Virtual reassembly buffers in 6LoWPAN
draft-bormann-lwig-6lowpan-virtual-reassembly-00

Abstract

When employing adaptation layer fragmentation in 6LoWPAN, it may be
beneficial for a forwarder not to have to reassemble each packet in
its entirety before forwarding it.

This has been always possible with the original fragmentation design
of RFC 4944. Apart from a brief mention of the way to do this in
Section 2.5.2 of the 6LoWPAN book, this has not been extensively
described in the literature. The present document attempts to fill
that gap.

Status of This Memo

This Internet-Draft is submitted in full conformance with the
provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF). Note that other groups may also distribute
working documents as Internet-Drafts. The list of current Internet-
Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months
and may be updated, replaced, or obsoleted by other documents at any
time. It is inappropriate to use Internet-Drafts as reference
material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 29, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents
(https://trustee.iETF.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
1. Introduction

(TO DO: Insert an extended form of the abstract first here, expanding the references to [RFC4944] and [BOOK] in the process.)

2. Reassembly buffers

An adaptation layer implementation for 6LoWPAN needs to perform reassembly of every fragmented packet received in order to be able to forward the packet (re-fragmenting it in the process).

A reassembly buffer for 6LoWPAN contains:

- datagram_size,
- datagram_tag and L2 sender and receiver addresses (to which the datagram_tag is local),
- 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 timer that allows discarding the partial packet after a timeout.

This requires a reassembly buffer for each fragmented packet the reception of which is in progress. Since the forwarder may be receiving fragments for multiple packets concurrently (e.g., from different senders), this means that multiple reassembly buffers are needed, easily dominating the memory requirements in a 6LoWPAN
implementation. Worse, as this space may still be limited, any lack of reassembly buffers may lead to an increased loss rate for fragmented packets (which already have to cope with a higher compound loss rate).

3. Virtual reassembly

To reduce the memory requirement for reassembly buffers, the implementation may opt to not keep the actual packet data in the reassembly buffer. Instead, it may attempt to send out the data for a fragment in the form of a forwarded fragment, as soon as all necessary information for that is available. Obviously, all fragments need to be sent with the same outgoing address (otherwise a full reassembly implementation would discard the fragments) and the same datagram_tag.

To this end, the reassembly buffer now also stores, as soon as enough of the packet is available to make a forwarding decision (i.e., as soon as the first fragment has been received):

- L2 destination address used for forwarding,
- outgoing datagram_tag chosen for this packet.

A simple implementation may do away with any attempt to keep packet data in the virtual reassembly buffer. It then has to discard all non-first fragments for which a reassembly buffer is not already available (penalizing reordering, which however may be rare).

Note that the decision to do local processing of a packet needs to be taken with the first fragment - such packets of course do need to be fully reassembled (unless transport and application also can cope with fragments, which they rarely can in the presence of security).

4. Header compression

[ RFC6282 ] defines the header compression format for 6LoWPAN. One important impact of header compression is that the header is no longer of a fixed length. In particular, changes made by a forwarder may gain or lose the ability to use a more highly compressed variant, changing the length of the header in the packet. If the change increases the size, the maximum frame size may be exceeded, leading to the need to re-fragment in the forwarder. This is less of a problem with full reassembly, but with virtual reassembly can lead to the need for sending an additional frame for each packet.

The well-known approach to minimize the probability of this need is for the original sender to put all slack in the frame sizes into the
_first_ packet, making this the smallest fragment and not the last one as would be done in a naive implementation. (This also has other consequences related to delivery probability, which are not discussed here.) This makes sure an additional fragment only needs to be sent if the header expansion during forwarding would have created an additional fragment with full reassembly as well.

5. IANA Considerations

This document makes no requests of IANA.

6. Security considerations

TBD

7. References

7.1. Normative References


7.2. Informative References


Acknowledgements

Many people have mentioned that it would be good to have a description of virtual reassembly in 6LoWPAN. Finally, Thomas Watteyne assembled a design team that intends to work on 6Lo fragmentation. Writing up the present document has been motivated by that work.

Author’s Address