6LoWPAN Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 5, 2012

C. Bormann
Universitaet Bremen TZI

October 03, 2011

6LoWPAN Generic Compression of Headers and Header-like Payloads

draft-bormann-6lowpan-ghc-03

Abstract

This short I-D provides a complete design for a simple addition to 6LoWPAN Header Compression that enables the compression of generic headers and header-like payloads.

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 April 5, 2012.

Copyright Notice

Table of Contents

1. Introduction

1.1. The Header Compression Coupling Problem

6LoWPAN-HC [RFC6282] defines a scheme for header compression in 6LoWPAN [RFC4944] packets. As with most header compression schemes, a new specification is needed for every new kind of header that needs to be compressed. In addition, [RFC6282] does not define an extensibility scheme like the ROHC profiles defined in ROHC [RFC3095] [RFC5795]. This leads to the difficult situation that 6LoWPAN-HC tended to be reopened and reexamined each time a new header receives consideration (or an old header is changed and reconsidered) in the 6lowpan/roll/core cluster of IETF working groups. While [RFC6282] finally got completed, the underlying problem remains unsolved.

The purpose of the present contribution is to plug into [RFC6282] as is, using its NHC (next header compression) concept. We add a slightly less efficient, but vastly more general form of compression for headers of any kind and even for header-like payloads such as those exhibited by routing protocols, DHCP, etc. The objective is to arrive at something that can be defined on a single page and implemented in a couple of lines of code, as opposed to a general data compression scheme such as that defined in [RFC1951].

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The term "byte" is used in its now customary sense as a synonym for "octet".

2. 6LoWPAN-GHC

The format of a compressed header or payload is a simple bytecode. A compressed header consists of a sequence of pieces, each of which begins with a code byte, which may be followed by zero or more bytes as its argument. Some code bytes cause bytes to be laid out in the destination buffer, some simply modify some decompression variables.

At the start of decompressing a header or payload within a L2 packet (= fragment), variables "sa" and "na" are initialized as zero.

The code bytes are defined as follows:

code byte	Action	Argument
0kkkkkkk	Append k = 0b0kkkkkkk bytes of data in the bytecode argument (k < 96)	k bytes of data
1000nnnn	Append 0b0000nnnn+2 bytes of zeroes	
10010000	STOP code (end of compressed data, see Section 3.2)	
101nssss	Set up extended arguments for a backreference: sa += 0b0ssss000, na += 0b0000n000	
11nnnkkk	Backreference: n = na+0b00000nnn+2; s = 0b00000kkk+sa+n; append n bytes from previously output bytes, starting s bytes to the left of the current output pointer; set sa = 0, na = 0	

Note that the following bit combinations are reserved at this time: 011xxxxx (possibly for Appendix B.1), and 1001nnnn (where nnnn > 0, possibly for Appendix B.2).

For the purposes of the backreferences, the expansion buffer is initialized with the pseudo-header as defined in [RFC2460], at the end of which the target buffer begins. These pseudo-header bytes are therefore available for backreferencing, but not copied into the final result.

3. Integrating 6LoWPAN-GHC into 6LoWPAN-HC

6LoWPAN-GHC is intended to plug in as an NHC format for 6LoWPAN-HC [RFC6282]. This section shows how this can be done (without supplying the detailed normative text yet, although it could be implemented from this page).

3.1. Compressing payloads (UDP and ICMPv6)

GHC is by definition generic and can be applied to different kinds of packets. All the examples given in Appendix A are for ICMPv6 packets; a single NHC value suffices to define an NHC format for ICMPv6 based on GHC (see below).

In addition it may be useful to include an NHC format for UDP, as many headerlike payloads (e.g., DHCPv6) are carried in UDP. [RFC6282] already defines an NHC format for UDP (11110CPP). What remains to be done is to define an analogous NHC byte formatted, e.g. as shown in Figure 1, and simply reference the existing specification, indicating that for 0b11010cpp NHC bytes, the UDP payload is not supplied literally but compressed by 6LoWPAN-GHC.

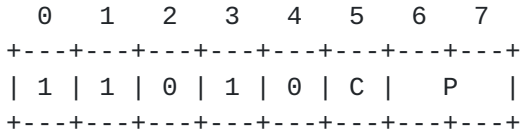

Figure 1: Proposed NHC byte for UDP GHC

To stay in the same general numbering space, we propose 0b11011111 as the NHC byte for ICMPv6 GHC (Figure 2).

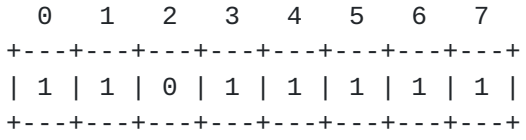

Figure 2: Proposed NHC byte for ICMPv6 GHC

3.2. Compressing extension headers

If the compression of specific extension headers is considered desirable, this can be added in a similar way, e.g. as in Figure 3 (however, probably only EID 0 to 3 need to be assigned). As there is no easy way to extract the length field from the GHC-encoded header before decoding, this would make detecting the end of the extension header somewhat complex. The easiest (and most efficient) approach is to completely elide the length field (in the same way NHC already

elides the next header field in certain cases) and reconstruct it only on decompression. To serve as a terminator for the extension header, the reserved bytecode 0b10010000 has been assigned as a stop marker -- this is only needed for extension headers, not for final payloads.

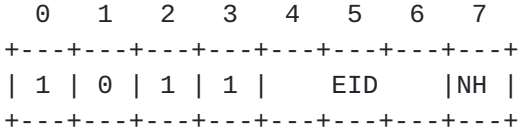

Figure 3: Proposed NHC byte for extension header GHC

3.3. Indicating GHC capability

The 6LoWPAN baseline includes just [RFC4944], [RFC6282], [I-D.ietf-6lowpan-nd] (see [I-D.bormann-6lowpan-roadmap]). To enable the use of GHC, 6LoWPAN nodes need to know that their neighbors implement it. While this can also simply be administratively required, a transition strategy as well as a way to support mixed networks is required.

One way to know a neighbor does implement GHC is receiving a packet from that neighbor with GHC in it ("implicit capability detection"). However, there needs to be a way to bootstrap this, as nobody ever would start sending packets with GHC otherwise.

To minimize the impact on [I-D.ietf-6lowpan-nd], we propose adding an ND option 6LoWPAN Capability Indication (6CIO), as illustrated in Figure 4.

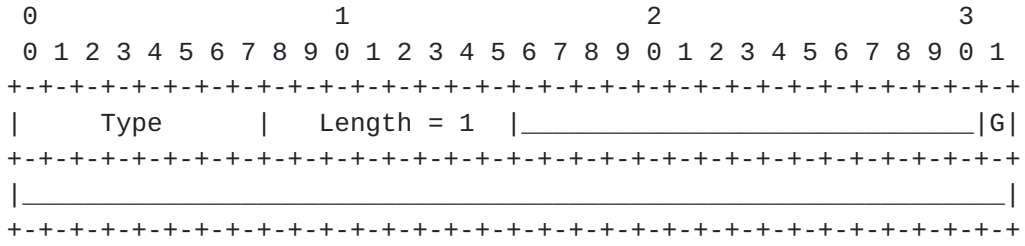

Figure 4: 6LoWPAN Capability Indication Option (6CIO)

The G bit indicates whether the node sending the option is GHC capable.

The 6CIO option will typically only be ever sent in 6LoWPAN-ND RS packets (it then cannot itself be GHC compressed unless the host desires to limit itself to talking to GHC capable routers); the resulting 6LoWPAN-ND RA can already make use of GHC and thus indicate

GHC capability implicitly, which in turn allows the nodes to use GHC in the 6LoWPAN-ND NS/NA exchange.

6CIO can also be used for future options that need to be negotiated between 6LoWPAN peers; an IANA registry will administrate the flags. (Bits marked by underscores in Figure 4 are reserved for future allocation, i.e., they MUST be sent as zero and MUST be ignored on reception until allocated. Length values larger than 1 MUST be supported for future extensions; the additional bits in the option are then reserved in the same way. For the purposes of the IANA registry, the bits are numbered in msb-first order from the 16th bit of the option onwards, i.e., the G bit is flag number 15.) (Additional bits may also be used by a followon version of this document if some bit combinations that have been left reserved here are then used in an upward compatible manner.)

4. IANA considerations

In the IANA registry for the 6LOWPAN_NHC header type, IANA would need to add the assignments in Figure 5.

```
 10110IIN: Extension header GHC                [RFCthis]
 11010CPP: UDP GHC                             [RFCthis]
 11011111: ICMPv6 GHC                          [RFCthis]
```

Figure 5: IANA assignments for the NHC byte

IANA needs to allocate an ND option number for 6CIO.

An IANA registry is needed for 6LoWPAN capability flags. (Policy TBD.)

5. Security considerations

The security considerations of [RFC4944] and [RFC6282] apply. As usual in protocols with packet parsing/construction, care must be taken in implementations to avoid buffer overflows and in particular (with respect to the back-referencing) out-of-area references during decompression.

One additional consideration is that an attacker may send a forged packet that makes a second node believe a third victim node is GHC-capable. If it is not, this may prevent packets sent by the second node from reaching the third node.

No mitigation is proposed (or known) for this attack, except that a node that does implement GHC is not vulnerable. However, with unsecured ND, a number of attacks with similar outcomes are already possible, so there is little incentive to make use of this additional attack. With secured ND, 6CIO is also secured; nodes relying on secured ND therefore should use 6CIO bidirectionally (and limit the implicit capability detection to secured ND packets carrying GHC) instead of basing their neighbor capability assumptions on receiving any kind of unprotected packet.

6. Acknowledgements

Colin O'Flynn has repeatedly insisted that some form of compression for ICMPv6 and ND packets might be beneficial. He actually wrote his own draft, [I-D.oflynn-6lowpan-icmphc], which compresses better, but addresses basic ICMPv6/ND only and needs a much longer spec (around 17 pages of detailed spec, as compared to the single page of core spec here). This motivated the author to try something simple, yet general. Special thanks go to Colin for indicating that he indeed considers his draft superseded by the present one.

The examples given are based on pcap files that Colin O'Flynn and Owen Kirby provided.

7. References

7.1. Normative References

[I-D.ietf-6lowpan-nd]
Shelby, Z., Chakrabarti, S., and E. Nordmark, "Neighbor Discovery Optimization for Low Power and Lossy Networks (6LoWPAN)", draft-ietf-6lowpan-nd-17 (work in progress), June 2011.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.

7.2. Informative References

[I-D.bormann-6lowpan-roadmap]
Bormann, C., "6LoWPAN Roadmap and Implementation Guide", draft-bormann-6lowpan-roadmap-00 (work in progress), March 2011.

[I-D.ietf-core-link-format]
Shelby, Z., "CoRE Link Format", draft-ietf-core-link-format-07 (work in progress), July 2011.

[I-D.oflynn-6lowpan-icmphc]
O'Flynn, C., "ICMPv6/ND Compression for 6LoWPAN Networks", draft-oflynn-6lowpan-icmphc-00 (work in progress), July 2010.

[RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header

Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.

[RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, March 2010.

Appendix A. Examples

This section demonstrates some relatively realistic examples derived from actual PCAP dumps taken at previous interops. Unfortunately, for these dumps, no context information was available, so the relatively powerful effect of context-based compression is not shown. (TBD: Add a couple DHCP examples.)

Figure 6 shows an RPL DODAG Information Solicitation, a quite short RPL message that obviously cannot be improved much.

```
IP header:
 60 00 00 00 00 08 3a ff fe 80 00 00 00 00 00 00
 02 1c da ff fe 00 20 24 ff 02 00 00 00 00 00 00
 00 00 00 00 00 00 00 1a
Payload:
 9b 00 6b de 00 00 00 00
Pseudoheader:
 fe 80 00 00 00 00 00 00 02 1c da ff fe 00 20 24
 ff 02 00 00 00 00 00 00 00 00 00 00 00 00 00 1a
 00 00 00 08 00 00 00 3a
copy: 04 9b 00 6b de
4 nulls: 82
Compressed:
 04 9b 00 6b de 82
Was 8 bytes; compressed to 6 bytes, compression factor 1.33
```

Figure 6: A simple RPL example

Figure 7 shows an RPL DODAG Information Object, a longer RPL control message that is improved a bit more (but would likely benefit additionally from a context reference). Note that the compressed output exposes an inefficiency in the simple-minded compressor used to generate it; this does not devalue the example since constrained nodes are quite likely to make use of simple-minded compressors.

```
   IP header:
    60 00 00 00 00 5c 3a ff fe 80 00 00 00 00 00 00
    02 1c da ff fe 00 30 23 ff 02 00 00 00 00 00 00
    00 00 00 00 00 00 00 1a
   Payload:
    9b 01 7a 5f 00 f0 01 00 88 00 00 00 20 02 0d b8
    00 00 00 00 00 00 00 ff fe 00 fa ce 04 0e 00 14
    09 ff 00 00 01 00 00 00 00 00 00 00 08 1e 80 20
    ff ff ff ff ff ff ff ff 00 00 00 00 20 02 0d b8
    00 00 00 00 00 00 00 ff fe 00 fa ce 03 0e 40 00
    ff ff ff ff 20 02 0d b8 00 00 00 00
   Pseudoheader:
    fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23
    ff 02 00 00 00 00 00 00 00 00 00 00 00 00 00 1a
    00 00 00 5c 00 00 00 3a
   copy: 09 9b 01 7a 5f 00 f0 01 00 88
   3 nulls: 81
   copy: 04 20 02 0d b8
   7 nulls: 85
   ref(52): ff fe 00 -> ref 101nssss 0 6/11nnnkkk 1 1: a6 c9
   copy: 08 fa ce 04 0e 00 14 09 ff
   2 nulls: 80
   copy: 01 01
   7 nulls: 85
   copy: 06 08 1e 80 20 ff ff
   ref(2): ff ff -> ref 11nnnkkk 0 0: c0
   ref(4): ff ff ff ff -> ref 11nnnkkk 2 0: d0
   4 nulls: 82
   ref(48): 20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 fa ce
    -> ref 101nssss 1 4/11nnnkkk 6 0: b4 f0
   copy: 03 03 0e 40
   ref(9): 00 ff -> ref 11nnnkkk 0 7: c7
   ref(28): ff ff ff -> ref 101nssss 0 3/11nnnkkk 1 1: a3 c9
   ref(24): 20 02 0d b8 00 00 00 00
    -> ref 101nssss 0 2/11nnnkkk 6 0: a2 f0
   Compressed:
    09 9b 01 7a 5f 00 f0 01 00 88 81 04 20 02 0d b8
    85 a6 c9 08 fa ce 04 0e 00 14 09 ff 80 01 01 85
    06 08 1e 80 20 ff ff c0 d0 82 b4 f0 03 03 0e 40
    c7 a3 c9 a2 f0
   Was 92 bytes; compressed to 53 bytes, compression factor 1.74
```

Figure 7: A longer RPL example

Similarly, Figure 8 shows an RPL DAO message. One of the embedded addresses is copied right out of the pseudoheader, the other one is effectively converted from global to local by providing the prefix FE80 literally, inserting a number of nulls, and copying (some of) the IID part again out of the pseudoheader. Note that a simple implementation would probably emit fewer nulls and copy the entire IID; there are multiple ways to encode this 50-byte payload into 27 bytes.

```
IP header:
 60 00 00 00 00 32 3a ff 20 02 0d b8 00 00 00 00
 00 00 00 ff fe 00 33 44 20 02 0d b8 00 00 00 00
 00 00 00 ff fe 00 11 22
Payload:
 9b 02 58 7d 01 80 00 f1 05 12 00 80 20 02 0d b8
 00 00 00 00 00 00 00 ff fe 00 33 44 06 14 00 80
 f1 00 fe 80 00 00 00 00 00 00 00 00 ff fe 00
 11 22
Pseudoheader:
 20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 33 44
 20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 11 22
 00 00 00 32 00 00 00 3a
copy: 0c 9b 02 58 7d 01 80 00 f1 05 12 00 80
ref(52): 20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 33 44
 -> ref 101nssss 1 4/11nnnkkk 6 4: b4 f4
copy: 08 06 14 00 80 f1 00 fe 80
9 nulls: 87
ref(58): ff fe 00 11 22 -> ref 101nssss 0 6/11nnnkkk 3 5: a6 dd
Compressed:
 0c 9b 02 58 7d 01 80 00 f1 05 12 00 80 b4 f4 08
 06 14 00 80 f1 00 fe 80 87 a6 dd
Was 50 bytes; compressed to 27 bytes, compression factor 1.85
```

Figure 8: An RPL DAO message

Figure 9 shows the effect of compressing a simple ND neighbor solicitation (again, no context-based compression).

```
IP header:
 60 00 00 00 00 30 3a ff 20 02 0d b8 00 00 00 00
 00 00 00 ff fe 00 3b d3 fe 80 00 00 00 00 00 00
 02 1c da ff fe 00 30 23
Payload:
 87 00 a7 68 00 00 00 00 fe 80 00 00 00 00 00 00
 02 1c da ff fe 00 30 23 01 01 3b d3 00 00 00 00
 1f 02 00 00 00 00 00 06 00 1c da ff fe 00 20 24
Pseudoheader:
 20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 3b d3
 fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23
 00 00 00 30 00 00 00 3a
copy: 04 87 00 a7 68
4 nulls: 82
ref(32): fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23
 -> ref 101nssss 1 2/11nnnkkk 6 0: b2 f0
copy: 04 01 01 3b d3
4 nulls: 82
copy: 02 1f 02
5 nulls: 83
copy: 02 06 00
ref(24): 1c da ff fe 00 -> ref 101nssss 0 2/11nnnkkk 3 3: a2 db
copy: 02 20 24
Compressed:
 04 87 00 a7 68 82 b2 f0 04 01 01 3b d3 82 02 1f
 02 83 02 06 00 a2 db 02 20 24
Was 48 bytes; compressed to 26 bytes, compression factor 1.85
```

Figure 9: An ND neighbor solicitation

Figure 10 shows the compression of an ND neighbor advertisement.

```
IP header:
 60 00 00 00 00 30 3a fe fe 80 00 00 00 00 00 00
 02 1c da ff fe 00 30 23 20 02 0d b8 00 00 00 00
 00 00 00 ff fe 00 3b d3
Payload:
 88 00 26 6c c0 00 00 00 fe 80 00 00 00 00 00 00
 02 1c da ff fe 00 30 23 02 01 fa ce 00 00 00 00
 1f 02 00 00 00 00 00 06 00 1c da ff fe 00 20 24
Pseudoheader:
 fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23
 20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 3b d3
 00 00 00 30 00 00 00 3a
copy: 05 88 00 26 6c c0
3 nulls: 81
ref(48): fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23
 -> ref 101nssss 1 4/11nnnkkk 6 0: b4 f0
copy: 04 02 01 fa ce
4 nulls: 82
copy: 02 1f 02
5 nulls: 83
copy: 02 06 00
ref(24): 1c da ff fe 00 -> ref 101nssss 0 2/11nnnkkk 3 3: a2 db
copy: 02 20 24
Compressed:
 05 88 00 26 6c c0 81 b4 f0 04 02 01 fa ce 82 02
 1f 02 83 02 06 00 a2 db 02 20 24
Was 48 bytes; compressed to 27 bytes, compression factor 1.78
```

Figure 10: An ND neighbor advertisement

Figure 11 shows the compression of an ND router solicitation. Note that the relatively good compression is not caused by the many zero bytes in the link-layer address of this particular capture (which are unlikely to occur in practice): 7 of these 8 bytes are copied from the pseudo header (the 8th byte cannot be copied as the universal/ local bit needs to be inverted).

```
IP header:
 60 00 00 00 00 18 3a ff fe 80 00 00 00 00 00 00
 ae de 48 00 00 00 00 01 ff 02 00 00 00 00 00 00
 00 00 00 00 00 00 00 02
Payload:
 85 00 90 65 00 00 00 00 01 02 ac de 48 00 00 00
 00 01 00 00 00 00 00 00
Pseudoheader:
 fe 80 00 00 00 00 00 00 ae de 48 00 00 00 00 01
 ff 02 00 00 00 00 00 00 00 00 00 00 00 00 00 02
 00 00 00 18 00 00 00 3a
copy: 04 85 00 90 65
ref(33): 00 00 00 00 01 -> ref 101nssss 0 3/11nnnkkk 3 4: a3 dc
copy: 02 02 ac
ref(42): de 48 00 00 00 00 01
 -> ref 101nssss 0 4/11nnnkkk 5 3: a4 eb
6 nulls: 84
Compressed:
 04 85 00 90 65 a3 dc 02 02 ac a4 eb 84
Was 24 bytes; compressed to 13 bytes, compression factor 1.85
```

Figure 11

Figure 12 shows the compression of an ND router advertisement. The indefinite lifetime is compressed to four bytes by backreferencing; this could be improved (at the cost of minor additional decompressor complexity) by including some simple runlength mechanism.

```
IP header:
 60 00 00 00 00 60 3a ff fe 80 00 00 00 00 00 00
 10 34 00 ff fe 00 11 22 fe 80 00 00 00 00 00 00
 ae de 48 00 00 00 00 01
Payload:
 86 00 55 c9 40 00 0f a0 1c 5a 38 17 00 00 07 d0
 01 01 11 22 00 00 00 00 03 04 40 40 ff ff ff ff
 ff ff ff ff 00 00 00 00 20 02 0d b8 00 00 00 00
 00 00 00 00 00 00 00 00 20 02 40 10 00 00 03 e8
 20 02 0d b8 00 00 00 00 21 03 00 01 00 00 00 00
 20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 11 22
Pseudoheader:
 fe 80 00 00 00 00 00 00 10 34 00 ff fe 00 11 22
 fe 80 00 00 00 00 00 00 ae de 48 00 00 00 00 01
 00 00 00 60 00 00 00 3a
copy: 0c 86 00 55 c9 40 00 0f a0 1c 5a 38 17
2 nulls: 80
copy: 06 07 d0 01 01 11 22
4 nulls: 82
copy: 06 03 04 40 40 ff ff
ref(2): ff ff -> ref 11nnnkkk 0 0: c0
ref(4): ff ff ff ff -> ref 11nnnkkk 2 0: d0
4 nulls: 82
copy: 04 20 02 0d b8
12 nulls: 8a
copy: 04 20 02 40 10
ref(38): 00 00 03 -> ref 101nssss 0 4/11nnnkkk 1 3: a4 cb
copy: 01 e8
ref(24): 20 02 0d b8 00 00 00 00
 -> ref 101nssss 0 2/11nnnkkk 6 0: a2 f0
copy: 02 21 03
ref(84): 00 01 00 00 00 -> ref 101nssss 0 9/11nnnkkk 3 7: a9 df
ref(40): 00 20 02 0d b8 00 00 00 00 00 00 00
 -> ref 101nssss 1 3/11nnnkkk 2 4: b3 d4
ref(120): ff fe 00 11 22
 -> ref 101nssss 0 14/11nnnkkk 3 3: ae db
Compressed:
 0c 86 00 55 c9 40 00 0f a0 1c 5a 38 17 80 06 07
 d0 01 01 11 22 82 06 03 04 40 40 ff ff c0 d0 82
 04 20 02 0d b8 8a 04 20 02 40 10 a4 cb 01 e8 a2
 f0 02 21 03 a9 df b3 d4 ae db
Was 96 bytes; compressed to 58 bytes, compression factor 1.66
```

Figure 12: An ND router advertisement

Appendix B. Things we probably won't do

This appendix documents parts of the proposal that so far have not proven themselves sufficiently using real-life packets. They may come back if they turn out to be useful; otherwise, they are to be removed on the way to RFC.

B.1. Context References

A previous version of GHC also allowed the use of context references. However, it appears that context references are more useful at the IPv6/NHC level than here - contexts that are useful often already have been unpacked into the pseudoheader, so they can be used by backreferences. So none of the examples in Appendix A strongly need this capability. Context references might be more useful if we find good ways to populate the 6LoWPAN context with certain strings that are likely to turn up in a certain LoWPAN.

code byte	Action	Argument
0110iiii	Append all bytes (possibly filling an incomplete byte with zero bits) from Context i	
0111iiii	Append 8 bytes from Context i; i.e., the context value truncated/zero-extended to 8 bytes, and then append 0000 00FF FE00 (i.e., 14 bytes total)	

B.2. Nibblecode

(It is to be decided whether the mechanism described in this section is worth its additional complexity. To make this decision, it would be useful to obtain more packet captures, in particular those that do include ASCII data - the packet-capture-based examples in Appendix A currently do not include nibblecode.)

Some headers/header-like structures, such as those used in CoAP or DNS, may use ASCII data. There is very little redundancy by repetition in these (DNS actually has its own compression mechanism for repetition), so the backreferencing mechanism provided in the bytecode is not very effective.

Efficient stateless compression for small amounts of ASCII data of this kind is pretty much confined to Huffman (or, for even more

complexity, arithmetic) coding. The complexity can be reduced significantly by moving to n-ary Huffman coding, i.e., optimizing not to the bit level, but to a larger level of granularity. Informal experiments by the author show that a 16ary Huffman coding is close to optimal at least for a small corpus of URI data. In other words, basing the encoding on nibbles (4-bit half-bytes) is both nearly optimal and relatively inexpensive to implement.

The actual letter frequencies that will occur in more general 6LoWPAN ASCII data are hard to predict. As a first indication, the author has analyzed an HTTP-based URI corpus and found the following lower case letters to be the ASCII characters that occur with highest frequency: aeinorst - it is therefore most useful to compress these.

In the encoding proposed, each byte representing one of these eight highly-compressed characters is represented by a single 4-bit nibble from the range 0x8 to 0xF. Bytes representing printable ASCII characters, more specifically bytes from 0x20 to 0x7F, are represented by both of their nibbles. Bytes from 0x00 to 0x1F and from 0x80 to 0xFF are represented by a 0x1 nibble followed by both nibbles of the byte. An 0x0 nibble terminates the nibblecode sequence and returns to bytecode on the next byte boundary.

The first nibble of the nibblecode is transmitted right in the "enter nibblecode" bytecode (0x9x - note that since it is never useful to immediately return to bytecode, the bytecode 0x90 is allocated for a different purpose). All other nibbles of the nibblecode are transmitted as a sequence of bytes in most-significant-nibble-first order; any unused nibble in the last byte of a nibblecode sequence is set to 0x0.

The encoding is summarized in Figure 13.

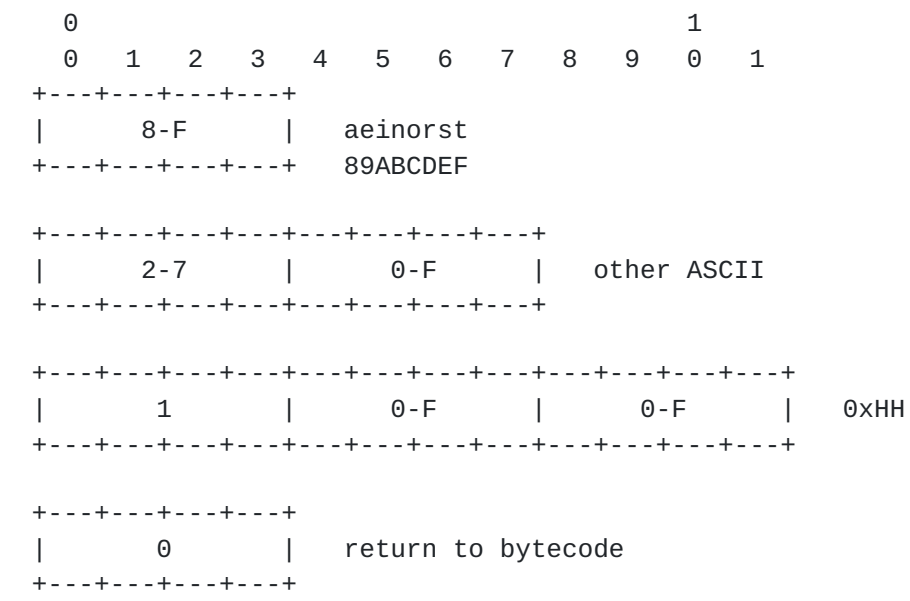

Figure 13: A nibble-based encoding

As an example for what level of compression can be expected, the 121 bytes of ASCII text shown in Figure 14 (taken from [I-D.ietf-core-link-format]) are compressed into 183 nibbles of nibblecode, which (including delimiter and padding overhead) need 93 bytes, resulting in a net compression factor of 1.30. (Note that RFC 4944/6LoWPAN-HC supports compression only in the first of a sequence of adaptation layer fragments; 93 bytes may not all fit into the first fragment, so any remaining payload would be sent without the benefit of compression.)

```
<http://www.example.com/sensors/temp123>;anchor="/sensors/temp"
;rel=describedby,
</t>;anchor="/sensors/temp";rel=alternate
```

Figure 14: Example input text (line-wrapped)

Author's Address

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

Phone: +49-421-218-63921
Fax: +49-421-218-7000
Email: cabo@tzi.org