Authors: T. Watteyne, Ed.   P. Thubert, Ed.
         Analog Devices   Cisco Systems
         C. Bormann
         Universitaet Bremen TZI

On Forwarding 6LoWPAN Fragments over a Multihop IPv6 Network

Abstract

This document introduces the capability to forward 6LoWPAN fragments. This method reduces the latency and increases end-to-end reliability in route-over forwarding. It is the companion to using virtual reassembly buffers which is a pure implementation technique.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The original 6LoWPAN fragmentation is defined in [RFC4944] and it is
implicitly defined for use over a single IP hop through possibly
multiple Layer-2 (mesh-under) hops in a meshed 6LoWPAN Network.
Although [RFC6282] updates [RFC4944], it does not redefine 6LoWPAN
fragmentation.

This means that over a Layer-3 (route-over) network, an IP packet is
expected to be reassembled at every hop at the 6LoWPAN sublayer,
pushed to Layer-3 to be routed, and then fragmented again if the
next hop is another similar 6LoWPAN link. This draft introduces an
alternate approach called 6LoWPAN Fragment Forwarding (FF) whereby
an intermediate node forwards a fragment as soon as it is received
if the next hop is a similar 6LoWPAN link. The routing decision is

made on the first fragment, which has all the IPv6 routing
information. The first fragment is forwarded immediately and a state
is stored to enable forwarding the next fragments along the same
path.

Done right, 6LoWPAN Fragment Forwarding techniques lead to more
streamlined operations, less buffer bloat and lower latency. It may
be wasteful if some fragments are missing after the first one since
the first fragment will still continue until the 6LoWPAN endpoint
that will attempt to perform the reassembly, and may be misused to
the point that the end-to-end latency falls behind that of per-hop
recomposition. This specification provides a generic overview of FF,
discusses advantages and caveats, and introduces a particular
6LoWPAN Fragment Forwarding technique called Virtual Reassembly
Buffer that can be used while conserving the message formats defined
in [RFC4944].

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. Referenced Work

Past experience with fragmentation, e.g., as described in "IPv4
Reassembly Errors at High Data Rates" [RFC4963] and references
therein, has shown that mis-associated or lost fragments can lead to
poor network behavior and, occasionally, trouble at application
layer. That experience led to the definition of "Path MTU discovery"
[RFC8201] (PMTUD) protocol that limits fragmentation over the
Internet.

"IP Fragmentation Considered Fragile" [FRAG-ILE] discusses security
threats that are linked to using IP fragmentation. The 6LoWPAN
fragmentation takes place underneath, but some issues described
there may still apply to 6LoWPAN fragments.

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].

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.

2.3. New Terms

This specification uses the following terms:

6LoWPAN endpoints: The 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.

3. Overview of 6LoWPAN Fragmentation

We use Figure 1 to illustrate 6LoWPAN fragmentation. We assume node
A forwards a packet to node B, possibly as part of a multi-hop route
between IPv6 source and destination nodes which are neither A nor B.

```
             +---+                    +---+
    ... ---| A |------------------->| B |--- ...
             +---+                    +---+
                         # (frag. 5)

        123456789                 123456789
        +---------+               +---------+
        |   #  ###|               |###  #   |
        +---------+               +---------+
           outgoing                 incoming
        fragmentation              reassembly
            buffer                   buffer
```
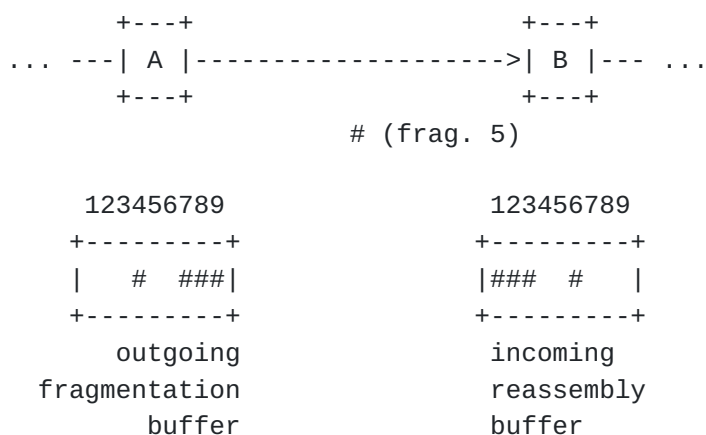
        Figure 1: Fragmentation at node A, reassembly at node B.

Node A starts by compacting the IPv6 packet using the header
compression mechanism defined in [RFC6282]. If the resulting 6LoWPAN
packet does not fit into a single Link-Layer frame, node A's 6LoWPAN
sublayer cuts it into multiple 6LoWPAN fragments, which it transmits
as separate Link-Layer frames to node B. Node B's 6LoWPAN sublayer
reassembles these fragments, inflates the compressed header fields
back to the original IPv6 header, and hands over the full IPv6
packet to its IPv6 layer.

In Figure 1, a packet forwarded by node A to node B is cut into nine
fragments, numbered 1 to 9 as follows:

  *Each fragment is represented by the '#' symbol.

  *Node A has sent fragments 1, 2, 3, 5, 6 to node B.

  *Node B has received fragments 1, 2, 3, 6 from node A.

  *Fragment 5 is still being transmitted at the link layer from node
   A to node B.

The reassembly buffer for 6LoWPAN is indexed in node B by:

  *a unique Identifier of Node A (e.g., Node A's Link-Layer address)

  *the datagram_tag chosen by node A for this fragmented datagram

Because it may be hard for node B to correlate all possible Link-
Layer addresses that node A may use (e.g., short vs. long
addresses), node A must use the same Link-Layer address to send all
the fragments of the same datagram to node B.

Conceptually, the reassembly buffer in node B contains:

  *a datagram_tag as received in the incoming fragments, associated
   to Link-Layer address of node A for which the received
   datagram_tag is unique,

  *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,

  *a state indicating the fragments already received,

  *a datagram_size,

  *a timer that allows discarding a partially reassembled packet
   after some timeout.

A fragmentation header is added to each fragment; it indicates what
portion of the packet that fragment corresponds to. Section 5.3 of
[RFC4944] defines the format of the header for the first and
subsequent fragments. All fragments are tagged with a 16-bit
"datagram_tag", used to identify which packet each fragment belongs
to. Each datagram can be uniquely identified by the sender Link-
Layer addresses of the frame that carries it and the datagram_tag
that the sender allocated for this datagram. [RFC4944] also mandates
that the first fragment is sent first and with a particular format
that is different than that of the next fragments. Each fragment but
the first one can be identified within its datagram by the datagram-
offset.

Node B's typical behavior, per [RFC4944], is as follows. Upon
receiving a fragment from node A with a datagram_tag previously
unseen from node A, node B allocates a buffer large enough to hold
the entire packet. The length of the packet is indicated in each
fragment (the datagram_size field), so node B can allocate the
buffer even if the first fragment it receives is not fragment 1. As
fragments come in, node B fills the buffer. When all fragments have
been received, node B inflates the compressed header fields into an
IPv6 header, and hands the resulting IPv6 packet to the IPv6 layer
which performs the route lookup. This behavior typically results in
per-hop fragmentation and reassembly. That is, the packet is fully
reassembled, then (re)fragmented, at every hop.

4. Limits of Per-Hop Fragmentation and Reassembly

There are at least 2 limits to doing per-hop fragmentation and
reassembly. See [ARTICLE] for detailed simulation results on both
limits.

4.1. Latency

When reassembling, a node needs to wait for all the fragments to be
received before being able to generate the IPv6 packet, and possibly
forward it to the next hop. This repeats at every hop.

This may result in increased end-to-end latency compared to a case
where each fragment is forwarded without per-hop reassembly.

4.2. Memory Management and Reliability

Constrained nodes have limited memory. Assuming a reassembly buffer
for a 6LoWPAN MTU of 1280 bytes as defined in section 4 of
[RFC4944], typical nodes only have enough memory for 1-3 reassembly
buffers.

To illustrate this we use the topology from Figure 2, where nodes A,
B, C and D all send packets through node E. We further assume that
node E's memory can only hold 3 reassembly buffers.

```
          +---+         +---+
 ... --->| A |------>| B |
          +---+         +---+\
                             \
                    +---+     +---+
                    | E |--->| F | ...
                    +---+     +---+
                     /
                    /
          +---+         +---+
 ... --->| C |------>| D |
          +---+         +---+
```
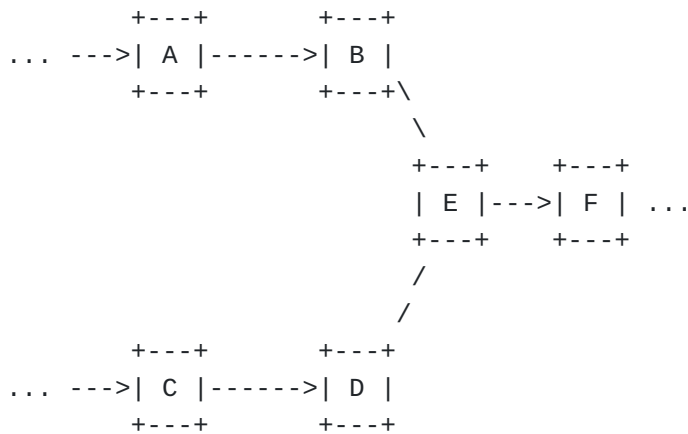
Figure 2: Illustrating the Memory Management Issue.

When nodes A, B and C concurrently send fragmented packets, all 3
reassembly buffers in node E are occupied. If, at that moment, node
D also sends a fragmented packet, node E has no option but to drop
one of the packets, lowering end-to-end reliability.

5. Forwarding Fragments

A 6LoWPAN Fragment Forwarding technique makes the routing decision
on the first fragment, which is always the one with the IPv6 address
of the destination. Upon a first fragment, a forwarding node (e.g.
node B in a A->B->C sequence) that does fragment forwarding MUST
attempt to create a state and forward the fragment. This is an
atomic operation, and if the first fragment cannot be forwarded then
the state MUST be removed.

Since the datagram_tag is uniquely associated to the source Link-Layer address of the fragment, the forwarding node MUST assign a new datagram_tag from its own namespace for the next hop and rewrite the fragment header of each fragment with that datagram_tag.

When a forwarding node receives a fragment other than a first fragment, it MUST look up state based on the source Link-Layer address and the datagram_tag in the received fragment. If no such state is found, the fragment MUST be dropped; otherwise the fragment MUST be forwarded using the information in the state found.

Compared to [Section 3](#), the conceptual reassembly buffer in node B now contains, assuming that node B is neither the source nor the final destination:

  *a datagram_tag as received in the incoming fragments, associated
   to Link-Layer address of node A for which the received
   datagram_tag is unique,

  *the Link-Layer address that node B uses as source to forward the
   fragments

  *the Link-Layer address of the next hop C that is resolved on the
   first fragment

  *a datagram_tag that node B uniquely allocated for this datagram
   and that is used when forwarding the fragments of the datagram

  *a buffer for the remainder of a previous fragment left to be
   sent,

  *a timer that allows discarding the stale FF state after some
   timeout. The duration of the timer should be longer than that
   which covers the reassembly at the receiving end point.

A node that has not received the first fragment cannot forward the next fragments. This means that if node B receives a fragment, node A was in possession of the first fragment at some point. In order to keep the operation simple, it makes sense to be consistent with [RFC4944] and enforce that the first fragment is always sent first. When that is done, if node B receives a fragment that is not the first and for which it has no state, then node B treats this as an error and refrain from creating a state or attempting to forward. This also means that node A should perform all its possible retries on the first fragment before it attempts to send the next fragments, and that it should abort the datagram and release its state if it fails to send the first fragment.

One benefit of Fragment Forwarding is that the memory that is used to store the packet is now distributed along the path, which limits

the buffer bloat effect. Multiple fragments may progress in parallel
along the network as long as they do not interfere. An associated
caveat is that on a half duplex radio, if node A sends the next
fragment at the same time as node B forwards the previous fragment
to a node C down the path then node B will miss the next fragment
from node A. If node C forwards the previous fragment to a node D at
the same time and on the same frequency as node A sends the next
fragment to node B, this may result in a hidden terminal problem at
B whereby the transmission from C interferes with that from A
unbeknownst of node A. It results that consecutive fragments must be
reasonably spaced in order to avoid the 2 forms of collision
described above. A node that has multiple packets or fragments to
send via different next-hop routers may interleave the messages in
order to alleviate those effects.

6. Virtual Reassembly Buffer (VRB) Implementation

Virtual Reassembly Buffer (VRB) is the implementation technique
described in [LWIG-VRB] in which a forwarder does not reassemble
each packet in its entirety before forwarding it.

VRB overcomes the limits listed in Section 4. Nodes do not wait for
the last fragment before forwarding, reducing end-to-end latency.
Similarly, the memory footprint of VRB is just the VRB table,
reducing the packet drop probability significantly.

There are, however, limits:

Non-zero Packet Drop Probability: The abstract data in a VRB table
   entry contains at a minimum the Link-Layer address of the
   predecessor and that of the successor, the datagram_tag used by
   the predecessor and the local datagram_tag that this node will
   swap with it. The VRB may need to store a few octets from the
   last fragment that may not have fit within MTU and that will be
   prepended to the next fragment. This yields a small footprint
   that is 2 orders of magnitude smaller compared to needing a 1280-
   byte reassembly buffer for each packet. Yet, the size of the VRB
   table necessarily remains finite. In the extreme case where a
   node is required to concurrently forward more packets that it has
   entries in its VRB table, packets are dropped.

No Fragment Recovery: There is no mechanism in VRB for the node
   that reassembles a packet to request a single missing fragment.
   Dropping a fragment requires the whole packet to be resent. This
   causes unnecessary traffic, as fragments are forwarded even when

the destination node can never construct the original IPv6
packet.

No Per-Fragment Routing: All subsequent fragments follow the same
sequence of hops from the source to the destination node as the
first fragment, because the IP header is required to route the
fragment and is only present in the first fragment. A side effect
is that the first fragment must always be forwarded first.

The severity and occurrence of these limits depends on the Link-
Layer used. Whether these limits are acceptable depends entirely on
the requirements the application places on the network.

If the limits are present and not acceptable for the application,
future specifications may define new protocols to overcome these
limits. One example is [FRAG-RECOV] which defines a protocol which
allows fragment recovery.

7. Security Considerations

Secure joining and the Link-Layer security that it sets up protects
against those attacks from network outsiders.

"IP Fragmentation Considered Fragile" [FRAG-ILE] discusses security
threats that are linked to using IP fragmentation. The 6LoWPAN
fragmentation takes place underneath, but some issues described
there may still apply to 6LoWPAN fragments.

  *Overlapping fragment attacks are possible with 6LoWPAN fragments
   but there is no known firewall operation that would work on
   6LoWPAN fragments at the time of this writing, so the exposure is
   limited. An implementation of a firewall SHOULD NOT forward
   fragments but recompose the IP packet, check it in the
   uncompressed form, and then forward it again as fragments if
   necessary.

  *Resource exhaustion attacks are certainly possible and a
   sensitive issue in a constrained network. An attacker can perform
   a Denial-of-Service (DoS) attack on a node implementing VRB by
   generating a large number of bogus first fragments without
   sending subsequent fragments. This causes the VRB table to fill
   up. When hop-by-hop reassembly is used, the same attack can be
   more damaging if the node allocates a full datagram_size for each
   bogus first fragment. With the VRB, the attack can be performed
   remotely on all nodes along a path, but each node suffers a
   lesser hit. this is because the VRB does not need to remember the
   full datagram as received so far but only possibly a few octets
   from the last fragment that could not fit in it. An
   implementation MUST protect itself to keep the number of VRBs
   within capacity, and that old VRBs are protected by a timer of a

reasonable duration for the technology and destroyed upon
timeout.

 *Attacks based on predictable fragment identification values are
  also possible but can be avoided. The datagram_tag SHOULD be
  assigned pseudo-randomly in order to defeat such attacks.

 *Evasion of Network Intrusion Detection Systems (NIDS) leverages
  ambiguity in the reassembly of the fragment. This sounds
  difficult and mostly useless in a 6LoWPAN network since the
  fragmentation is not end-to-end.

8. IANA Considerations

No requests to IANA are made by this document.

9. Acknowledgments

The authors would like to thank Carles Gomez Montenegro, Yasuyuki
Tanaka, Ines Robles and Dave Thaler for their in-depth review of
this document and improvement suggestions. Also many thanks to
Georgies Papadopoulos and Dominique Barthel for their own reviews,
and to Sarah Banks, Joerg Ott and Francesca Palombini For their
constructive reviews through the IESG process.

10. 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>.

[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>.

[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>.

11. Informative References

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[FRAG-ILE] 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>.

[LWIG-VRB] 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>.

[FRAG-RECOV] Thubert, P., "6LoWPAN Selective Fragment Recovery",
            Work in Progress, Internet-Draft, draft-ietf-6lo-
            fragment-recovery-08, 28 November 2019, <https://
            tools.ietf.org/html/draft-ietf-6lo-fragment-recovery-08>.

   [ARTICLE] Tanaka, Y., Minet, P., and T. Watteyne, "6LoWPAN Fragment
              Forwarding", IEEE Communications Standards Magazine ,
              2019.

Authors' Addresses

   Thomas Watteyne (editor)
   Analog Devices
   32990 Alvarado-Niles Road, Suite 910
   Union City, CA 94587
   United States of America

   Email: thomas.watteyne@analog.com

   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

   Carsten Bormann
   Universitaet Bremen TZI
   Postfach 330440
   D-28359 Bremen
   Germany

   Email: cabo@tzi.org