SCHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 26 September 2024

G. Papadopoulos
A. Bruniaux
IMT Atlantique

25 March 2024

Forward Error Correction (FEC) for SCHC framework
draft-papadopoulos-schc-fec-00

Abstract

This document describes a Forward Error Correction (FEC) method that is applied over the SCHC framework to improve the network performance under certain range of loss/error rates.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://example.com/LATEST. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-todo-yourname-protocol/.

Discussion of this document takes place on the WG Working Group mailing list (mailto:WG@example.com), which is archived at https://example.com/WG.

Source for this draft and an issue tracker can be found at https://github.com/USER/REPO.

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 26 September 2024.

Table of Contents

1. Introduction

In Low-Power Wide Area Network (LPWAN) technologies, the L2 MTU typically ranges from tens to hundreds of bytes.

The RFC 8724 standard defines the Static Context Header Compression and fragmentation (SCHC) framework, which provides header compression and optional fragmentation mechanisms to enable LPWAN technologies, that do not come with internal fragmentation/reassembly functionalities, to comply with the IPv6 MTU requirement of 1280 bytes [RFC8200].

However, this standardized framework struggles in low link-quality scenarios. This document describes a Forward Error Correction (FEC) method that is applied over the SCHC framework to improve the network performance under certain range of loss/error rates.

2. Conventions and Definitions

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.

3. Terminology

4. FEC in SCHC

FEC is a method employed to control errors in packet transmission by embedding additional redundant information within transmitted fragments, thereby reducing the chances for the destination node to request retransmission of missing fragments. Employed in satellite communications and mobile networks, FEC mechanisms use encoding algorithms that allow the destination node to detect errors and often to recover missing components (i.e., to correct the errors).

FEC can be classified into intra-frame, where error correction codes add redundancy inside a packet to correct errors on individual packets, and inter-frame (or inter-fragment), where additional redundant frames are transmitted. LoRa technology employed the intra-frame FEC. Indeed, the intra-frame FEC of LoRa uses Coding Rates (CR) 4/5 to 4/8. In this document, a generic inter-frame FEC mechanism is presented in order to obtain higher Data Delivery Rate (DDR).

SCHC framework can be applied over lossy radio links such as LPWAN where some of the fragments of a SCHC packet can be lost, which may lead to the failure of the reception of the whole SCHC packet (notably in the case of No-ACK mode). Therefore, incorporating FEC into SCHC allows the destination node to increase the chances for the destination node to recover the missing SCHC fragments without the need for the sender to retransmit the missing SCHC fragments.

While FEC mechanisms increase network reliability in lossy networks, they also introduce additional costs. This is because sending additional fragments demands energy and bandwidth. Furthermore, the increase in traffic can ultimately lead to overflow in the transmission queues of relay nodes when such nodes exist. Consequently, the implementation of FEC schemes in networks with constrained resources warrants careful consideration.

4.1. XORFEC Algorithm

XORFEC employs the Exclusive OR (XOR) operator () within its FEC mechanism to produce an extra fragment for a fragmented IPv6 SCHC packet. This supplementary fragment contains the redundant information. This additional fragment is sent after the original fragments of the SCHC packet and allows the destination node to detect a potential loss of an original fragment and to recover it, mitigating thus the scenario where the loss of one fragment leads to the entire packet being lost and/or to reduce the number of fragment retries required to avoid the entire packet being lost.

4.1.1. The XOR Operator

XOR is a logical operator employed in encoding mechanisms, blending information from multiple fragments during encoding and subsequently decoding the encoded fragments upon reception. XOR is a binary operator, and when applied to fragments that consist of series of bits, is applied bitwise. The key property of XOR utilized in XORFEC for fragment recovery is that applying XOR to the result of an initial XOR operation and one of its input values (i.e., of the first XOR) yields the other input value, see an eample below:

B = A (A B)
A = B (A B)

Indeed, if a SCHC packet is fragmented into two fragments A and B, the additional fragment C generated by the source node will be:

C = A B

In this case, if the destination node receives the A and C fragments but does not receive the B fragment, it can recover B fragment by applying the XOR operator to the successfully received fragments.

Note that this function can be generalised to SCHC packets that consists of more than two fragments. Indeed, with k original fragments (F1, F2, F3, ..., Fk), the additional fragment F_additional will be:

F_additional = F1 F2 F3 ... Fk

In a scenario where the destination node receives successfully all fragments except Fi, then it can recover the latter by applying the XOR operator to the successfully received fragments, as it is shown below:

Fi = (F1 ... Fi1 Fi+1 ... Fm) F_additional

The main limitation of the XORFEC algorithm is that the loss tolerance is one missing fragment. Indeed, in the previous example of k fragments, the recovery of Fi is only possible if no more than one fragment is lost.

4.1.2. XORFEC Operation Example in LPWAN

4.1.2.1. XORFEC over No-ACK mode

In No-ACK mode, a SCHC Packet is first fragmented into k original fragments and the additional fragment (i.e., F_additional) is generated by applying the XOR operator to these k fragments.

In Figure 1, the example (i.e., Figure 29) from [RFC8724] of No-ACK mode of a SCHC Packet that needs 5 SCHC Fragments (and where FCN is 1 bit wide) is adapted when XORFEC is applied to all 5 SCHC Fragments.

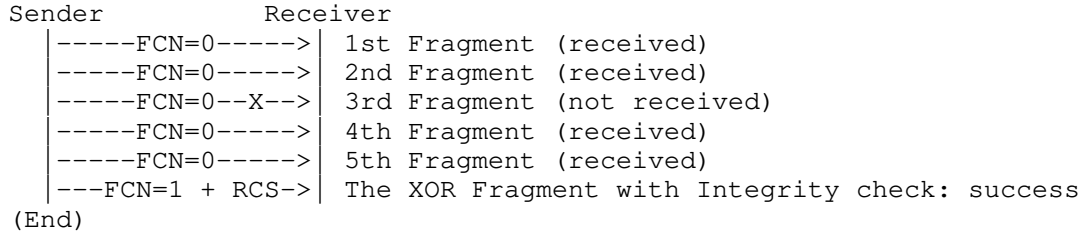

Figure 1: Successful transmission of a fragmented SCHC Packet with XORFEC over No-ACK mode: even though one fragment was lost (i.e., 3rd Fragment), it is recovered thanks to the additional XOR fragment.

Thus, even if with No-ACK mode there is no feedback from the receiver, by employing XORFEC, the receiver may successfully reassemble the original SCHC Packet. As a result, both the network reliability and the spectrum/bandwidth utlization efficiency are increased for a certain range of loss/error rates.

4.1.2.2. XORFEC over ACK-on-Error mode

In ACK-on-Error mode, the XOR is applied per Window. In case, when there is one Tile per Fragment, then one additional fragment is introduced per Window.

In Figure 2, the example (i.e., Figure 31) from [RFC8724] of ACK-on-Error mode of a SCHC Packet fragmented in 11 tiles is adapted when XORFEC is applied on ACK-on-Error mode is illustrated. A SCHC Packet is fragmented in 11 Tiles, with one Tile per SCHC Fragment, N=3, WINDOW_SIZE=7, and two lost SCHC Fragments.

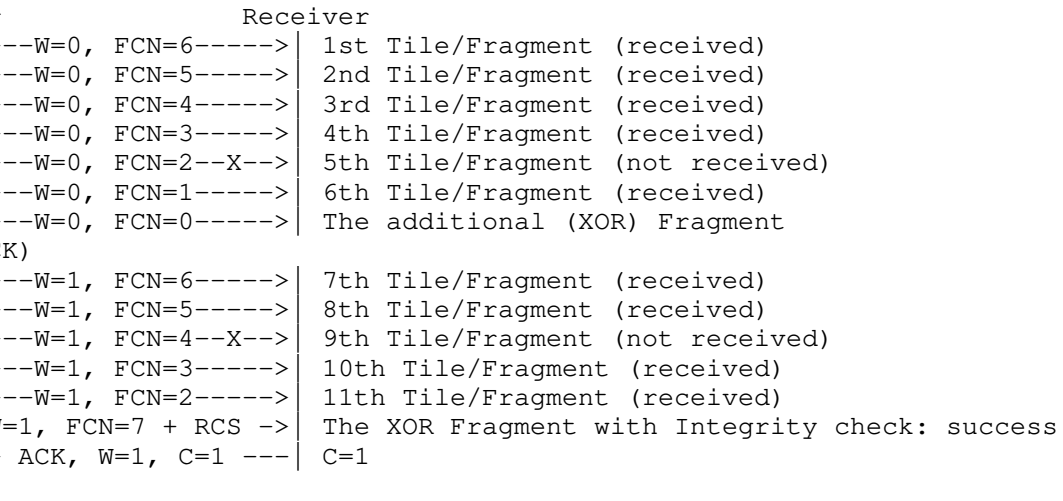

Figure 2: Successful transmission of a fragmented SCHC Packet with XORFEC over ACK-on-Error mode (11 Tiles, One Tile per SCHC Fragment, Two Lost SCHC Fragments): even though 2 fragments were lost (i.e., 5th and 9th Fragments), they were recovered thanks to the additional XOR fragments.

As it can be calculated, in the original example, there were in total 16 transmissions with two fragment losses, i.e., 11 original transmissions from the Sender, two retransmissions from the Sender, and three acknowledgments from the Receiver. In this XORFEC based approach, there are in total 14 transmissions, i.e., 11 original fragment transmissions from the Sender, two additional XOR transmissions from the Sender, and the ACK at the end from the Receiver. As a result, thanks to the XORFEC, the communication was reduced by two transmissions. Indeed, the ACK transmissions with the Bitmap of the missing fragments was not transmitted, and consequently the retransmissions of the missing fragments.

4.1.2.3. XORFEC over ACK-Always mode

Similar to ACK-on-Error mode, in ACK-Always, the XOR is applied per Window. In case, when there is one Tile per Fragment, then one additional fragment is introduced per Window.

In Figure 3, the example (i.e., Figure 34) from [RFC8724] when XORFEC is applied on ACK-Always is illustrated. A SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, N=3, WINDOW_SIZE=7 and two lost SCHC Fragments.

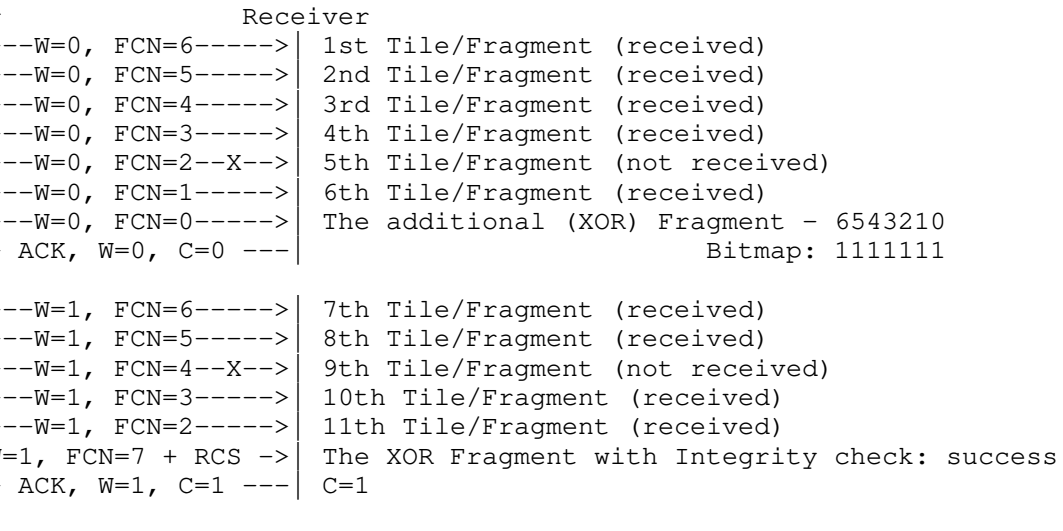

Figure 3: Successful transmission of a fragmented SCHC Packet with XORFEC over ACK-Always mode (11 Tiles, One Tile per SCHC Fragment, Two Lost SCHC Fragments): even though 2 fragments were lost (i.e., 5th and 9th Fragments), they were recovered thanks to the additional XOR fragments.

5. Security Considerations

TODO Security

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. 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/rfc/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/rfc/rfc8174>.

7.2. Informative References

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <https://www.rfc-editor.org/info/rfc8724>.

Acknowledgments

Thanks to Carles Gomez for useful design considerations, reviews and comments.

Authors' Addresses

Georgios Papadopoulos
IMT Atlantique
Email: georgios.papadopoulos@imt-atlantique.fr

Amaury Bruniaux
IMT Atlantique
Email: amaury.bruniaux@imt-atlantique.fr