6LPWA Static Context Header Compression (SCHC) for CoAP

draft-toutain-lpwan-coap-static-context-hc-00

Abstract

This draft discusses the way SCHC can be applied to CoAP headers and extend the number of functions (CDF) to optimize compression.

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 May 1, 2017.

Copyright Notice

Copyright (c) 2016 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 (http://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 to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

[I-D.toutain-lpwan-ipv6-static-context-hc] defines a compression technique for LPWA network based on static context. This context is said static since the element values composing the context are not learned during packet exchanges but previously installed. The context is known by both ends. A context is composed of a set of rules (referenced by rule ids). A rule describes the header fields with some associated Target Values (TV). A Matching Operator (MO) is associated to each field. The rule is selected if all the MO matches. A Compression Decompression Function is associated to each field to define the link between the compressed and decompressed value for a specific field.

This draft discusses the way SCHC can be applied to CoAP headers and extend the number of functions (CDF) to optimize compression.

2. Compressing CoAP

CoAP [RFC7252] is an implementation of the REST architecture for constrained devices. Gateway between CoAP and HTTP can be easily build since both protocol uses the same address space (URL), caching mechanisms and methods.

Nevertheless, if limited, the size of a CoAP header may be incompatible with LPWAN constraints and some compression may be needed to reduce the header size. CoAP compression is not straightforward. Some differences between IPv6/UDP and CoAP can be enlighten. CoAP differs from IPv6 and UDP protocols:

- IPv6 and UDP are symmetrical protocols. The same fields are found in the request and in the answer, only location in the header may change (e.g. source and destination fields). A CoAP request is different from an answer. For instance, the URI-path option is mandatory in the request and may not be found in the response.

- CoAP also obeys to the client/server paradigm and the compression rate can be different if the request is issued from a LPWAN node or from an external device. For instance in the former case the token size may be set to one byte. In the latter case, the token size cannot be constraint and be up to 15 byte long.

- In IPv6, main header and UDP fields have a fixed size. In CoAP, Token size may vary from 0 to 15 bytes, length is given by a field in the header. More systematically, the options are described using the Type-Length-Value principle. Evenmore regarding the option size value, the coding will be different.
options type in CoAP are not defined with the same value. The Delta TLV coding makes that the type is not independent of previous option and may vary regarding the options contained in the header.

2.1. CoAP usages

A LPWAN node can either be a client or a server and sometimes both. In the client mode, the LPWAN node sends request to a server and expected answer or acknowledgements. Acknowledgements can be at 2 different levels:

- transport level, a CON message is acknowledged by an ACK message. NON confirmable messages are not acknowledged.
- REST level, a REST request is acknowledged by an "error" code. [RFC7967] defines an option to limit the number of acknowledgements.

Note that acknowledgement can be optimized and a REST level acknowledgement can be used as a transport level acknowledgement.

2.2. CoAP protocol analysis

CoAP defines the following fields:

- version (2 bits): this field can be elided during a compression.
- type (2 bits): defines the type of the transport messages, 4 values are defined. Regarding the type of exchange, if only NON messages are sent or CON/ACK messages, this field can be reduced to 0 or 1 bit.
- token length (4 bytes). The standard allows up to 15 bytes for a token length. If a fixed token size is chosen, then this field can be elided. If some variation in length are needed then 1 or 2 bits could be enough for most of LPWAN applications.
- code (8 bits). This field codes the request and the response values. CoAP represents in a more compact way, coding used in HTTP, but the coding is not optimal.
- message id (16 bits). This value is used to acknowledge CON frames. The size of this field is computed to allow the anticipation (how many frames can be sent without acknowledgement). When a value is used, [RFC7252] defines the time before it can be reused without ambiguities. This size may
be too large for a LPWAN node sending or receiving few messages a day.

- Token (0 to 15 bytes). Token identifies active flows. Regarding the usage (stability of in time and limited number), a short token (1 Byte) can be enough.

- options are coded through delta-TLV. The delta-T depends of previous values, length is encoded inside the option. [RFC7252] distinguishes repeatable options that can appear several time in the header. Among them we can distinguish:
  
  * list options which appear several time in the header but are exclusive such as the Accept option.
  
  * cumulative options which appear several time in the header but are part of a more generic value such as Uri-Path and Uri-Query.

For a given flow some value options are stable through time. Observe, ETag, If-Match, If-None-Match and Size varies in each message. Options can be stored in a SCHC context and cumulative options can be stored globally.

The CoAP protocol must not be altered by the compression/decompression phase, but if no semantic is attributed to a value, it may be changed during this phase. For instance the compression phase may reduce the size of a token but must maintain its unicity. The decompressor will not be able to restore the original value but behavior will remain the same. If no special semantic is assigned to the token, this will be transparent. If a special semantic is assigned to the token, its compression may not be possible.

This implies that the compressor/decompressor must be aware of the protocol state machine and do not processes request and response the same way.

A conservative compression leaves the field value unchanged. Non conservative compression can be used when a CoAP implementation has not been defined to work specifically with LPWAN network and uses large values for fields.

2.2.1. CoAP Compression Decompression Function

To compress more efficiently CoAP message, several Compression/Decompression Function (CDF) must be defined.
2.2.1.1. Static-mapping

The goal of static-mapping is to reduce the size of a field by allocating shorter value. The mapping is known by both ends and stored in a table in both end context. The Static-mapping is conservative.

Static-mapping may be applied to several fields. For instance the type field may be reduced from 2 bits to 1 bit if only CON/ACK type is used, but the main benefit is compressing the code field.

The CoAP code field defines a tricky way to ensure compatibility with HTTP values. Nevertheless only 21 values are defined by [RFC7252] compared to the 255 possible values. So it could efficiently be coded on 5 bits. To allow flexibility and evolution if new codes are introduced, a static mapping table is associated to each instance of this function.

Figure 1 gives a possible mapping, it can be changed to add new codes or reduced if some values are never used by both ends.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td>0.01</td>
<td>GET</td>
<td>0x01</td>
</tr>
<tr>
<td>0.02</td>
<td>POST</td>
<td>0x02</td>
</tr>
<tr>
<td>0.03</td>
<td>PUT</td>
<td>0x03</td>
</tr>
<tr>
<td>0.04</td>
<td>DELETE</td>
<td>0x04</td>
</tr>
<tr>
<td>0.05</td>
<td>FETCH</td>
<td>0x05</td>
</tr>
<tr>
<td>0.06</td>
<td>PATCH</td>
<td>0x06</td>
</tr>
<tr>
<td>0.07</td>
<td>iPATCH</td>
<td>0x07</td>
</tr>
<tr>
<td>2.01</td>
<td>Created</td>
<td>0x08</td>
</tr>
<tr>
<td>2.02</td>
<td>Deleted</td>
<td>0x09</td>
</tr>
<tr>
<td>2.03</td>
<td>Valid</td>
<td>0x0A</td>
</tr>
<tr>
<td>2.04</td>
<td>Changed</td>
<td>0x0B</td>
</tr>
<tr>
<td>2.05</td>
<td>Content</td>
<td>0x0C</td>
</tr>
<tr>
<td>4.00</td>
<td>Bad Request</td>
<td>0x0D</td>
</tr>
<tr>
<td>4.01</td>
<td>Unauthorized</td>
<td>0x0E</td>
</tr>
<tr>
<td>4.02</td>
<td>Bad Option</td>
<td>0x0F</td>
</tr>
<tr>
<td>4.03</td>
<td>Forbidden</td>
<td>0x10</td>
</tr>
<tr>
<td>4.04</td>
<td>Not Found</td>
<td>0x11</td>
</tr>
<tr>
<td>4.05</td>
<td>Method Not Allowed</td>
<td>0x12</td>
</tr>
<tr>
<td>4.06</td>
<td>Not Acceptable</td>
<td>0x13</td>
</tr>
<tr>
<td>4.12</td>
<td>Precondition Failed</td>
<td>0x14</td>
</tr>
<tr>
<td>4.13</td>
<td>Request Entity Too Large</td>
<td>0x15</td>
</tr>
<tr>
<td>4.15</td>
<td>Unsupported Content-Format</td>
<td>0x16</td>
</tr>
<tr>
<td>5.00</td>
<td>Internal Server Error</td>
<td>0x17</td>
</tr>
<tr>
<td>5.01</td>
<td>Not Implemented</td>
<td>0x18</td>
</tr>
<tr>
<td>5.02</td>
<td>Bad Gateway</td>
<td>0x19</td>
</tr>
<tr>
<td>5.03</td>
<td>Service Unavailable</td>
<td>0x1A</td>
</tr>
<tr>
<td>5.04</td>
<td>Gateway Timeout</td>
<td>0x1B</td>
</tr>
<tr>
<td>5.05</td>
<td>Proxying Not Supported</td>
<td>0x1C</td>
</tr>
</tbody>
</table>

Figure 1: CoAP code mapping

This CDF can also be applied to path to send a reference instead of the path value.

2.2.1.2. Remapping

With dynamic mapping, the mapping is done dynamically, which means that the other end has no way to learn the original value. This function is not conservative. The mapping context must be stored in a reliable way on the compressor, if lost the session with LPWAN node
will be lost, which can generate a traffic increase on the LPWA network.

This function converts a large number to a smaller one and maintain bi-directional mapping. If the field has no semantic, such as a CoAP token or a message ID, this will reduce the size of the information sent on the link. This mapping only applies for request compression, answers must keep the value original value.

For instance a compression receives a CoAP request with a large token. The compressor reduces the token size by allocating a unused value in a smaller space. When the response come back, the compressor exchange the smallest token with the original one.

This mean that the compressor must be aware of the CoAP state machine, to identify a request and its associated response, but also determine when a token value can be reused.

2.2.1.3. Reduce-entropy

Reduce-entropy is a non-conservative function. the goal is to minimize the increase in a field value. It may be used for the observe option, all increase in the original sequence number will lead to an increase of 1 in the compressed value.

For instance a LPWAN node is a CoAP server and receives Observe responses coming from an outside client. The client uses a clock to generate Observe sequence number. If that value has non particular meaning for the CoAP server, increase of 1 will not change the protocol behavior. Reordering works the same way as for original Observe.

2.2.2. CoAP mandatory header

Figure 2 proposes some function assignments to the CoAP header fields.
2.2.3.3.1. Mandatory header with CON message

In this first scenario, the LPWAN compressor receives from outside client a POST message, which is immediately acknowledged by the ES. For this simple scenario, the rules are described Figure 3
Figure 3: CoAP Context to compress header without token

Figure 3 gives a simple compression rule for CoAP headers without tokens.

The version fields and Token Length are elided. Code has shrunk to 5 bits using the static-mapping function. Message-ID has shrunk to 9 bits to preserve alignment on byte boundary.

Figure 4 shows the time diagram of the exchange. A LPWAN Application Server sends a CON message. Compression reduces the header sending only the Type, a mapped code and the Message ID is change to a value on 9 bits. The receiver decompress the header. The message ID value is changed.

The CON message is a request, therefore the LC process to a dynamic mapping. When the ES receives the ACK message, this will not initiate locally a the message ID mapping since it is a response. The LC receives the ACK and uncompress it to restore the original value. Dynamic Mapping context lifetime follows the same rules as message ID duration.
Figure 4: Compression with global addresses

Note that the compressor and decompressor must understand the CoAP protocol:

- The LC compressor detects a new transport request and allocate a new dynamic mapping value.

- When receiving a response the ES compressor ES detects that this is a response (type=2) therefore the message ID value in unchanged.

- The upstream compressor detects that is an REST answer (code 2.05) therefore the path option is not inserted in the uncompress header

2.2.3.2. Exchange with token

The following scenario introduces tokens. The LC manages two remapping contexts. One for Message ID and the other for token. ES manages one context for Message ID. Mapping is trigged by the reception of CON messages to compress or CoAP requests to compress. Note that the compressed message ID size has been reduced to 7 bits, compared to the previous example, to maintain byte boundary alignment.
<table>
<thead>
<tr>
<th>Field</th>
<th>Function</th>
<th>Ctxt Value</th>
<th>Sent compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoAP version</td>
<td>not-sent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoAP Type</td>
<td>value-sent</td>
<td></td>
<td>TT</td>
</tr>
<tr>
<td>CoAP TKL</td>
<td>compute-token-length</td>
<td></td>
<td>LL</td>
</tr>
<tr>
<td>CoAP Code</td>
<td>map-code</td>
<td>mapping table</td>
<td>CCCC C</td>
</tr>
<tr>
<td>CoAP MID</td>
<td>remapping</td>
<td>7 bits</td>
<td>M-ID</td>
</tr>
<tr>
<td>CoAP Token</td>
<td>remapping</td>
<td>8 bits</td>
<td>token</td>
</tr>
<tr>
<td>CoAP Path</td>
<td>not-sent</td>
<td>/data/humidity</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5: CoAP Context to compress header with token
Figure 6: Compression with token

3. Normative References

[I-D.toutain-lpwan-ipv6-static-context-hc]


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