the receiver 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 arms a retry timer to cover the fragment that carries the Acknowledgment request. Upon time out, the sender assumes that all the fragments on the way are received or lost. The process must have completed within an acceptable time that is within the boundaries of upper layer retries. The method detailed in [RFC2988] (Paxson, V. and M. Allman, “Computing TCP's Retransmission Timer,” November 2000.) is recommended for the computation of the retry timer. 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 datagram_offset, sequence and datagram_size all set to zero, 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 acknowledgement 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 recomposition buffers, or considers that this packet is already fully recomposed and passed to the upper layer. In that case, the receiver SHOULD indicate so to the sender with a NULL bitmap. Upon an acknowledgement with a NULL bitmap, the sender MUST drop the datagram.

---

8. Forwarding Fragments

TOC

This specification enables intermediate routers to forward fragments with no intermediate reconstruction of the entire packet. Upon the first fragment, the routers lay an label along the path that is followed by that fragment (that is IP routed), and all further 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.

8.1. Upon the first fragment

TOC

In route over the L2 source changes at each hop. The label that is formed adnd 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_prv (prv as previous)
>
> Datagram_tag= DT_prv

The intermediate router that forwards individual fragments does the following:

> a route lookup to get Next hop IPv6 towards IP_B, which resolves as IP_nxt (nxt as next)
>
> a ND resolution to get the MAC address associated to IP_nxt, which resolves as MAC_nxt

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

> cleans up any leftover resource associated to the tupple (MAC_prv, DT_prv)
>
> allocates a new label for that flow, DT_nxt, from a Least Recently Used pool or some siumilar procedure.
>
> allocates a Label swap structure indexed by (MAC_prv, DT_prv) that contains (MAC_nxt, DT_nxt)
>
> allocates a Label swap structure indexed by (MAC_nxt, DT_nxt) that contains (MAC_prv, DT_prv)
>
> swaps the MAC info to from self to MAC_nxt
>
> Swaps the datagram_tag to DT_nxt

At this point the router is all set and can forward the packet to nxt.

8.2. Upon the next fragments

TOC

Upon next fragments (that are not first fragment), the router expects to have already Label swap structure indexed by (MAC_prv, DT_prv). The router:

> lookups up the Label swap entry for (MAC_prv, DT_prv), which resolves as (MAC_nxt, DT_nxt)
>
> swaps the MAC info to from self to MAC_nxt;
>
> Swaps the datagram_tag to DT_nxt

At this point the router is all set and can forward the packet to nxt.
if the Label swap entry for (MAC_src, DT_src) is not found, the router builds an RFRAG-ACK to indicate the error. The acknowledgment message has the following information:

> MAC info set to from self to MAC_prv as found in the fragment
>
> Swaps the datagram_tag set to DT_prv
>
> Bitmap of all zeroes to indicate the error

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

8.3. Upon the fragment acknowledgements

TOC

Upon fragment acknowledgements next fragments (that are not first fragment), the router expects to have already Label swap structure indexed by (MAC_nxt, DT_nxt). The router:

> lookups up the Label swap entry for (MAC_nxt, DT_nxt), which resolves as (MAC_prv, DT_prv)
>
> swaps the MAC info to from self to MAC_prv;
>
> Swaps the datagram_tag to DT_prv

At this point the router is all set and can forward the RFRAG-ACK to prv.
if the Label swap entry for (MAC_nxt, DT_nxt) is not found, it simply drops the packet.
if the RFRAG-ACK indicates either an error or that the fragment was fully receive, 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 fragments, which will result as an error RFRAG-ACK from the first router on the way that has already cleaned up.

9. Security Considerations

TOC

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" (Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” September 2007.) [RFC4944].

10. IANA Considerations

TOC

Need extensions for formats defined in "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" (Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” September 2007.) [RFC4944].

11. Acknowledgments

TOC

The author wishes to thank Jay Werb, Christos Polyzois, Soumitri Kolavennu and Harry Courtice for their contribution and review.

12. References

TOC

12.1. Normative References

TOC

[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2988]	Paxson, V. and M. Allman, “Computing TCP's Retransmission Timer,” RFC 2988, November 2000 (TXT).
[RFC4944]	Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” RFC 4944, September 2007 (TXT).

12.2. Informative References

TOC

[I-D.ietf-roll-rpl]	Winter, T., Thubert, P., and R. Team, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks," draft-ietf-roll-rpl-08 (work in progress), May 2010 (TXT).
[I-D.mathis-frag-harmful]	Mathis, M., "Fragmentation Considered Very Harmful," draft-mathis-frag-harmful-00 (work in progress), July 2004 (TXT).
[RFC1191]	Mogul, J. and S. Deering, "Path MTU discovery," RFC 1191, November 1990 (TXT).
[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, August 2007 (TXT).
[RFC4963]	Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates," RFC 4963, July 2007 (TXT).
[RFC5405]	Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers," BCP 145, RFC 5405, November 2008 (TXT).

Authors' Addresses

TOC

	Pascal Thubert (editor)
	Cisco Systems
	Village d'Entreprises Green Side
	400, Avenue de Roumanille
	Batiment T3
	Biot - Sophia Antipolis 06410
	FRANCE
Phone:	+33 4 97 23 26 34
Email:	pthubert@cisco.com
	Jonathan W. Hui
	Arch Rock Corporation
	501 2nd St. Ste. 410
	San Francisco, California 94107
	USA
Phone:	+415 692 0828
Email:	jhui@archrock.com