Network Working Group
INTERNET-DRAFT
TO UPDATE: RFC 3095, 3241, 3843, 4019, 4362
Expires: May 2007

L-E. Jonsson
K. Sandlund
G. Pelletier
P. Kremer
Ericsson

November 6, 2006

RObust Header Compression (ROHC): Corrections and Clarifications to RFC 3095

<draft-ietf-rohc-rtp-impl-guide-22.txt>

Status of this memo

This document is a submission of the IETF ROHC WG. Comments should be directed to the ROHC WG mailing list, rohc@ietf.org.

Abstract

RFC 3095 defines the RObust Header Compression (ROHC) framework and profiles for IP (Internet Protocol), UDP (User Datagram Protocol), RTP (Real-Time Transport Protocol), and ESP (Encapsulated Security Payload). Some parts of the specification are unclear or contain errors that may lead to misinterpretations that may impair interoperability between different implementations. This document provides corrections, additions and clarifications to RFC 3095; this document thus updates RFC 3095. In addition, other clarifications related to RFC 3241 (ROHC over PPP), RFC 3843 (ROHC IP profile) and RFC 4109 (ROHC UDP-Lite profiles) are also provided.

Table of Contents

1. Introduction and terminology

RFC 3095 [1] defines the RObust Header Compression (ROHC) framework and profiles for IP (Internet Protocol) [8][9], UDP (User Datagram Protocol) [10], RTP (Real-Time Transport Protocol) [11], and ESP (Encapsulated Security Payload) [12]. During implementation and interoperability testing of RFC 3095 some ambiguities and common misinterpretations have been identified, as well as a few errors.

This document summarizes identified issues and provides corrections needed for implementations of RFC 3095 to interoperate, i.e. it constitutes an update to RFC 3095. This document also provides other

clarifications related to common misinterpretations of the specification. References to RFC 3095 should therefore also include this document.

In addition, some clarifications and corrections are also provided for RFC 3241 (ROHC over PPP) [2], RFC 3843 (ROHC IP-only profile) [4], and RFC 4019 (ROHC UDP-Lite profiles) [5], which are thus also updated by this document. Furthermore, RFC 4362 (ROHC Link-Layer Assisted Profile) [7] is implicitly updated by this document, since also RFC 4362 is based on RFC 3095.

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 [6].

When a section of this document makes formal corrections, additions or invalidations to text in RFC 3095, this is clearly summarized. The text from RFC 3095 that is being addressed is given and labeled "INCOMPLETE", "INCORRECT" or "INCORRECT AND INVALIDATED", followed by the correct text labeled "CORRECTED", where applicable. When text is

added that does not simply correct text in previous specifications, it is given with the label "FORMAL ADDITION".

In this document, a reference to a section in RFC 3095 [1] is written as a prefixed section reference, RFC3095-<section number>.

2. CRC calculation and coverage

2.1. CRC calculation

Section RFC3095-5.9 defines polynomials for 3, 7 and 8-bit CRCs, but it does not specify what algorithm is used. The 3, 7 and 8-bit CRCs are calculated using the CRC algorithm defined in [3].

A Perl implementation of the algorithm can be found in Appendix A of this document.

2.2. Padding octet and CRC calculations

Section RFC3095-5.9.1 is incomplete, as it does not mention how to handle the padding octet in CRC calculations for IR and IR-DYN

packets. Padding isn't meant to be a meaningful part of a packet and is not included in the CRC calculation. As a result, the CRC does not cover the Add-CID octet for CID 0, either.

INCOMPLETE RFC 3095 TEXT (section RFC3095-5.9.1):

"The CRC in the IR and IR-DYN packet is calculated over the entire IR or IR-DYN packet, excluding Payload and including CID or any Add-CID octet."

CORRECTED TEXT:

"The CRC in the IR and IR-DYN packet is calculated over the entire IR or IR-DYN packet, excluding Payload, Padding and including CID or any Add-CID octet, except for the add-CID octet for CID 0."

2.3. CRC coverage in CRC feedback options

Section RFC3095-5.7.6.3 is incomplete, as it does not mention how the "size" field is handled when calculating the 8-bit CRC used in the CRC feedback option. Since the "size" field is an extension of the "code" field, it must be treated in the same way.

INCOMPLETE RFC 3095 TEXT (section RFC3095-5.7.6.3):

"The CRC option contains an 8-bit CRC computed over the entire feedback payload, without the packet type and code octet, but including any CID fields, using the polynomial of section 5.9.1."

CORRECTED TEXT:

"The CRC option contains an 8-bit CRC computed over the entire feedback payload including any CID fields but excluding the packet type, the 'Size' field and the 'Code' octet, using the polynomial of section 5.9.1."

2.4. CRC coverage of the ESP NULL header

Section RFC3095-5.8.7 gives the CRC coverage of the ESP NULL [13] header as "Entire ESP header". This must be interpreted as including

only the initial part of the header (i.e. SPI and Sequence number), and not the trailer part at the end of the payload. Therefore, the ESP NULL header has the same CRC coverage as the ESP header used in the ESP profile (section RFC3095-5.7.7.7).

3. Mode transition

3.1. Feedback during mode transition to U- and O-mode

Section RFC3095-5.6.1 states that during mode transitions, while the D_TRANS parameter is I, the decompressor send feedback for each received packet. This restrictive behavior prevents a decompressor from using a sparse feedback algorithm during mode transitions.

To reduce transmission overhead and computational complexity (including CRC calculation) associated with feedback packets sent for each decompressed packet during mode transition, a decompressor MAY be implemented with slightly modified mode transition procedures compared to those defined in [1], as described in this section.

These enhanced procedures should be considered only as a possible improvement to a decompressor implementation, since interoperability is not affected in any way. A decompressor implemented according to the optimized procedures will interoperate with an RFC3095 compressor, as well as a decompressor implemented according to the procedures described in RFC3095.

3.1.1. Mode transition procedures allowing sparse feedback

The purpose of these enhanced transition procedures is to allow the decompressor to sparsely send feedback for packets decompressed during the second half of the transition procedure, i.e. after an appropriate IR/IR-DYN/UOR-2 packet has been received from the compressor. This is achieved by allowing the decompressor transition parameter (D_TRANS) to be set to P (Pending) at that stage, as shown in the transition diagrams of sections 3.1.2 and 3.1.3 below.

This enhanced transition, where feedback need not be sent for every decompressed packet, does however introduce some considerations in case feedback messages would be lost. Specifically, there is a risk

for a deadlock situation when a transition from R-mode is performed,

if no feedback message successfully reaches the compressor the transition is never completed. For transition between U-mode and O-mode, there is also a small risk for reduced compression efficiency.

To avoid this, the decompressor MUST continue to send feedback at least periodically, also when in Pending transition state. This is equivalent to enhancing the definition of the D_TRANS parameter in section RFC3095-5.6.1, to include the definition of a Pending state:

- D_TRANS:
  Possible values for the D_TRANS parameter are (I)NITIATED, (P)ENDING and (D)ONE. D_TRANS MUST be initialized to D, and a mode transition can be initiated only when D_TRANS is D. While D_TRANS is I, the decompressor sends a NACK or ACK carrying a CRC option for each packet received. When D_TRANS is set to P, the decompressor do not have to send a NACK or ACK for each packet received, but it MUST continue to send feedback with some periodicity, and all feedback packets sent MUST include the CRC option. This ensures that all mode transitions will be completed also in case of feedback losses.

The modifications affect transitions to Optimistic and Unidirectional modes of operation, i.e. the transitions described in sections RFC3095-5.6.5 and RFC3095-5.6.6, and make those transition diagrams more consistent with the diagram describing the transition to R-mode.

3.1.2. Transition from Reliable to Optimistic mode

The enhanced procedure for transition from Reliable to Optimistic mode is shown below:

```
            Compressor                     Decompressor
           ----------------------------------------------
                 |                               |
                 |        ACK(O)/NACK(O) +-<-<-<-| D_TRANS = I
                 |       +-<-<-<-<-<-<-<-<-+     |
     C_TRANS = P |-<-<-<-+                       |
     C_MODE = O  |                               |
                 |->->->-+ IR/IR-DYN/UOR-2(SN,O) |
                 |       +->->->->->->->->-+     |
                 |->-..                  +->->->-| D_TRANS = P
                 |->-..                          | D_MODE = O
                 |           ACK(SN,O)   +-<-<-<-|
                 |       +-<-<-<-<-<-<-<-<-+     |
     C_TRANS = D |-<-<-<-+                       |
                 |                               |
                 |->->->-+  UO-0, UO-1*          |
                 |       +->->->->->->->->-+     |
                 |                       +->->->-| D_TRANS = D
                 |                               |
```

3.1.3. Transition to Unidirectional mode

The enhanced procedure for transition to Unidirectional mode is shown on the following figure:

```
                Compressor                     Decompressor
              ----------------------------------------------
                |                              |
                |        ACK(U)/NACK(U) +-<-<-<-| D_TRANS = I
                |       +-<-<-<-<-<-<-<-+       |
  C_TRANS = P   |-<-<-<-+                      |
  C_MODE = U    |                              |
                |->->->-+ IR/IR-DYN/UOR-2(SN,U) |
                |       +->->->->->->->-+       |
                |->-..                  +->->->-| D_TRANS = P
                |->-..                         |
                |           ACK(SN,U)   +-<-<-<-|
                |       +-<-<-<-<-<-<-<-+       |
  C_TRANS = D   |-<-<-<-+                      |
                |                              |
                |->->->-+  UO-0, UO-1*         |
                |       +->->->->->->->-+       |
                |                       +->->->-| D_TRANS = D
                |                              | D_MODE= U
```

3.2. Feedback during mode transition

Section RFC3095-5.6.1 states that feedback is always used during mode transitions. However, the text then continues by making concrete applications of the rule in an inconsistent way, making it unclear when CRCs are used. Further, the text does not define how the compressor should act during mode transitions based on feedback not protected by CRCs, i.e. whether to carry out mode transition actions or not. The proper behavior from the compressor is to perform any action related to mode transitions only when the feedback is protected by the CRC option.

INCOMPLETE RFC 3095 TEXT (section RFC3095-5.6.1):

> "As a safeguard against residual errors, all feedback sent during a mode transition MUST be protected by a CRC, i.e., the CRC option MUST be used."

CORRECTED TEXT:

"As a safeguard against residual errors, all feedback sent by the decompressor during a mode transition MUST be protected by a CRC, i.e., the CRC option MUST be used. The compressor MUST ignore feedback information related to mode transition if the feedback is not protected by the CRC option."

One more related issue that requires clarifications comes from the following text at the end of section RFC3095-5.6.1:

"While D_TRANS is I, the decompressor sends a NACK or ACK carrying a CRC option for each received packet."

However, Section RFC3095-5.5.2.2 already stated that for R-mode, feedback is never sent for packets that do not update the context, i.e. for packets that do not carry a CRC such as R-0 and R-1*.

This means that when D_TRANS=I during mode transition, a decompressor operating in R-mode sends an acknowledgement for each packet it receives and MUST use the sequence number that corresponds to the packet that last updated the context, i.e. the decompressor MUST NOT use the sequence number of the R-0 or the R-1* packet."

3.3. Packet decoding during mode transition

The purpose of a mode transition is to ensure that the compressor and the decompressor coherently move from one mode of operation to another using a three-way handshake. At one point during the mode transition, the decompressor acknowledges the reception of one (or more) IR, IR-DYN or UOR-2 packet(s) that have mode bits set to the new mode. Packets of type 0 or 1 that are received up to this point are decompressed using the old mode, while afterwards they are decompressed using the new mode. If the enhanced transition procedures described in section 3.1 are used, the setting of the D_TRANS parameter to P represents this breakpoint. The successful decompression of a packet of type 0 or type 1 completes the mode transition.

4. Timestamp encoding

4.1. Encoding used for compressed TS bits

RTP Timestamp values (TS) are always encoded using W-LSB encoding, both when sent scaled and when sent unscaled. When no TS bits are transmitted in a compressed packet, TS is always scaled. If a compressed packet carries an extension 3 and field(Tsc)=0, the compressed packet must thus always carry unscaled TS bits. For TS values sent in Extension 3, W-LSB encoded values are sent using the self-describing variable-length format (section RFC3095-4.5.6), and this applies to both scaled and unscaled values.

4.2. (De)compression of TS without transmitted TS bits

When ROHC RTP operates using its most efficient packet types, apart from packet type identification and the error detection CRC, only RTP sequence number (SN) bits are transmitted in RTP compressed headers. All other fields are then omitted either because they are unchanged or because they can be reconstructed through a function from the SN

(i.e. by combining the transmitted SN bits with state information from the context). Fields that can be inferred from the SN are the IP Identification (IP-ID) and the RTP Timestamp (TS).

IP-ID compression and decompression, both with and without transmitted IP-ID bits in the compressed header, are well defined in section RFC3095-4.5.5 (see section 8.2). For the TS field, however, RFC 3095 only defines how to decompress based on actual TS bits in the compressed header, either scaled or unscaled, but not how to infer the TS from the SN when there are no TS bits present in the compressed header.

When no TS bits are received in the compressed header, the scaled TS value is reconstructed assuming a linear extrapolation from the SN, i.e. delta_TS = delta_SN * default-slope, where delta_SN and delta_TS are both signed integers. Section RFC3095-5.7 defines the potential values for default-slope.

INCOMPLETE RFC 3095 TEXT (section RFC3095-5.7):

> "If value(Tsc) = 1, Scaled RTP Timestamp encoding is used before
> compression (see section 4.5.3), and default-slope(TS) = 1.

If value(Tsc) = 0, the Timestamp value is compressed as-is, and default-slope(TS) = value(TS_STRIDE)."

CORRECTED TEXT:

"When a compressed header with no TS bits is received, the scaled TS value is reconstructed assuming a linear extrapolation from the SN, i.e. delta_TS = delta_SN * default-slope(TS).

If value(Tsc) = 1, Scaled RTP Timestamp encoding is used before compression (see section 4.5.3), and default-slope(TS) = 1.

If value(Tsc) = 0, the Timestamp value is compressed as-is, and default-slope(TS) = value(TS_STRIDE). If a packet with no TS bits is received with Tsc=0, the decompressor MUST discard the packet."

INCORRECT AND INVALIDATED RFC 3095 TEXT (section RFC3095-5.5.1.2):

"For example, in a typical case where the string pattern has the form of non-SN-field = SN * slope + offset, one ACK is enough if the slope has been previously established by the decompressor (i.e., only the new offset needs to be synchronized). Otherwise, two ACKs are required since the decompressor needs two headers to learn both the new slope and the new offset."

Consequently, there is no other slope value than the default-slope, as defined in section RFC3095-5.7.

4.3. Interpretation intervals for TS encoding

Section RFC3095-4.5.4 defines the interpretation interval, p, for timer-based compression of the RTP timestamp. However, Section RFC3095-5.7 defines a different interpretation interval, which is defined as the interpretation interval to use for all TS values. It is thus unclear which p-value to use, at least for timer-based compression.

The way this should be interpreted is that the p-value differs depending on whether timer-based compression is enabled or not.

For timer-based compression (TIME_STRIDE set to a non-zero value), the interpretation interval is:

p = 2^(k-1) - 1 (as per section RFC3095-4.5.4)

Otherwise, the interpretation interval is:

p = 2^(k-2) - 1 (as per section RFC3095-5.7)

4.4. Scaled RTP timestamp encoding

This section redefines the algorithm for scaled RTP timestamp encoding, defined as a 5-step procedure in section RFC3095-4.5.3. Two formal errors have been corrected, as described in subsections 4.4.1 and 4.4.2 below, and the whole algorithm has been reworked to be more concise and use well-defined terminology. The resulting text can be found in 4.4.3 below.

4.4.1. TS_STRIDE for scaled timestamp encoding

RFC 3095 defines the timestamp stride (TS_STRIDE) as the expected increase in the timestamp value between two RTP packets with consecutive sequence numbers. TS_STRIDE is set by the compressor and explicitly communicated to the decompressor, and it is used as the scaling factor for scaled TS encoding.

The relation between TS and TS_SCALED, given by the following equality in section RFC3095-4.5.3, defines the mathematical meaning of TS_STRIDE:

```
TS = TS_SCALED * TS_STRIDE + TS_OFFSET
```

TS_SCALED is incompletely written as TS / TS_STRIDE in the compression step following the above core equality. This formula is incorrect both because it excludes TS_OFFSET and because it would prevent a TS_STRIDE value of 0, which is an alternative not excluded by the definition or by the core equality above. If "/" were a generally unambiguously defined operation meaning "the integral part of the result from dividing X by Y", the absence of TS_OFFSET could be explained, but the formula would still lack a proper output for TS_STRIDE equal to 0. The formula of "2. Compression" is thus valid only with the following requirements:

a) "/" means "the integral part of the result from dividing X by Y"
b) TS_STRIDE>0 (TS is never sent scaled when TS_STRIDE=0)

4.4.2. TS wraparound with scaled timestamp encoding

Section RFC3095-4.5.3 states in point 4 and 5 that the compressor is not required to initialize TS_OFFSET at wraparound, but that it is required to increase the number of bits sent for the scaled TS value when there is a TS wraparound. The decompressor is also required to detect and cope with TS wraparound, including updating TS_OFFSET.

This method is not interoperable and not robust. The gain is also insignificant, as TS wraparound happens very seldom. Therefore, the compressor should reinitialize TS_OFFSET upon TS wraparound, by sending unscaled TS.

4.4.3. Algorithm for scaled timestamp encoding

INCORRECT RFC 3095 TEXT (section RFC3095-4.5.3):

```
   "1. Initialization: The compressor sends to the decompressor the
       value of TS_STRIDE and the absolute value of one or several TS
       fields. The latter are used by the decompressor to initialize
       TS_OFFSET to (absolute value) modulo TS_STRIDE. Note that
       TS_OFFSET is the same regardless of which absolute value is
       used, as long as the unscaled TS value does not wrap around;
       see 4) below.

    2. Compression: After initialization, the compressor no longer
       compresses the original TS values. Instead, it compresses the
       downscaled values: TS_SCALED = TS / TS_STRIDE. The
       compression method could be either W-LSB encoding or the
       timer-based encoding described in the next section.

    3. Decompression: When receiving the compressed value of
       TS_SCALED, the decompressor first derives the value of the
       original TS_SCALED.  The original RTP TS is then calculated as
       TS = TS_SCALED * TS_STRIDE + TS_OFFSET.

    4. Offset at wraparound: Wraparound of the unscaled 32-bit TS
       will invalidate the current value of TS_OFFSET used in the
       equation above. For example, let us assume TS_STRIDE = 160 =
       0xA0 and the current TS = 0xFFFFFFF0. TS_OFFSET is then 0x50
       = 80.  Then if the next RTP TS = 0x00000130 (i.e., the
       increment is 160 * 2 = 320), the new TS_OFFSET should be
       0x00000130 modulo 0xA0 = 0x90 = 144.  The compressor is not
       required to re-initialize TS_OFFSET at wraparound. Instead,
       the decompressor MUST detect wraparound of the unscaled TS
       (which is trivial) and update TS_OFFSET to
          TS_OFFSET = (Wrapped around unscaled TS) modulo TS_STRIDE"
```

CORRECTED TEXT:

"1. Initialization and updating of RTP TS scaling function: The compressor sends to the decompressor the value of TS_STRIDE along with an unscaled TS. These are both needed by the decompressor to initialize TS_OFFSET as hdr(TS) modulo field(TS_STRIDE). Note that TS_OFFSET is the same for any TS as long as TS_STRIDE does not change and as long as the unscaled TS value does not wrap around; see 4) below.

2. Compression: After initialization, the compressor no longer compresses the unscaled TS values. Instead, it compresses the scaled values. The compression method can be either W-LSB encoding or timer-based encoding.

3. Decompression: When receiving a (compressed) TS_SCALED, the field is first decompressed, and the unscaled RTP TS is then calculated as TS = TS_SCALED * TS_STRIDE + TS_OFFSET.

4. Offset at wraparound: If the value of TS_STRIDE is not equal to a power of two, wraparound of the unscaled 32-bit TS will change the value of TS_OFFSET. When this happens, the compressor SHOULD reinitialize TS_OFFSET by sending unscaled TS, as in 1 above."

INCORRECT AND INVALIDATED RFC 3095 TEXT (section RFC3095-4.5.3):

The entire point 5, i.e. the entire text starting from "5. Interpretation interval at wraparound ...", down to and including the block of text that starts with "Let a be the number of LSBs" and that ends with "...interpretation interval is b." is incorrect and is thus invalid.

4.5. Recalculating TS_OFFSET

TS can be sent unscaled if the TS value change does not match the established TS_STRIDE, but the TS_STRIDE might still stay unchanged. To ensure correct decompression of subsequent packets, the decompressor MUST therefore always recalculate TS_OFFSET (RTP TS modulo TS_STRIDE) when a packet with an unscaled TS value is received.

4.6. TS_STRIDE and the Tsc flag in Extension 3

The Tsc flag in Extension 3 indicates whether TS is scaled or not. The value of the Tsc flag thus applies to all TS bits, also if there are no TS bits in the extension itself. When TS is scaled, it is always scaled using context(TS_STRIDE). The legend for Extension 3 in section RFC3095-5.7.5 incorrectly states that value(TS_STRIDE) is used for scaled TS, which is incorrect.

If TS_STRIDE is present in Extension 3, as indicated by the Tss flag being set, the compressed header SHOULD carry unscaled TS bits, i.e. the Tsc flag SHOULD NOT be set when Tss is set since an unscaled TS is needed together with TS_STRIDE to recalculate the TS_OFFSET. If TS_STRIDE is included in a compressed header with scaled TS, the decompressor must ignore and discard field(TS_STRIDE).

INCORRECT RFC 3095 TEXT (section RFC3095-5.7.5):

```
   "Tsc: Tsc = 0 indicates that TS is not scaled;
         Tsc = 1 indicates that TS is scaled according to section
          4.5.3, using value(TS_STRIDE).
          Context(Tsc) is always 1.  If scaling is not desired, the
          compressor will establish TS_STRIDE = 1."
```

CORRECTED TEXT:

```
   "Tsc: Tsc = 0 indicates that TS is not scaled;
         Tsc = 1 indicates that TS is scaled according to section
         4.5.3, using context(TS_STRIDE).

         Context(Tsc) is always 1.  If scaling is not desired, the
         compressor will establish TS_STRIDE = 1.

         If field(Tsc) = 1, and if TSS = 1 (meaning that TS_STRIDE is
         present in the extension), field(TS_STRIDE) MUST be ignored
         and discarded."
```

When the compressor re-establishes a new value for TS_STRIDE using Extension-3, it should send unscaled TS bits together with TS_STRIDE.

4.7. Using timer-based compression

Timer-based compression of the RTP timestamp, as described in section RFC3095-4.5.4, may be used to reduce the number of transmitted timestamp bits (bytes) needed when the timestamp can not be inferred from the SN. Timer-based compression is only used for decompression of compressed headers that contains a TS field, otherwise when no timestamp bits are present the timestamp is linearly inferred from the SN (see section 4.2 of this document).

Whether to use timer-based compression or not is controlled by the TIME_STRIDE control field, which can be set either by an IR, an IR-DYN, or by a compressed packet with extension 3. Before timer-based compression can be used, the decompressor has to inform the compressor (on a per-channel basis) about its clock resolution by sending a CLOCK feedback option for any CID on the channel. The compressor can then initiate timer-based compression by sending (on a per-context basis) a non-zero TIME_STRIDE to the decompressor. When the compressor is confident that the decompressor has received the TIME_STRIDE value, it can switch to timer-based compression.

5. List compression

5.1. CSRC list items in RTP dynamic chain

Section RFC3095-5.7.7.6 defines the static and dynamic parts of the RTP header. This section indicates a 'Generic CSRC list' field in the dynamic chain, which has a variable length (see section RFC3095-5.8.6). This field is always at least one octet in size, even if the list is empty (as opposed to the CSRC list in the uncompressed RTP header, which is not present when the RTP CC field is set to 0).

5.2. Multiple occurrences of the CC field

The static and the dynamic parts of the RTP header are defined in section RFC3095-5.7.7.6. In the dynamic part, a CC field indicates the number of CSRC items present in the 'Generic CSRC list'. Another CC field also appears within the 'Generic CSRC list' (section RFC3095-5.8.6.1), because Encoding Type 0 is always used in the dynamic chain. Both CC fields have the same meaning: the value of the CC field determines the number of XI items in the CSRC list for Encoding Type 0, and it is not used otherwise. Therefore, the following applies:

FORMAL ADDITION TO RFC 3095:

"The first octet in the dynamic part of the RTP header contains a CC field, as defined in section 5.7.7.6. A second occurrence appears in the 'Generic CSRC list', which is also in the dynamic part of the RTP header, where Encoding Type 0 is used according to the format defined in RFC3095-5.8.6.1.

The compressor MUST set both occurrences of the CC field to the same value.

The decompressor MUST use the value of the CC field from the Encoding Type 0 within the Generic CRSC list, and it MUST thus ignore the first occurrence of the CC field."

5.3. Bit masks in list compression

The insertion and/or removal schemes, described in sections RFC3095-5.8.6.2 - 5.8.6.4, use bit masks to indicates insertion or removal positions within the reference list. The size of the bit mask can be 7-bit or 15-bit.

The compressor MAY use a 7-bit mask, even if the reference list has more than 7 items, provided that changes to the list are only applied to items within the first 7 items of the reference list, leaving items with an index not covered by the 7-bit mask unchanged.

The decompressor MUST NOT modify items with an index not covered by the 7-bit mask, when a 7-bit mask is received for a reference list that contains more than 7 items.

5.4. Headers compressed with list compression

In section RFC3095-5.8, it is stated that headers which can be part of extension header chains "include" AH [14], ESP NULL [13], minimal encapsulation (MINE) [15], GRE [16][17], and IPv6 [9] extensions. This list of headers which can be compressed is correct, but the word "include" should not be there, since only the header types listed can actually be handled. It should further be noted that for the Minimal Encapsulation (MINE) header, there is no explicit discussion of how

to compress it, as the header is either sent uncompressed or fully compressed away.

5.5. ESP NULL header list compression

Due to the offset of the fields in the trailer part of the ESP header, a compressor MUST NOT compress packets containing more than one NULL ESP [13] header, unless the second-outermost header is treated as a regular ESP [12] header and the packets are compressed using profile 0x0003.

5.6. Translation tables and indexes for IP extension headers

Section RFC3095-5.8.4 describes how list indexes are associated to list items and how table lists are built for IP extension headers. The text incorrectly states that one index per type is used, since the same type can appear several times with different content in one single chain.

In IP extension header list compression, an index is associated with each individual extension header of an extension header chain. When there are multiple non-identical occurrences of the same extension type (Protocol Number) within a header chain, each MUST be given its own index.

In the case where there are multiple identical occurrences of the same extension type, the compressor can associate them to the same index. When the value of an item whose index occurs more than once in the list is updated, the compressor MUST send the value for each occurrence of that index in the list.

When content of extension headers changes, an implementation can choose to either use a different index, or update the existing one. Some extensions can be compressed away even when some fields change, as those changes can be conveyed to the decompressor implicitly (e.g. sequence numbers in extension headers that can be inferred from the RTP SN) or explicitly (e.g. as part of the 'IP extension header(s)' field in extension 3).

When there is more than one IP header, there is more than one list of extension headers, and a translation table is maintained for each

list independently of one another.

5.7. Reference list

A list compressed using encoding type 1 (insertion), type 2 (removal) or type 3 (removal/insertion) uses a coding scheme that is based on the use of a reference list in the context (identified as ref_id).

While it could seem a fair choice to send a type 1 list when ref_id is an empty list, there is no gain in doing so with respect to using a type 0 list. Sending a type 2 list when ref_id is an empty list would lead to a failure, while sending a type 3 list has very little meaning. All these alternatives could be seen as possible, based on how list compression is specified in RFC 3095.

If these alternatives were allowed, a decompressor would become required to maintain a sliding window of ref_id lists in R-mode, even for the case where no items are sent in the compressed list, and this is not a desirable requirement. Using list encoding type 1, type 2, and type 3 is therefore only allowed for non-empty reference lists.

FORMAL ADDITION TO RFC 3095:

> "Regardless of the operating mode, for list encoding of type 1, type 2, and type 3 lists, ref_id MUST refer to a non-empty list."

5.8. Compression of AH and GRE sequence numbers

Section RFC3095-5.8.4.2 and section RFC3095-5.8.4.4 describes how to compress the Authentication Header (AH) [14] and the Generic Routing Encapsulation (GRE) [16][17] header. Both these sections present a possibility to omit the AH/GRE sequence number in the compressed header, under certain circumstances. However, the specific conditions for omitting the AH/GRE sequence number, as well as the concrete compression and decompression procedures to apply, are not clearly defined to guarantee robustness and facilitate interoperable implementation.

Proper rules are provided for the ESP case, i.e.:

> "Sequence Number: Not sent when the offset from the sequence number of the compressed header is constant, when the compressor has confidence that the decompressor has established the correct offset. When the offset is not constant, the sequence number may be compressed by sending LSBs"

The same logic applies to the AH/GRE sequence numbers.

INCORRECT RFC 3095 TEXT (section RFC3095-5.8.4.2):

"If the sequence number in the AH linearly increases as the RTP Sequence Number increases, and the compressor is confident that the decompressor has obtained the pattern, the sequence number in AH need not be sent. The decompressor applies linear extrapolation to reconstruct the sequence number in the AH."

CORRECTED TEXT:

"The AH sequence number can be omitted from the compressed header when the offset from the sequence number (SN) of the compressed header is constant, when the compressor has confidence that the decompressor has established the correct offset."

INCORRECT RFC 3095 TEXT (section RFC3095-5.8.4.4):

"If the sequence number in the GRE header linearly increases as the RTP Sequence Number increases and the compressor is confident that the decompressor has received the pattern, the sequence number in GRE need not be sent. The decompressor applies linear extrapolation to reconstruct the sequence number in the GRE header."

CORRECTED TEXT:

"The GRE sequence number can be omitted from the compressed header when the offset from the sequence number (SN) of the compressed header is constant, when the compressor has confidence that the decompressor has established the correct offset."

6. Updating properties

6.1. Implicit updates

A context updating packet that contains compressed sequence number information may also carry information about other fields; in such cases, these fields are updated according to the content of the packet. The updating packet also implicitly updates inferred fields (e.g. RTP timestamp) according to the current mode and the appropriate mapping function of the updated and the inferred fields.

An updating packet thus updates the reference values of all header

fields, either explicitly or implicitly, except for the UO-1-ID packet (see section 6.2 of this document). In UO-mode, all packets are updating packets, while in R-mode all packets with a CRC are updating packets.

For example, a UO-0 packet contains the compressed RTP sequence number (SN). Such a packet also implicitly updates RTP timestamp, IPv4 ID, and sequence numbers of IP extension headers.

6.2. Updating properties of UO-1*

Section RFC3095-5.7.3 states that the values provided in extensions carried by a UO-1-ID packet do not update the context, except for SN, TS, or IP-ID fields. However, section RFC3095-5.8.1 correctly states that the translation table in the context is updated whenever an (Index, item) pair is received, something that is contradicted by the statement in RFC3095-5.7.3 because the UO-1-ID packet can carry extension 3 with (Index, item) pair items within the 'Compressed CSRC list' field. In addition to this contradiction, the text does not mention what to do with the other sequence numbers inferred from the SN, which are also to be implicitly updated. The updating properties of UO-1* as stated by section RFC3095-5.7.3 are thus incomplete.

INCOMPLETE RFC 3095 TEXT (section RFC3095-5.7.3):

> "Values provided in extensions, except those in other SN, TS, or IP-ID fields, do not update the context."

CORRECTED TEXT:

> "UO-1-ID packets only updates TS, SN, IP-ID, and sequence numbers of IP extension headers. Other values provided in extensions do not update the context.
>
> The decompressor MUST update its translation table whenever an (Index, item) pair is received, as per Section RFC3095-5.8.1, and this rule applies also to UO-1-ID packets."

6.3. Context updating properties for IR packets

IR packets do not clear the whole context, but update all fields

carried in the IR header. Similarly, an IR without a dynamic chain simply updates the static part of the context, while the rest of the context is left unchanged.

A consequence of this is that fields that are not updated by the IR packet, e.g. the translation tables for list compression, MUST NOT be invalidated by the decompressor when it assumes context damage.

6.4. RTP padding field (R-P) in extension 3

Section RFC3095-5.7.5 defines the properties of RTP header flags and fields in extension 3. These get updated when the rtp flag of the extension 3 is set, i.e. when rtp = 1, otherwise they are not updated. However, it is unclear how extension 3 updates the R-P bit in the context.

INCOMPLETE RFC 3095 TEXT (section RFC3095-5.7.5):

"R-P: RTP Padding bit, absolute value (presumed zero if absent)."

CORRECTED TEXT:

"R-P: RTP Padding bit. If R-PT = 1, R-P is the absolute value of the RTP padding bit and this value updates context(R-P). If R-PT = 0, context(R-P) is updated to zero."

6.5. RTP eXtension bit (X) in dynamic part

Section RFC3095-5.7.7.6 defines the properties of the RTP header flags and fields in the RTP part of the dynamic chain of IR and IR-DYN packets. However, it is unclear how the X bit is updated in the context.

INCOMPLETE RFC 3095 TEXT (section RFC3095-5.7.7.6):

"X: Copy of X bit from RTP header (presumed 0 if RX = 0)"

CORRECTED TEXT:

"X: X bit from RTP header. If RX = 1, X is the X bit from the RTP header and this value updates context(X). If RX = 0, context(X) is updated to zero."

7. Context management and CID/context re-use

7.1. Persistence of decompressor contexts

As part of the negotiated channel parameters, compressor and decompressor have through the MAX_CID parameter agreed on the highest context identification (CID) number to be used. By agreeing on MAX_CID, the decompressor also agrees to provide memory resources to host at least MAX_CID+1 contexts, and an established context with a CID within this negotiated space MUST be kept by the decompressor until either the CID gets re-used, or the channel is taken down or re-negotiated.

7.2. CID/context re-use

As part of the channel negotiation, the maximal number of active contexts supported is negotiated between the compressor and the decompressor through the MAX_CID parameter. The value of MAX_CID can differ significantly from one link application to another, as well as the load in terms of the number of packet streams to compress. The lifetime of a ROHC channel can also vary, from almost permanent to rather short-lived. However, in general it is not expected that resources will be allocated for more contexts than what can reasonably be expected to be active concurrently over the link. As a

consequence hereof, context identifiers (CIDs) and context memory are resources that will have to be re-used by the compressor as part of what can be considered normal operation.

How context resources are re-used is left unspecified in RFC 3095 [1] and subsequent 3095-based ROHC specifications. This document does not intends to change that, i.e. ROHC resource management is still considered an implementation detail. However, re-using a CID and its allocated memory is not always as simple as initiating a context with a previously unused CID. Because some profiles can be operating in various modes where packet formats vary depending on current mode, care has to be taken to ensure that the old context data will be completely and safely overwritten, eliminating the risk of undesired

side effects from interactions between old and new context data. This document therefore points out some important core aspects to consider when implementing resource management in ROHC compressors and decompressors.

On a high level, CID/context re-use can be of two kinds, either re-use for a new context based on the same profile as the old context, or for a new context based on a different profile. These cases, are discussed separately in the following two subsections.

7.2.1. Re-using a CID/context with the same profile

For multi-mode profiles, such as those defined in RFC 3095 [1], mode transitions are performed using a decompressor-initiated handshake procedure, as defined in section RFC3095-5.6. When a CID/context is re-used for a new context based on the same profile as the old context, the current mode of operation SHOULD be inherited from the old to the new context. Specifically, the compressor SHOULD continue to operate using the mode of operation of the old context also with the new context. The reason for this is that there is no reliable way for the compressor to inform the decompressor that a CID/context re-use is happening. The decompressor can thus not be expected to clear the context memory for the CID (see section 6.3), and there is no way to trigger a safe mode switching (which requires the decompressor-initiated handshake procedure).

The rule of mode inheritance applies also when the CONTEXT_REINITIALIZATION signal (section RFC3095-6.3.1) is used to reinitiate an entire context.

7.2.2. Re-using a CID/context with a different profile

When a CID is re-used for a new context based on a different profile than the old context, both the compressor and the decompressor MUST start operation with that context in the initial mode of the profile (if it is a multi-mode profile). This applies both to IR-initiated new contexts and profile downgrades with IR-DYN (e.g. the profile 0x0001 -> profile 0x0002 downgrade in section RFC3095-5.11.1).

Type 0 and type 1 packets have different formats in U/O- and R-mode, and these R-mode packets have no CRC. When initiating a new context

on a re-used R-mode CID, there is a risk that the decompressor will misinterpret compressed packets, if the initiating IR packets are lost.

A CID for a context currently operating in R-mode SHOULD therefore not be re-used for a new context based on a different profile than the old context. A compressor doing otherwise should minimize the risk for misinterpretation of R-0/R-1 by e.g. not using packets of types beginning with 00 or 10 before it is highly confident that the new context has successfully been initiated at the decompressor.

8. Other protocol clarifications

8.1. Meaning of NBO

In IPv4 dynamic part (section RFC3095-5.7.7.4), if the 'NBO' bit is set, it means that network byte order is used.

8.2. IP-ID

According to section RFC3095-5.7, IP-ID means the compressed value of the IPv4 header's 'Identification' field. Compressed packets contain this compressed value (IP-ID), while IR packets with dynamic chain and IR-DYN packets transmit the original, uncompressed Identification field value. The IP-ID field always represents the Identification value of the innermost IPv4 header whose corresponding RND flag is not 1.

If RND or RND2 is set to 1, the corresponding IP-ID(s) is(are) sent as 16-bit uncompressed Identification value(s) at the end of the compressed base header, according to the IP-ID description (see the beginning of section RFC3095-5.7). When there is no compressed IP-ID, i.e. for IPv6 or when all IP Identification information is sent as-is (as indicated by RND/RND2 being set to 1), the decompressor ignores IP-ID bits sent within compressed base headers.

When RND=RND2=0, IP-ID is compressed, i.e. expressed as an SN offset and byte-swapped if NBO=0. This is the case also when 16 bits of IP-ID is sent in extension 3.

When RND=0 but no IP-ID bits are sent in the compressed header, the SN offset for IP-ID stays unchanged, meaning that Offset_m equals Offset_ref, as described in Section 4.5.5. This is further expressed in a slightly different way (with the same meaning) in Section 5.7, where it is said that "default-slope(IP-ID offset) = 0", meaning that if no bits are sent for IP-ID, its SN offset slope defaults to 0.

8.3. Extension-3 in UOR-2* packets

Some flags of the IP header in the extension (e.g. NBO or RND) may change the interpretation of fields in UOR-2* packets. In such cases, when a flag changes in Extension-3, a decompressor MUST re-parse the UOR-2* packet.

8.4. Multiple occurrences of the M bit

The RTP header part of Extension 3, as defined by section RFC3095-5.7.5, includes a one-bit field for the RTP Marker bit. This field is also present in all compressed base header formats except for UO-1-ID, meaning there may be two occurrences of the field within one single compressed header. In such cases, the two M fields must have the same value.

FORMAL ADDITION TO RFC 3095:

"When there are two occurrences of the M field in a compressed header (both in the compressed base header and in the RTP part of Extension 3), the compressor MUST set both these occurrences of the M field to the same value.

At the decompressor, if the two M field values of such a packet are not identical, the packet MUST be discarded."

8.5. Multiple SN options in one feedback packet

The length of the sequence number field in the original ESP [12] header is 32 bits. The format of the SN feedback option (section RFC3095-5.7.6.6) allows for 8 additional SN bits to the 12 SN bits of the FEEDBACK-2 format (section RFC3095-5.7.6.1). One single SN feedback option is thus not enough for the decompressor to send back all the 32 bits of the ESP sequence number in a feedback packet, unless it uses multiple SN options in one feedback packet. Section RFC3095-5.7.6.1 declares that a FEEDABCK-2 packet can contain variable number of feedback options and the options can appear in any order.

When processing multiple SN options in one feedback packet, the SN would be given by concatenating the fields.

8.6. Multiple CRC options in one feedback packet

Although it is not useful to have more than one single CRC option in a feedback packet, having multiple CRC options is still allowed. If multiple CRC options are included, all such CRC options MUST be identical, as they will be calculated over the same header, the compressor MUST otherwise discard the feedback packet.

8.7. Responding to lost feedback links

Although this is neither desirable or expected, it may happen that a link used to carry feedback between two associated instances becomes unavailable. If the compressor can be notified of such event, the compressor SHOULD restart compression for each flow that is operating in R-mode. When restarting compression, the compressor SHOULD use a different CID for each flow being restarted; this is useful to avoid that packet types for which both U/O-mode and R-mode share the same type identifier gets misinterpreted when restarting the flow in U-mode (see also section 7.2).

Generally, feedback links are not expected to disappear when once present, but it should be noted that this might be the case for certain link technologies.

8.8. UOR-2 in profile 0x0002 (UDP) and profile 0x0003 (ESP)

One single new format is defined for UOR-2 in profile 0x0002 and profile 0x0003, which replaces all three (UOR-2, UOR-2-ID, UOR-2-TS) formats from profile 0x0001. The same UOR-2 format is thus used independent of whether there are IP headers with a corresponding RND=1 or not. This also applies to the IP profile [4] and the IP/UDP-Lite profile [5].

8.9. Sequence number LSB's in IP extension headers

In section RFC3095-5.8.5, formats are defined for compression of IP extension header fields. These include compressed sequence number fields, and these fields contain "LSB of sequence number". These sequence numbers are not "LSB-encoded" as e.g. the RTP sequence number, but are the LSB's of the uncompressed fields.

8.10. Expecting UOR-2 ACKs in O-mode

Usage of UOR-2 ACKs in O-mode, as discussed in section RFC3095-5.4.1.1.2, is optional. A decompressor can also send ACKs for purposes other than to acknowledge the UOR-2, without having to continue sending ACKs for all UOR-2. Similarly, a compressor implementation can ignore UOR-2 ACKs for the purpose of adapting the optimistic approach strategies.

It is thus NOT RECOMMENDED to use of the optional ACK mechanism in O-mode, neither in compressor nor in decompressor implementations.

Using an incorrect expectation on UOR-2 ACKs as a basis for compressor behavior will significantly degrade the compression performance. This is because UOR-2 ACKs can be sent from a decompressor for other purposes than to acknowledge the UOR-2 packet, e.g. to send feedback such as clock resolution, or to initiate a mode transition. If an implementation does use the optional acknowledgment

algorithm described in Section 5.4.1.1.2, it must make sure to set the k_3 and n_3 parameters to much larger values than one to ensure that the compressor performance is not degraded due to the problem described above.

8.11. Context repairs, TS_STRIDE and TIME_STRIDE

The 7-bit CRC used to verify the outcome of the decompression attempt covers the original uncompressed header. The CRC verification thus excludes TS_STRIDE and TIME_STRIDE, as these fields are not part of the original uncompressed header.

The UOR-2 packet type can be used to update the value of the TS_STRIDE and/or the TIME_STRIDE, with the extension 3. However, these fields are not used for decompression of the RTP TS field for this packet type and their respective value is thus not verified, either implicitly or explicitly.

When the compressor receives a negative acknowledgement, it can thus not determine if the failure may be caused by an unsuccessful update to the TS_STRIDE and/or the TIME_STRIDE field(s), for which a previous header that last attempted to update their value had previously been acknowledged.

FORMAL ADDITION TO RFC 3095:

"When the compressor receives a NACK and uses the UOR-2 header type to repair the decompressor context, it SHOULD include fields that update the value of both the TS_STRIDE and the TIME_STRIDE whose value it has updated at least once since the establishment of that context, i.e. since the CID was first associated with its current profile.

When the compressor receives a static-NACK, it MUST include in the IR header fields for both the TS_STRIDE and the TIME_STRIDE whose value it has updated at least once since the establishment of that context, i.e. since the CID was first associated with its current profile."

9. ROHC negotiation

Section RFC3095-4.1 states that the link layer must provide means to negotiate e.g. the channel parameters listed in section RFC3095-5.1.1. One of these parameters is the PROFILES parameter, which is a set of non-negative integers where each integer indicates a profile supported by the decompressor.

Each profile is identified by a 16-bit value, where the 8 LSB bits indicate the actual profile, and the 8 MSB bits indicate the variant of that profile (see chapter RFC3095-8). In the ROHC headers sent over the link, the profile used is identified only with the 8 LSB

bits, which means that the compressor and decompressor must have agreed on which variant to use for each profile.

The negotiation protocol must thus be able to communicate to the compressor the set of profiles supported by the decompressor, and when multiple variants of the same profile are available, also provide means for the decompressor to know which variant will be used by the compressor. This basically means that the PROFILES set after negotiation MUST NOT include more than one variant of a profile.

10. PROFILES suboption in ROHC-over-PPP

The logical union of suboptions for IPCP and IPV6CP negotiations, as

specified by ROHC over PPP [2], can not be used for the PROFILES suboption, as the whole union would then have to be considered within each of the two IPCP negotiations, to avoid getting an ambiguous profile set. An implementation of RFC 3241 MUST therefore ensure the same profile set is negotiated for both IPv4 and IPv6 (IPCP/IPV6CP).

11. Constant IP-ID encoding in IP-only and UPD-Lite profiles

In the ROHC IP-only profile, section 3.3 of RFC 3843 [4], a mechanism for encoding of a constant Identification value in IPv4 (constant IP-ID) is defined. This mechanism is also used by the ROHC UDP-Lite profiles, RFC 4019 [5].

The "Constant IP-ID" mechanism applies to both the inner and the outer IP header, when present, meaning that there will be both a SID and a SID2 context value.

12. Security considerations

This document provides a number of corrections and clarifications to [1], but it does not make any changes with regards to the security aspects of the protocol. As a consequence, the security considerations of [1] apply without additions.

13. IANA considerations

This document does not require any IANA actions.

14. Acknowledgment

The authors would like to thank Vicknesan Ayadurai, Carsten Bormann, Mikael Degermark, Zhigang Liu, Abigail Surtees, Mark West, Tommy Lundemo, Alan Kennington, Remi Pelland, Lajos Zaccomer, Endre Szalai, Mark Kalmanczhelyi, and Arpad Szakacs for their contributions and comments. Thanks also to the committed document reviewers, Carl Knutsson and Biplab Sarkar, who reviewed the document during working group last-call.

15. References

15.1. Normative References

[1] C. Bormann, et al., "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.

[2] C. Bormann, "Robust Header Compression (ROHC) over PPP", RFC 3241, April 2002.

[3] W. Simpson, "PPP in HDLC-like Framing", RFC 1662, July 1994.

[4] L-E. Jonsson & G. Pelletier, "RObust Header Compression (ROHC): A Compression Profile for IP", RFC 3843, June 2004.

[5] G. Pelletier, "RObust Header Compression (ROHC): Profiles for User Datagram Protocol (UDP) Lite", RFC 4019, April 2005.

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

15.2. Informative References

[7] L-E. Jonsson, G. Pelletier & K. Sandlund, "RObust Header Compression (ROHC): A Link-Layer Assisted Profile for IP/UDP/RTP", RFC 4362, June 2004.

[8] J. Postel, "Internet Protocol", STD 5, RFC 791, September 1981.

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

[10] J. Postel, "User Datagram Protocol", STD 6, RFC 768, August 1980.

[11] H. Schulzrinne, S. Casner, R. Frederick & V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[12] S. Kent & R. Atkinson, "IP Encapsulating Security Payload", RFC 2406, November 1998.

[13] R. Glenn & S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, November 1998.

[14] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[15] Perkins, C., "Minimal Encapsulation within IP", RFC 2004, October 1996.

[16] D. Farinacci, T. Li, S. Hanks, D. Meyer & P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[17] G. Dommety, "Key and Sequence Number Extensions to GRE", RFC 2890, August 2000.

16. Authors' Addresses

Lars-Erik Jonsson
Ericsson AB
Box 920
SE-971 28 Lulea, Sweden
Phone: +46 8 404 29 61
EMail: lars-erik.jonsson@ericsson.com

Kristofer Sandlund
Ericsson AB
Box 920
SE-971 28 Lulea, Sweden
Phone: +46 8 404 41 58
EMail: kristofer.sandlund@ericsson.com

Ghyslain Pelletier
Ericsson AB
Box 920
SE-971 28 Lulea, Sweden
Phone: +46 8 404 29 43
EMail: ghyslain.pelletier@ericsson.com

Peter Kremer
Conformance and Software Test Laboratory
Ericsson Hungary
H-1300 Bp. 3., P.O. Box 107, HUNGARY
Phone: +36 1 437 7033
EMail: peter.kremer@ericsson.com

Appendix A - Sample CRC algorithm

```
#!/usr/bin/perl -w
use strict;
#=================================
#
# ROHC CRC demo - Carsten Bormann cabo@tzi.org 2001-08-02
#
# This little demo shows the four types of CRC in use in RFC 3095,
# the specification for robust header compression. Type your data in
# hexadecimal form and then press Control+D.
#
#---------------------------------
#
# utility
#
sub dump_bytes($) {
    my $x = shift;
    my $i;
    for ($i = 0; $i < length($x); ) {
  printf("%02x ", ord(substr($x, $i, 1)));
  printf("\n") if (++$i % 16 == 0);
    }
    printf("\n") if ($i % 16 != 0);
}

#---------------------------------
#
# The CRC calculation algorithm.
#
sub do_crc($$$) {
    my $nbits = shift;
```

```
    my $poly = shift;
    my $string = shift;

    my $crc = ($nbits == 32 ? 0xffffffff : (1 << $nbits) - 1);
    for (my $i = 0; $i < length($string); ++$i) {
      my $byte = ord(substr($string, $i, 1));
      for( my $b = 0; $b < 8; $b++ ) {
        if (($crc & 1) ^ ($byte & 1)) {
          $crc >>= 1;
          $crc ^= $poly;
        } else {
        $crc >>= 1;
        }
        $byte >>= 1;
      }
    }
    printf "%2d bits, ", $nbits;
    printf "CRC: %02x\n", $crc;
```



```
  }

  #--------------------------------
  #
  # Test harness
  #
  $/ = undef;
  $_ = <>;         # read until EOF
  my $string = ""; # extract all that looks hex:
  s/([0-9a-fA-F][0-9a-fA-F])/$string .= chr(hex($1)), ""/eg;
  dump_bytes($string);

  #--------------------------------
  #
  # 32-bit segmentation CRC
  # Note that the text implies this is complemented like for PPP
  # (this differs from 8, 7, and 3-bit CRC)
  #
  #      C(x) = x^0 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^10 +
  #             x^11 + x^12 + x^16 + x^22 + x^23 + x^26 + x^32
  #
  do_crc(32, 0xedb88320, $string);
```

```
#--------------------------------
#
# 8-bit IR/IR-DYN CRC
#
#       C(x) = x^0 + x^1 + x^2 + x^8
#
do_crc(8, 0xe0, $string);

#--------------------------------
#
# 7-bit FO/SO CRC
#
#       C(x) = x^0 + x^1 + x^2 + x^3 + x^6 + x^7
#
do_crc(7, 0x79, $string);

#--------------------------------
#
# 3-bit FO/SO CRC
#
#       C(x) = x^0 + x^1 + x^3
#
do_crc(3, 0x6, $string);
```

attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Copyright Statement

Disclaimer of Validity

This Internet-Draft expires May 6, 2007.