Network Working Group A. Minaburo
Internet-Draft Acklio
Intended status: Informational L. Toutain
Expires: March 27, 2017 Institut MINES TELECOM ; TELECOM Bretagne
September 23, 2016

LPWAN Static Context Header Compression (SCHC) for IPv6 and UDP
draft-toutain-lpwan-ipv6-static-context-hc-00

Abstract

This document describes a header compression scheme for IPv6, IPv6/UDP based on static contexts. This technique is especially tailored for LPWA networks and could be extended to other protocol stacks.

During the IETF history several compression mechanisms have been proposed. First mechanisms, such as RoHC, are using a context to store header field values and send smaller incremental differences on the link. Values in the context evolve dynamically with information contained in the compressed header. The challenge is to maintain sender's and receiver's contexts synchronized even with packet losses. Based on the fact that IPv6 contains only static fields, 6LoWPAN developed an efficient context-free compression mechanisms, allowing better flexibility and performance.

The Static Context Header Compression (SCHC) combines the advantages of RoHC context which offers a great level of flexibility in the processing of fields, and 6LoWPAN behavior to elide fields that are known from the other side. Static context means that values in the context field do not change during the transmission, avoiding complex resynchronization mechanisms, incompatible with LPWA characteristics. In most of the cases, IPv6/UDP headers are reduced to a small identifier.

This document focuses on IPv6/UDP headers compression, but the mechanism can be applied to other protocols such as CoAP. It will be described in a separate document.

1. Introduction

Headers compression is mandatory to bring the internet protocols to the node within a LPWA network [I-D.minaburo-lp-wan-gap-analysis].

Nevertheless, LPWA networks offer good properties for an efficient header compression:

- Topology is star oriented, therefore all the packets follows the same path. For the needs of this draft, the architecture can be summarized to End-Systems (ES) exchanging information with LPWAN Application Server (LA). The exchange goes trhough a single LPWA Compressor (LC). In most of the cases, End Systems and LC form a star topology. ESs and LC maintain a static context for compression. Static context means that context information is not learned during the exchange.

- Traffic flows are mostly deterministic, since End-Systems embed built-in applications. Contrary to computers or smartphones, new applications cannot be easily installed.

First mechanisms such as RoHC use a context to store header field values and send smaller incremental differences on the link. The first version of RoHC targeted IP/UDP/RTP stack. RoHCv2 extends the principle to any protocol and introduces a formal notation [RFC4997] describing the header and associating compression functions to each

field. To be efficient the sender and the receiver must check that the context remains synchronized (i.e. contains the same values). Context synchronization imposes to periodically send a full header or at least dynamic fields. If fully compressed, the header can be compatible with LPWA constraints. However, the first exchanges or context resynchronisations impose to send uncompressed headers, which may be bigger than the original one. This will force the use of inefficient fragmentation mechanisms. For some LPWA technologies, duty cycle limits can also delay the resynchronization. Figure 1 illustrates this behavior.

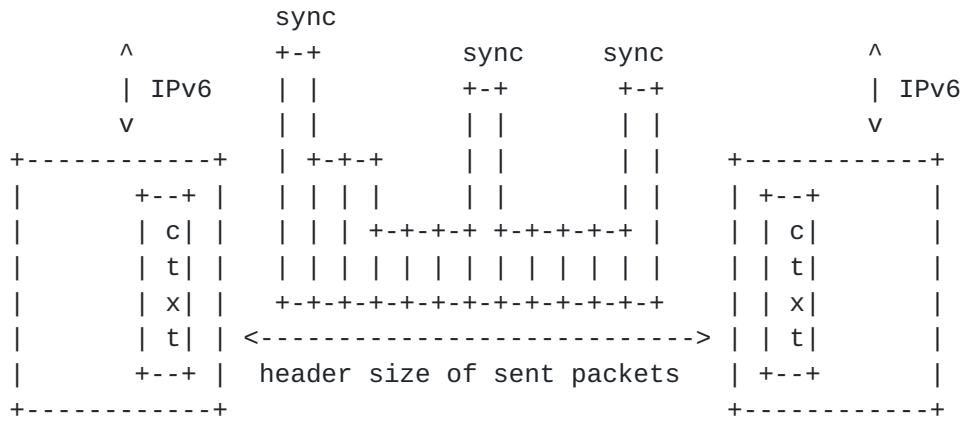

Figure 1: RoHC Compressed Header size evolution.

On the other hand, 6LoWPAN [RFC4944] is context-free based on the fact that IPv6, its extensions or UDP headers do not contain incremental fields. The compression mechanism described in [RFC6282] is based on sending a 2-byte bitmap, which describes how the header should be decompressed, either using some standard values or sending information after this bitmap. [RFC6282] also allows for UDP compression.

In the best case, when Hop limit is a standard value, flow label, DiffServ fields are set to 0 and Link Local addresses are used over a single hop network, the 6LoWPAN compressed header is reduced to 4 bytes. This compression ratio is possible because the IID are derived from the MAC addresses and the link local prefix is known from both sides. In that case, the IPv6 compression is 4 bytes and UDP compression is 2 bytes, which fills half of the payload of a SIGFOX frame, or more than 10% of a LoRaWAN payload (with spreading factor 12).

The Static Context Header Compression (SCHC) combines the advantages of RoHC context, which offers a great level of flexibility in the

processing of fields, and 6LoWPAN behavior to elide fields that are known from the other side. Static context means that values in the context field do not change during the transmission, avoiding complex resynchronization mechanisms, incompatible with LPWA characteristics. In most of the cases, IPv6/UDP headers are reduced to a small context identifier.

2. Static Context Header Compression

Static Context Header Compression (SCHC) avoids context synchronization, which is the most bandwidth-consuming operation in RoHC. Based on the fact that the nature of data flows is highly predictable in LPWA networks, a static context may be stored on the End-System (ES). The other end, the LPWA Compressor (LC) can learn the context through a provisioning protocol during the identification phase (for instance, as it learns the encryption key).

The context contains a list of rules (cf. Figure 2). Each rule contains itself a list of field descriptions composed of a target value (TV), a matching operator (MO) and a Compression/Decompression Function (CDF).

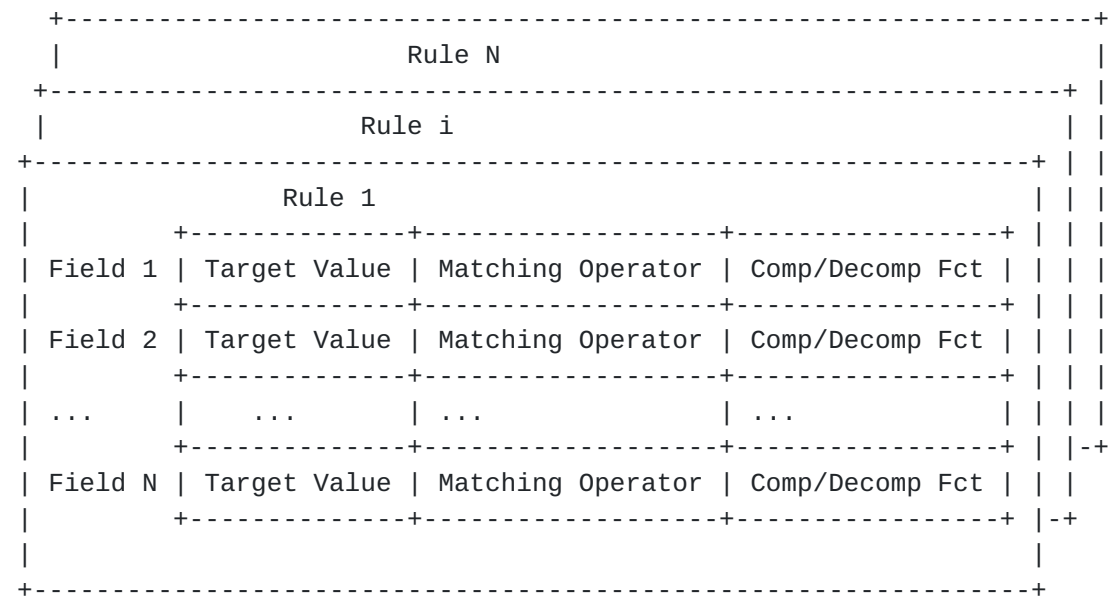

Figure 2: Compression Decompression Context

The rule does not describe the compressed/decompressed packet format which must be known from the compressor/decompressor. The rule just describes the compression/decompression behavior for a field.

The main idea of the compression scheme is to send the rule number (or rule id) to the other end instead of known field values.

Matching a field with a value and header compression are related operations; If a field matches a rule containing the value, it is not necessary to send it on the link. Since contexts are synchronized, reading the rule's value is enough to reconstruct the field's value at the other end.

On some other cases, the value need to be sent on the link to inform the other end. The field value may vary from one packet to another, therefore the field cannot be used to select the rule id.

2.1. Simple Example

A simple header is composed of 3 fields (F1, F2, F3). The compressor receives a packet containing respectively [F1:0x00, F2:0x1230, F3:0xABC0] in those fields. The Matching Operators (as defined in Section 3) allow to select Rule 5 as represented in Figure 3; F1 value is ignored and F2 and F3 packet field values are matched with those stored in the rule Target Values.

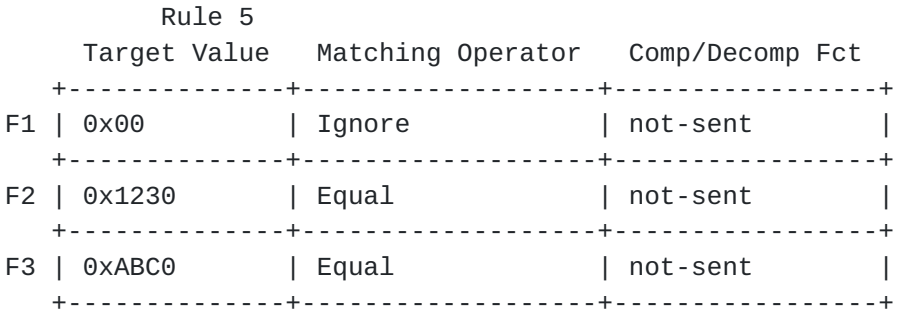

Figure 3: Matching Rule

The Compression/Decompression Function (as defined in Section 4 describes how the fields are compressed. In this example, all the fields are elided and only the rule number has to be sent to the other end.

The decompressor receives the rule number and reconstructs the header using the values stored in the Target Value column.

Note that F1 value will be set to 0x00 by the decompressor, even if the original header field was carrying a different value.

To allow a range of values for field F2 and F3, the MSB() Matching Operator and LSB() Compression/Decompression Function can be used (as defined in Section 3 and Section 4). In that case the rule will be rewritten as defined in Figure 4.

	Rule 5 Target Value	Matching Operator	Comp/Decomp Fct
F1	0x00	Ignore	not-sent
F2	0x1230	MSB(12)	LSB(4)
F3	0xABC0	MSB(12)	LSB(4)

Figure 4: Matching Rule

In that case, if a packet with the following header fields [F1:0x00, F2:0x1234, F3:0xABCD] arrives to the compressor, the new rule 5 will be selected and sent to the other end. The compressed header will be composed of the single byte [0x4D]. The decompressor receives the compressed header and follows the rule to reconstruct [0x00, 0x1234, 0xABCD] applying a OR operator between the target value stored in the rule and the compressed field value sent.

2.2. Packet processing

The compression/decompression process follows several steps:

- compression rule selection: the goal is to identify which rule will be used to compress the headers. To each field is associated a matching rule for compression. Each header field's value is compared to the corresponding target value stored in the rule for that field using the matching operator. If all the fields satisfied the matching operator, the packet is processed using this Compression Decompression Function functions. Otherwise the next rule is tested. If no eligible rule is found, then the packet is dropped.

- sending: The rule number is sent to the other end followed by data resulting from the field compression. The way the rule number is sent depends of the layer two technology and will be specified in a specific document. For exemple, it can either be included in a Layer 2 header or sent in the first byte of the L2 payload.

- decompression: The receiver identifies the sender through its device-id (e.g. MAC address) and select the appropriate rule through the rule number. It applies the compression decompression function to reconstruct the original header fields.

3. Matching operators

It may exist some intermediary cases, where part of the value may be used to select a field and a variable part has to be sent on the link. This is true for Least Significant Bits (LSB) where the most significant bit can be used to select a rule id and the least significant bits have to be sent on the link.

Several matching operators are defined:

- equal: a field value in a packet matches with a field value in a rule if they are equal.
- ignore: no check is done between a field value in a packet and a field value in the rule. The result is always true.
- MSB(length): a field value of length T in a packet matches with a field value in a rule if the most significant "length" bits are equal.

4. Compression Decompression Functions (CDF)

The Compression Decompression Functions (CDF) describe the action taken during the compression and inversely the action taken by the decompressor to restore the original value.

Function	Compression	Decompression
not-sent	elided	use value stored in ctxt
value-sent	send	build from received value
LSB(length)	send LSB	ctxt value OR rcvd value
compute-IPv6-length	elided	compute IPv6 length
compute-UDP-length	elided	compute UDP length
compute-UDP-checksum	elided	compute UDP checksum
ESiid-DID	elided	build IID from L2 ES addr
LAiid-DID	elided	build IID from L2 LA addr

Figure 5: Compression and Decompression Functions

Figure 5 lists all the functions defined to compress and decompress a field. The first column gives the function's name. The second and third columns outlines the compression/decompression process.

As with 6LoWPAN, the compression process may produce some data, where fields that were not compressed (or were partially compressed) will be sent in the order of the original packet. Information added by the compression phase must be aligned on byte boundaries, but each individual compression function may generate any size.

Field	Comp Decomp Fct	Behavior
IPv6 version IPv6 DiffServ IPv6 FL	not-sent	The value is not sent, but each end agrees on a value, which can be different from 0.
IPv6 NH	value-sent	Depending on the matching operator, the entire field value is sent or an adjustment to the context value
IPv6 Length	compute-IPv6-length	Dedicated fct to reconstruct value
IPv6 Hop Limit	not-sent+MO=ignore	The receiver takes the value stored in the context. It may be different from one originally sent, but in a star topology, there is no risk of loops
	not-sent+matching	Receiver and sender agree on a specific value.
	value-sent	Explicitly sent
IPv6 ESPrefix IPv6 LAPrefix	not-sent	The 64 bit prefix is stored on the context
	value-sent	Explicitly send 64 bits on the link
IPv6 ESiid IPv6 LAiid	not-sent	IID is not sent, but stored in the context
	ESiid-DID|LAiid-DID	IID is built from the ES/LA Dev. ID
	value-sent	IID is explicitly sent on the link. Size depends of the L2 technology
UDP ESport UDP LAport	not-sent	In the context
	value-sent	Send the 2 bytes of the port number
	LSB(length)	or least significant bits if MSB matching is specified in the matching operator.
UDP length	compute-UDP-length	Dedicated fct to reconstruct value
UDP Checksum	compute-UDP-checksum	Dedicated fct to reconstruct value

Figure 6: SCHC functions' example assignment for IPv6 and UDP

Figure 6 gives an example of function assignment to IPv6/UDP fields.

4.1. Compression Decompression Functions (CDF)

4.1.1. not-sent

The compressor do not sent the field value on the link. The decompressor restore the field value with the one stored in the matched rule.

4.1.2. value-sent

The compressor send the field value on the link, if the matching operator is "=". Otherwise the matching operator indicates the information that will be sent on the link. For a LSB operator only the Least Significant Bits are sent.

4.1.3. LSB(length)

The compressor sends the "length" Least Significant Bits. The decompressor combines with a OR operator the value received with the Target Value.

4.1.4. ESiid-DID, LAiid-DID

These functions are used to process respectively the End System and the LA Device Identifier (DID). The IID value is computed from device ID present in the Layer 2 header. The computation depends on the technology and the device ID size.

4.1.5. Compute-*

These functions compute the field value based on received information. They are elided during the compression and reconstructed during the decompression.

- compute-ipv6-length: compute the IPv6 length field as described in [RFC2460].
- compute-udp-length: compute the IPv6 length field as described in [RFC0768].
- compute-udp-checksum: compute the IPv6 length field as described in [RFC0768].

5. Examples

This section gives some scenarios of the compression mechanism for IPv6/UDP. The goal is to illustrate the SCHC behavior.

5.1. IPv6/UDP compression in a star topology

The most common case will be a LPWA end-system embeds some applications running over CoAP. In this example, the first flow is for the device management based on CoAP using Link Local addresses and UDP ports 123 and 124. The second flow will be a CoAP server for measurements done by the end-system (using ports 5683) and Global Addresses alpha::IID/64 to beta::1/64. The last flow is for legacy applications using different ports numbers, the destination is gamma::1/64.

Figure 7 presents the protocol stack for this end-system. IPv6 and UDP are represented with dotted lines since these protocols are compressed on the radio link.

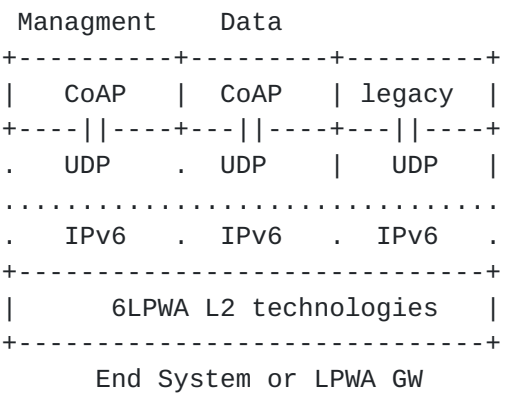

Figure 7: Simplified Protocol Stack for LP-WAN

Note that in some LPWA technologies, only End Systems have a device ID . Therefore it is necessary to define statically an IID for the Link Local address for the LPWA Compressor.

Rule 0

Field	Value	Match	Function	Sent
IPv6 version	6	equal	not-sent	
IPv6 DiffServ	0	equal	not-sent	
IPv6 Flow Label	0	equal	not-sent	
IPv6 Length		ignore	comp-IPv6-l	
IPv6 Next Header	17	equal	not-sent	
IPv6 Hop Limit	255	ignore	not-sent	
IPv6 ESprefix	FE80::/64	equal	not-sent	
IPv6 ESiid		ignore	ESiid-DID	
IPv6 LCprefix	FE80::/64	equal	not-sent	

IPv6 LAiid	::1	equal	not-sent		
UDP ESport	123	equal	not-sent		
UDP LAport	124	equal	not-sent		
UDP Length		ignore	comp-UDP-l		
UDP checksum		ignore	comp-UDP-c		

Rule 1

Field	Value	Match	Function	Sent
IPv6 version	6	equal	not-sent	
IPv6 DiffServ	0	equal	not-sent	
IPv6 Flow Label	0	equal	not-sent	
IPv6 Length		ignore	comp-IPv6-l	
IPv6 Next Header	17	equal	not-sent	
IPv6 Hop Limit	255	ignore	not-sent	
IPv6 ESprefix	alpha/64	equal	not-sent	
IPv6 ESiid		ignore	ESiid-DID	
IPv6 LAprefix	beta/64	equal	not-sent	
IPv6 LAiid	::1000	equal	not-sent	
UDP ESport	5683	equal	not-sent	
UDP LAport	5683	equal	not-sent	
UDP Length		ignore	comp-UDP-l	
UDP checksum		ignore	comp-UDP-c	

Rule 2

Field	Value	Match	Function	Sent
IPv6 version	6	equal	not-sent	
IPv6 DiffServ	0	equal	not-sent	
IPv6 Flow Label	0	equal	not-sent	
IPv6 Length		ignore	comp-IPv6-l	
IPv6 Next Header	17	equal	not-sent	
IPv6 Hop Limit	255	ignore	not-sent	
IPv6 ESprefix	alpha/64	equal	not-sent	
IPv6 ESiid		ignore	ESiid-DID	
IPv6 LAprefix	gamma/64	equal	not-sent	
IPv6 LAiid	::1000	equal	not-sent	
UDP ESport	8720	MSB(12)	LSB(4)	lsb
UDP LAport	8720	MSB(12)	LSB(4)	lsb
UDP Length		ignore	comp-UDP-l	
UDP checksum		ignore	comp-UDP-c	

```
+================+=========+========+=============++======+
```

Figure 8: Context rules

All the fields described in the three rules Figure 8 are present in the IPv6 and UDP headers. The ESDevice-ID value is found in the L2 header.

The second and third rules use global addresses. The way the ES learn the prefix is not in the scope of the document. One possible way is to use a management protocol to set up in both end rules the prefix used on the LPWA network.

The third rule compresses port numbers on 4 bits. This value is selected to maintain alignment on byte boundaries for the compressed header.

6. Acknowledgements

Thanks to Dominique Barthel, Arunprabhu Kandasamy, Antony Markovski, Alexander Pelov, Juan Carlos Zuniga for useful design consideration.

7. Normative References

[I-D.minaburo-lp-wan-gap-analysis]
Minaburo, A., Pelov, A., and L. Toutain, "LP-WAN GAP Analysis", draft-minaburo-lp-wan-gap-analysis-01 (work in progress), February 2016.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[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>.

[RFC4997] Finking, R. and G. Pelletier, "Formal Notation for RObust Header Compression (ROHC-FN)", RFC 4997, DOI 10.17487/RFC4997, July 2007, <http://www.rfc-editor.org/info/rfc4997>.

[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>.

Authors' Addresses

Ana Minaburo
Acklio
2bis rue de la Chataigneraie
35510 Cesson-Sevigne Cedex
France

Email: ana@ackl.io

Laurent Toutain
Institut MINES TELECOM ; TELECOM Bretagne
2 rue de la Chataigneraie
CS 17607
35576 Cesson-Sevigne Cedex
France

Email: Laurent.Toutain@telecom-bretagne.eu